APPLYING PERSONA CONCEPT AND ONTOLOGY-BASED APPROACH TO SUPPORT THE REQUIREMENTS ENGINEERING PROCESS

by

Wee Wee Sim
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____ Dr. Peggy Brouse, Dissertation Director

_____ Dr. Gheorghe Tecuci, Committee Member

_____ Dr. Chun-Hung Chen, Committee Member

_____ Dr. Edward Huang, Committee Member

_____ Dr. Stephen Nash, Senior Associate Dean

_____ Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date:_____ Summer Semester 2015
George Mason University
Fairfax, VA

Applying Persona Concept and Ontology-Based Approach to Support the Requirements Engineering Process

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Wee Wee Sim
Master of Science
Wichita State University, 2004
Bachelor of Science
University of Washington, 1995

Director: Peggy Brouse, Associate Professor
Systems Engineering and Operations Research

Summer Semester 2015
George Mason University
Fairfax, VA

Copyright: 2015 Wee Wee Sim
All Rights Reserved

DEDICATION

To all my professors who had taught, inspired, and believed in me, my family in Singapore, and my best friend in Seattle whose love sustained and supported me, and most of all, exceptionally patient throughout my pursuit of Ph.D. studies.

ACKNOWLEDGEMENTS

I want to thank all my professors who had taught me during the course of my Ph.D. studies at George Mason University. In particular, I want to thank my dissertation advisor, Dr. Peggy Brouse for her wisdom, flexibility, and patience throughout my long journey of pursuing the Ph.D. studies. I also want to thank my P.D. committee members, Dr. Gheorghe Tecuci, Dr. Chun- Hung Chen, Dr. Edward Huang, and not forgetting the late Dr. Andrew Sage for showing their support and patience. Thank you all!

I want to thank my family in Singapore for giving me all the encouragement and the time to let me pursue my dream of getting a Ph.D. Thank you!

I also want to thank my best friend in Seattle who has always been supportive all these years and always says "*Peter, you can make it.*" Thank you!

Last but not least, I want to thank all those people who had assisted me in some way and in some degree during my Ph.D. studies at George Mason University. Thank you all!

TABLE OF CONTENTS

Page

List of Tables……viii
List of Figures……ix
List of Abbreviations……xii
Abstract……xiii

1. INTRODUCTION……1
1.1. Motivation ……2
1.1.1. Persona……2
1.1.2. Ontology ……5
1.2. Research Problem Statement……6
1.3. Significance of Research Problem……9
1.4. Research Hypotheses……11
1.5. Research Methodology……12
1.6. Research Contributions ……14
1.7. Organization of the Dissertation……17

2. LITERATURE REVIEW ……19
2.1. Requirements Engineering ……20
2.2. Persona-Based Methods ……21
2.3. Scenario-Based Method……27
2.4. Goal-Based Methods ……29
2.5. Task-Based Methods ……32
2.6. Ontology-Based Methods……34

3. PERSONA CONCEPT……39
3.1. What is a Persona?……39
3.2. Benefits of Personas ……40

4. ONTOLOGIES ……43
4.1. What is an Ontology? ……43
4.2. Benefits of Ontologies……44
4.3. The Protégé Knowledge Model……45
4.4. Constraint, Consistency, and Completeness Check ……48

5. PERSONA-URM: PERSONA-DRIVEN USER REQUIREMENTS MODELING METHODOLOGICAL FRAMEWORK……51
5.1. Introduction ……51
5.2. Concept Development Process (CDP) Model ……53
5.2.1. Process 1 (ring 1) – Personas Construction ……56

5.2.2. Process 2 (ring 2) – Viewpoints Identification and Construction.....59
5.2.3. Process 3 (ring 3) – Concepts Modeling.....60
5.2.4. Process 4 (ring 4) – Analysis and Evaluation.....62
5.2.5. Integration into Requirements Engineering Activities.....69
5.3. Ontology-Based Persona-Driven User Requirements Modeling (OntoPersonaURM) Model.....72
5.3.1. Ontology Development Process.....74
5.3.2. Persona Ontology.....77
5.3.3. Behavioral Goal-Scenario-Task (GST) Ontology.....81
5.3.4. Requirements Ontology.....84
6. CASE STUDY — GMU PATRIOTWEB WEBSITE.....86
6.1. Introduction.....86
6.2. CDP Model.....88
6.3. OntoPersonaURM Model.....102
7. PROTOTYPE PATRIOTWEB WEBSITE.....114
7.1. Introduction.....114
7.2. Prototype PatriotWeb Design.....117
8. RESEARCH EVALUATION, FINDINGS, AND RESULTS.....129
8.1. Introduction.....129
8.2. PART 1 Evaluation – Persona Concept.....131
8.2.1. Design of Empirical Evaluation.....132
8.2.2. PART 1 Empirical Evaluation Analysis, Findings, and Results.....135
8.3. PART 2 Evaluation – Ontology-Based Approach.....150
8.3.1. Design of Evaluation (query and rule-based).....150
8.3.2. PART 2 Evaluation Findings and Results.....155
9. CONCLUSIONS AND FUTURE WORK.....165
9.1. Conclusions.....165
9.1.1. Research Contributions.....168
9.1.2. Limitations/Challenges.....177
9.2. Future Research.....179
APPENDICES.....183
Appendix A: List of Associated Publications.....184
Appendix B: Concept Development Process (CDP) Templates.....185
Appendix C: Protégé-Frames Ontology Class Descriptions.....192
C.1. Persona Ontology Class Descriptions.....192
C.2. Behavioral-GST Ontology Class Descriptions.....206
C.3. Requirements Ontology Class Descriptions.....223
Appendix D: Case Study Screenshots.....227
Appendix E: Prototype PatriotWeb Website New Features and Improvements ..... Screenshots.....241
Appendix F: Evaluation – PART 1 Survey.....248
Appendix G: Evaluation – PART 1 Survey Screenshots.....258

Appendix H: Evaluation– PART 1 Survey Summarized Data 277
Appendix I: Evaluation– PART 1 Survey Evaluation Results 284
Appendix J: Evaluation– PART 2 PAL Constraints, Jess Consistency and Completeness Rules 285
J.1. PAL Constraints and Jess Rules Natural Language Statements..285
J.2. PAL Constraints and Jess Rules 299
Appendix K: Evaluation – PART 2 Evaluation Tests Results.....376
List of References 377
Biography.....386

LIST OF TABLES

Table **Page**

Table 1. Eight Usability Factors for Scenarios-Usability Goals Evaluation (SUGE)….66
Table 2. Integration of CDP Processes into RE Activities 69
Table 3. Ontology Development Process.....75
Table 4. Criteria of Categorizing Survey Respondents (Group A, Group B).....136
Table 5a. Initial Categorization of Survey Respondents (Group A, Group B).....138
Table 5b. Final Categorization of Survey Group A 139
Table 6. PART 2 Evaluation Tests Results (First Round) 164
Table 7. PART 2 Evaluation Tests Results (Second Round).....164

LIST OF FIGURES

Figure **Page**
Figure 1. "AS-IS" Relationships among Role, Scenarios, and Personas 3
Figure 2. "TO-BE" Relationships among Role, Personas, and Scenarios 4
Figure 3. The Composition of Research Background 19
Figure 4. Concept Development Process (CDP) Model 54
Figure 5. Persona Profile Document (PPD) Template 57
Figure 6. Persona Definition Document (PDD) Template 58
Figure 7. Viewpoint Document (VPD) Template 61
Figure 8. Personas-Viewpoints-Requirements Matrix 63
Figure 9. Personas-Viewpoints-Scenarios Matrix Template 65
Figure 10. Scenarios-Usability Goals Matrix Template 67
Figure 11. Scenarios-Tasks-Requirements Matrix Template 68
Figure 12. OntoPersonaURM Model – Persona Ontology, Behavioral-GST Ontology, Requirements Ontology 74
Figure 13. UML Class Diagram – Persona Ontology 78
Figure 14. Protégé-Frames – Persona Ontology 79
Figure 15. Interest Class – Persona Ontology 81
Figure 16. UML Class Diagram – Behavioral-GST Ontology 82
Figure 17. Protégé-Frames – Behavioral-GST Ontology 83
Figure 18. UML Class Diagram – Requirements Ontology 85
Figure 19. Protégé-Frames – Requirements Ontology 85
Figure 20. Case Study – PatriotWeb Login Page 87
Figure 21. Case Study – PatriotWeb Homepage 87
Figure 22. Case Study – PPD for Primary Persona Linda Rose 89
Figure 23. Case Study – PDD for Primary Persona Linda Rose 89
Figure 24. Case Study – VPD for Primary Persona Linda Rose 93
Figure 25. Case Study – Personas-Viewpoints-Requirements Matrix for PVRA Activity 95
Figure 26. Case Study – Scenarios-Usability Goals Matrix for SUGE Activity 101
Figure 27. Case Study – UML Instance Diagram for Persona Ontology 104
Figure 28. Case Study – UML Instance Diagram for Behavioral-GST Ontology 105
Figure 29. Case Study – Protégé-Frames View of the Instances of Persona Class (Persona Ontology) 105
Figure 30. Case Study – Protégé-Frames View (Partial View) of the Instances of Goal Class (Behavioral-GST Ontology) 106
Figure 31. Case Study – Protégé-Frames View (Partial View) of the Instances of

Requirement Class (Requirements Ontology) 106
Figure 32. Case Study – Protégé-Frames' Project Menu.....107
Figure 33. Case Study – Protégé-Frames' Manage Included Projects.....107
Figure 34. Case Study – Persona Ontology and Behavioral-GST Ontology Class Relationship.....109
Figure 35. Case Study – Persona Class and Viewpoint Class isPersonaOf/hasPersona Relationships.....110
Figure 36. Case Study – Combined Persona Ontology (Current Ontology) and Behavioral-GST Ontology (Included Ontology) in Protégé-Frames.....111
Figure 37. Case Study – Combined Behavioral-GST Ontology (Current), Persona Ontology (Included), and Requirements Ontology (Included) in Protégé-Frames 112
Figure 38. Case Study – Combined Requirements Ontology (Current) and Behavioral-GST Ontology (Included) in Protégé-Frames.....113
Figure 39. Case Study – Prototype PatriotWeb Website – New Improved Navigational System.....118
Figure 40. Case Study – Prototype PatriotWeb Website – New Improved Buttons (Registration Page).....119
Figure 41. Case Study – Prototype PatriotWeb Website – New Improved Buttons (Student Records Page).....120
Figure 42. Case Study – Prototype PatriotWeb Website – New Improved Visibility and Layout of Services Links (Main Menu Page) 121
Figure 43. Case Study – Prototype PatriotWeb Website – New Improved Visibility and Layout of Services Links (Registration Page) 122
Figure 44. Case Study – Prototype PatriotWeb Website – New Improved Visibility and Layout of Services Links (Student Records Page) 122
Figure 45. Case Study – Prototype PatriotWeb Website – New "View and Update Personal Information" Link (Personal Information Page) 124
Figure 46. Case Study – Prototype PatriotWeb Website – "Select Term" Link.....125
Figure 47. Case Study – Prototype PatriotWeb Website – New "Return to Courses Page" Button.....126
Figure 48. Case Study – Prototype PatriotWeb Website – New "Return to Subjects Page" Button.....127
Figure 49. Case Study – Prototype PatriotWeb Website – Advanced Search Field Options Panel.....128
Figure 50. Evaluation and Hypotheses Validation 130
Figure 51. Evaluation – "Student Services" Page of Prototype PatriotWeb.....133
Figure 52. Evaluation – "Registration" Page of Prototype PatriotWeb.....134
Figure 53. Evaluation – Survey Question 21 142
Figure 54. Evaluation – Survey Question 21 Responses 143
Figure 55. Evaluation – Survey Question 22 Responses 145
Figure 56. Evaluation – Survey Question 20 Responses 146
Figure 57. Evaluation – Normalized Scores for Survey Question 21 and

Question 22 149
Figure 58 Evaluation – PAL Constraint Statement Example 152
Figure 59. Evaluation – Jess Consistency Rule Example 1 153
Figure 60. Evaluation – Jess Consistency Rule Example 2 153
Figure 61. Evaluation – Jess Completeness Rule Example 154
Figure 62. Evaluation – PAL Constraint Violation Example 1a 155
Figure 63. Evaluation – PAL Constraint Violation Example 1b 157
Figure 64. Evaluation – Inconsistent Class Instances 158
Figure 65. Evaluation – Jess Rule Execution Result for Inconsistency 159
Figure 66. Evaluation – Incomplete Class Attribute Instance 161
Figure 67. Evaluation – Jess Rule Execution Result for Incompleteness 161

LIST OF ABBREVIATIONS

CDP ..... Concept Development Process
CRR ..... Conflict Requirements Resolution
CTT ..... ConcurTaskTrees
DL ..... Description Logic
GBRAM ..... Goal-Based Requirements Analysis Model
GST ..... Goal-Scenario-Task
HCL ..... Human-Computer Interaction
HTA ..... Hierarchical Task Analysis
IEEE ..... Institute of Electrical and Electronics Engineers
Jess ..... Java Expert System Shell
KAOS ..... Knowledge Acquisition in autOmated Specification
OntoPersonaURM ..... Ontology-Based Persona-Driven User Requirements Modeling
OWL ..... Web Ontology Language
PAL ..... Protégé Axiom Language
PPD ..... Persona profile Definition
PDD ..... Persona Definition Document
Persona-URM ..... Persona-Driven User Requirements Modeling
PSG ..... Persona-Scenario-Goal
PVRA ..... Personas-Viewpoints-Requirements Analysis
PVSA ..... Personas-Viewpoints-Scenarios Analysis
RE ..... Requirements Engineering
SDLC ..... Software Development Life Cycle
SGI ..... Scenario-Goal-Issue
SRS ..... System Requirements Specification
STRE ..... Scenarios-Tasks-requirements Evaluation
SUGE ..... Scenarios-Usability Goals Evaluation
SWRL ..... Semantic Web Rule Language
UI ..... User Interface
UCM ..... Use Case Maps
UML ..... Unified Modeling Language
VP ..... Viewpoint
VPB ..... Viewpoint Block
VPD ..... Viewpoint Definition Document

ABSTRACT

APPLYING PERSONA CONCEPT AND ONTOLOGY-BASED APPROACH TO SUPPORT THE REQUIREMENTS ENGINEERING PROCESS

Wee Wee Sim, Ph.D.

George Mason University, 2015

Dissertation Director: Dr. Peggy S. Brouse

One of the challenges in requirements engineering is how to achieve a better understanding of the target users' needs and behaviors, and refine requirements early in the requirements engineering process to meet the projects' intended results. There is a poor shared semantic agreement among engineers on target users that hinders the requirements engineering activities. Poor or inadequate understanding of users' requirements increases the chance of not meeting users' needs. Well understanding, explicit specification, and common sharing of users' knowledge and information are crucial in the success of the requirements engineering projects.

The objective and contribution of this research is to create a framework to: (1) enhance the requirements engineering process by applying the persona concept to the requirements engineering activities in the context of the concepts of viewpoint, goal, scenario, task, and requirement that can be integrated in a unified environment to enable

developers to gain a better understanding of target users' needs and behaviors and refine requirements early in the requirements engineering process, and (2) apply an ontology-based approach to explicitly represent specification of concepts to improve the design model quality requirements of the ontologies through applying constraint, consistency, and completeness check on concepts, properties, and relationships between concepts in the knowledge base.

The effectiveness of persona concept and the ontology approach is evaluated and validated through conducting qualitative evaluation, empirical experiment, and query-rule-based evaluation techniques via the GMU PatriotWeb website case study and the implementation of the prototype PatriotWeb system. The results of the case study evaluations indicate that (1) the application of the concept of persona contributes positively to a better understanding of target users' needs and behaviors and the refinement of requirements, and (2) the explicit specification of concepts in the ontologies through the application of the ontology-based approach enables checking of constraints, consistencies, and completeness of concepts, properties, and relationships between concepts in the knowledge base and contributes to improving the design model quality requirements of the ontologies.

CHAPTER 1: INTRODUCTION

It has been widely acknowledged that one of the contributing factors in most requirements engineering projects is the failure to meet intended results because of poor understanding of target users' needs and behaviors to achieve projects' requirements [GAO 08, VERNER 05]; mainly, there is a poor semantic agreement and understanding among users that hinder requirements engineering activities. Majority of software applications place too much emphasis on the tasks or functionalities these applications have to perform or provide, while overlooking the actual users' needs and goals [RANDOLPH 04]. This is evident in most mass-users web-based application systems, as well as in digital consumer products such as cell phones, digital cameras, remote control, and so forth.

Poor understanding of users' requirements increases the risk of missing users' needs. Focusing solely on the tasks or functionalities without considering the real users' needs and goals is a recipe for failure. As such, well understood, explicit formal specifications, and the common sharing of users' knowledge and information are crucial in the success of the requirements engineering projects. The users' knowledge and information are captured as *personas*, i.e. user profiles such as knowledge, skills, abilities, goals, and usage patterns (scenarios, tasks).

1.1. Motivation

1.1.1. Persona

The concept of persona, originally presented by Cooper [COOPER 99] focused on the use of personas, their goals and scenarios on design, is becoming a promising and an emergent new paradigm in user requirements modeling. Personas are fictitious but concrete representations of a group of target users. They are constructed to resemble *real* people, i.e. they contain information such as names, ages, educational backgrounds, occupations, skills, goals, concerns, usage patterns on the system, and so forth.

Personas are defined primarily not only by the goals the users have on the system, products or services, but also by the behaviors and attitudes of the users. Personas capture rich behavior models of users and help requirements engineers to obtain a deeper understanding of the target users and make better decisions based on these personas. Instead of asking "*How should a feature on the application work*?" engineers can ask "*What would John Doe want to do in this application?*"

In the traditional software development process in **UML** (**U**nified **M**odeling **L**anguage) modeling, actors are used to represent users' *roles*. A role represents a homogeneous view of the users of the system; it describes the relationships (or interactions) between the users and the system. This is in contrast to persona, which describes rich information about user, i.e. age, gender, knowledge, skills, abilities, goals, usage patterns, and so forth. The users' roles in the UML use case model can be further refined by providing personas. For example, a user role "developer" can be refined to two personas, i.e. *novice persona*, *expert persona*. These personas have different scenarios,

tasks, and requirements. Personas are complimentary to the scenario-based approach to requirements engineering. Scenarios are often constructed around personas. For example, Figure 1 illustrates the "*AS-IS*" relationships among a user role, scenarios, and personas for the "*Perform Module Test*" use case.

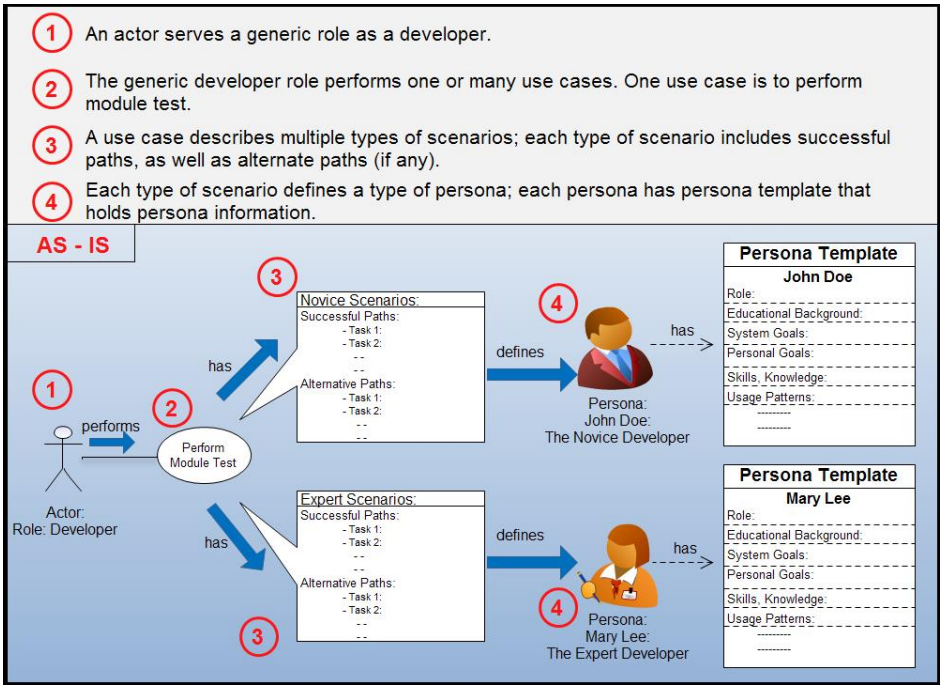


Figure 1. "*AS-IS*" Relationships among Role, Scenarios, and Personas

Figure 1 shows that there are two different types of developers, a novice developer and an expert developer, which correspond to novice persona and expert persona respectively. These two personas have different scenarios (i.e. novice scenario and expert scenario), but the goal for both scenarios is the same, i.e. to "*Perform Module Test*". However the implementation of the two scenarios may be very different.

In the UML use case role-model, the knowledge level of a generic role "Developer" is unclear. There is one actor but multiple personas. The knowledge level of the generic developer actor may not be the same as one of the many personas that plays the role, and thus the scenarios generated may be different. In this case, the use case role-model does not provide sufficient set or coverage of scenarios. By refining the role "Developer" into personas, for examples: "*John Doe, the Novice Developer*" and "*Mary Lee, the Expert Developer*", a greater understating of the needs and behaviors of the users can emerge **early** during the requirements analysis process. Figure 2 illustrates the persona-model, the "*TO-BE*" model to be adopted in this research. It illustrates the "*TO-BE*" relationships among user role, personas, and scenarios for the same "*Perform Module Test*" use case.

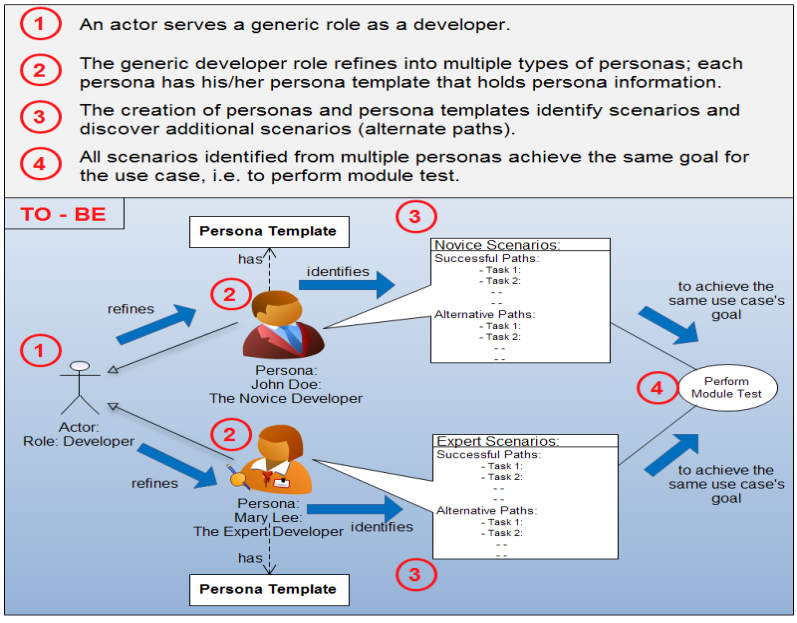


Figure 2. "*TO-BE*" Relationships among Role, Personas, and Scenarios

Both user roles and personas are effective means for capturing and conveying understanding about users, but they differ in one important aspect in that personas describe users while user roles describe relationships between users and systems; personas represent real people while user roles do not. By exploring personas, we can understand our target users' behaviors better and thus help refine requirements.

1.1.2. Ontology

In user-centered requirements engineering design, models guide the designers or developers throughout the process. The nature of requirements engineering involves capturing knowledge from many sources. The common term used in the field of knowledge representation is known as *ontology*, which is a formal representation of the entities and relationships that exist in some domain of interest. One of the most concise and widely cited ontology definitions is provided by Gruber [GRUBER 93a]:

> **Definition 1 –** An *ontology is a formal, explicit specification of a shared conceptualization. Conceptualization means an abstract simplified view of the world, explicit means all concepts and constraints are clearly defined, formal means that the ontology should be machine processable, and shared means the ontology captures consensual knowledge.*

The study of ontology has become a very popular field, not only in AI but also in other disciplines of science and technology. Ontology is all about defining the domain vocabularies, the essential concepts in the domain, their classifications, taxonomies (concept hierarchies), relationships among the concepts (including constraints), and

domain axioms related to a particular application domain. Nowadays, ontology technologies are frequently used in many application domains, as the concepts, relationships, and their categorizations in a real world can be represented in ontology which can be used as resources of domain knowledge in a specific application domain.

As ontologies are useful for representing and interrelating many types of knowledge, ontology-based modeling approach in requirements engineering has become very useful and beneficial in representing requirements model, the application domain, and the environment. In requirements engineering, ontologies not only offer the capability of representing knowledge and interrelating different types of knowledge, but also provide inference mechanism to detect inconsistency and incompleteness to help improve the design model quality requirements. The commonly identified principles in the evaluation of ontology quality literature [Gomez-Perez 94, Guasrino 02] are: consistency, completeness, conciseness, and reusability, as well as the IEEE standard [IEEE 11 a]: correctness, completeness, consistency, unambiguity, and so forth. In this dissertation, we applied consistency, completeness, and constraint checks as the measures for the quality of the underlying ontologies in the knowledge base.

1.2. Research Problem Statement

As mentioned in the introductory paragraph in this chapter, one of the main reasons why most software requirements projects fail to meet the customer's requirements and expectations is because of the failure to identify and understand real users' needs and behaviors in the early phase of the requirements engineering process [GAO 08, VERNER

05]. There is a poor semantic agreement and understanding among users that hindered the requirements engineering activities. The poor understanding of users' needs and behaviors increases the chance of not meeting users' needs and consequently results in project failure.

In requirements engineering and software development processes, a role captures the specific behavior of an entity (i.e. user) participating in a given domain. In UML software development, actors are used in the use case models to represent users' roles. However, within a given role, there may be many different types of users. These users' roles can be further refined by providing personas. For example, a user role "developer" can be refined to two personas, i.e. *novice persona*, *expert persona*; or it may be refined to *X persona*, representing those developers who like to build systems by looking at and following examples of codes, and *Y persona*, representing those developers who like to build systems by writing algorithms from scratch. These personas exhibit different scenarios, tasks, goals, and/or requirements. By exploring the concept of persona, we can understand the users better and thus help refine requirements.

The issues addressed above lead to the following main research problem (**RP**):

RP – ***The poor understanding of target users' needs and behaviors and the poor semantic agreement among developers on users' knowledge lead to an increased risk of failure to meet project's requirements.***

The main research problem addressed above is a general research problem. To investigate this problem further, the general research problem can be broken down into the following three sub-research problems (**S-RP**):

S-RP1 – ***The poor understanding of target users' needs and behaviors leads to the risk of not meeting project's intended results.***

S-RP2 – ***The failure to discover real users' needs and behaviors leads to the chance of failure to refine requirements.***

S-RP3 – ***The poor explicit formal specification and common understanding of users' knowledge leads to the risk of producing inconsistent and incomplete design model and requirements.***

The identification of the above research problems enables us to formulate the critical research questions. In this research, the following three research questions (**RQ**) are observed and addressed:

RQ1 – ***How can personas, in the context of scenarios, tasks, goals, and requirements, help to understand target users' needs and behaviors?***

RQ2 – ***How can personas, in the context of scenarios, tasks, goals, and requirements, help to refine requirements?***

RQ3 – ***How can ontology-based approach of defining an explicit specification of users' knowledge helps to enhance or improve the design model quality requirements of the ontologies by providing constraint, consistency, and completeness check?***

1.3. Significance of Research Problem

In requirements engineering and software development process, the "role" plays a primary vehicle for understanding the users. The role is captured through the use of actors in the use case model in UML modeling. In some application domains such as mobile phones, where there are potentially a wide array of unknown users, the UML role model may not be enough to allow engineers, analysts, and developers to develop a deeper understanding of the users of the application. This is because a role is a homogeneous view of the users of the system. Within a given role, there may be many different types of users. For example, in a role "*gamer*" (i.e. users who play computer games), the "role" view would treat all gamers the same manner; yet some gamers are heavy game players whereas some are light game players. By using role model to model users, there are potential risks in producing missing or incomplete scenarios, tasks, goals, and/or requirements, and consequently resulting in the failure to meet the project's intended results. Therefore, there is a need to develop a better understanding of the target's users in order to refine requirements.

The concept of persona is an emergent new paradigm in user modeling. Unlike roles which describe the relationships between users and system, personas describe users, i.e. they contain information about the users' knowledge, skills, abilities, goals, concerns,

usage patterns, and so forth. The roles in the use case model can be further refined by providing personas. For the "*gamer*" example, there may be a "*heavy gamer*" persona and a "*light gamer*" persona. These personas present different scenarios, tasks, goals, and/or requirements. Therefore, by exploring personas, we can find missing pieces in a scenario during the gathering of requirements and thus help refine requirements.

In a software requirements project, missing critical requirements (functional or nonfunctional requirements) could well be the greatest risk in requirements engineering. Example of a missing functional requirement could be an important user class or user need. The cost of committing this mistake is usually larger than what most stakeholders can tolerate, and in some worst cases, the mistake of overlooking missing critical requirements could cause the project to start over from scratch. However, requirements defects such as conflicting, inconsistence, incomplete, and/or missing requirements do exist and are often unavoidable in most software projects. An effective way to identify these requirements defects is to inspect and detect them at the earliest possible time, as the cost to remove requirements defects once the software product is delivered into the market increases geometrically with time.

Ontologies not only offer knowledge representation and interrelating different types of knowledge, but also provide inference mechanism to detect inconsistency and incompleteness to help improve the design model quality requirements of the ontologies. As the nature of requirements engineering involves capturing knowledge from different sources, ontologies are deemed to be very useful and beneficial in representing the requirements model, the application domain, and requirements traceability (i.e. detecting

inconsistency and incompleteness) via inference engine. The application of ontology enables engineers to represent, organize, share, reuse, and reason over the complex sets of knowledge that requirement documents embody.

This research is motivated by the significance of the problems addressed above. By carefully exploring the concept of persona and investigating its relationship with scenarios, tasks, goals, and requirements, engineers, analysts, and developers will be able to gain a better understanding of the target users' needs and behaviors. By adopting an ontological representation and specification of users' knowledge, engineers, analysts, and developers will be able to not only share common understanding of this common knowledge, but may also detect constraint violations, inconsistencies, and incompleteness in the design model of the ontologies using constraints and inference mechanism.

1.4. Research Hypotheses

The primary objective of this research is to investigate persona driven user requirements modeling using an ontology-based approach. Specifically, this research asserts the following three research hypotheses (**H**):

> **H1 –** ***The concept of persona and its relationships with scenarios, tasks, goals, and requirements help to understand target users' needs and behaviors.***

> **H2 –** ***The concept of persona, in the context of scenarios, tasks, goals, and requirements helps to refine requirements.***

H3 – ***Using an ontology-based approach to define an explicit specification of users' knowledge (i.e. personas, scenarios, tasks, goals, and requirements) helps to improve the design model quality requirements of the ontologies by providing checking of constraints, consistencies, and completeness of concepts, properties, and relationships between concepts in the knowledge base.***

Qualitative, empirical, and query-and-rule-based evaluations (chapter 8) were conducted to validate the three research hypotheses via the George Mason University PatriotWeb case study (chapter 6) and the implementation of the prototype PatriotWeb system (chapter 7) to determine whether the research approach could be supported by these hypotheses.

1.5. Research Methodology

With its knowledge representation and inference capability, ontology-based approach to user modeling is emerging and a good choice to represent knowledge about users, such as users' behaviors, scenarios, tasks, goals, and requirements. Over the past several years, there have been several works conducted by researchers on scenarios [ALSPAUGH 08, LIU 04, SUTCLIFFE, 98] and goal modeling [ANTON 96, DARDENNE 93, ROLLAND 98, VAN LAMSWEERDE 01, VAN LAMSWEERDE 09, YU 05, YU 11a, YU 11b], as well as ontology-based scenarios and goal requirements modeling [SHIBAOKA 07, KAIYA 02, KAIYA 05, KAIYA 06]. Limited work has been done on investigating the concept of persona in requirements engineering. A few have proposed

techniques to identify personas and investigated their relationships with scenarios and goals [GRUDIN 02, PRUITT 03, AOYAMA 05, AOYAMA 07]. However, there has been no similar work using an ontology-based approach to represent and integrate the concepts of persona, scenario, task, goal, and requirement.

This research presents the **Persona**-Driven **U**ser **R**equirements **M**odeling (**Persona-URM**) methodological framework for user requirements analysis and modeling in the requirements engineering process. The Persona-URM framework distinguishes from previous work conducted by other researchers in that it aims to investigate how the concepts of persona, scenario, task, goal, and requirement can be integrated in a unified environment and how the relationships among these concepts can be specified explicitly using an ontology-based approach that will help (1) to provide a better understanding of users' needs and behaviors and help refine requirements, and (2) to improve the quality requirements of design model of the ontologies in the knowledge base through checking of constraints, consistencies, and completeness of concepts, properties, and relationships between concepts in the knowledge base. The three generic interrelated ontologies developed in this research are: Persona Ontology, Behavioral **G**oal-**S**cenario-**T**ask (**GST**) Ontology, and Requirements Ontology. The three domain independent ontologies are designed and constructed to provide an explicit specification of the concepts and their properties, i.e. attributes, cardinalities, and relationships among personas, scenarios, tasks, goals, requirements, and other relevant concepts. In addition, constraints and inference rules are applied to the concepts, properties, and relationships between concepts to detect constraint violations, inconsistencies, and incompleteness of concepts,

properties, and relationships between concepts so as to improve the underlying design model quality requirements of the ontologies.

This research captures some of the existing and past well-known user requirements modeling techniques conducted by other researchers. These techniques include scenario-based, goal-based, and task-based modeling methods. This research applies some of these relevant techniques and ideas with the goal that the result of this research will serve as a complimentary or enhancement to these works.

In this research, to validate the stated three hypotheses, a George Mason University PatriotWeb case study was studied and evaluated against a prototype PatriotWeb on selected scenarios. This case study was chosen because of its wide variety of users with different backgrounds, requirement needs, as well as potential changing requirements. The case study demonstrated the research methodology and ontological specifications of the concepts. Experimental investigations and execution of constrains and inference rules were conducted on the case study to validate the effectiveness of the persona concept (hypotheses H1 and H2) and the ontology-based approach (hypothesis H3).

1.6. Research Contributions

The underlying themes for this research focus on two aspects:

(1). The analysis and modeling of user requirements in requirements engineering, i.e. personas, viewpoints, scenarios, tasks, goals, and requirements modeling.

(2). An explicit ontological specification of users' knowledge (i.e. personas, scenarios, tasks, goals, requirements, and other pertaining concepts).

As the concept of persona is a relatively new paradigm in user modeling and software requirements engineering, limited work has been conducted on proposing techniques to identify personas and investigate its relationships with scenarios and goals. However, there has been no similar work using an ontology approach to represent the concepts of persona, viewpoint, scenario, task, goal, and requirement.

This research distinguishes from other previous works conducted by other researchers in that:

(1). It investigates and integrates the relationships of the concepts of persona, viewpoint, scenario, task, goal, and requirement;

(2). It represents and specifies these concepts using an ontology-based approach by constructing three domain independent ontologies; specifically, the design and construction of Persona Ontology, Behavioral **G**oal-**S**cenario-**T**ask (**GST**) Ontology, and Requirements Ontology, and applies constraints and inference rules to detect constraint violations, inconsistencies, and incompleteness of concepts, properties, and relationships between concepts to help improve the design model quality requirements of the ontologies in the knowledge base.

Formally, this research aims to offer two major contributions:

Contribution 1 – *An improvement in user requirements modeling by applying the concept of persona, in the context of scenarios, tasks, goals, and*

requirements, to achieve a better understanding of users' needs and behaviors and help in refining requirements .

Contribution 2 – *An ontology-based methodology to user requirements modeling by building ontologies to provide explicit ontological specifications of user's knowledge that will enhance shared understanding of common knowledge and allow for constraints, consistencies, and completeness validation to improve the design model quality requirements of the ontologies in the knowledge base.*

The outcome of this research is the development of **Persona**-Driven **U**ser **R**equirements **M**odeling (**Persona-URM**) methodological framework for user requirements analysis and modeling to accomplish the following:

(1). A **C**oncept **D**evelopment **P**rocess (**CDP**) model to help guide the development of the concepts of persona, viewpoint, scenario, task, goal, and requirement.

(2). An **Onto**logy-Based **Persona**-Driven **U**ser **R**equirements **M**odeling (**OntoPersonaURM**) model that includes the design and construction of three domain independent interrelated ontologies: Persona Ontology, Behavioral **G**oal-**S**cenario-**T**ask (**GST**) Ontology, and Requirements Ontology to represent explicitly the specifications and capture the relationships among the concepts.

(3). The ability to detect constraint violations, inconsistencies, and incompleteness of concepts, properties, and relationships between concepts in the knowledge base with the application of constraints check and inference rules to help improve the underlying design model quality requirements of the ontologies.

(4). An evaluation of the effectiveness of applying the concept of persona and ontology-based approach via the GMU PatriotWeb case study and prototype PatriotWeb system to validate the research hypotheses asserted in this dissertation.

The contributions that this research brings are explained more in detail in chapter 9 of this dissertation.

1.7. Organization of the Dissertation

The dissertation is organized in eight chapters (including this Introduction chapter) as follows:

- Chapter 2 provides a survey of the current and past relevant research in persona, goal, scenario, task, ontology, and requirements engineering.
- Chapter 3 describes the concept of persona. Specifically, what is a persona and what are the benefits of applying persona.
- Chapter 4 describes ontologies. Specifically, the definitions of ontology, the benefits of using an ontology-based approach, and the inference mechanisms to check consistency and completeness in the knowledge base.
- Chapter 5 introduces the Persona-Driven User Requirements Modeling (Persona-URM) methodological framework developed in this research; specifically, the Concept Development Process (CDP) model and Ontology-Driven User Requirements Modeling (OntoPersonaURM) model.
- Chapter 6 presents the George Mason University PatriotWeb case study which served as the base for the evaluations and validations of the research hypotheses.

- Chapter 7 presents the prototype of the PatriotWeb system and demonstrates the design of the new features and improvements of the current PatriotWeb system.
- Chapter 8 describes the research evaluation methods (via the GMU PatriotWeb case study and prototype PatriotWeb system) and the evaluation results to support the stated hypotheses.
- Chapter 9 discusses conclusions, research contributions, research limitations/challenges, and highlights some future research work.

CHAPTER 2: LITERATURE REVIEW

The main theme of this research is concentrated in two categories:

(1). User requirements analysis and modeling in requirements engineering, i.e. personas, viewpoints, scenarios, tasks, goals, and requirements analysis and modeling.

(2). Ontological representation and specification of users' and domain knowledge (i.e. personas, scenarios, tasks, goals, requirements, and other pertaining concepts).

As shown in Figure 3, the dissertation builds upon current and past research from a variety of methods surrounding the above two categories. These methods include persona-based methods, scenario-based methods, task-based methods, goal-based methods, and ontology-based methods.

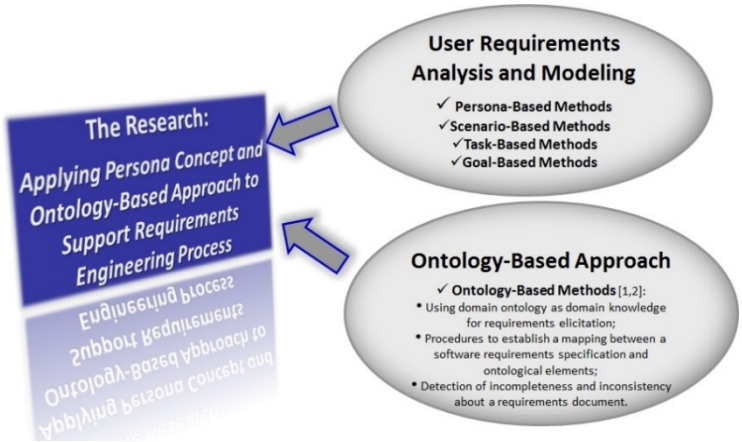


Figure 3. The Composition of Research Background

2.1. Requirements Engineering

Requirements engineering (RE) is a multi-disciplinary, human-centered process. It is viewed both at the software level and systems level. Due to its multi-disciplinary nature, there is no standardized or commonly accepted definition of requirements engineering. Zave [ZAVE 97] provides one of the clearest definitions of *requirements engineering (RE)*:

> **Definition 2 –** *Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specification of software behavior, and to their evolution over time and across software families.*

The IEEE standards [IEEE 10] define *requirements engineering (RE)* as:

> **Definition 3 –** *(1) The process of studying user needs to arrive at a definition of system, hardware, or software requirements. (2) The process of studying and refining system, hardware, or software requirements.*

A formal definition of *requirement*, defined by IEEE standards [IEEE 10] as:

> **Definition 4 –** *(1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. (3) A documented representation of a condition or capability as in (1) or (2).*

Requirements engineering (RE) is thus a process of eliciting, understanding, specifying, validating customers' and users' requirements, and identifying constraints on the application to be constructed. The RE process is commonly divided into a set of intertwined activities:

- Requirements Elicitation
- Requirements Analysis and Modeling
- Requirements Specification
- Requirements Validation
- Requirements Management

This research focuses mainly on the requirements analysis and modeling activity of requirements engineering; in particular, the analysis and modeling of user requirements, with some guidance from the requirements elicitation methods. Several techniques on user requirements analysis and modeling will be investigated and applied in this research; namely, scenario-based, task-based, and goal-based methods.

2.2. Persona-Based Methods

The notion of persona, originally introduced by Alan Cooper [COOPER 99] as a practical interaction design tool in his Goal-Directed Design approach, has rapidly gained popularity in the software industry due to its effectiveness and powerful concept. Persona is becoming a promising and an emergent new paradigm in user modeling, and is gaining much attention in requirements engineering for better understanding of users.

Personas are fictitious and concrete representations of a group of target users. They are constructed to resemble *real* people, i.e. they contain information such as names, age, educational background, occupations, skills, goals, concerns, usage patterns on the system, and so forth. The use of abstract representation of users originated in the field of marketing, but Cooper's use of personas, their goals, and scenarios are focused on design. Cooper does not describe in detail how personas are constructed nor endeavor to explain the power of personas. Cooper contends that focusing solely on tasks a particular application has to be performed without meeting user needs and goals is a recipe for failure. Cooper's Goal-Directed Design approach provides focus through the creation of fictional personas whose goals form the basis for scenario creation.

Randolph [RANDOLPH 04] thinks of personas as "hypothetical users – fictional people who represent classes of users". He suggests that the Goal-Directed Design approach presented by Cooper in the concept of personas could help software developers think about their users early in the Software Development Life Cycle (SDLC) and integrate the user interface (UI) design without huge cost. These personas are populated with personal attributes and personal goals and each UI would be designed for at least one primary persona. Randolph suggests that this primary persona's needs and goals must be satisfied to declare the system a success.

The use of personas has also been suggested in user-centered requirements engineering by RedHat (www.redhat.com) [DANTIN 05]. RedHat's user-centered requirements approach of using personas consists of the following steps:

- What personas are going to use this software component?

- What are the goals of these personas when using this software?
- For each primary persona, write all relevant scenarios, each telling the story of the persona achieving a goal.
- For each scenario, determine the individual tasks involved.
- Do the matrix of tasks and scenarios.

Personas are complimentary to scenario-based approach to requirements engineering. Scenarios are often constructed around personas. Grudin and Pruitt [GRUDIN 02, PRUITT 03] have suggested using personas together with scenarios as a participatory design technique. Personas are used to provide guidance on product design and development decisions, as well as prioritization of requirements via a feature-persona weighted priority matrix to help determine what features are built in the product development cycle. In the feature-persona weighted priority matrix, each persona is assigned a weight according to the proportion of the market each represents. Each requirement is assigned a value for each persona:

-1: The persona is confused, annoyed, or in some way harmed by the requirement.

0: The persona does not care about the requirement.

+1: The feature provides some value to the persona.

+2: The persona loves the feature or the feature does something wonderful for the persona even if the persona does not realize it.

A weighted sum for each requirement is computed according to the market each represents, and requirements with the highest weighted sum are given the highest priority in development.

Aoyama [AOYAMA 05] proposes Hanako method of persona-scenario based requirements analysis methodology combining with conjoint analysis based on statistical techniques for embedded software on digital consumer products. The methodology provides techniques to identify persona based on conjoint analysis theory and requirements value and hot spots analysis between personas and scenarios. The process of Hanako method consists of two stages:

Stage 1: Persona analysis, which is divided into two processes of (1) simultaneous disjoint clustering of users and services (or requirements) using conjoint analysis, and (2) identification of personas and service hot spots.

Stage 2: Requirements value analysis between persona and scenarios of services.

The first step of conjoint analysis is to simultaneously decompose the users and services into a set of disjoint groups. The primary criteria of disjoint decomposition of users are user profiles, based on demographic variables of age, gender, occupation, and generation. The next step is to identify the preference of each user group to services in terms of usage (frequency) of the services rated from 1 to 5 through written questionnaire and interviews. The primary persona is then identified from the user groups that have the highest coverage of service frequency. For the identified primary persona, the usage scenarios of several use cases of preferred services are analyzed via service transition diagram, a simplified state transition diagram, as a representation of scenarios. Hot spots in the requirements (i.e. high usage) can be identified by analyzing the interaction between primary persona and scenario.

Aoyama also proposes Persona-Scenario-Goal (PSG) methodology [AOYAMA 07] as an extension of Hanako method (Persona-Scenario methodology) by integrating Persona-Scenario model with goal orientation. The PSG methodology follows the Hanako method of persona identification and scenario analysis. To identify goals, PSG employs seven goals of usability: *Favorability*, *Usefulness*, *Credibility*, *Operability*, *Understandability*, *Readability*, and *Responsiveness*. Using Scenario-Goal-Issue (SGI) Matrix, requirements issues can be identified and evaluated by walking through each scenario (or use case) from the viewpoint of the primary persona in light of the seven goals of usability. Conflicts of requirements issues identified by the SGI matrix can then be resolved to the level of fulfillment to the goals.

Sim and Brouse propose a **C**oncept **D**evelopment **P**rocess (**CDP**) model [Sim 14] to guide engineers, analysts, and developers in the development of the concept of persona, in the context of the concepts of viewpoint, goal, scenario, task, and requirement. The CDP model distinguishes from other previous models proposed by other researchers in that it investigates and integrates the relationships of the concepts of persona, viewpoint, goal, scenario, task, and requirement in a unified environment to provide an improvement in the understanding of the target users' needs and behaviors early in the requirements engineering process.

The CDP model [SIM 14a] consists of four main processes: 1 – Personas Construction, 2 – Viewpoints Identification and Construction, 3 – Concepts Modeling, and 4 – Analysis and Evaluation. Unlike other models proposed by other researchers, the

CDP model [SIM 14] provides templates and matrices that help and guide engineers and developers in the development of the concepts:

(1). **P**ersona **D**efinition **D**ocument (**PDD**) (deliverable of process 1) contains a set of general attributes defining the persona. The attributes are extracted from the description of the persona profile.

(2). View**p**oint **D**ocument (**VPD**) (deliverable of process 2), uniquely identified by three components <Persona, Role, Environment>, contains information that defines one or more views of the persona at a certain level of detail and addresses certain design concerns by the persona playing a particular role in a particular environment.

(3). Personas-Viewpoints-Requirements Matrix (deliverable of process 4) provides a summary of the importance of each requirement, relative to the importance of each persona. The Personas-Viewpoints-Requirements Matrix is a modification of the Persona-Weighted Features Matrix [PRUITT 06] by including personas' viewpoints in the matrix and modifying the rating scale: 5 – Critical, 4 – Very Important, 3 – Important 2 – Somewhat important, 1 – Unimportant. The Personas-Viewpoints-Scenarios Matrix (deliverable of process 4) provides information on the number of scenarios that refer to each viewpoint of each persona and the frequency count of each scenario. The Personas-Viewpoints-Scenarios Matrix is an extended version in [PRUITT 06, MIKKELSON 00] by incorporating viewpoints in the matrix. The Scenarios-Usability Goals Matrix (deliverable of process 4) identifies any requirement issues with respect to the eight usability factors chosen for the matrix: *Accuracy*, *Attractiveness*, *Efficiency*, *Learnability*, *Reliability*, *Comprehensibility*,

Clarity, and *Rememberability*. The Scenarios-Usability Goals Matrix is a modification of the Scenario-Goal-Issue Matrix approach in Aoyama's Persona-Scenario-Goal (PSG) methodology [AOYAMA 07] by modifying the usability factors.

2.3. Scenario-Based Methods

Scenario-based approaches originated in the HCI field in the late 1980s, but they are now being recognized as an important conceptual tool for requirements engineering. Nowadays, scenarios are a widely used technique to support the requirements analysis, design, and evaluation of interactive systems, and are often used in conjunction with task analysis and modeling. Alexander [ALEXANDER 04] defines scenario as:

> **Definition 5 –** *A scenario describes the system's behavior through a sequence of concrete interactions with its users who are trying to achieve some goal.*

The sequence of interactions in a scenario contains detailed interactions that illustrate one of many infinitely ways of interacting with a system. Scenarios reflect a concrete path or set of steps performed by the user towards achieving a goal. The use of scenarios often complements goal modeling because, although goals focus on abstractions that describe users' intentions, scenarios make abstract intentions explicit and easy to understand by giving concrete examples of how the application might actually fulfill users' goals. Due to their narrative, simple, and intuitive structures,

scenario descriptions provide an effective means of communicating stakeholders' goals and expectations through vivid examples of use.

Scenarios can be expressed in different ways including narrative text, structured text, images, maps (i.e. Use Case Maps, UCM), or graphical forms, i.e. use cases, activity diagrams in UML. **U**se **C**ase **M**aps (**UCM**) [BUHR 98] provides a visual notation for scenarios for describing and reasoning about behavior patterns in systems, as well as the coupling of these patterns. The UCM notation uses scenario paths to illustrate causal relationships among responsibilities. The UCM notation consists of the following basic elements:

- Start points (filled circles) represent pre-conditions or triggering causes.
- End points (bars) represent post-conditions or resulting effects.
- Responsibilities (crosses) represent actions, tasks, or functions to be performed.
- Components (boxes) represent entities or objects composing the system.

Rolland [ROLLAND 98] proposes the coupling of goals and scenarios in requirements engineering in CREWS-L'Ecritoire approach within the CREWS project in which scenarios are used as a means to elicit requirements and goals of the system-to-be. The approach centered on the notion of a requirement chunk (RC) which is a pair <Goal, Scenario>. When a goal is discovered, a scenario is authored for it and once a scenario has been authored, it is analyzed to yield goals. The goal-scenario relationship is then examined in the reversed direction, i.e. from scenario to goals. In this process, goal discovery and scenario authoring are complimentary steps and goals are incrementally discovered by repeating the goal-discovery, scenario-authoring cycle.

In [ALSPAUGH 08], Aspaugh and Antón introduce SMaRT (***S***cenario ***M***anagement ***a***nd ***R***equirements ***T***ool) tool that assists analysts working with scenarios by supporting scenario specification tasks:

- *Episode Reference Diagram* shows which scenarios use other scenarios as episodes in a compact and informative way. An episode is a scenario that is used as an event of several scenarios. Each node in the diagram represents a scenario, and each edge connects a scenario to another scenario. Each episode shared by two or more scenarios represents a dependency between those scenarios.
- *Scenario Context Diagram* describes the contexts needed by and produced by a scenario.
- *Shared Event Diagram* shows which events recur in which scenarios.

2.4. Goal-Based Methods

Goal-oriented methods have long been intensively studied in the requirements engineering community. Traditional systems analysis focuses on *what* features (i.e. activities, functionalities) a system will support. Goal-based approaches focus on *why* systems are constructed providing the motivation and rationale to justify software requirements. Goal-oriented methods provide specific support for coping with high level stakeholders' goals, facilitates the exploration of design alternatives, and the definition of requirements at a suitable level of abstraction.

A goal is a state of affairs that the user wishes to achieve; it denotes *intention* but not the *means* that materialize these intentions. Dardenne [DARDENNE 93] provides a formal definition of goal as:

> **Definition 6 –** *A non-operational objective to be achieved by the composite system. Non-operational means that the objective is not formulated in terms of objects and actions available to some agent in the system. Agent is an object which is a processor for some actions.*

Antón [ANTON 96] defines goal as:

> **Definition 7 –** *Goals are high level objectives of the business, organization, or system. They capture the reasons why a system is needed and guide decisions at various levels within the enterprise.*

KAOS (**K**nowledge **A**cquisition in aut**O**mated **S**pecification) [DARDENNE 93] is one of the most well-known software engineering approaches that highlights the importance of explicitly representing and modeling organizational goals. It consists of a conceptual meta-model that provides a formal language for describing functional and non-functional requirements to facilitate the requirements acquisition and elaboration process. In the KAOS model, goals represent the targets of achievement for the system. They are formally defined and analyzed such that conflicts between goals are identified. The goals are systematically decomposed into sub-goals through a refinement process modeled in AND/OR graphs, and eventually operationalized into requirements, which describe the characteristics the system should meet.

Cockburn [COCKBURN 97a, COCKBURN 97b] introduces goals to object-oriented analysis. He defines use cases to satisfy goals. Cockburn defines use cases as:

> **Definition 8 –** *A collection of possible scenarios between the system under discussion and external actors, characterized by the goal the primary actor has toward the system's declared responsibilities, showing how the primary actor's goal might be delivered or might fail.*

In [COCKBURN 97a, COCKBURN 97b], Cockburn cites five opportunities for goals:

- Attach non-functional requirements to goals.
- Track the project by goals.
- Get subtle requirements from goal failures.
- Use goals with responsibility-based design.
- Match user goals to operational concepts.

Other notable goal-oriented approaches to requirements analysis include:

- **GBRAM** (**G**oal-**B**ased **R**equirements **A**nalysis **M**odel) framework [ANTON 96]: GBRAM provides effective support for the elaboration and pruning of goals by combining goal-based methods with scenario-based approaches for software-intensive information systems.
- i* framework [YU 93, YU 11a, YU 11b]: i*framework enables modeling early requirements by identifying the actors (users and main stakeholders) involved, their goals, their dependencies, and the rationale for their intentions. It defines various types of agent dependency links to model situations where an agent depends on

another for a goal to be achieved, a task to be achieved, or a resource to become available.

2.5. Task-Based Methods

Task-based analysis and modeling is a research area which has received supports and contributions from computer science, cognitive science, and HCI communities. It focuses on:

- Task analysis, i.e. how to characterize and identify tasks.
- Task modeling, i.e. using notations suitable to represent tasks and their relationships.

A *task* is the course of action (or activity) the user goes through in order to achieve a goal. Tasks and goals are thus closely related; each task can be associated with one goal (i.e. the goal achieved by performing the task), and one goal can be achieved by performing (or choosing) one or more multiple tasks. Task analysis is usually performed in the requirements elicitation phase of requirements engineering to identify, describe, and evaluate the activities (or tasks) necessary or required for a user to achieve a goal. The analysis focuses on precisely-defined user interactions. The result of task analysis is the input for the task modeling phase.

In web application analysis and design, tasks are usually associated with one or more user profiles, which represent the target audiences. User profiles and relative tasks may be defined through scenarios, or user elicitation techniques such as structured interviews, questionnaires, surveys, and direct observation.

Hierarchical **T**ask **A**nalysis (**HTA**) technique [ANNETT 67] is one of the oldest general purpose task description techniques. In HTA, tasks are described in terms of *operations* and *plans*. Operations are actions or activities users perform to reach their goals and plans are statements of conditions for each operation is to be carried out. A hierarchy of tasks and subtasks is constructed in which tasks are gradually decomposed into subtasks and eventually into actions. With each new subtask, a new plan is created.

Task models for user-centered interactive design have been developed by Paternó [PATERNO 00]. Paternó introduces a task diagramming notation for task model specifications, known as **C**oncur**T**ask**T**rees (**CTT**). CTT distinguishes four categories of tasks: User Tasks, Application Tasks, Interaction Tasks, and Abstract Tasks. CTT provides an expressive and flexible notation capable of representing concurrent and interactive activities, with the possibility of supporting co-operations among multiple users and possible interactions.

An extension of the **U**nified **M**odeling **L**anguage (**UML**) for modeling tasks has been proposed by Markopoulos [MARKOPOULOS 01] to specify tasks model using use cases. In [MARKOPOULOS 01], tasks are modeled as stereotyped use cases. By modeling tasks as use case stereotypes, task attributes may be written as attributes of these use cases. Alternative task decompositions (OR relationship) for the same task can be represented with the generalization relationship. The stereotyped <<includes>> can be used to represent task-subtask relationship, where the task consists in the composition of its subtasks.

2.6. Ontology-Based Methods

The emergence of the semantic web has attracted great interest and attention in ontologies for the requirements engineering community. Ontologies offer the benefits for requirements of explicitly modeling domain knowledge in a machine-understandable way, i.e. allowing traceability and consistency check of requirements.

Over the past two decades, numerous works have addressed the use of ontologies in requirements engineering that built on knowledge representation; well-known examples include KAOS [DARDENNE 93] and i* framework [YU 93, YU 11a, YU 11b]. Tecuci [TECUCI 98] develops the Disciple approach for building intelligent agents in which an expert teaches the agent how to perform domain specific tasks in a way that resembles the way the expert would teach an apprentice. Disciple provides various tools for ontology creation, modeling, and management. These tools include a feature editor, an object editor, a task editor, a rule editor, an object browser, a hierarchical browser, an association browser, a feature browser, and so forth.

Today, the emphasis is placed on sharing ontologies via the web, utilizing semantic web technologies in software engineering, in particular requirements engineering. The W3C has finalized **OWL** (**W**eb **O**ntology **L**anguage) [W3C 04] as the standard format for ontologies to be represented on the web. OWL is an RDF language, which is a language for representing information about resources in the web suitable for processing by applications. OWL is built on the foundation of **D**escription **L**ogic (**DL**), which is a logic that has semantics that can be translated to first-order predicate logic. The nature of DL means that classification, subsumption, and satisfiability can be

automatically computed by a reasoner. This enables OWL the capability to find inconsistencies and maintain complex class-subclass relationships by inference machine. A number of ontology tools have been created to support ontology construction and editing. Among them, Protégé-2000 [PROTÉGÉ 00] is a popular graphical tool for ontology editing and knowledge acquisition. Protégé with OWL Plugin provides a rapid prototyping environment to assist ontology engineers to create ontology instantly and perform semantic restrictions due to OWL powerful inference engine. Chimaera [CHIMAERA 00] is an ontology environment that supports the creation and maintenance of multiple ontologies.

Several works on ontology-based requirements analysis have been conducted over the past few years. Kaiya [KAIYA 05, KAIYA 06] proposes a software requirements analysis method based on domain ontology technique. The method allows mapping between a software requirements specification and the domain ontology that represents semantic components. It also enables detection of requirements incompleteness and inconsistency, measurement of the quality of a specification with respect to its meaning, and prediction of requirements changes based on semantic analysis on a change history. The ontology system is written in the form of class diagrams. The detections of incomplete and inconsistent requirements elements are achieved by applying inference rules. Measurement of quality is done by computing metrics for the characteristics of a good software requirements specification based on IEEE standard, [IEEE 11 a] i.e. correctness, completeness, consistency, unambiguity, and so forth.

Koay [KOAY 09] provides ontological support for managing non-functional requirements in pervasive healthcare systems. The ontological solution includes a remote patient monitoring system (RPMS) for patients suffering from complications of strokes and ontologies to create a reasoning mechanism to choose the best monitoring device for a particular patient when using the RPMS. Req-ONTO and Dev-ONTO ontologies are created which store semantics of non-functional requirements imposed on the RPMS and the characteristics of devices/sensors respectively. The ontological model is implemented using the Protégé 2000 ontology editor. OWL-DL ontology language is used to model the semantics of ontological schemas and **SWRL** (**S**emantic **W**eb **R**ule **L**anguage) [SWRL] and **SQWRL** (**S**emantic **Q**uery-Enhanced **W**eb **R**ule **L**anguage) [SQWRL] to model the rules needed for reasoning. SQWRL is a SWRL-based query language that allows for more detailed querying of OWL ontologies.

Sim and Brouse [SIM 14b] proposes an **O**ntology-Based **Persona**-Driven **U**ser **R**equirements **M**odeling (**OntoPersonaURM**) model that consists of a 5-step iterative ontology development process to help guide engineers and developers in the representations and specifications of the concepts of persona, viewpoint, scenario, task, requirement, and other relevant concepts in an ontological environment using Protégé-Frames [NOY 00a], a widely popular frame-based open source ontology editor tool and knowledge model developed by Stanford Medical Informatics. The OntoPersonaURM model is developed in the Concepts Modeling process of the **C**oncept **D**evelopment **P**rocess (**CDP**) model proposed by SIM and Brouse [SIM 14a].

The OntoPersonaURM model [SIM 14b] is composed of three domain independent interrelated ontologies specifying general yet broad set of concepts: (1) Persona Ontology: specifies general concepts pertaining to person characteristics including education, abilities, interests, knowledge, viewpoints, environments, and so forth. (2) Behavioral **G**oal-**S**cenario-**T**ask (**GST**) Ontology: captures and defines the needs and behaviors of the personas and the system-to-be, i.e. viewpoint, goal, scenario, and task concepts. (3) Requirements Ontology: specifies general concepts for the representation of the requirements and their properties. The OntoPersonaURM model helps to establish a knowledge repository and foster a shared common understanding of target users' needs and behaviors among developers and stakeholders during the requirements analysis and modeling activity. The OntoPersonaURM model proposed by SIM and Brouse [SIM 14b] distinguishes from other ontological models developed by other researchers [DARDENNE 93, KAIYA 05, KAIYA 06, KOAY 09, YU 93, YU 11a, YU 11b] in that:

(1). The OntoPersonaURM model provides insights and helps guide ontology engineers and developers in the construction of ontologies for explicit specifications of the concept of persona in representing users' characteristics and its relationships with the concepts of viewpoint, goal, scenario, task, requirement, and other pertaining concepts in a unified environment.

(2). The ontologies constructed in the OntoPersonaURM model are developed as general as possible yet encompassed a broad set of concepts that are not domain dependent

and thus can be applied, modified, extended to other applicable domains, or used as a referenced model.

CHAPTER 3: PERSONA CONCEPT

3.1. What is a Persona?

A persona is a fictitious character that represents the needs and requirements of a specific user group who will be using the system in terms of his or her goals and personal characteristics [COOPER 07; COOPER 99; PRUITT 06]. Although personas are fictitious, they are based on comprehensive user research on real users. A persona consists of a persona description which focuses on information such as background, occupations, knowledge and skills, goals, interests, workflow, context, concerns, emotions and attitudes, and other personal traits. The persona is brought to life by being given a name, a life, a personality as well as a portrait.

Personas and complimentary to scenario-based approach to requirements engineering; scenarios are often constructed around personas. The use of personas and scenarios are intended to help designers and project stakeholders focus on the users and their needs during system development. The most common interpretation of the persona method centers on the fact that creating personas and scenarios is an effective way of communicating the results from user research. The persona material(s) can be used to understand and focus on user needs and desires and to communicate these among the stakeholders in a project: designers, developers, project managers, clients and others [PRUITT 03]. The personas method can be a useful and valuable tool to help design team

gather, analyze, and synthesize information related to the users interacting with the system [COOPER 07]. Personas help you to understand the user. Not as a part of a group or a demographic but as a person, an individual with goals, a history, interests and a relation to the product. [QUESENBERY 06]. When creating personas one prioritizes their needs, and the most important persona is called the primary persona, while the others are secondary [Cooper 99]. When discussing functionality as well as designing the interface, the primary persona is prioritized and is the one that, above all others, needs to be able to use the system.

Personas and other methods of user representations have been used quite extensively in user-centered design as well as marketing and branding, although the exact construction and usage of the persona has varied in detail and description. Usability specialists and researchers working with system development have also used representations of users to a considerable extent, within both usability engineering and user-centered design [Pruitt 06]. All these efforts have the common goal of increasing the focus on the needs of users instead of focusing primarily on the technology. There are two main aspects of the usage of the persona method: on the one hand, they are used as a communication device and, on the other, they are used as a design aid.

3.2. Benefits of Personas

According to Grudin and Pruitt [GRUDIN 02], the creation and use of personas can be used inappropriately, but it has potential benefits:

- Personas allows designers to focus on and design for a small set of specific (intended) users or subset of (intended) users who have the highest priority goals and needs through the fictionalized setting. Personas focus attention on a specific target users, establish who is and who is not being designed for. In user-centered design, it is easy to extrapolate on the basis of our own individual needs and desires [PRUITT 03]. Personas are meant to help us avoid our unconscious, individual biases and focus on the needs and desires of the intended users. By using personas, the project group can unite around a set of a small number of individuals that represent the anonymous group of target users for whom the system is being designed and developed. Personas make it easier for the project group focus on the specific intended users instead of on everyone. In fact, designing for a single and specific user rather than vaguely for everyone creates a focused design process that still can target a broader audience [COOPER 07].
- The creation of personas helps to make assumptions and knowledge about the target users and decision-making criteria explicit. Personas encourage consensus as they bring the design team together to establish one common shared vision of who the target users are and what the target users want in the system. When a design team talks about users in general, each team member's ideas of the user's needs and goals may be different. In addition, the word *user* is a fairly general term that can mean entirely different things to different people. Members of the project group may have various assumptions about the "user" and they may have personal, cultural and corporate biases that are neither apparent to colleagues nor to the persons themselves

[Cooper 99]. The detailed description of a persona helps the design team establishes a common understanding of the target users and makes assumptions about target users explicit: what they are designing for, why they are designing this feature, why they are designing it in this way, and so forth.

- Personas provide a medium for communications to convey a wide range of quantitative and qualitative data [PRUITT 03]. Personas utilize narrative and storytelling to enhance attention, memory, and organization of detailed user data. The narrative and storytelling aspect of the persona method taps into a basic cognitive process in human beings [QUESENBERY 06], a process that relates to memory and reconstruction. Clear communication in turn facilitates consensus and efficiency in team decision-making [MULDER 07]. Personas alone can aid design, but they are more powerful when complementing other quantitative and qualitative methods rather than totally replacing them [GRUDIN 02, PRUITT 03].

CHAPTER 4: ONTOLOGIES

4.1. What is an Ontology?

Ontology was originally part of a branch of philosophy known as metaphysics, which is the study of the nature of being, becoming, existence, or reality, the entities that exist or can be said to exist, the categorization of such entities, as well as the relationship of these entitles within a hierarchy. The word "ontology" has its roots in Greek words *ontos* (for "being") and *logos* (for "word" or "reason") and refers to the subject of existence in philosophy or the "reason of being".

The term "ontology" has widely been used during the past several years, especially in several fields of informatics such as Artificial Intelligence, Agent Systems, Information Systems, Database or Web Technology. There are many definitions of the term ontology in the Artificial Intelligence literature. Gasevic et al. [GASEVIC 06] suggest that "informally, the ontology of a certain domain is about its terminology (domain vocabulary), all essential concepts in the domain, their classification, their taxonomy, their relations (including all important hierarchies and constraints), and domain axioms". The most widely used definition of ontology is given by Gruber [GRUBER 93b], which defined ontology as "*a formal specification of a conceptualization*". In this context, *conceptualization* refers to an abstract simplified view of a domain that describes the objects, concepts, and relationships between them that

exist in that domain of discourse. *Formal specification* implies the existence of a representational vocabulary in which the objects in the universe of discourse and their relationships can be formally represented and machine understandable. Borst [BORST 97] extends the definition given by Gruber by emphasizing that an ontology should be reusable and shared across several applications. He defined ontology as "*a formal specification of a shared conceptualization*". Studer et al. [STUDER 98] expand the former definition by emphasizing that an ontology should make the domain assumptions explicit, i.e. all concepts identified and constraints used are explicitly defined. Studer et al's definition of ontology is: "*An ontology is a formal, explicit specification of a shared conceptualization*". In this dissertation, we adopt the definition of ontology by Studer et al.

4.2. Benefits of Ontologies

Developing a new ontology is often a tedious and time consuming effort. There is no single correct ontology for any domain. However, the effort spent in building an ontology can be worthwhile in the end, as an ontology can offer a number of benefits which can be summarized as follows:

- Ontologies promote share common understanding of a domain by enabling knowledge sharing among humans, as well as software applications. When a group of people or several software applications commit to an ontology, the group of people or the software applications are guaranteed to use terms with the same meaning as specified in the ontology.

- Ontologies provide a representation vocabulary by sharing the terminology defined in the ontology. Ontologies provide a declarative, machine readable representation that allow an unambiguous communication among software agents and reduce conceptual and terminological mismatches.
- Ontologies allow explicit formal representation of domain knowledge and assumptions. By having a formal representation, domain knowledge can be used in formal algorithms or logical reasoning to solve different tasks.
- Ontologies enable reuse of domain knowledge due to the explicit representation of domain knowledge and assumptions. Domain assumptions and implementation may be checked for correctness and standards may be introduced to allow interoperability.
- Ontologies enhance the design model quality through explicit representation of knowledge (concepts, relationships, and constraints) and logical reasoning to allow semi or automated consistency checking.

4.3. The Protégé Knowledge Model

In this dissertation, the knowledge representation formalism that has been chosen for the construction of the ontologies was a frame-based formalism, Protégé-Frames [NOY 00a], which is a widely popular frame-based open source ontology editor tool and knowledge model developed by Stanford Medical Informatics. We chose Protégé-Frames ontology editing tool as a knowledge representation environment for the construction of ontologies because (1) it has an intuitive and easy-to-use graphical user interface that does not demand too much learning curve, and (2) it provides our research needs of defining

classes (and sub-classes), describing properties and relationships of classes, populating classes with instances, and performing common queries to check constraints on the classes.

In a frame-based knowledge representation system, a frame is the principal building block of a knowledge model, which represents an entity in the domain of discourse. The Protégé frame-based knowledge model [NOY 00a] is built upon classes, slots, facets, and instances:

- **Classes**

  Classes are a concrete representation of concepts in a domain of discourse, which are also called entities. Classes are organized in taxonomic hierarchies formed by the sub-class-super-class hierarchy (or specialization/generalization hierarchy or child-parent relationship) and linked by means of transitive *is-a* relationship. A class may have multiple sub-classes (or child) that represent concepts that are more specific than the super-class. Inversely, a class may have super-classes (or parents) that represent concepts that are more general than the sub-class. For example, a CheesePizza is a sub-class of a more general class Pizza.

- **Slots**

  A class has properties or slots describing various features (attributes, relationships) of the concept in the domain. For example, a slot age may describe the age of a person. Slots are also known as roles in description logic and relations in object-oriented language such as UML. Each slot has a type and constraints (also referred

to facets) are attached to the slot that restrict the values that the slot is allowed to take. The properties of the slots that can be constrained are:

- Value Type, e.g. String, Float, Integer, Boolean, Symbol, Instance, and Class.
- Cardinality, e.g. required, at least, at most, and multiple.
- Minimum and maximum value for numerical slot.

- **Facets**

  Facets describe the properties of the slots. They are used to define constraints on the values the slots are allowed to take. Examples of facets are the cardinality property of a slot which constrains how many values the slot may have, the type of a slot which constrains the valid value type for the slot, the minimum and maximum values for a numeric slot, the allowed values when the value type of a slot is specified as type Symbol (i.e. enumerations), the allowed classes (also called range of a slot) from which the instances can come when the value type of a slot is specified as type Instance, and so on.

- **Instances**

  Classes may have one or more instances, which correspond to individual objects in the domain of discourse. Each instance of the class has a concrete value assigned for each slot of that class that conforms to the definition of the properties of the class. Based on the semantics of sub-class-super-class relationship in the knowledge model, all instances of a sub-class are also instances of its super-class. An ontology with an explicit specification of concepts and relationships in a domain combines with a set of individual instances of classes constitutes a knowledge base.

4.4. Constraint, Consistency, and Completeness Check

As described in previous section, the frame-based Protégé knowledge model is built upon four main components: classes, slots, facets, and instances. While these built-in components and their relationships cover the overall definition of a knowledge base, there are limitations on the things that can be expressed, as most frame-based knowledge systems lack the ability to express or capture complex types of constraints, rules, or axioms using disjunction, existential quantification about frames, or relationships between properties of the same frame or different frames. For example, one cannot relate or constraint the values of different slots attached to a given class, nor relate or constraint the slot values of different instances of a class. One approach to remedy for this limitation is the use of a more expressive constraint language, **P**rotégé **A**xiom **L**anguage (**PAL**) [PAL].

PAL extends the Protégé frame-based knowledge modeling environment to support the definition of logical constraints and queries about frames in a knowledge base. PAL is based on a subset of first-order logic and is implemented as a tab-widget plugin to Protégé-Frames editing tool that allows users to check for constraints and run queries on slot values in a knowledge base, and find instances in the knowledge base that violate the PAL constraints. PAL is a model-checking language used for writing restrictions on existing knowledge and not for asserting new knowledge. The primary goals of PAL are to detect incomplete entry of information and to check entered information for inconsistencies [PAL], which in turn help to improve the quality requirements of the design model.

Another approach to extend the Protégé knowledge base with additional axioms for checking completeness and consistency in the knowledge base model to improve the quality requirements of design model is to apply rule reasoning that can be interpreted by a logical reasoner. In this dissertation, **completeness** in an ontological design model refers to all required or important concepts, including attributes and relations have been specified in the knowledge based, and **consistency** in an ontological design model refers to all concept specifications in the knowledge base are specified with no contradiction, i.e. no conflict concepts and invalid relations (between concepts). In order to check the semantic completeness and consistency in a design model, an explicit formal representation of the meaning and assumptions of the concepts in a domain is therefore necessary and then the use of logical reasoning to check for incompleteness and inconsistencies.

In this dissertation, we applied constraints query and rule-based techniques to evaluate the quality of the domain independent ontologies created in the knowledge base and validate the research hypothesis, H3 (chapter 1). The rule reasoning and inference engine used in this dissertation is **JESS** (**J**ava **E**xpert **S**ystem **S**hell) [JESS], a light, fast rule engine and scripting environment written entirely in Sun's Java language. Jess is integrated in the Protégé ontology editor via a tab-widget plugin called JessTab [ERIKSSON 03], which allows users to write Jess rules to implement operations on the knowledge base. In our case, Jess rules are written to identify inconsistency and incompleteness on our knowledge base. JessTab integrates the knowledge representation models by mapping the Protégé instances to Jess facts. This representation of instances as

facts in Jess enables users to effectively write Jess rules that match instance patterns [ERIKSSON 03]. Reasoning in Jess is based on a list of known facts (or values) and a set of rules that try to match on these facts in its fact base.

CHAPTER 5: PERSONA-URM: PERSONA-DRIVEN USER REQUIREMENTS MODELING METHODOLOGICAL FRAMEWORK

5.1. Introduction

As mentioned in chapter 1, the primary objective of this research is to investigate the concept of persona and ontology-based approach in user requirements modeling. The objectives of this research are:

(1). To enhance the requirements engineering process by incorporating the persona concept into the requirements engineering activities through investigating how the concept of persona, in the context of goals, scenarios, tasks, and requirements may be integrated in a unified environment to enable developers gain a better understanding of target users' needs and behaviors and refine requirements early in the requirements engineering process.

(2). To examine how can ontology-based approach to explicit concepts specification of users' knowledge helps to improve the design model quality requirements of the ontologies by providing checking of constraints, consistencies, and completeness.

To accomplish the research objectives, we proposed a Persona-Driven User Requirements Modeling (Persona-URM) methodological framework that is comprised of:

- A Concept Development Process (CDP) model to help guide developers in the development of the concepts and the integration of the concepts into the requirements

engineering process. The CDP model consists of four main processes: 1 – Personas Construction 2 – Viewpoints Identification and Construction 3 – Concepts Modeling 4 – Analysis and Evaluation.

- An Ontology-Based Persona-Driven User Requirements Modeling (OntoPersonaURM) model consists of a five-step iterative ontology development process to help guide developers in the construction of three generic interrelated ontologies: Persona Ontology, Behavioral Goal-Scenario-Task (GST) Ontology, and Requirements Ontology. The construction of ontologies are developed using Protégé-Frames, an open source knowledge model and ontology editing tool.

It is to be noted that the development of the Persona-URM framework, i.e. CDP and OntoPersonaURM models, are the result of utilizing some of the techniques and ideas that have been investigated by other researchers [AOYAMA 05, AOYAMA 07, KAIYA 05, KAIYA 06, ALSPAUGH 08] in the past. However, the Persona-URM framework is distinguished from past work by other researchers in that it applies all the relevant techniques and ideas and investigates how the concepts of persona, scenario, task, goal, and requirement may be integrated in a unified environment and how the relationships among these concepts may be represented explicitly using an ontology-based approach that will help to provide a better understanding of users' needs and behaviors and help refine requirements.

5.2. Concept Development Process (CDP) Model

The Concept Development Process (CDP) model is proposed to guide engineers, analysts, and developers in the development of the concepts and the integration of the concepts into the requirements engineering process. In the CDP model, the persona concept plays the central role in driving the development of other concepts, i.e. concepts of viewpoints, scenarios, tasks, goals, and requirements. From the personas, valuable information such as the personas' viewpoints, goals, environments, concerns, scenarios, tasks, and requirements are elicited.

Figure 4 shows the graphical representation of the CDP model. The CDP model is pictorially viewed as a model composing of four concentric rings representing the four main processes:

- Process 1 (ring 1) – Personas Construction
- Process 2 (ring 2) – Viewpoints Identification and Construction
- Process 3 (ring 3) – Concepts Modeling
- Process 4 (ring 4) – Analysis and Evaluation

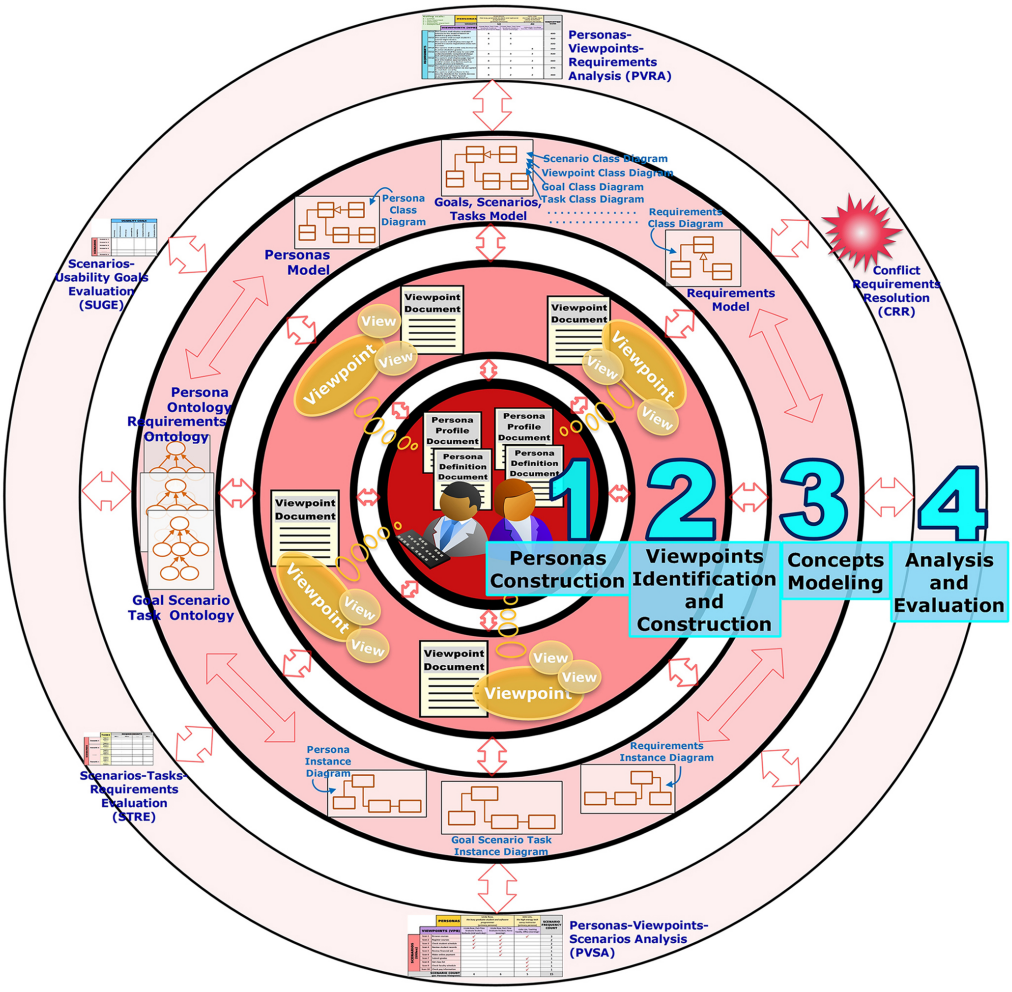


Figure 4. Concept Development Process (CDP) Model

As the concept of persona plays the center role in driving the development process, the Personas Construction process occupies the center of the CDP model (i.e. ring 1) and creates persona profile and definition documents for the personas. The Viewpoints Identification and Construction process (ring 2) identifies one or more

viewpoints for each persona and constructs viewpoint documents that provides specification for the construction of the views and modeling methods. The Concepts Modeling process (ring 3) conducts UML modeling and ontology concepts representation by constructing personas model, scenarios model, tasks model, goals model, and requirements model. Each model is comprised of UML class diagrams, UML instance diagrams, and ontology specifications. The Analysis and Evaluation process (ring 4) may be conducted during or immediately after process 2. One or more of the following activities may be conducted in the Analysis and Evaluation process which are described in section 5.2.4:

- **P**ersonas-**V**iewpoints-**R**equirements **A**nalysis (**PVRA**)
- **P**ersonas-**V**iewpoints-**S**cenarios **A**nalysis (**PVSA**)
- **S**cenarios-**U**sability **G**oals **E**valuation (**SUGE**)
- **S**cenarios-**T**asks-**R**equirements **E**valuation (**STRE**)
- **C**onflict **R**equirements **R**esolution (**CRR**)

In Figure 4, the bi-directional arrows between two rings signify the fact that the inner ring provides inputs to the outer ring and the outer ring requires inputs from the inner ring. For example, the Persons Construction process (ring 1) provides Persona Profile Document and Persona Definition Document as inputs to the Viewpoints Identification and Construction process (ring 2). The bi-directional arrows *within* a ring (i.e. ring 3 – Concepts Modeling) indicate the relationships between two models which are captured in UML class diagrams and ontology specifications.

The processes of the CDP model may be incorporated in the requirements engineering (RE) activities through establishing relationships with one or more RE activities including Requirements Elicitation, Requirements Analysis and Modeling, and Requirements Specifications. Section 5.2.5 describes the integration of the processes of CDP model into the requirements engineering activities.

5.2.1. Process 1 (ring 1) – Personas Construction

The objective of the Personas Construction process is to develop a concise definition of persona. The identification of personas is not the focus in this dissertation. In most practices, potential personas are first identified based on common techniques used by a marketing team. These techniques include one or more of the following: observations, user interviews, demographic market surveys, usability testing, field studies, and site traffic/log files analysis. There have also been efforts conducted in the past by a few researchers on developing methodologies or techniques to identify personas [AOYAMA 07, CASTRO 08, COOPER 99. GRUDIN 02, PRUITT 03]. In the Personas Construction process, we develop a general yet wide-ranging set of personas' attributes, such as personas' needs, behaviors, knowledge, goals, roles, usage patterns, the environments the personas engage in, and so forth that can be applied, extended to other domains, or used as a reference. Based on input information from the marketing team on common users in the context of a given domain, potential personas' profiles are crafted and relevant attributes are extracted from the profiles and defined.

The deliverables of the Personas Construction process are: (1) **P**ersona **P**rofile

Document (**PPD**) and (2) **P**ersona **D**efinition **D**ocument (**PDD**). The PPD is a one-page narrative description or mini-biography of a persona describing (summarizing) information such as who this person is, what is this person's typical day like, what is the person's lifestyle, how this person interacts with the system, what are this person's attitudes, concerns, goals, and interests toward the system. The PPD gives an overall image of each persona on which the development team can focus. Depending on available time and resources of the development team, as well as the needs of the system, the PPD, at a minimum, should cover the information as mentioned earlier. We recommend writing the PPD in a one-page narrative story-like style rather than a bulleted list of facts, as stories are engaging and memorable which can help the development team focus on and build empathy towards the potential users. For more detail information and guidance in writing the persona document (or profile), we recommend readers to consult [PRUITT 06]. A sample PPD template is shown in Figure 5 and included in Appendix B.

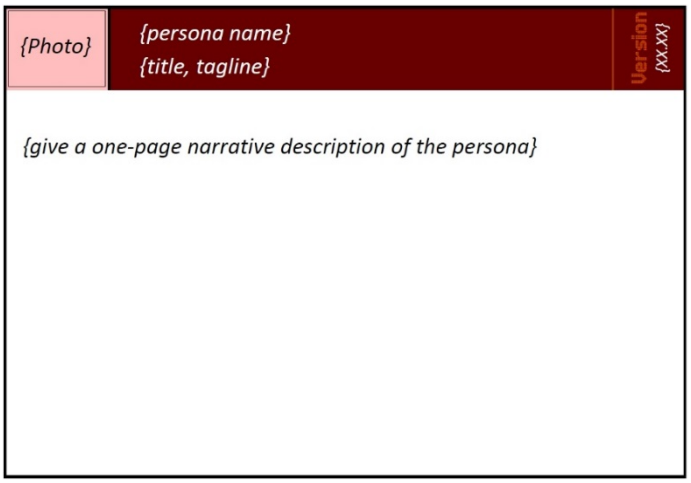


Figure 5. Persona Profile Document (PPD) Template

Next, for each persona, a PDD is created which contains a set of attributes defining the persona. The attributes are extracted from information described in the PPD. These attributes form the core information essential to the persona, as well as other information that may or may not be included in the PDD, depending on the application domain. More importantly, the PDD provides useful input to facilitate the identification and construction of viewpoints in the Viewpoints Identification and Construction process and also the development of conceptual models in the Concepts Modeling process. Engineers and developers are free to modify the PDD (i.e. add, delete, or edit attributes) to best suit the needs of the application domain. A sample PDD template is shown in Figure 6 and included in Appendix B.

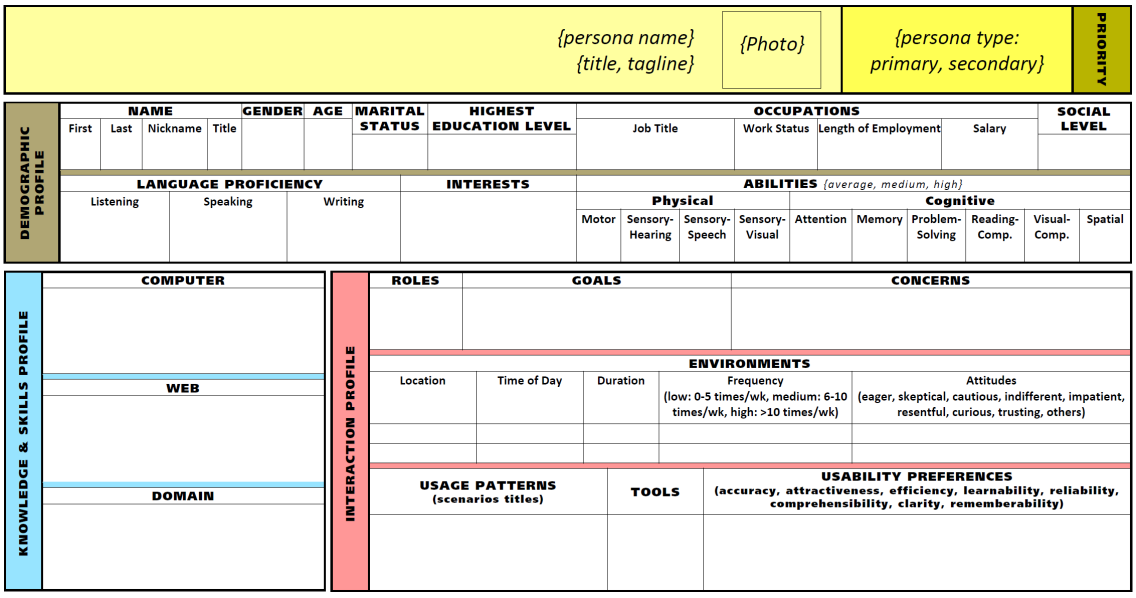


Figure 6. Persona Definition Document (PDD) Template

5.2.2. Process 2 (ring 2) – Viewpoints Identification and Construction

Viewpoint and view are systems engineering concepts and according to [IEEE 00]:

> "*A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views.*", and
>
> "*A view is a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders.*"

A viewpoint thus provides a means to define views at a certain level of detail of design concepts and address certain design concerns by the persona playing a particular role. Each view thus conforms to a specific persona's viewpoint playing a certain role in a particular environment to address the persona's concerns. The relationship between viewpoint and view is analogous to that of a class and object (or instance) of the class in UML [IEEE 00]. Thus, a view can be regarded as an instance of a viewpoint. Based on the characteristics of the persona profile, a persona possesses varied viewpoints depending on the roles they play and the environments they engage in at a particular moment in time. These personas' viewpoints define views that composed of various design models including scenarios, tasks, goals, requirements, and so forth.

Based on the PDD defined above, one or more viewpoints are identified for each persona. Since a persona personifies some roles and each role participates in some environments, each viewpoint, denoted as VP, is identified uniquely by three

components: the persona name, the role the persona plays, and the environment the persona engages in at a particular moment in time, i.e. location name and time of day. The viewpoint is represented as **<Persona, Role, Environment>** or **<Per, Rol, Env>**, which we call a Viewpoint Block (VPB). By synthesizing information from the PPD and PDD, the Viewpoints Identification and Construction process enables the development team to refine requirements.

The deliverable of the Viewpoint Identification and Construction process is the **V**iew**p**oint **D**ocument (**VPD**), created for each identified viewpoint which contains information that defines one or more views (i.e. instances of VP) at a certain level of detail and addresses certain design concerns by the persona playing a particular role in a particular environment. As information such as persona, role, and environment may change over time as the CDP development proceeds, the Viewpoints Identification and Construction process may be repeated throughout the CDP to generate new or revised viewpoints. A sample VPD template is shown in Figure 7 and included in Appendix B.

5.2.3. Process 3 (ring 3) – Concepts Modeling

In the Concepts Modeling process, an Ontology-Based Persona-Driven User Requirements Modeling (OntoPersonaURM) model is proposed that consisted of a five-step iterative ontology development process to help guide engineers and developers in the construction of ontologies for explicit specifications of the concept of persona in representing users' characteristics and the concepts of viewpoint, goal, scenario, task, and requirement. Based on the VPD created in process 2, Viewpoints Identification and

Construction, the view consisting of models of the concepts of persona, goal, scenario, task, and requirement are represented graphically using the UML class and instance diagrams notation.

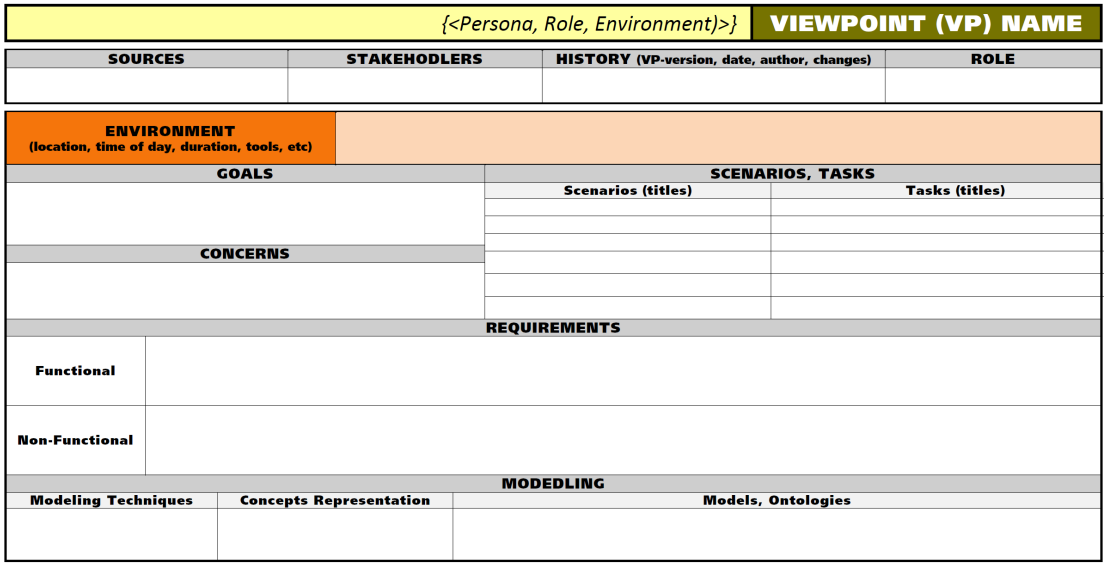


Figure 7. Viewpoint Document (VPD) Template

The OntoPersonaURM model is composed of three generic interrelated ontologies:

- **Persona Ontology:**

  The Persona Ontology covers general concepts pertaining to person characteristics including education, abilities, interests, knowledge, viewpoints, environments, and so forth.

- **Behavioral Goal-Scenario-Task (GST) Ontology:**

  The Behavioral GST Ontology captures and defines the needs and behaviors of the personas and the system-to-be, i.e. viewpoint, goal, scenario, and task concepts.

- **Requirements Ontology:**

  The Requirements Ontology specifies general concepts for the representation of the requirements and their properties.

The three ontologies are developed to be as general as possible yet encompasses a broad set of concepts that can be applied, modified, extended to other domains, or used as a referenced model. The construction of ontologies, i.e. the specifications of concepts and their relationships among concepts, are represented using Protégé-Frames, a widely popular frame-based open source ontology editor tool and knowledge model developed by Stanford Medical Informatics. The OntoPersonaURM model is described in detail in Section 5.3.

5.2.4. Process 4 (ring 4) – Analysis and Evaluation

In the Analysis and Evaluation process, engineer and developers may conduct one or more of the following activities during process 2 (or immediately after process 2):

- **Personas-Viewpoints-Requirements Analysis (PVRA)**

  In the PVRA activity, a Personas-Viewpoints-Requirements Matrix may be used to provide a summary of the importance of each requirement, relative to the importance of each persona with respect to each persona's viewpoint. The Personas-Viewpoints-Requirements Matrix is a modification of the Persona-Weighted Feature Matrix

provided in [PRUITT 06] by adding personas' viewpoints in the matrix and simplifying the rating scale. A persona may have more than one viewpoint and a requirement in one viewpoint may have varying degrees of importance compared to another viewpoint. Each persona is assigned a meaningful weight according to the relative importance of the persona based on the proportion of the market the persona represents. For each persona, a numeric rating scale of 1-5 is assigned for each requirement: 5 – Critical, 4 – Very Important, 3 – Important 2 – Somewhat important, 1 – Unimportant. A weighted sum for each requirement is computed by multiplying the persona's weight by the requirement's numeric rating and then adding all results across each requirement's row. Requirement with the highest weighted sum is given the highest priority for the development team. A sample Personas-Viewpoints-Requirements Matrix template is shown in Figure 8 and included in Appendix B. An application of the Personas-Viewpoints-Requirements matrix is demonstrated in an online university course registration system case study in chapter 6.

Rating scale: 5 – Critical 4 – Very Important 3 – Important 2 – Somewhat important 1 – Unimportant		PERSONAS	Persona 1			Persona ...			Persona n			WEIGHTED SUM
		WEIGHTS										
		VIEWPOINTS (VPB)	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	
REQUIREMENTS	REQ 1											
	REQ 2											
	REQ 3											
	REQ ..											
	REQ ..											
	REQ n											

Figure 8. Personas-Viewpoints-Requirements Matrix Template

- **Personas-Viewpoints-Scenarios Analysis (PVSA)**

  In the PVSA activity, a Personas-Viewpoints-Scenarios Matrix may be used to provide information on (1) the number of scenarios that refer to each viewpoint (i.e. viewpoint block VPB) of each persona and (2) the frequency count of each scenario. The Personas-Viewpoints-Scenarios Matrix is an extended version of [PRUITT 07, MIKKELSON 00] by the addition of viewpoints in the matrix. The scenario count (per persona-viewpoint) on the last column sums the number of scenarios appears for each VPB. The scenario count information provides insights on which VPB has more referred scenarios versus which VPB has less referred scenario.

  The analysis conducted in the PVSA activity allows engineers and developers to further review the viewpoints, if necessary. In addition, there should be a proportional number of scenarios relative to the importance of each persona, i.e. a primary persona should have a higher number of total scenario count (of all VPBs of that persona) compared to a secondary persona. If a secondary persona shows higher total scenario count than the primary persona, then this may indicate that the primary persona needs may need to be reviewed again. The scenario frequency count (last row) sums the frequency of scenarios referred by all the personas. A high frequency count of a scenario indicates that this scenario is heavily utilized and thus may need special attention. A sample Personas-Viewpoints-Scenarios Matrix template is shown in Figure 9 and included in Appendix B. An application of the Personas-Viewpoints-Scenarios matrix is demonstrated in the GMU PatriotWeb case study described in chapter 6.

		PERSONAS	Persona 1			Persona ...			Persona n			SCENARIO FREQUENCY COUNT
		VIEWPOINTS (VPB)	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	
SCENARIOS (titles)	Scen 1	title 1										
	Scen 2	title 2										
	Scen 3	title 3										
	Scen ..	title ..										
	Scen ..	Title ..										
	Scen ..	title ..										
	Scen n	title n										
SCENARIO COUNT (per Persona-Viewpoint)												

Figure 9. Personas-Viewpoints-Scenarios Matrix Template

- **Scenarios-Usability Goals Evaluation (SUGE)**

  Usability is a non-functional requirement, as it specifies how the user perceives the functionality of the application. There are various usability factors defined by multiple sources in the past [CONSTANTINE 99, ISO 06, NIELSEN 93]. In our research, we have selected eight of these factors (Table 1) that are relevant to our work, i.e. Accuracy, Attractiveness, Efficiency, Learnability, Reliability, Comprehensibility, Clarity, and Rememberability. It is to be noted that these eight chosen usability factors are applied in the PatriotWeb prototype design (chapter 7) and also used to categorize the usability questions (written in non-technical terms) in the questionnaire (Appendix F) to seek respondents' opinions on usability of the prototype PatriotWeb system. The usability factors, prototype PatriotWeb system, and the questionnaire help to support the empirical evaluation on the effectiveness of persona concept (chapter 8). Hence, it is not the scope of this research to conduct formal usability tests on the prototype PatriotWeb system.

Table 1. Eight Usability Factors for Scenarios-Usability Goals Evaluation (SUGE)

Accuracy	The degree to which specified users can achieve specified goals with accuracy and completeness in a specified context of use. Operations (menus/links/buttons) produce the correct result pages.
Attractiveness	The page content view, layout, color, font style size are appealing, impressive.
Clarity	Texts and images are readable. Content representations do not cause eyestrain.
Comprehensibility	Interface elements (menus/links/buttons) are visible and easy to understand. Easy to understand what each menu/link/button does. Interface elements and content representations are uniform, consistent, not clutter the page, and easily accessible.
Efficiency	Ease of use. How long it takes or quickly for a user to complete a particular task. Indicators of tasks completion time (slow/fast, less time/more time) include the number of (more/less) mouse clicks or the number of (more/less) webpages a user needs to take to reach the target page, the number of (more/less) repeated steps to complete the task.
Learnability	Ease of learning. How well or easy a user learns how to use a particular function, how to navigate the site, how to carry out a particular task?
Reliability	Information presented on the site is credible, useful.
Rememberability	The system is easy to remember for the casual user. How well a user remembers the system interface after a period of time (an hour/day/week/month)? How well a user able to remember how the system looks and what the system does?

We consulted with the Scenario-Goal-Issue Matrix approach provided in [AOYAMA 07] and created a Scenarios-Usability Goals Matrix in the SUGE activity to identify any requirement issues with respect to the eight usability factors chosen in this research. The Scenarios-Usability Goals Matrix is a slight modification of Scenario-

Goal-Issue Matrix approach in [AOYAMA 07] in the selection of the eight usability factors by removing "Responsiveness" and "Usefulness" factors used in [AOYAMA 07], adding "Accuracy" and "Efficiency" with revised descriptions and metrics, and editing some usability factors' names and descriptions. A sample Scenarios-Usability Goals Matrix template is shown in Figure 10 and included in Appendix B. From the viewpoint of each persona, engineers review the scenarios with respect to the eight usability factors and for each empty cell in Figure 10, record any requirement issues and/or any improvements needed to resolve the issues.

		USABILITY GOALS							
		Accuracy	Attractiveness	Efficiency	Learnability	Reliability	Comprehensibility	Clarity	Rememberability
SCENARIOS	Scenario 1								
	Scenario 2								
	Scenario 3								
	Scenario 4								
	.....								
	Scenario n								

Figure 10. Scenarios-Usability Goals Matrix Template

- **Scenarios-Tasks-Requirements Evaluation (STRE)**

  In the STRE activity, a Scenarios-Tasks-Requirements Matrix may be used to provide a summary of the task of a scenario that fulfills a particular requirement. A sample Scenarios-Tasks-Requirements Matrix template is shown in Figure 11 and included in

Appendix B. As shown in Figure 11, all the requirements are listed horizontally on the top row and all the tasks for each scenario are listed vertically in the left columns. A qualitative indication such as “Meet”, “Partial Meet”, or “Not Meet” can be filled in an empty cell to correspond to the task and requirement. Additional information may also be added in the empty cells to indicate any issue that needs to be addressed.

		TASKS	REQUIREMENTS			
			REQ 1	RRQ 2	.....	REQ n
SCENARIOS	Scenario 1	Task 1				
		Task 2				
		.....				
		Task n				
	Scenario 2	Task 1				
		Task 2				
		.....				
		Task n				
	.....	Task 1				
		Task 2				
		.....				
		Task n				
	Scenario n	Task 1				
		Task 2				
		.....				
		Task n				

Figure 11. Scenarios-Tasks-Requirements Matrix Template

- **Conflict Requirements Resolution (CRR)**

  In the CRR activity, through conducting various analysis and evaluation activities in the Analysis and Evaluation process, identified requirement conflicts are reviewed against the personas’ needs and behaviors, as well as the overall business’ goals. The reconciliation of requirement conflicts is not an easy task and requires the development team to go through one or more of the processes of the CDP model.

5.2.5. Integration into Requirements Engineering Activities

Requirements engineering (RE) is a process of eliciting, understanding, specifying, validating customers' and users' requirements, and identifying constraints on the application to be constructed. The RE process is generally divided into a set of inter-twined activities [SOMMERVILLE 10]: Requirements Elicitation, Requirements Analysis and Modeling, Requirements Specification, Requirements Validation, and Requirements Management.

As the concept of persona captures rich behavior model of users interacting with an application system and helps requirements engineers identify functionalities of an application system that meet the target users' needs and goals, the proposed CDP model in which the persona concept plays the central role can thus be incorporated into one or more activities of the RE process. Table 2 presents the integration of the proposed CDP processes with the RE activities. It shows the processes and deliverables of the proposed CDP model, the RE activity(ies) with which each CDP process can be incorporated, and the justification of the incorporation.

Table 2. Integration of CDP Processes into RE Activities

Requirements Engineering (RE)	**Concept Development Process (CDP)**		**Integration of CDP Processes into RE Activities**
Activities	**Processes**	**Deliverables**	**Justification**
Requirements Elicitation	Personas Construction	• Persona Profile Document (PPD) • Persona Definition Document (PDD)	The Personas Construction process gathers, analyzes, and synthesizes information related to the target users that interact

			with the system. Information synthesized to elicit requirements include users' characteristics, needs, behaviors, goals, and types (primary persona, secondary persona).
	Viewpoints Identification and Construction	• Viewpoint Document (VPD)	Viewpoints Identification and Construction process encapsulates some, but not all information about a system's requirements. This information may be derived from an analysis of the PPDs and PDDs created in the Personas Construction process, consultations with the marketing team on target users' profiles, or discussions with system stakeholders. The Viewpoints Identification and Construction process helps structuring the requirements to represent the perspectives of different personas.
Requirements Analysis	Personas Construction	• Persona Profile Document (PPD) • Persona Definition Document (PDD)	The personas' information gathered and recorded in the PPDs and PDDs helps in creating the VPDs in the Viewpoints Identification and Construction process and in facilitating the construction of models; in particular, the personas model.
	Viewpoints Identification and Construction	• Viewpoint Document (VPD)	The Viewpoints Identification and Construction process aims to model the domain from

			different perspectives of the identified personas by defining one or more views representing the various viewpoints of the personas. Each VPD contains information that defines one or more views at a certain level of detail and addresses certain design concerns by the persona playing a particular role in a particular environment.
	Concepts Modeling	• UML class and instance diagrams - Persona class and instance diagrams - Behavioral-GST class and instance diagrams - Requirements class and instance diagrams	Based on the VPDs produced in the Viewpoints Identification and Construction process, one or more views are created for each viewpoint and are graphically represented by constructing UML models (class diagrams and instance diagrams) representing design concepts of personas, viewpoints, environments, goals, scenarios, tasks, requirements, and so forth.
Requirements Specification	Personas Construction	• Persona Profile Document (PPD) • Persona Definition Document (PDD)	The PPDs and PDDs highlight information on the requirements the personas have on the system and assist in providing personas' concerns input to the VPDs in the Viewpoints Identification and Construction process.
	Viewpoints Identification and Construction	• Viewpoint Document (VPD)	The VPDs created in the Viewpoints Identification and Construction process provide requirements input

			to the Software Requirements Specifications document.
	Concepts Modeling	• Ontology Specifications (Protégé-Frames) - Persona Ontology - Behavioral-GST Ontology - Requirements Ontology	The specifications of concepts and their properties (attributes, cardinalities, relationships) of the ontologies are specified explicitly using a widely popular Protégé-Frames ontology tool.
Requirements Validation	Analysis and Evaluation	• Analysis and Evaluation Matrices - Personas-Viewpoints Requirements Analysis (PVRA) - Personas-Viewpoints-Scenarios Analysis (PVSA) - Scenarios-Tasks-Requirements Evaluation (STRE) - Conflict Requirements Resolution (CRR)	One or more matrices are performed in the Analysis and Evaluation process to help engineers in evaluating personas, viewpoints, scenarios, tasks, and goals, as well as checking for requirements issues and redundancy.

5.3. Ontology-Based Persona-Driven User Requirements (OntoPersonaURM) Model

An Ontology-Based Persona-Driven User Requirements Modeling (OntoPersonaURM) model is proposed in process 3 (Concepts Modeling process) of the CDP model. Within the OntoPersonaURM model, a five-step iterative ontology development process has been developed to help guide engineers and developers in the process of constructing the

ontologies for explicit specifications of the concept of persona in representing users' characteristics and the concepts of viewpoint, goal, scenario, task, and requirement. The five-step iterative ontology development process is a revision of the typical ontology development process [NOY 01] with the addition of Step 5: Combine Ontologies in the typical ontology process. The added step (Step 5) in the five-step ontology development process helps ontology engineers to combine two or more ontologies on the Protégé-Frames ontology editing environment. The ontology development process is described in Table 3 Section 5.3.1. Based on the VPD created in process 2 of the CDP model, i.e. Viewpoints Identification and Construction process, the view consisting of models of the concepts of persona, goal, scenario, task, and requirement are represented graphically using the UML class and instance diagrams notation.

The OntoPersonaURM model (Figure 12) is composed of three generic interrelated domain independent ontologies that are developed to be as general as possible yet encompasses a broad set of concepts that can be applied, modified, extended to other domains, or used as a referenced model:

- **Persona Ontology:**

  The Persona Ontology covers general concepts pertaining to person characteristics including education, abilities, interests, knowledge, viewpoints, environments, and so forth.

- **Behavioral Goal-Scenario-Task (GST) Ontology:**

  The Behavioral GST Ontology captures and defines the needs and behaviors of the personas and the system-to-be, i.e. viewpoint, goal, scenario, and task concepts.

- **Requirements Ontology:**

  The Requirements Ontology specifies general concepts for the representation of the requirements and their properties.

The Persona Ontology, Behavioral-GST Ontology, and Requirements Ontology are described in detail in Section 5.3.2, 5.3.3, and 5.3.4 respectively.

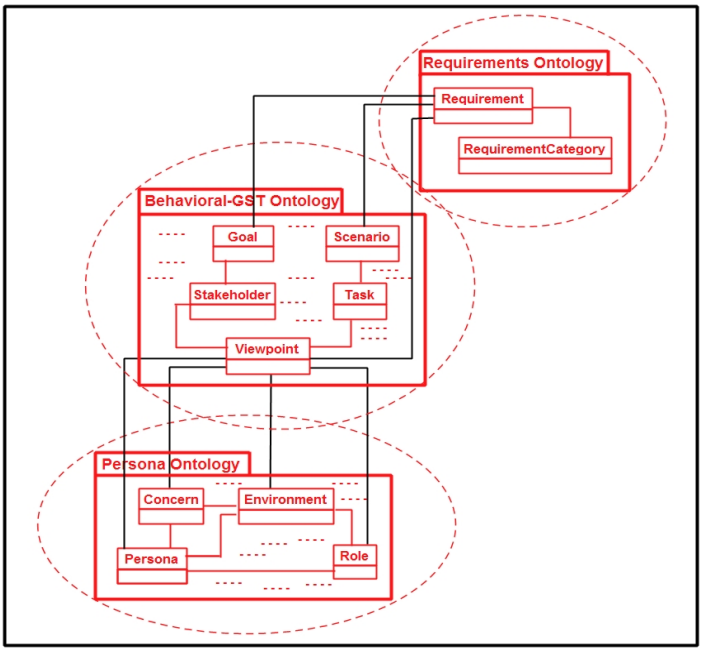


Figure 12. OntoPersonaURM Model – Persona Ontology, Behavioral-GST Ontology, Requirements Ontology

5.3.1. Ontology Development Process

It is to be emphasized that developing a new ontology is often tedious and time consuming; it normally requires engineers and developers to have sufficient knowledge

in ontology specifications and familiar with ontology development environment. There is no single correct ontology for any domain [NOY 00a]. In building our ontologies for the OntoPersonaURM model, we consulted with the guidelines suggested in [NOY 00a] and developed the five-step iterative ontology development process. The five-step process differs from [NOY 00a] in that we have included a step of the process of combining ontologies as part of the ontology development process in [NOY 00a]. The five-step iterative ontology development process is outlined in Table 3.

Table 3. Five-Step Ontology Development Process

Step 1: Synthesize Information Collected
Information gathered and described in the persona and viewpoint documents created through collaboration with marketing analysts, ontology engineers, and requirement engineers during the requirements elicitation process are analyzed and synthesized. Terms extracted from these documents are candidates for the definition of classes and properties in the ontology(ies).
Step 2: Consult Existing Ontologies
There are extensive libraries of reusable ontologies available on the Web. For examples, the Protégé ontology library [PROTÉGÉ a] maintains a good collection of ontologies, the DAML ontology library [DAML], user profile ontology [GOLEMATI 07], personal ontology [KATIFORI 08], and so forth. As building a new ontology from scratch is a time consuming process, if an existing solution ontology is available and is relevant to the application domain in hand, then it is suggested to consult with the existing ontology to determine if we can reuse, refine, or extend existing classes and properties.
Step 3: Define Classes and Properties
A top-down approach is adopted to define the class hierarchy (super-sub-class), i.e. from most general concepts to specialized concepts. A set of potential classes, class hierarchy, and class properties (i.e. attributes, cardinalities, and relationships with other classes) are identified, defined, and specified in the Protégé-Frames editor tool. This step occupies the most time. Some approaches in creating classes and

relationships:

(1). Classes are not duplicated across one or more ontologies in the Protégé-Frames tool. If class A is related with one or more classes in two or more ontologies, then class A is specified explicitly (in Protégé-Frames tool) only in the ontology that has the most number of its classes related with class A.
(2). If a class in one ontology is related to another class in another ontology, the relationship property of these two classes is specified in the Protégé-Frames tool ***after*** the two ontologies are combined (in Step 5) in the Protégé-Frames tool.

Step 4: Create Instances

Instances of the classes are created in the Protégé-Frames editor. Creating class instances can help to correct mistakes and fine-tune the classes and properties in the ontology.

Step 5: Combine Ontologies

If one or more relationships exist between classes of two ontologies, the ontologies may be combined by including the related ontology into the current ontology via Protégé-Frames' "Manage Included Projects" menu [PROTÉGÉ b]. For example, in the OntoPersonaURM model, the Persona Ontology contains classes that have relationships with classes of the Behavioral-GST Ontology, thus the Behavioral-GST Ontology may be included in the Persona Ontology. Downsides of including ontologies directly in Protégé-Frames via the "Manage Included Projects" menu are (1) if two same classes are created in two ontologies before the "Manage Included Projects" operation, then the editor may alter the structure of the class hierarchy, i.e. a sub-class of a class may not still be a sub-class after the ontologies are included; thus manual checking and editing of the class hierarchy is needed, and (2) it does not offer consistency check on the classes. The included ontology remains as a separate ontology which can be opened and edited separately [PROTÉGÉ b].

Two ontologies may also be merged to form one single ontology in Protégé-Frames via "Merge Included Projects" menu [PROTÉGÉ b], i.e. all classes in the included ontology become part of the members in the current ontology and thus can be edited as with the current ontology.

Another approach in combining (or merging) ontologies is via third party tool such as Prompt [NOY 00b], which offers a toolset for merging and mapping ontologies. The toolset offers the capability of checking for consistency of concepts during merging of ontologies by offering suggestions to users. Unlike the Protégé-Frames' Included Project feature, Prompt does not alter the class hierarchy by itself. All changes are done through suggestions offered to users and users make the necessary decisions by accepting or rejecting the suggestions. Prompt is available as a plugin in the Protégé-

Frames ontology editor.

Combining (or merging) related ontologies help ontology engineers better understand the relationships of classes between ontologies, identify conflicts, and make necessary changes. Ontology engineers may need to revisit one or more previous steps to refine the ontologies.

5.3.2. Persona Ontology

The Persona Ontology provides a broad set of concepts pertaining to person characteristics and environment. Unlike other ontology research conducted in the past by other researchers [GOLEMATI 07, KATIFORI 08] in modeling user characteristics and preferences, the concepts chosen in the Persona Ontology in the proposed OntoPersonaURM model not only capture the basic characteristics and preferences of a person such as age, gender, name, education, occupation, abilities, expertise, interests and so forth, but also the relationships to the environment in which the person engages in. The Environment concept plays an important role in the Personal Ontology as it relates to the persona's viewpoint which specifies the concerns of the persona, the scenarios the persona have and the tasks the persona performed to achieve the goals of the persona interacting in a particular environment. The Environment concept thus captures the (partial) dynamic aspect of the persona, such as the location, time, and frequency the persona is interacting in a particular environment.

The modeling elements in the Persona Ontology are mainly class and association which are represented graphically using the UML class diagram as shown in Figure 13. The concepts are explicitly specified ontologically in Protégé-Frames as class hierarchy,

i.e. super-sub-class as shown (partial view) in Figure 14. The Persona Ontology encompasses a basic set of general concepts to allow the ontology to be enriched or modified by the ontology designer through addition or elimination of the classes, based on the needs of the domain. The addition of new classes may be accomplished through super-sub-class relationship (or "is-a" association), as well as class instantiation (or instances of classes). Detailed descriptions of the classes and properties of the Persona Ontology are provided in Appendix C.

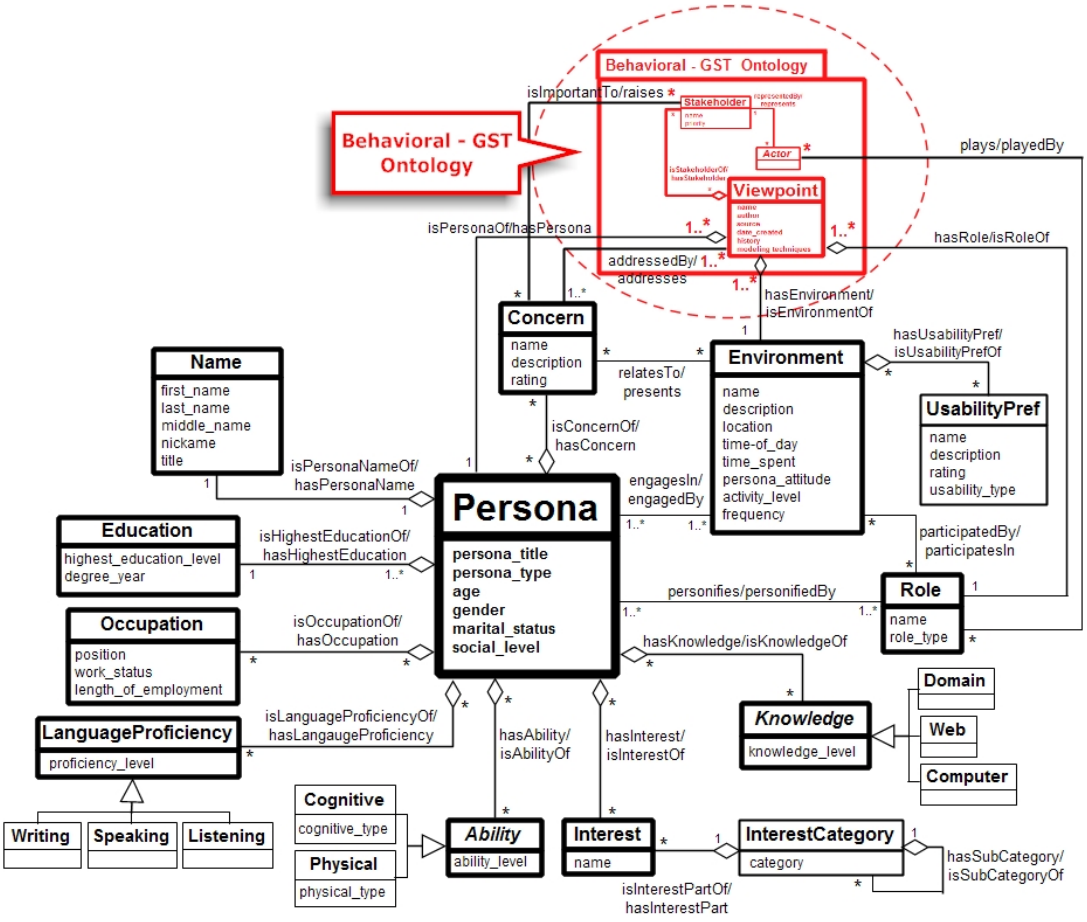


Figure 13. UML Class Diagram – Persona Ontology

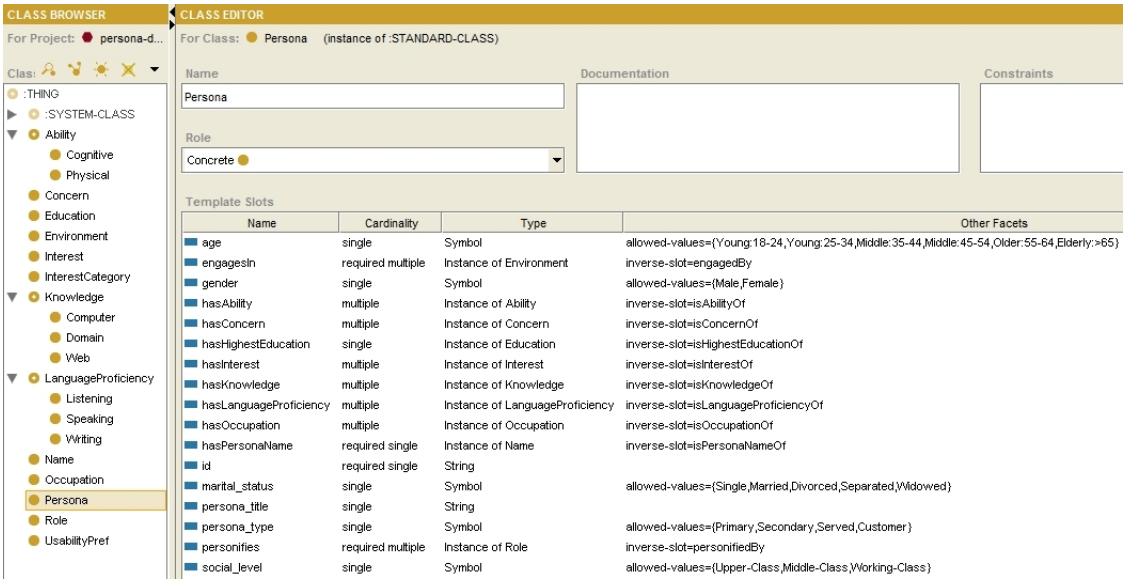


Figure 14. Protégé-Frames – Persona Ontology

In the Persona Ontology, the Persona class is the central class containing the general characteristics of a person. There are certain areas to be highlighted in regard to the Persona Ontology:

- Some entities in the Persona class are represented as separate distinct classes via associations rather than attributes as these entities have internal structures or complex data types that may be useful for the ontology designer to apply validation or formatting rules to be recognized by the ontology reasoner. For examples, a person's name is represented as a Name class, since it contains internal structures such as first_name, last_name, middle_name, title, and nickname; a person's education is represented as an Education class as it captures internal structures such as

degree_year, degree_title, highest_education_level. If these entities (person's name, education, occupation) were represented as attributes of String type in the Persona class, then the internal structure of these complex data values and their semantics could be lost and thus could not be made available in the ontology for further processing, filtering, sorting, etc. However, the decision to represent an entity as a class or an attribute is a design choice to be decided by the ontology designer, based on the application domain in hand.

- The Environment class is represented in the Persona Ontology rather than the Behavioral-GST Ontology (Section 5.3.3), since the Environment class is directly related with several classes in the Persona Ontology, namely, Persona, Role, and Concern classes. Constraint check and query execution on the classes in the Persona Ontology can thus be executed easily with an appropriate plugin tool such as PAL [PAL].
- The Interest class (Figure 15) is associated with the InterestCategory class via aggregation, i.e. an InterestCategory class is an aggregation of an Interest class. As it is common that there are various interest names that may belong to a same interest category, we chose to place a one-to-many relationship between InterestCategory and Interest classes, i.e. an interest category hasInterestPart one or more interests. For example, an interest category "*Entertainment*" hasInterestPart "*Listening music*", "*Playing guitar*". For Simplicity, we chose to specify that an interest is part of one and only one interest category. For example, an interest "*Listening music*" isInterestPartOf "*Entertainment*". The cardinalities between Interest and

InterestCategory classes (and vice versa) is a design choice and thus may be modified by the ontology designer.

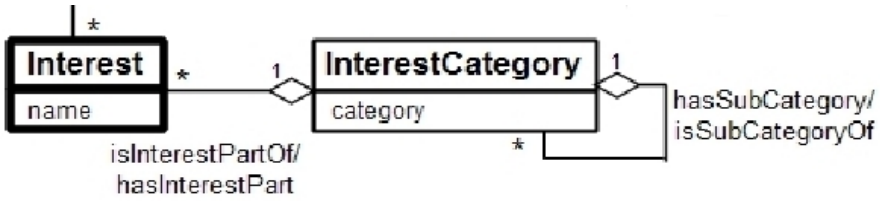


Figure 15. Interest Class – Persona Ontology

5.3.3. Behavioral-GST Ontology

The Behavioral-GST Ontology captures the behavior of the system-to-be and defines main concepts including Viewpoint, Goal, Scenario, and Task concepts. The supporting concepts include Constraint, Obstacle, Action, Tool, Stakeholder, and Actor concepts. Similar to the Persona Ontology described in Section 5.3.2, the modeling elements used in the Behavioral-GST Ontology are mainly class and association represented graphically using the UML Class Diagram as shown in Figure 16. The various concepts are represented in the Behavioral-GST Ontology as class hierarchy, i.e. super-sub-classes displayed in the Protégé-Frames as shown (partial view) in Figure 17. Detailed descriptions of the classes and properties of the Behavioral-GST Ontology are provided in Appendix C. It is to be emphasized that one or more attributes in the Behavioral-GST

Ontology may be eliminated and many other attributes may be defined and added by the ontology designer, depending on the nature of the application domain.

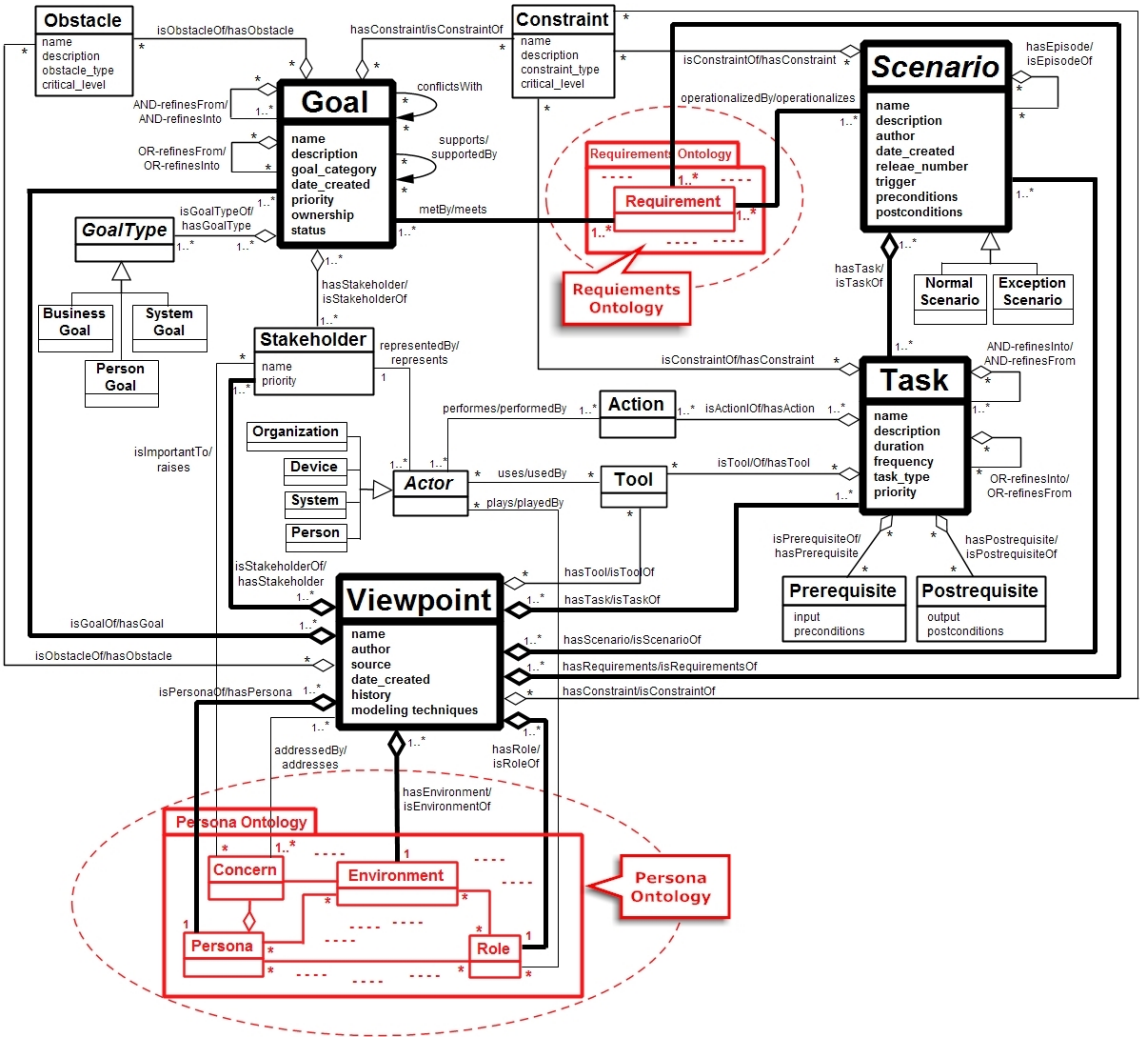


Figure 16. UML Class Diagram – Behavioral-GST Ontology

As shown in Figure 16, the Viewpoint class, the Goal class, and the Scenario class are the central classes of the Behavioral-GST ontology as they relate with other ontologies in the OntoPersonaURM model in the following ways:

- The Behavioral-GST Ontology is related to the Persona Ontology via the Viewpoint class (of the Behavioral-GST Ontology).
- The Behavioral-GST Ontology is related to the Requirements Ontology via three classes (of the Behavioral-GST Ontology): Viewpoint class, Goal class, and Scenario class.

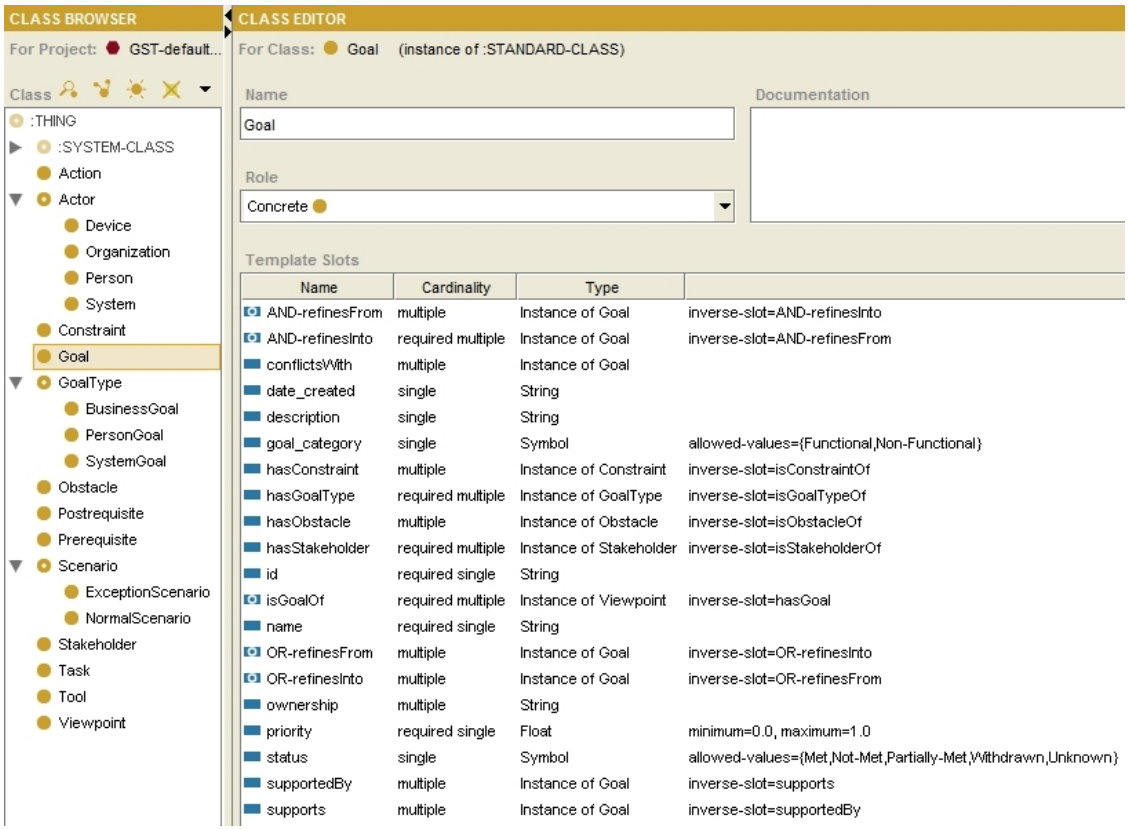


Figure 17. Protégé-Frames – Behavioral-GST Ontology

5.3.4. Requirements Ontology

The Requirements Ontology contains general concepts that are considered applicable to most domains for the representation of the requirements and their properties. In the Requirements Ontology, the Requirement class is the central class which defines a typical yet broad set of requirement properties. Supporting classes include RequirementCategory, class and SRS (Systems or Software Requirements Specification) class. The UML Class Diagram for the Requirements Ontology and the ontological representation of the requirements classes in Protégé-Frames are depicted in Figure 18 and Figure 19 (partial view) respectively. Detailed descriptions of the classes and properties of the Requirements Ontology are provided in Appendix C. The attributes listed in Appendix C are by no means exhaustive. As with the Persona Ontology and Behavioral-GST Ontology presented in previous sections, one or more attributes may be eliminated and many other attributes may be defined and added by the ontology designer.

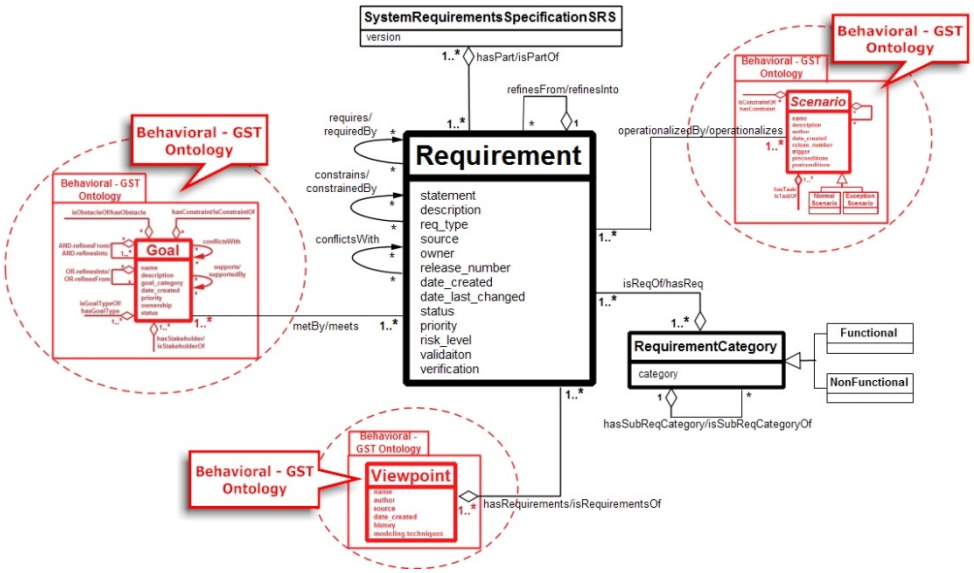


Figure 18. UML Class Diagram – Requirements Ontology

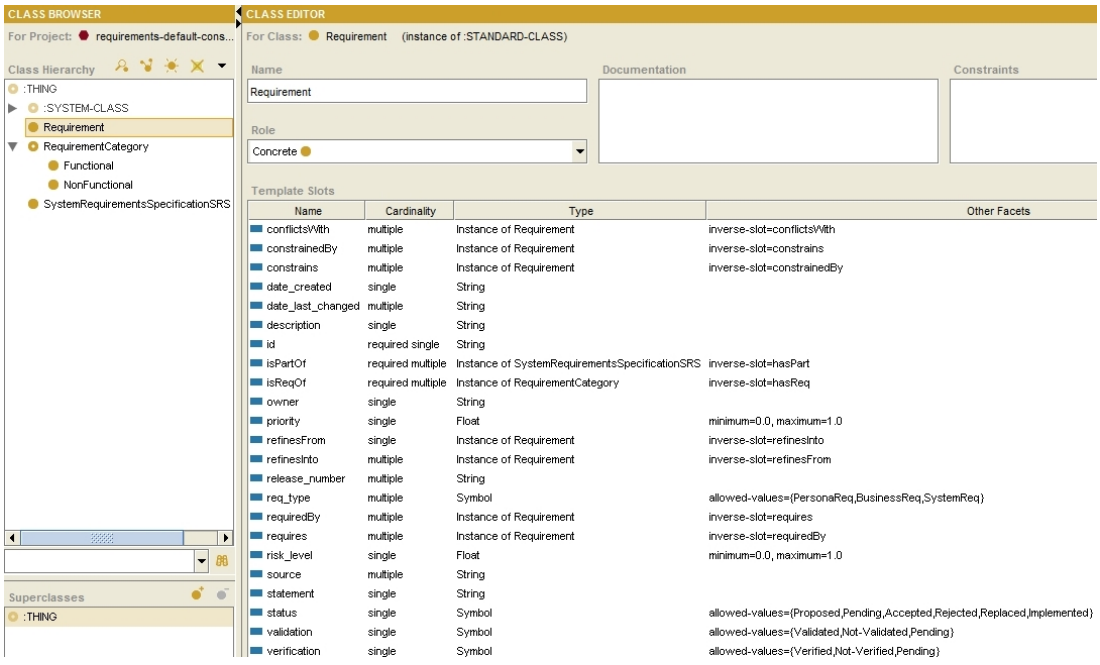


Figure 19. Protégé-Frames – Requirements Ontology

CHAPTER 6: CASE STUDY – GMU PATRIOTWEB WEBSITE

6.1 Introduction

To demonstrate the application of persona concept and ontology-based approach in the Persona-URM methodological framework developed in this research, the George Mason University (GMU) PatriotWeb website case study was selected and studied. The GMU PatriotWeb [PATRIOTWEB a] is a self-service website for students, faculty, and staff providing online services including schedule of classes, class registration and tuition payment, financial aid applications, timesheets, paystubs, class list, and grading. The PatriotWeb self-service website is accessible on various platforms: desktop, laptop, tablet, and smartphone (with Internet connection). As of this writing, there is no dedicated mobile version of the streamlined PatriotWeb self-service website.

The GMU PatriotWeb case study was chosen for this research because of its variety of users with different backgrounds, roles, requirement needs, as well as its common daily usage by university students, faculty, and staff. This case study thus served as a base for evaluating the effectiveness of applying the concepts of persona and ontology-based approach to validate the research hypotheses. Evaluation procedures and results are presented in chapter 8. In this chapter, the application of the PersonaURM methodological framework (i.e. CDP and OntoPersonaURM models) are demonstrated step-by-step using the chosen PatriotWeb website as a base domain. Figure 20 and Figure

21 show the screenshots of the login page and homepage of PatriotWeb website respectively.

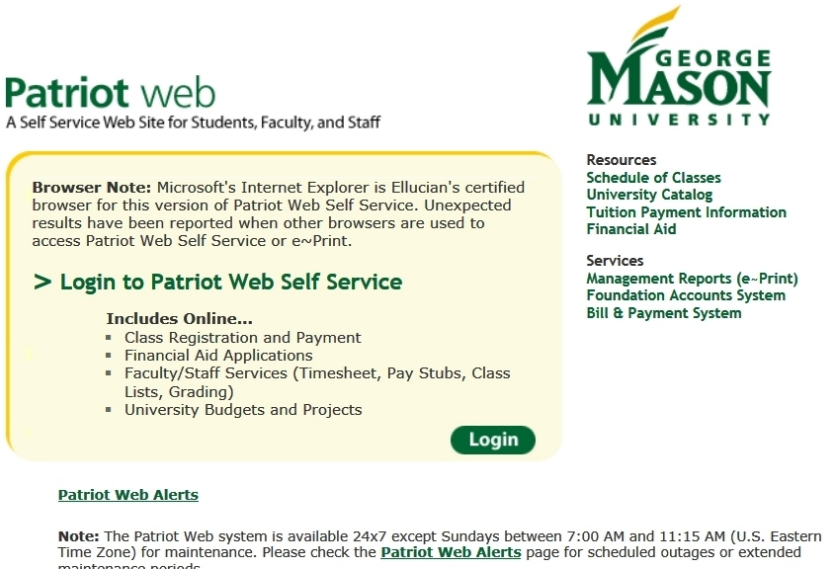


Figure 20. Case Study – PatriotWeb Login Page

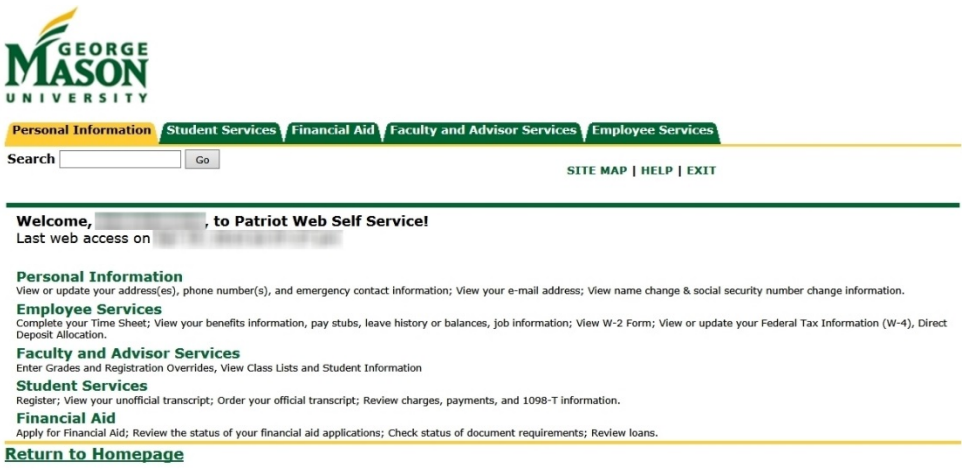


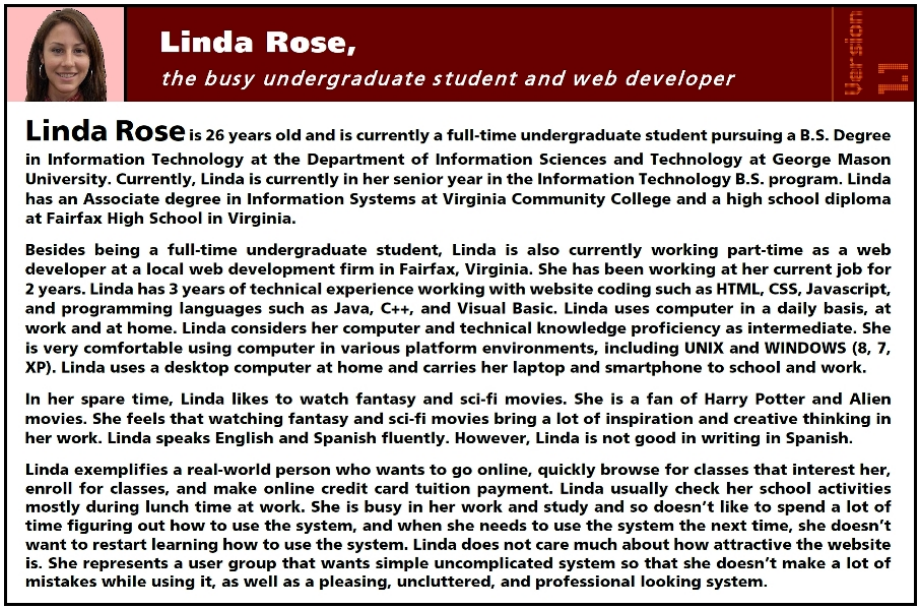
Figure 21. Case Study – PatriotWeb Homepage

6.2 CDP Model

Process 1 – Personas Construction

As the identification of personas is not the scope in this dissertation, we have made an assumption that input information on common users of the PatriotWeb website are already known, i.e. the primary target users are students that use the system to browse and register for courses, check student records/grades, review financial aid information, pay tuition and fees, and so forth, and the secondary users are the application developers and site administrator. In this dissertation, the primary users are the main focus. Thus, a primary persona, "***Linda Rose, the busy undergraduate student and web developer***" is created. It is to be highlighted that another potential primary persona that may be considered for the PatriotWeb case study is "*John Lim, the high energy tech savvy instructor*". John Lim is a teaching faculty who uses the PatriotWeb mainly for importing students' grades, submitting midterm evaluations, overriding students' course registration. However, using different personas may add an additional level of variability and complexity when evaluating the effectiveness of the persona concept. Hence, in this dissertation, we focus our attention on one primary persona Linda Rose and demonstrate the application of the persona Linda Rose throughout the case study.

The deliverables in this process consist of two documents: Persona Profile Document (PPD), and Persona Definition Document (PDD). Figure 22 and Figure 23 illustrate the PPD and PPD for persona Linda Rose respectively.



Linda Rose,

the busy undergraduate student and web developer

version 1.1

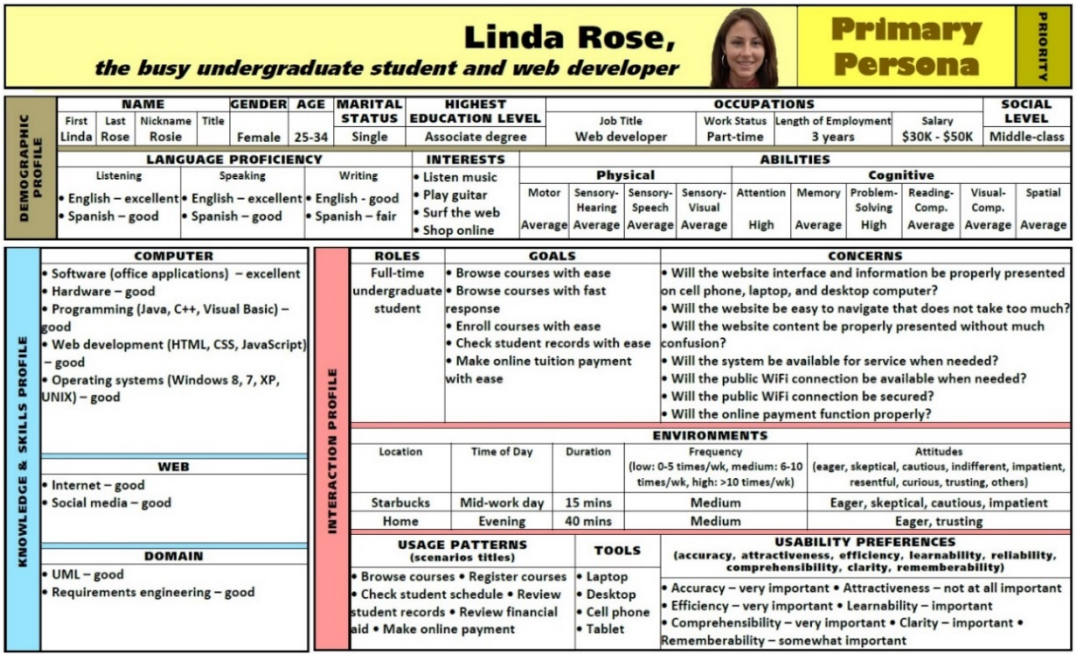
Linda Rose is 26 years old and is currently a full-time undergraduate student pursuing a B.S. Degree in Information Technology at the Department of Information Sciences and Technology at George Mason University. Currently, Linda is currently in her senior year in the Information Technology B.S. program. Linda has an Associate degree in Information Systems at Virginia Community College and a high school diploma at Fairfax High School in Virginia.

Besides being a full-time undergraduate student, Linda is also currently working part-time as a web developer at a local web development firm in Fairfax, Virginia. She has been working at her current job for 2 years. Linda has 3 years of technical experience working with website coding such as HTML, CSS, Javascript, and programming languages such as Java, C++, and Visual Basic. Linda uses computer in a daily basis, at work and at home. Linda considers her computer and technical knowledge proficiency as intermediate. She is very comfortable using computer in various platform environments, including UNIX and WINDOWS (8, 7, XP). Linda uses a desktop computer at home and carries her laptop and smartphone to school and work.

In her spare time, Linda likes to watch fantasy and sci-fi movies. She is a fan of Harry Potter and Alien movies. She feels that watching fantasy and sci-fi movies bring a lot of inspiration and creative thinking in her work. Linda speaks English and Spanish fluently. However, Linda is not good in writing in Spanish.

Linda exemplifies a real-world person who wants to go online, quickly browse for classes that interest her, enroll for classes, and make online credit card tuition payment. Linda usually check her school activities mostly during lunch time at work. She is busy in her work and study and so doesn't like to spend a lot of time figuring out how to use the system, and when she needs to use the system the next time, she doesn't want to restart learning how to use the system. Linda does not care much about how attractive the website is. She represents a user group that wants simple uncomplicated system so that she doesn't make a lot of mistakes while using it, as well as a pleasing, uncluttered, and professional looking system.

Figure 22. Case Study – PPD for Primary Persona Linda Rose



Linda Rose,

the busy undergraduate student and web developer

Primary Persona

PRIORITY

DEMOGRAPHIC PROFILE

NAME				GENDER	AGE	MARITAL STATUS	HIGHEST EDUCATION LEVEL	OCCUPATIONS				SOCIAL LEVEL
First	Last	Nickname	Title					Job Title	Work Status	Length of Employment	Salary	
Linda	Rose	Rosie		Female	25-34	Single	Associate degree	Web developer	Part-time	3 years	$30K - $50K	Middle-class

LANGUAGE PROFICIENCY			INTERESTS
Listening	Speaking	Writing	
• English – excellent • Spanish – good	• English – excellent • Spanish – good	• English - good • Spanish – fair	• Listen music • Play guitar • Surf the web • Shop online

ABILITIES									
Physical				Cognitive					
Motor	Sensory-Hearing	Sensory-Speech	Sensory-Visual	Attention	Memory	Problem-Solving	Reading-Comp.	Visual-Comp.	Spatial
Average	Average	Average	Average	High	Average	High	Average	Average	Average

KNOWLEDGE & SKILLS PROFILE

COMPUTER

- Software (office applications) – excellent
- Hardware – good
- Programming (Java, C++, Visual Basic) – good
- Web development (HTML, CSS, JavaScript) – good
- Operating systems (Windows 8, 7, XP, UNIX) – good

WEB

- Internet – good
- Social media – good

DOMAIN

- UML – good
- Requirements engineering – good

INTERACTION PROFILE

ROLES	GOALS	CONCERNS
Full-time undergraduate student	• Browse courses with ease • Browse courses with fast response • Enroll courses with ease • Check student records with ease • Make online tuition payment with ease	• Will the website interface and information be properly presented on cell phone, laptop, and desktop computer? • Will the website be easy to navigate that does not take too much? • Will the website content be properly presented without much confusion? • Will the system be available for service when needed? • Will the public WiFi connection be available when needed? • Will the public WiFi connection be secured? • Will the online payment function properly?

ENVIRONMENTS

Location	Time of Day	Duration	Frequency (low: 0-5 times/wk, medium: 6-10 times/wk, high: >10 times/wk)	Attitudes (eager, skeptical, cautious, indifferent, impatient, resentful, curious, trusting, others)
Starbucks	Mid-work day	15 mins	Medium	Eager, skeptical, cautious, impatient
Home	Evening	40 mins	Medium	Eager, trusting

USAGE PATTERNS (scenarios titles)	TOOLS	USABILITY PREFERENCES (accuracy, attractiveness, efficiency, learnability, reliability, comprehensibility, clarity, rememberability)
• Browse courses • Register courses • Check student schedule • Review student records • Review financial aid • Make online payment	• Laptop • Desktop • Cell phone • Tablet	• Accuracy – very important • Attractiveness – not at all important • Efficiency – very important • Learnability – important • Comprehensibility – very important • Clarity – important • Rememberability – somewhat important

Figure 23. Case Study – PDD for Primary Persona Linda Rose

An excerpt from the PPD is reproduced below with underlined words/sentences indicate the focus areas:

> "*Linda exemplifies a real-world person who wants to go online, quickly browse for classes that interest her, enroll for classes, and make online credit card tuition payment. Linda usually check her school activities mostly during lunch time at work. She is busy in her work and study and so doesn't like to spend a lot of time figuring out how to use the system, and when she needs to use the system the next time, she doesn't want to restart learning how to use the system. Linda does not care much about how attractive the website is. She represents a user group that wants simple uncomplicated system so that she doesn't make a lot of mistakes while using it, as well as a pleasing, uncluttered, and professional looking system*."

The excerpt from the PPD thus highlights or brings attention to developers the requirements that are important to Linda Rose. In particular, requirements that address usability issues such as "**Efficiency**", "**Comprehensibility**", "**Accuracy**", and "**Learnability**". For examples, requirements pertaining to (i) easy to find available services and without too much time taken, (ii) content and hyperlinks that do not cluttered on a webpage, (iii) the action of menus and hyperlinks that produce correct webpages, and (iv) easy to learn how to use the system. The PPD and PDD created in the Personas Construction process help in the Analysis and Evaluation process (process 4) in refining requirements on the PatriotWeb system with respect to usability issues.

Process 2 – Viewpoints Identification and Construction

Based on the PDD created in process 1, a Viewpoint Document (VPD) is created that contains information defining one or more views at a certain level of detail and addressing certain design concerns by the persona playing a particular role in a particular environment. For the PatriotWeb case study, a primary persona "*Linda Rose, the busy undergraduate student and web developer*" uses the course registration system to browse and register for courses in two different situations:

(1). In a public place (e.g. Starbucks) in mid-day lunch hour during work using her smartphone relying on WiFi internet connection.

(2). In a private place (e.g.at home) during evening time using her desktop computer with strong and reliable internet connection.

Therefore, two viewpoints or Viewpoint Blocks (VPB) are identified:

VPB 1 <*Linda Rose, Full-Time Undergraduate Student, Starbucks (mid - workday)*>

VPB 2 <*Linda Rose, Full-Time Undergraduate Student, Home (evening)*>

These two VPBs have slightly different goals, concerns, scenarios, goals, and requirements, as the environments are different. For examples:

- A public place where (i) internet connection depends on the availability and reliability of a strong WiFi connection, (ii) security and privacy issues are important factor especially when making payments using credit card, and (iii) constraint in time (e.g. during lunch work time) is a concern which makes accuracy and ease of use important usability requirements.

- A comfortable home environment where (i) internet connection is available and reliable, (ii) less concern for security and privacy, and (iii) less concern on time constraint which thus allows for more time to spend on the website or some mistakes to occur.

Hence, the Viewpoint Document (VPD) delivered in the Viewpoints Identification and Construction process highlights and brings attention to developers the requirements that may need refinements on the current PatriotWeb system. For examples, non-functional requirements pertaining to (i) content presentation layout on devices with smaller screen sizes such as mobile phones and tablets, (ii) navigational system that enables Linda Rose to find services options with ease and less time taken, and (iii) security and privacy requirements in public places. The VPD for persona Linda Rose for VPB 1 <*Linda Rose, Full-Time Undergraduate Student, Starbucks (mid-work day)*> is illustrated in Figure 24. The VPD, together with PPD and PDD produced in process 1, helps in the Analysis and Evaluation process (process 4) in refining requirements on the PatriotWeb system.

Process 3 – Concepts Modeling

Based on the VPD created in process 2, the view consisting of models of personas, goals, scenarios, tasks, and requirements are represented graphically using the UML class and instance diagrams notation. The application of OntoPersonaURM model developed in this process on the PatriotWeb case study is demonstrated in next section, section 6.3.

<Linda Rose, Full-Time Undergraduate Student, Starbucks (mid-work day)>	VIEWPOINT (VP) NAME

SOURCES	STAKEHODLERS	HISTORY (VP-version, date, author, changes)	ROLE
• Persona Profile Document (PPD) • Persona Definition Document (PDD)	• Administrators • Billing system • Instructors	VP-v1.0, 10-27-2013, John Deer	Full-time undergraduate student

ENVIRONMENT (location, time of day, duration, tools, etc)	• Starbucks coffee shop • Mid-day (lunch work time) • Limited available time (~15 mins) • Cell phone (internet WiFi capable) • Heavy human traffic • Rely on WiFi connection • Security and privacy issues

GOALS	SCENARIOS, TASKS	
	Scenarios (titles)	Tasks (titles)
• Browse courses with ease • Enroll courses with ease • Check student records with ease	• Search for classes	• Find a specific course • Browse course registration catalog
CONCERNS	• Register for classes	• Find a specific course • Browse course registration catalog • Add course to schedule • Select specific course
• The presentation of website interface and information on cell phone, laptop, and desktop computer • The navigational system and the time to complete the tasks • The accessibility of internet (WiFi) connection • The security and privacy of public internet (WiFi) connection • The availability of the system when needed	• View personal Information	....
	• View student records	....

REQUIREMENTS	
Functional	• The system shall display available courses to the student based on student's query criteria on courses • The system shall accept student's course registrations • The system shall validate based upon course availability • The system shall enable student to view her school records • The system shall display message if student's course registration does not succeed
Non-Functional	• The system shall provide uncomplicated navigational system that allows users to find available services options with ease and without spending too much time. • The system shall ensure that content does not clutter on a page.• The system shall display page layout and information appropriately for smaller screen size devices such as mobile phones and tablets • The system shall ensure that all links, menus, or buttons produce correct pages upon clicking. • The system shall ensure that all confidential information be encrypted to maintain security

MODEDLING		
Modeling Techniques	Concepts Representation	Models, Ontologies
• UML Class Diagram • UML Instance Diagram	• Ontology tool (Protégé-Frames)	• Persona Class Diagram, Persona Instance Diagram, Behavioral-GoalScenarioTask Class Diagram, Behavioral-GoalScenarioTask Instance Diagram, Requirements Class diagram, Requirements Instance Diagram • Persona Ontology, Behavioral-GoalScenarioTask Ontology, Requirements Ontology

Figure 24. Case Study – VPD for Primary Persona Linda Rose

Process 4 – Analysis and Evaluation

In this process, two activities were conducted two activities: Personas-Viewpoints-Requirements Analysis (PVRA) and Scenarios-Usability Goals Evaluation (SUGE) based on the profile and viewpoint of primary persona Linda Rose (i.e. PPD and PDD created in process 1, VPD created in process 2) when using the PatriotWeb website.

- **Personas-Viewpoints-Requirements Analysis (PVRA)**

  The Personas-Viewpoints-Requirements Matrix is used to analyze the PVRA activity. Two viewpoints of Linda Rose using the PatriotWeb website were analyzed:

  (1). In a public environment, Starbucks in mid-day during work day.

VPB1: ***<Linda Rose, Full-Time Undergraduate Student, Starbucks (mid-work day)>***

(2). In a private environment, home during evening time.

VPB2: ***<Linda Rose, Full-Time Undergraduate Student, Home (evening)>***

As mentioned in chapter 5 section 5.2.4, a meaningful weight is assigned to each persona according to the relative importance of the persona based on the proportion of the market the persona represents. Thus, for illustration purpose for the case study, a weight of 50 is assigned to Linda Rose. Samples of representative requirements are listed (REQ1 to REQ7) in the Personas-Viewpoints-Requirements Matrix. These sample requirements are based on information synthesized from the persona documents produced in the Personas Construction process (PPD and PDD) and Viewpoints Identification and Construction process (VPD) of the CDP model. For each viewpoint, each requirement is numerically rated based on the rating scale. The weight sum for each requirement is then calculated by multiplying the Linda Rose assigned weight of 50 by the requirement numeric rating. Figure 25 illustrates the analysis results.

Rating scale: 5 – Critical 4 – Very Important 3 – Important 2 – Somewhat important 1 – Unimportant		PERSONAS	Linda Rose, *the busy undergraduate student and web developer* (*primary persona*)		WEIGHTED SUM
		WEIGHTS	50		
		VIEWPOINTS (VPB)	<Linda Rose, Full-Time Undergraduate Student, Starbucks (mid-work day)>	<Linda Rose, Full-Time Undergraduate Student, Home (evening)>	
REQUIREMENTS	REQ1	The system shall provide uncomplicated navigational system that allows users to find available services options with ease and without spending too much time.	4	3	350
	REQ2	The system shall ensure that content does not clutter on a page.	4	3	350
	REQ3	The system shall ensure that all links, menus, or buttons produce correct pages upon clicking.	4	3	350
	REQ4	The system shall display message if student's course registration does not succeed.	3	3	300
	REQ5	The system shall display available courses to the student based on student's query criteria on courses.	4	3	350
	REQ6	The system shall display page layout and information appropriately for smaller screen size devices such as mobile phones and tablets.	4	2	300
	REQ7	The system shall ensure that all confidential information be encrypted to maintain security.	4	4	400

Figure 25. Case Study – Personas-Viewpoints-Requirements Matrix for PVRA Activity

Based on the analysis results from the Personas-Viewpoints-Requirements Matrix (Figure 25), it can be seen that requirement REQ7 (encryption of confidential information) has the highest weighted sum score of 400, followed by REQ1 (easy navigational system), REQ2 (uncluttered content layout), REQ3 (correct page result), and REQ5 (display of course availability) with same weighted sum score of 350 each. These are evident from information obtained in Linda Rose's PPD, PDD, and VPD created in process 1 and process 2. For examples: (i) "Security": since Linda Rose checks her school activities (including PatriotWeb) mostly in public places during lunch time at work where security may be vulnerable, especially when making payment using credit card, (ii) "Efficiency" in terms of time to complete the tasks (i.e.

slow/fast, less time/more time) and navigational system for quick access to available services, as Linda Rose wants to use the PatriotWeb to quickly complete her tasks and she does not want to figure how to use the PatriotWeb system, (iii) "Comprehensibility" in terms of easy-to-understand and uncluttered content representations, since Linda Rose wants a simple uncomplicated uncluttered system, (iv) "Accuracy" in terms of correct pages are displayed, as Linda Rose does not want to make a lot of mistakes and spend a lot time in using the PatriotWeb.

It can also be seen in Figure 25 that REQ6 (smaller screen sizes) is considered very important to Linda Rose when she is in a public place (i.e. VPB 1), since from Linda Rose's PPD, she uses laptop and smartphone to school and work, and thus a proper display of content on smaller screen size devices is very important. On the other hand, since Linda Rose's uses desktop computer at home (PPD), smaller screen size issue is thus not a big concern and is somewhat important to her.

The PVRA activity not only provides a summary of the importance of each requirement, but also highlights important but sometimes overlooked issues such as the importance of mobile design for mobile users and the significance of different environment settings. This is especially important to users who are constantly on the road or use the PatriotWeb website outdoor. Currently, there is no streamlined version of the PatriotWeb website available on mobile phones.

- **Scenarios-Usability Goals Evaluation (SUGE)**

  For the case study, five representative scenarios were selected to be evaluated using the Scenarios-Usability Goals Matrix, with the goal of improving the usability of the

PatriotWeb website by identifying any requirement issues with respect to the eight usability factors (chapter 5, section 5.2.4.) chosen for this research. Figure 26 illustrates the evaluation results in the Scenarios-Usability Goals Matrix. The five selected scenarios were:

- Scenario 1: View Student Services Page
- Scenario 2: View Registration Page
- Scenario 3: View Student Records Page
- Scenario 4: View Personal Information Page
- Scenario 5: Search for Classes

The five selected scenarios were reviewed in light of the chosen eight usability goals from the viewpoint of the primary persona Linda Rose using documents created in Personas Construction process (PPD and PDD) and Viewpoints Identification and Construction process (VPD) as inputs. For each cell in the matrix, any requirement issues and any improvements needed to resolve were recorded. The results of conducting the Scenarios-Usability Goals Matrix helped to refine requirements on the current PatriotWeb system.

From the results shown in the Scenarios-Usability Goals Matrix (Figure 26), it can be clearly seen that issues fell in the "**Efficiency**", "**Comprehensibility**", "**Learnability**", and "**Accuracy**" usability factors. These are evident from the Linda Rose's persona profile and definition documents created in process 1 (PPD and PDD, Figure 22 and Figure 23). These usability requirements issues have been highlighted in earlier sections of this chapter, i.e. the Personas Construction process (process 1),

Viewpoints Identification and Construction process (process 2), and Personas-Viewpoints-Requirements Matrix analysis.

Through walking-through each scenario against the eight usability factors with respect to the viewpoint of the primary persona Linda Rose, three specific requirements of the current PatriotWeb system in regard to the five scenarios were refined:

(1) "***The system shall enable users to return to the Courses page to continue to select other courses of the same subject.***"

Current PatriotWeb system brings users back to the page to select term when users click on the "New Search" button. This issue impacted the "Efficiency", "Accuracy", "Learnability", and "Comprehensibility" usability factors and were addressed in the Scenarios-Usability Goals Matrix (Figure 26).

(2) "***The system shall enable users to return to the Subjects page to continue to select other subjects of the same term.***"

Similar issue as in (1): current PatriotWeb system brings users back to the page to select term when users clicks on the "New Search" button. This issue impacted the "Efficiency", "Accuracy", "Learnability", and "Comprehensibility" usability factors and were addressed in the Scenarios-Usability Goals Matrix (Figure 26).

(3) "***The system shall ensure that most commonly used services links be visible and easily accessible.***"

On the current PatriotWeb system, two commonly used services on the

Registration page: “Register, Add or Drop Classes” and “Search for Classes” are placed along with other services links in the body of the page. Similarly, two commonly used services on the Student Records page: “View your Unofficial Transcript” and “Final Grades” are placed along with other services links in the body of the page. The visibility and location issue of commonly used services impacted the “Comprehensibility” usability factor and was addressed in the Scenarios-Usability Goals Matrix (Figure 26).

Requirements that were highlighted in process 1, process 2, and Personas-Viewpoints-Requirements Matrix analysis were further refined:

(4) “***The system shall provide main navigational system with drop-down menus that enable users with quick access to available services options.***”

Current PatriotWeb system requires users to click on a service link on the main navigational menu to display a page that reviews the available services options. The creation of the drop-down menus allows users to mouse over services links on the main navigational menu to quickly view available services options. The navigational system issue affected the “Efficiency” and “Learnability” usability factors and were addressed in the Scenarios-Usability Goals Matrix (Figure 26).

(5) “***The system shall ensure that similar services links be displaced as a single link.***”

This requirement was refined from the requirement related to uncluttered content layout which has been highlighted in process 1, process 2, and Personas-

Viewpoints-Requirements Matrix analysis. On the Personal Information page of the current PatriotWeb system, there are separate view and update links for a number of services on the page. This unnecessary cluttering of page issue impacted the "Efficiency" and "Comprehensibility" usability factors and were addressed in the Scenarios-Usability Goals Matrix (Figure 26).

Additional requirements that were highlighted in process 1, process 2, and Personas-Viewpoints-Requirements Matrix analysis are:

(6) "***The system shall ensure that all services links return correct page results.***"

The "New Search" button on the current PatriotWeb system brings users back to the page to select term when in fact users want to perform new search on the courses on the Courses page or new subjects on the Subjects page. This inaccurate page results impacted the "Accuracy" usability factor and was addressed in the Scenarios-Usability Goals Matrix (Figure 26).

(7) "***The system shall provide a mobile version for smaller screen devices.***"

Current PatriotWeb system is targeted for regular desktop or laptop users. It is beneficial to provide users who mostly spend time in outdoor environments a mobile version of the PatriotWeb system with streamlined services and appropriate presentation layout. This issue was not addressed in the Scenarios-Usability Goals Matrix (Figure 26) as the five selected scenarios in the matrix were reviewed based on regular desktop or laptop.

		USABILITY GOALS							
		Accuracy	Attractiveness	Efficiency	Learnability	Reliability	Comprehensibility	Clarity	Remembrability
SCENARIOS	SCEN 1: View Student Service Page			Extra click and page transition: needs to click on Student Services tab to go to next page, then clicks to select services, i.e. Registration, Student Records, Student Account – *it is better to have an effective main navigational system, e.g. a drop down menu for each menu tab.*	Navigational system not properly designed: needs to click on main menu tab in order to view other services for that menu tab – *it is better to incorporate a drop-down menu for each menu tab to allow viewing of available services for each menu tab.*		On the Main Menu page, services links on body content are a bit messy, order not in sequence with the order of the menu tabs – *it is better to reorder the services links on body content to be in same order as the menu tabs.*		
	SCEN 2: View Registration Page			(1) Same issue as in SCEN 1			(1) Services links on body content are in random order – *it is better to re-arrange the services links on body content, group relevant services links in same group.* (2) Visibility and location of commonly used links issue: two commonly used functions: "Search for Classes" and "Register, Add or Drop Classes" are tucked in the body content with other links – *it is better to be visible and easily accessible, e.g. big buttons.* (3) "Select Term" link on Registration page is redundant: needs to select the term again when search for classes, etc) – *it is better to remove the "Select Term" link.*		
	SCEN 3: View Student Records Page			(1) Same issue as in SCEN 1			(1) Services links on body content are in random order – *it is better to re-arrange the services links on body content, group relevant services links in same group.* (2) Visibility and location of commonly used links issue: two commonly used functions: "View Official Transcript" and "View Final Grades" are tucked in the body content with other links – *it is better to be visible and easily accessible, e.g. big buttons.*		
	SCEN 4: View Personal Information Page			(1) Same issue as in SCEN 1 (2) The unnecessary individual view and update services links results in unnecessary clicks and back and forth pages issues (affect task time) – *it is better to have one single link to view/update information on a single page.*			Cluttered content: too many unnecessary links to view and update personal information which cluttered the page. Needs to click the view addresses link to view addresses, return to Personal Information page and selects other services to view/update persona information – *it is better to combine all these individual view and update links into one link and view/update all in one same page.*		
	SCEN 5: Search for Classes	"New Search" button not accurately returns page that reflects what would be expected, i.e. return to previous page for new course search, rather than returning to the beginning for a brand new term search – *it is better to have another button to return to previous courses page and short instruction on top of the page body.*		(1) Same issue as in SCEN 1 (2) Repeated task: "New Search" button returns to the "Select Term" page. Needs to repeat the process again to select the term again – *it is better to have a back to previous page button.* (3) Advanced search field options are displayed in a separate page (i.e. extra page transition) – *it is better to display the field options on same page.*	(1) Same issue as in SCEN 1-4 navigational system issue. (2) "New Search" button issue: needs to know and learn that clicking the new search button will start from beginning, i.e. select new term. It does not mean to search new courses or subjects (of the same term) – *it is better to have short instruction highlighting what the 'New Search" button does.*		"New Search" button can be misleading and not understood correctly – this button original intention is to do a brand new search, starting at the Term selection. When searching for new courses of the same term and/or same subject, this button can thus be inconvenient – *it is better to have another button to return to previous course search page, rather than repeating the process starting at the Term selection. Also, short instruction on top of the page body to indicate what the buttons do.*		

Figure 26. Case Study – Scenarios-Usability Goals Matrix for SUGE Activity

6.3 OntoPersonaURM Model

The OntoPersonaURM model is developed in process 3 (Concept Modeling) of the CDP model to provide insights and help guide ontology engineers and developers into the construction of ontologies for explicit specifications of the concept of persona in representing users' characteristics, and the concepts of viewpoint, goal, scenario, task, and requirement. The OntoPersonaURM model is composed of three generic interrelated domain independent ontologies: Persona Ontology, Behavioral-GST (Goal-Scenario-Task) Ontology, and Requirements Ontology. The development team followed the five-step ontology development process (chapter 5 Table 3) developed in the OntoPersonaURM model to systematically construct the three ontologies for the case study:

Step 1: Synthesize Information Collected

This step is related to process 1 and process 2 of the CDP model, since the idea of the ontology development process is intended for standalone guide. The deliverables in this step are the Persona Profile Document (PPD), Persona Definition Document (PDD), and Viewpoint Document (VPD) which are described in previous section (section 6.2: CDP Model).

Step 2: Consult Existing Ontologies

In constructing the ontologies for the case study, there appeared to be no existing ontologies that are similar and are readily available for reusable purpose. However, we

have consulted some existing ontologies [GOLEMATI 07, KATIFORI 08, SHIBAOKA 07] on the approaches in specifying some of the class properties for the Persona class, Interest class, Goal class, and the Requirement class.

Step 3: Define Classes and Properties

The relevant classes, class hierarchy, and class properties of the three ontologies: Persona Ontology, Behavioral-GST Ontology, and Requirements Ontology are defined in a top-down approach and are represented pictorially in UML diagramming notations and specified ontologically in Protégé-Frames ontology editing environment. Descriptions of the classes and properties, as well as the UML class diagrams and the Protégé-Frames class specifications are presented in chapter 5 and also included in Appendix C.

Step 4: Create Instances

After the classes are defined and specified in Protégé-Frames (step 3), instances of the classes are created in Protégé-Frames editing environment. A representative name for an instance of a class is chosen and displayed in the "Instance Browser". The instance name may be selected from any one or combination of the values of the properties (or slots in Protégé-Frames). Values for the properties are filled in in the "Instance Editor" of the Protégé-Frames tool. One or more instances may be created for a class. In our case study, for example, we created an instance for the Persona class and selected the value of the attribute persona_title ("*Linda Rose, The Busy Undergraduate Student & Web*

Developer") as the representative instance name displayed in the "Instance Browser". An alternative for the instance name could be the id attribute.

Figure 27 and Figure 28 present the UML instance diagrams for Persona Ontology and Requirements Ontology respectively. Due to the number of classes and their instances for the Behavioral-GST Ontology, the UML instance diagram is omitted. Figure 29 illustrates the Protégé-Frames view of the instances of a representative class, Persona class of the Persona Ontology, Figure 30 illustrates the Protégé-Frames view (partial view) of the instances of a representative class, Goal class of the Behavioral-GST Ontology. And Figure 31 illustrates the Protégé-Frames of the instances of a representative class, Requirement class of the Requirements Ontology.

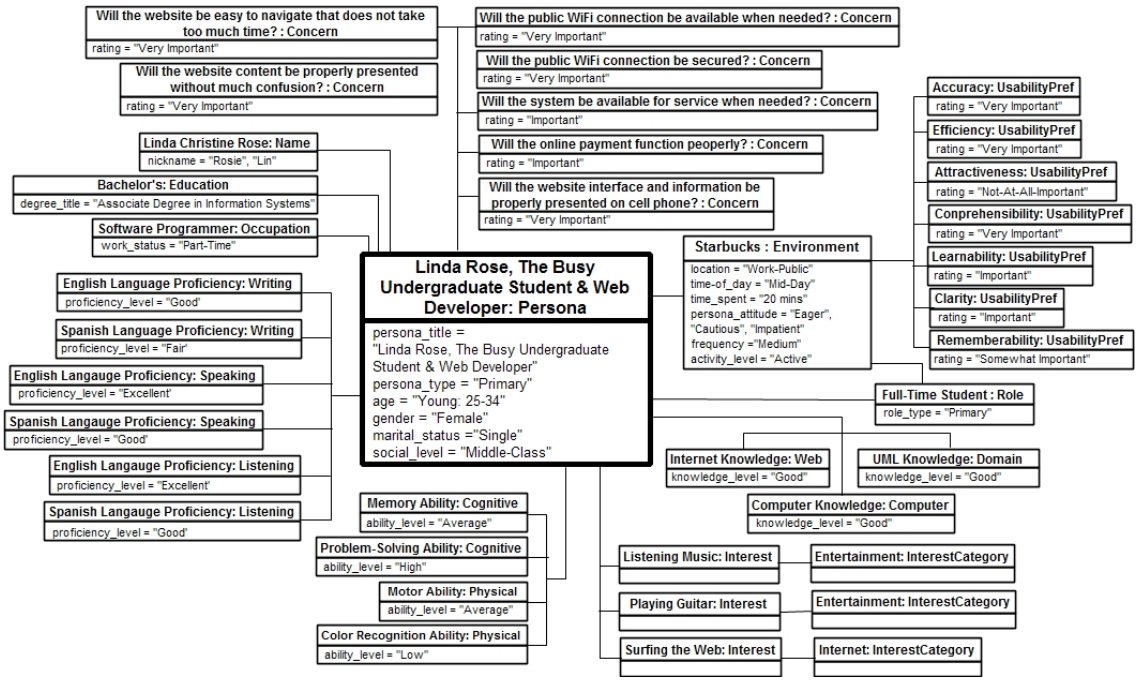


Figure 27. Case Study – UML Instance Diagram for Persona Ontology

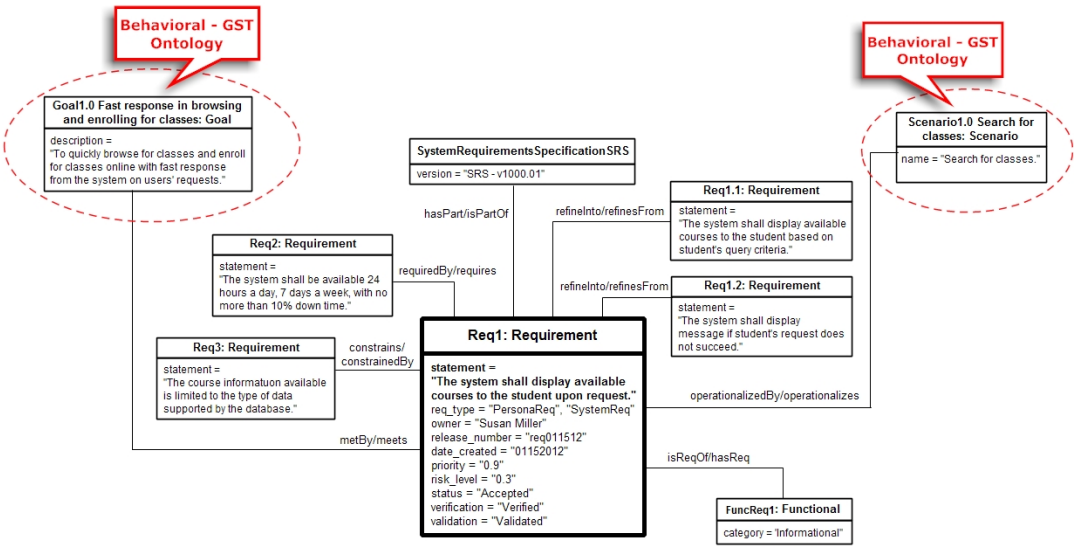


Figure 28. Case Study – UML Instance Diagram for Requirements Ontology

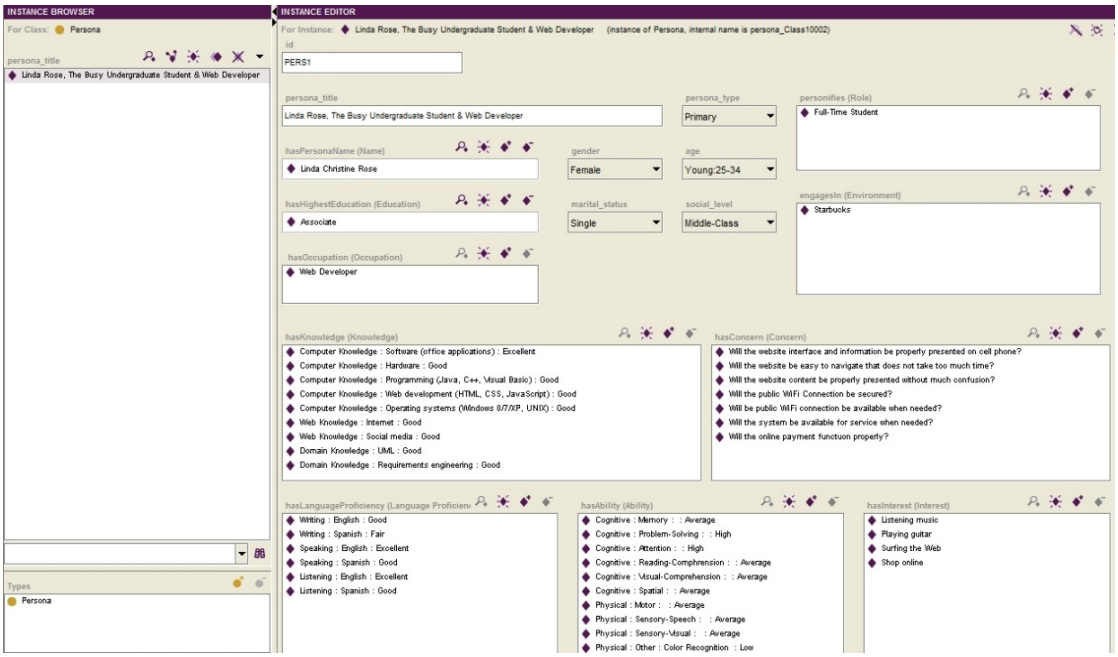


Figure 29. Case Study – Protégé-Frames View of the Instances of Persona Class (Persona Ontology)

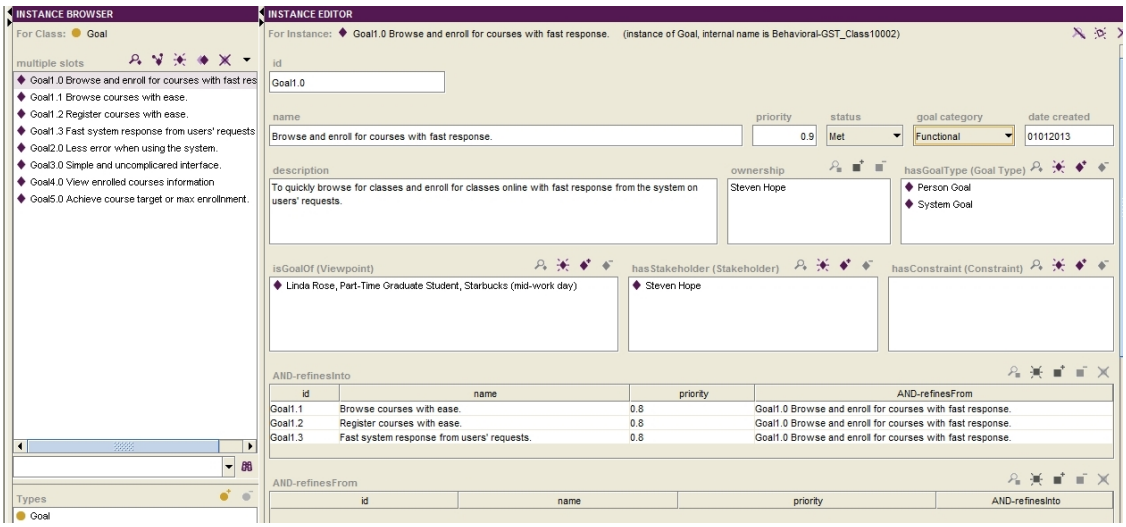


Figure 30. Case Study – Protégé-Frames View (Partial View) of the Instances of Goal Class (Behavioral-GST Ontology)

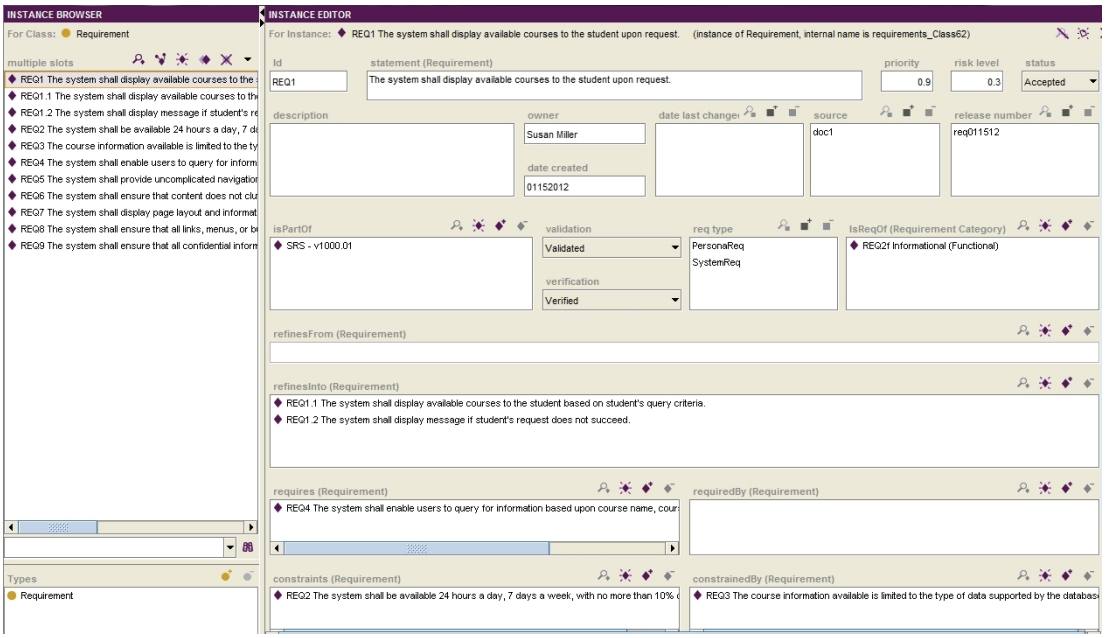


Figure 31. Case Study – Protégé-Frames View (Partial View) of the Instances of Requirement Class (Requirements Ontology)

Step 5: Combine Ontologies

The final step in our ontology development process is to combine related ontologies to help to better understand the relationships of classes between ontologies and make necessary corrections to refine the ontologies. If there exist one or more relationships between classes of two ontologies, then the ontologies are combined by including the related ontology into the current ontology via the "Manage Included Projects" selection of the "Project" menu in the Protégé-Frames ontology editing tool (Figure 32, Figure 33).

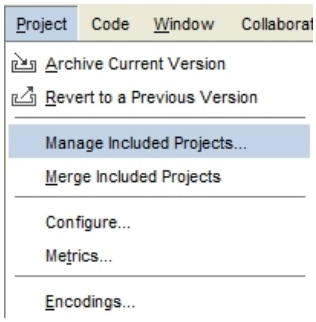


Figure 32. Case Study – Protégé-Frames' Project Menu

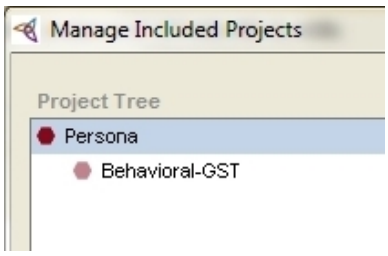


Figure 33. Case Study – Protégé-Frames' Manage Included Projects

The included classes and properties are displayed in Protégé-Frames as pale icons to distinguish from the classes in the current ontology. An included ontology may also be merged with the current ontology to form a single ontology via the "Merge Included Projects" and all classes and properties of the merged ontologies are then displayed as solid icons.

For the case study, the ontologies are combined (i.e. included) via the "Manage Included Projects" feature of the Protégé-Frames ontology editing tool. The approaches in combining the ontologies in this dissertation are to be noted as follows:

(1). No duplicate classes are created for one or more ontologies in the Protégé-Frames tool. For example, the Viewpoint class is designed to be a class in the Behavioral-GST Ontology, since the Viewpoint class relates with several classes of the Behavioral-GST Ontology. Although there are four classes in the Persona Ontology (Persona class, Environment class, Role class, and Concern class) that are related to the Viewpoint class of the Behavioral-GST Ontology (Figure 34), the Viewpoint class is not created (as a duplicate) in the Persona Ontology.

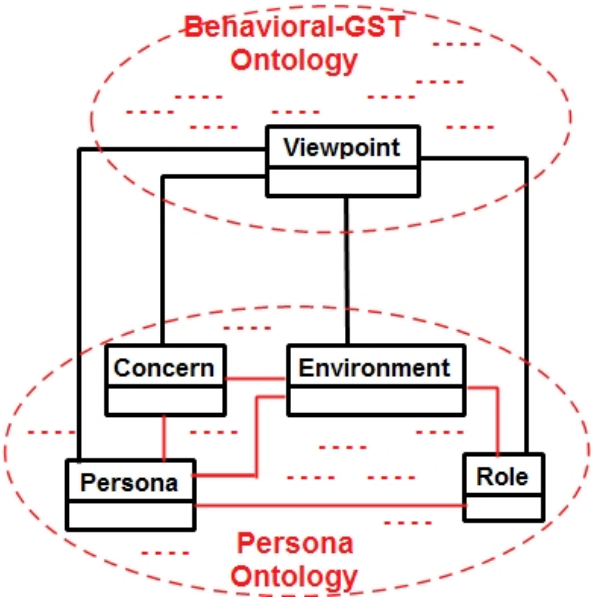


Figure 34. Case Study – Persona Ontology and Behavioral-GST Ontology Class Relationship

(2). If a class in one ontology is related to another class in another ontology, the relationship property of these two classes is specified in the Protégé-Frames tool ***after*** the two ontologies are combined (included) or merged (via "Merge Included Project" option) in the Protégé-Frames tool. For example, the Persona class (in Persona Ontology) is related to the Viewpoint class (in Behavioral-GST Ontology) via the isPersonaOf/hasPersona relationships (Figure 35) The isPersonaOf/hasPersona relationships are specified in the combined (included) or merged ontologies in the Protégé-Frames ***after*** both ontologies are combined (included) or merged (via "Merge Included Project" option).

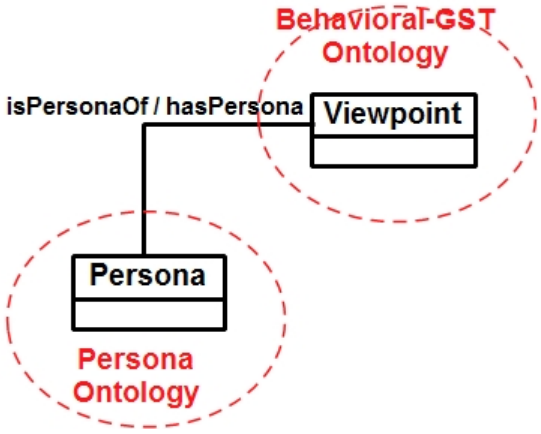


Figure 35. Case Study – Persona Class and Viewpoint Class isPersonaOf/hasPersona Relationships

(3). As the Persona Ontology (current ontology) contains classes that have established relationships with classes of the Behavioral-GST Ontology, the Behavioral-GST Ontology is included in (or combined with) the Persona Ontology. In a similar fashion, for the Behavioral-GST Ontology as a current ontology, as there are classes that are related with the Persona Ontology and the Requirements Ontology, the Persona Ontology and Requirements Ontology are included in (or combined with) the Behavioral-GST Ontology. For the Requirements Ontology as a current ontology, the Behavioral-GST Ontology is included in (or combined with) the Requirements Ontology. Figure 36, Figure 37, and Figure 38 are snapshots of the combined ontologies for Persona Ontology, Behavioral-GST Ontology, and Requirements Ontology respectively.

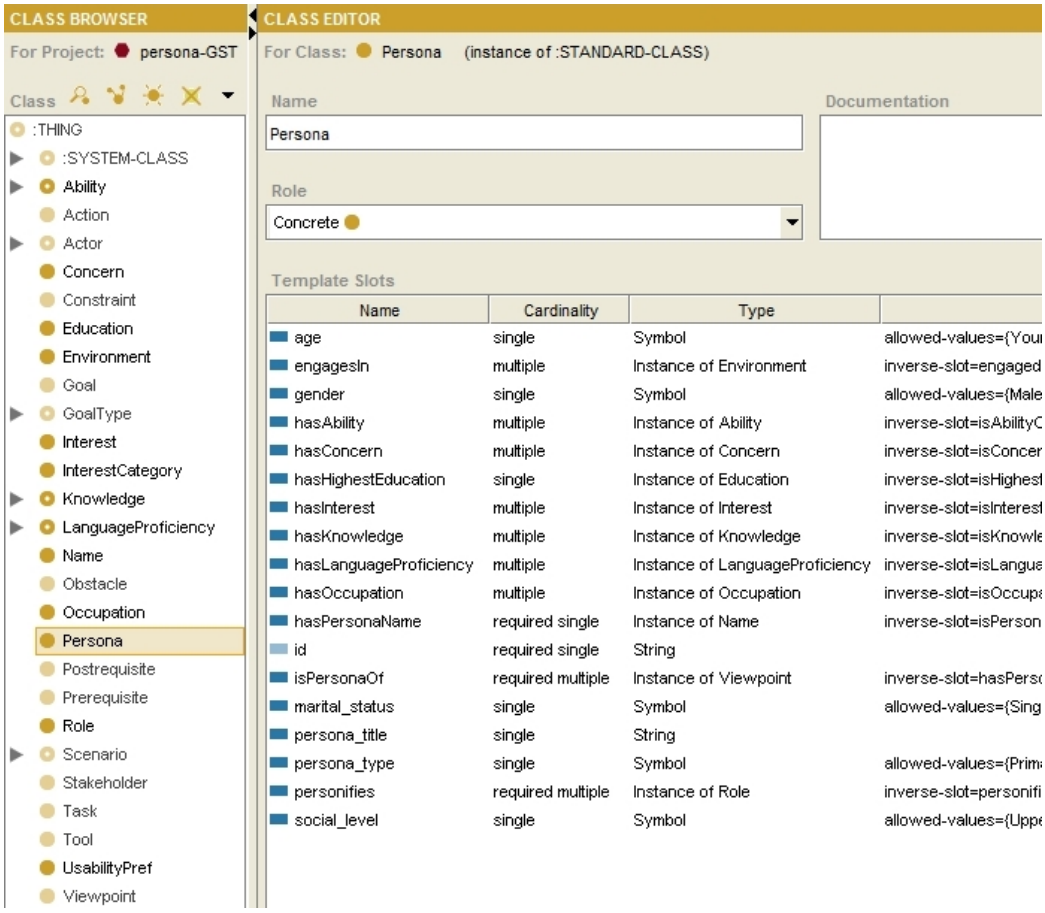


Figure 36. Case Study – Combined Persona Ontology (Current) and Behavioral-GST Ontology (Included) in Protégé-Frames

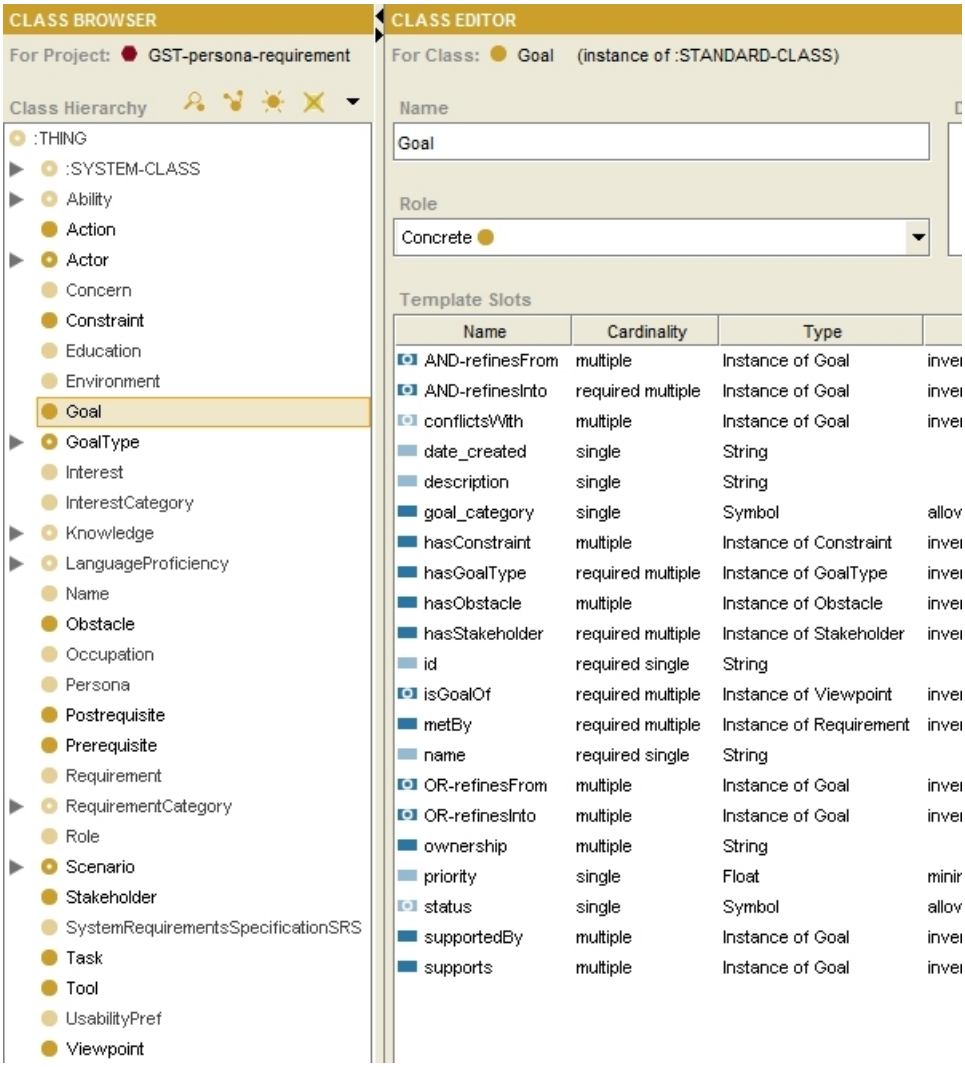


Figure 37. Case Study – Combined Behavioral-GST Ontology (Current), Persona Ontology (Included), and Requirements Ontology (Included) in Protégé-Frames

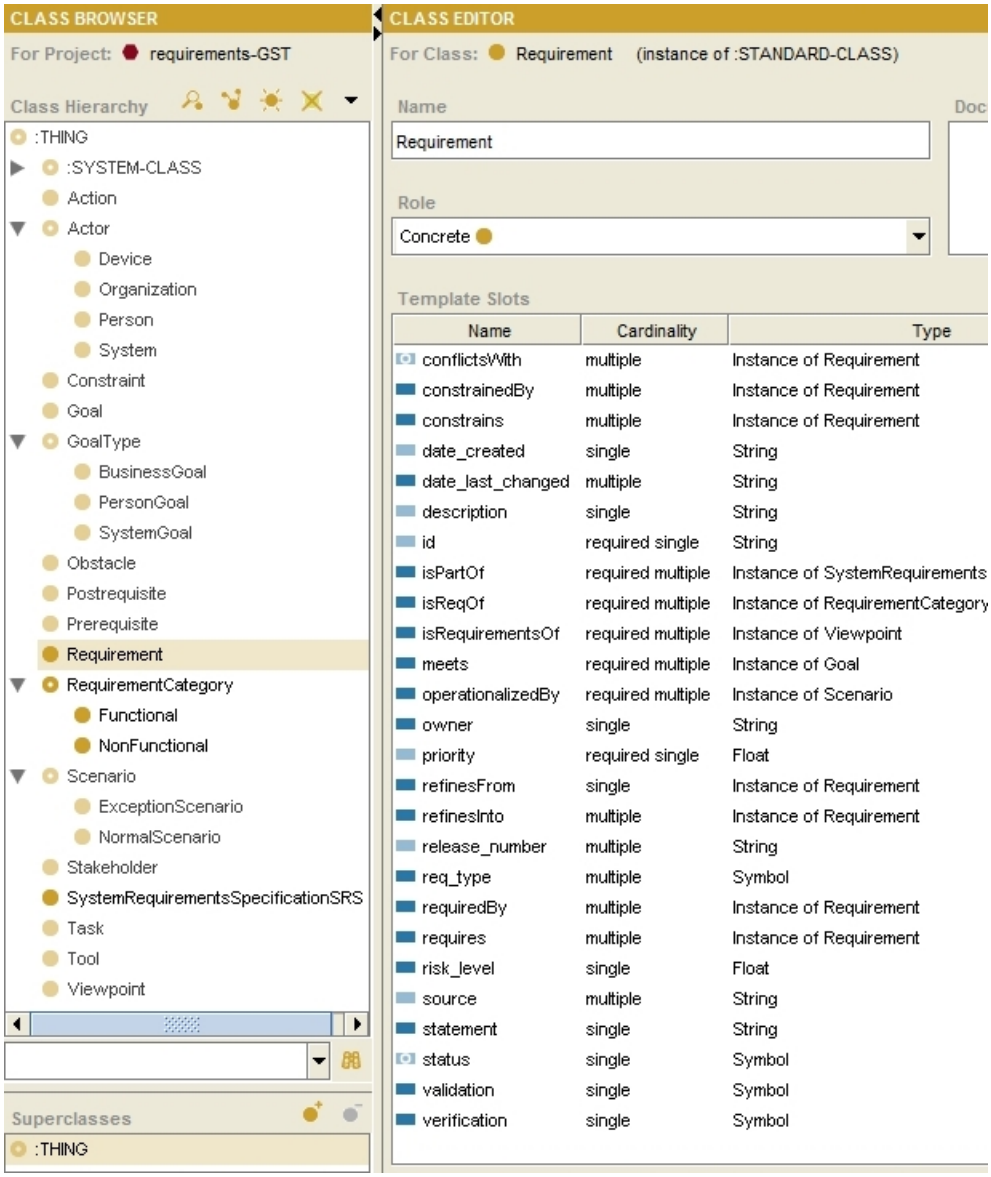


Figure 38. Case Study – Combined Requirements Ontology (Current) and Behavioral-GST Ontology (Included) in Protégé-Frames

CHAPTER 7: PROTOTYPE PATRIOTWEB WEBSITE

7.1 Introduction

As mentioned in chapter 6 section 6.2 “Scenarios-Usability Goals Matrix Evaluation (SUGE)”, five representative scenarios were selected and evaluated using the Scenarios-Usability Goals Matrix. The goal was to improve the usability of the current PatriotWeb system by identifying any requirement issues with respect to the eight usability factors chosen for this research (chapter 5 section 5.2.4) from the viewpoint of the primary persona Linda Rose based on the PPD, PDD, and VDP (chapter 6 section 6.2) created in process 1 and process 2 of the CDP mode, The identification of the requirement issues helped to refine requirements on the current PatriotWeb system. The five selected scenarios were:

- Scenario 1: View Student Services Page
- Scenario 2: View Registration Page
- Scenario 3: View Student Records Page
- Scenario 4: View Personal Information Page
- Scenario 5: Search for Classes

The steps of performing the five selected scenarios are provided in Appendix G. These five selected scenarios also served as the base scenarios for the research evaluation in the effectiveness of applying persona concept to validate research hypotheses H1 and

H2 (chapter 8 section 8.2). The five scenarios were reviewed with respect to the chosen eight usability goals from the viewpoint of the primary persona Linda Rose. The Scenarios-Usability Goals Matrix presented in Figure 26 of chapter 6 section 6.2 is reproduced here.

From the Scenarios-Usability Goals Matrix (Figure 26), it can be seen that most issues fell into the four usability factors: "**Efficiency**", "**Comprehensibility**", "**Learnability**", and "**Accuracy**". This is evident from the Linda Rose's persona profile and definition documents created in process 1 of the CDP model (Figure 22 and Figure 23, chapter 6 section 6.2). An excerpt from the Linda Rose's Personal Profile Document (PPD) is reproduced below with underlined words/sentences to highlight areas to focus on:

> "*Linda exemplifies a real-world person who wants to go online, quickly browse for classes that interest her, enroll for classes, and make online credit card tuition payment. Linda usually check her school activities mostly during lunch time at work. She is busy in her work and study and so doesn't like to spend a lot of time figuring out how to use the system, and when she needs to use the system the next time, she doesn't want to restart learning how to use the system. Linda does not care much about how attractive the website is. She represents a user group that wants simple uncomplicated system so that she doesn't make a lot of mistakes while using it, as well as a pleasing, uncluttered, and professional looking system*."

		USABILITY GOALS							
		Accuracy	Attractive-ness	Efficiency	Learnability	Reliability	Comprehensibility	Clarity	Remember-ability
SCENARIOS	SCEN 1: View Student Service Page			Extra click and page transition: needs to click on Student Services tab to go to next page, then clicks to select services, i.e. Registration, Student Records, Student Account – *it is better to have an effective main navigational system, e.g. a drop down menu for each menu tab.*	Navigational system not properly designed: needs to click on main menu tab in order to view other services for that menu tab – *it is better to incorporate a drop-down menu for each menu tab to allow viewing of available services for each menu tab.*		On the Main Menu page, services links on body content are a bit messy, order not in sequence with the order of the menu tabs – *it is better to reorder the services links on body content to be in same order as the menu tabs.*		
	SCEN 2: View Registration Page			(1) Same issue as in SCEN 1			(1) Services links on body content are in random order – *it is better to re-arrange the services links on body content, group relevant services links in same group.* (2) Visibility and location of commonly used links issue: two commonly used functions: "Search for Classes" and "Register, Add or Drop Classes" are tucked in the body content with other links – *it is better to be visible and easily accessible, e.g. big buttons.* (3) "Select Term" link on Registration page is redundant: needs to select the term again when search for classes, etc) – *it is better to remove the "Select Term" link.*		
	SCEN 3: View Student Records Page			(1) Same issue as in SCEN 1			(1) Services links on body content are in random order – *it is better to re-arrange the services links on body content, group relevant services links in same group.* (2) Visibility and location of commonly used links issue: two commonly used functions: "View Official Transcript" and "View Final Grades" are tucked in the body content with other links – *it is better to be visible and easily accessible, e.g. big buttons.*		
	SCEN 4: View Personal Information Page			(1) Same issue as in SCEN 1 (2) The unnecessary individual view and update services links results in unnecessary clicks and back and forth pages issues (affect task time) – *it is better to have one single link to view/update information on a single page.*			Cluttered content: too many unnecessary links to view and update personal information which cluttered the page. Needs to click the view addresses link to view addresses, return to Personal Information page and selects other services to view/update persona information – *it is better to combine all these individual view and update links into one link and view/update all in one same page.*		
	SCEN 5: Search for Classes	"New Search" button not accurately returns page that reflects what would be expected, i.e. return to previous page for new course search, rather than returning to the beginning for a brand new term search – *it is better to have another button to return to previous courses page and short instruction on top of the page body.*		(1) Same issue as in SCEN 1 (2) Repeated task: "New Search" button returns to the "Select Term" page. Needs to repeat the process again to select the term again – *it is better to have a back to previous page button.* (3) Advanced search field options are displayed in a separate page (i.e. extra page transition) – *it is better to display the field options on same page.*	(1) Same issue as in SCEN 1-4 navigational system issue. (2) "New Search" button issue: needs to know and learn that clicking the new search button will start from beginning, i.e. select new term. It does not mean to search new courses or subjects (of the same term) – *it is better to have short instruction highlighting what the 'New Search" button does.*		"New Search" button can be misleading and not understood correctly – this button original intention is to do a brand new search, starting at the Term selection. When searching for new courses of the same term and/or same subject, this button can thus be inconvenient – *it is better to have another button to return to previous course search page, rather than repeating the process starting at the Term selection. Also, short instruction on top of the page body to indicate what the buttons do.*		

Figure 26. Case Study – Scenarios-Usability Goals Matrix for SUGE Activity

7.2 Prototype PatriotWeb Design

The identified issues and suggested improvements highlighted in the Scenarios-Usability Goals Matrix (Figure 26) served to help improve the usability of the current PatriotWeb website. As a result, a prototype of the current PatriotWeb system was designed and built that included new features and improvements for the selected five scenarios. The prototype PatriotWeb website can be accessed at [PATRIOTWEB b]. Screenshots for the new features and improvements are included in Appendix E. The new features and improvements implemented in the prototype PatriotWeb system are described below:

(1). Improved navigational system

A new drop-down menu for each main menu tab was designed and built. There are five main menu tab on the current and prototype PatriotWeb system: "Personal Information", "Student Services", "Financial Aid", "Faculty and Advisor Services", and "Employee Services". For the prototype PatriotWeb system, by mousing over (instead of clicking) each main menu tab, the drop-down menu allowed Linda Rose to quickly see what services are available for each main menu tab. On the current PatriotWeb website, Linda Rose needs to click on a main menu tab to display the available services information on a new page. This new improved navigational system on the Prototype PatriotWeb system enabled Linda Rose to quickly find available services with ease by eliminating additional transition of page and thus helped to resolve and improve the "**Efficiency**" and "**Learnability**" usability issues. A representative screenshot for the drop-down menu for the "Student Services" menu tab is shown in Figure 39.

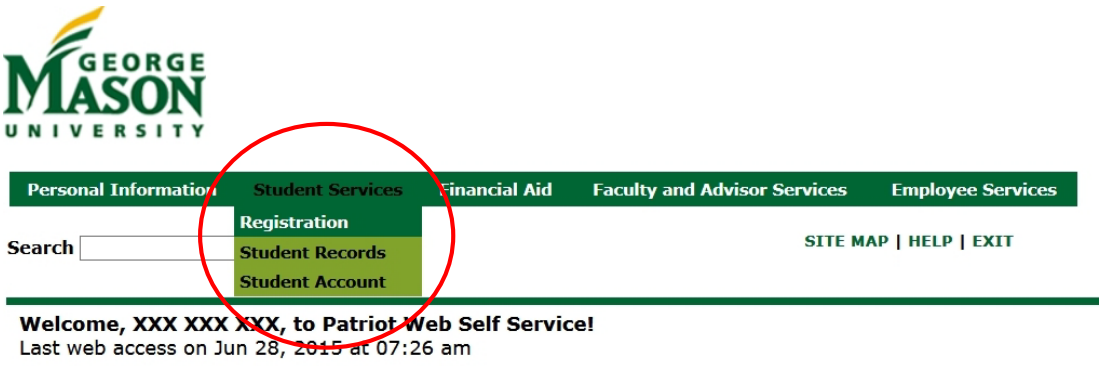


Figure 39. Case Study – Prototype PatriotWeb Website – New Improved Navigational System

(2). Improved visibility and location of commonly used services links

Links for the commonly used services were re-designed with big buttons and placed directly above other services links:

- On the **Registration** page of the Student Services tab, two highly visible buttons were designed and implemented for the two commonly used services: "Register, Add or Drop Classes" and "Search for Classes". These two services buttons were placed above other services links on the body content.
- On the **Student Records** page of the Student Services tab, two big buttons were designed and implemented for the two commonly used services: "View Unofficial Transcript" and "View Final Grades". The two services buttons were placed above other services on the body content.

Figure 40 and Figure 41 present the new improved design of the buttons for the Registration page and Student Records page respectively. Current PatriotWeb system

placed these commonly used links along with other services links in the body of the page. The high visibility of the two commonly used services links enabled Linda Rose to quickly find the two services links with ease and thus avoid making mistakes by clicking on other services links. The improved visibility of the services buttons on the prototype PatriotWeb system helped to resolve the "**Comprehensibility**" usability issue and improve "**Effficiency**" usability factor.

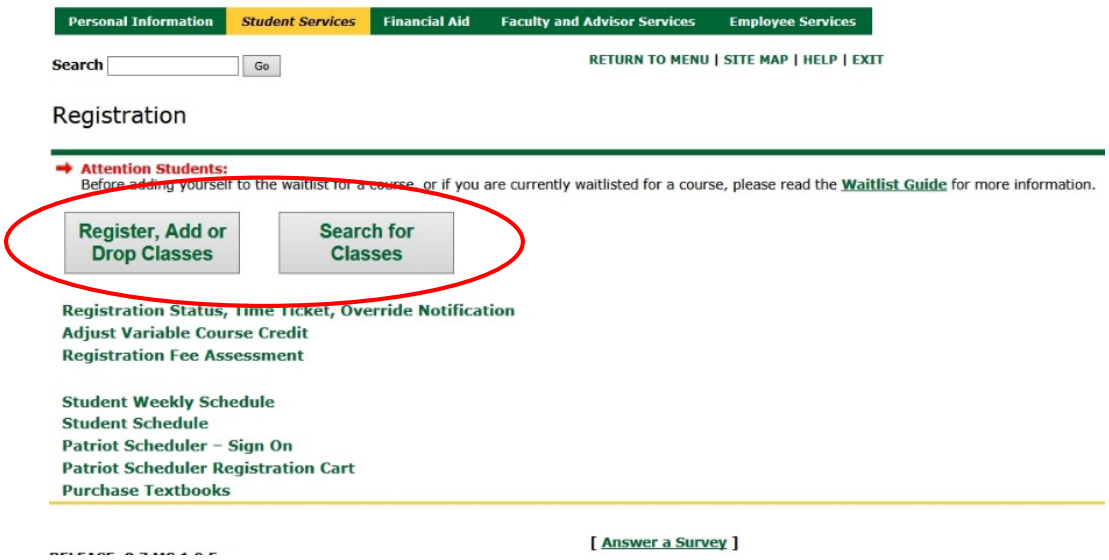


Figure 40. Case Study – Prototype PatriotWeb Website – New Improved Buttons (Registration Page)

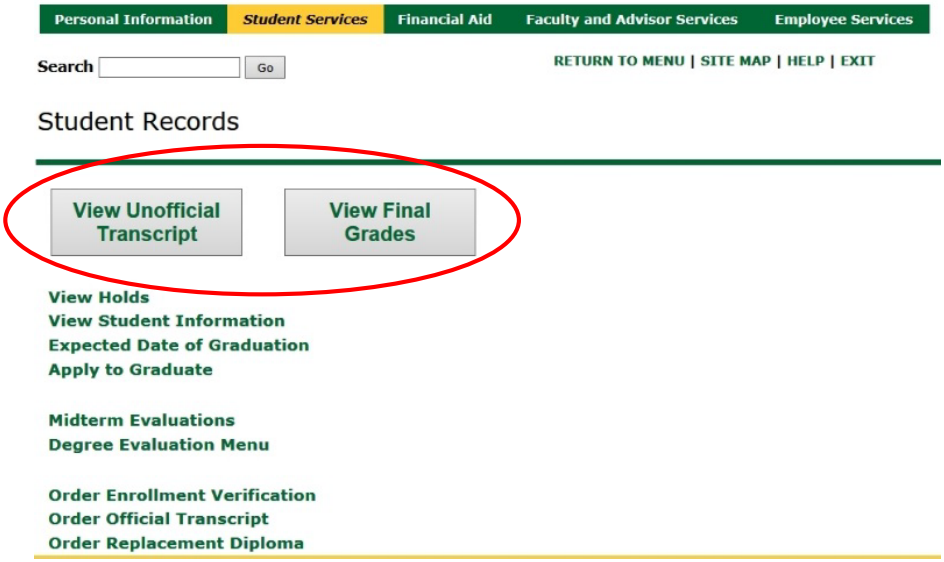


Figure 41. Case Study – Prototype PatriotWeb Website – New Improved Buttons (Student Records Page)

(3). Improved body content layout arrangement

The five primary services links on the body of the **Main Menu** page were re-ordered to reflect the same sequence as the main menu tabs. Similarly, the services links on the **Registration** page and the **Student Records** were re-ordered that grouped similar services together. Current PatriotWeb system shows these services links in a random order which can be inconvenient for Linda Rose who wants to quickly use the system to perform some tasks. Figure 42, Figure 43, and Figure 44 present the new layout of the services links in the body content for the Main Menu page, Registration page, and Student Services page respectively. The improved visibility and layout of the services

links on the prototype PatriotWeb system helped to reolve the "**Comprehensibility**" usability issue and improve the "**Efficiency**" usability factor.

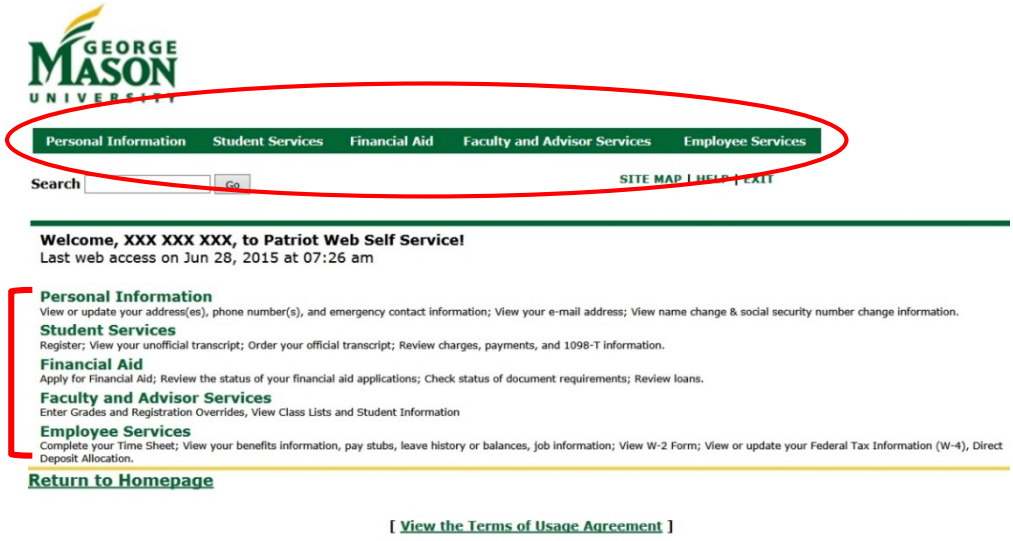


Figure 42. Case Study – Prototype PatriotWeb Website – New Improved Visibility and Layout of Services Links (Main Menu Page)

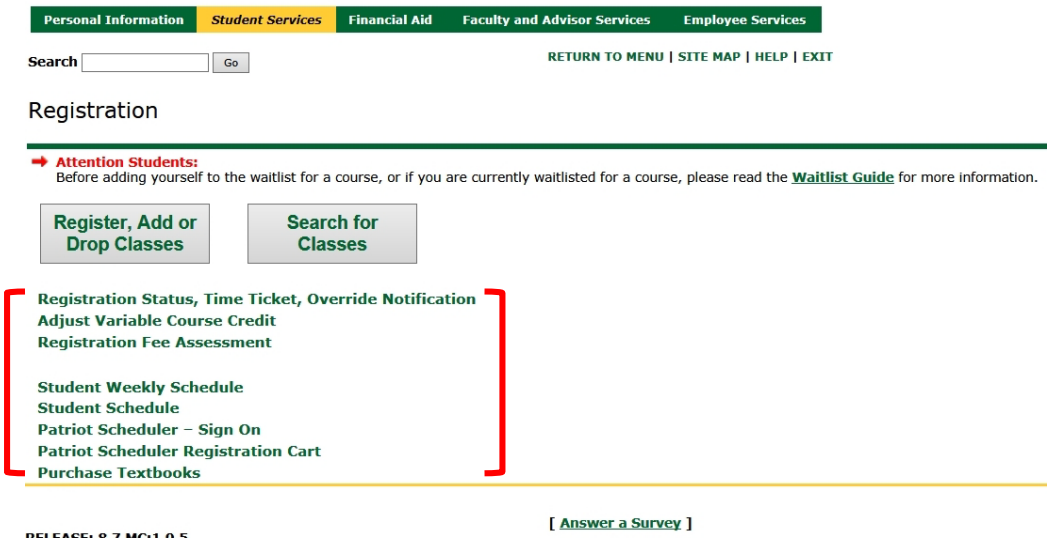


Figure 43. Case Study – Prototype PatriotWeb Website – New Improved Visibility and Layout of Services Links (Registration Page)

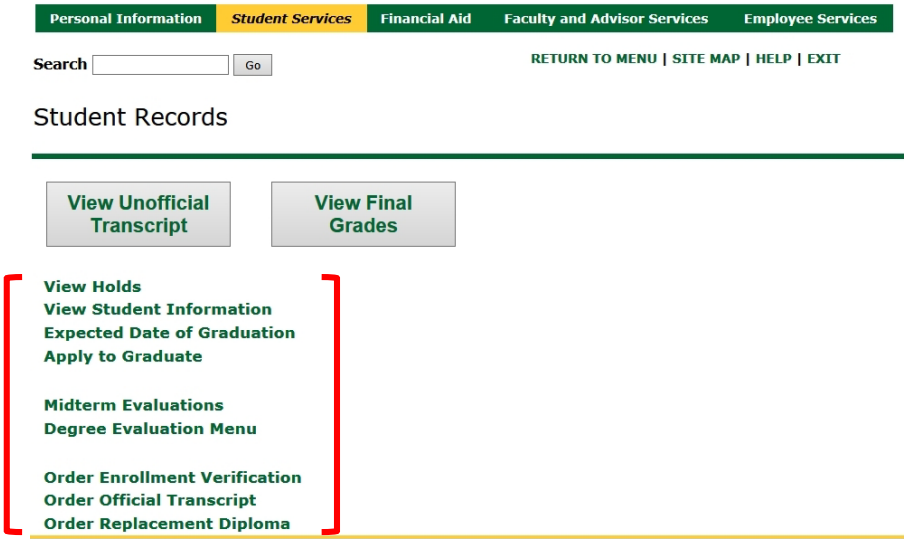


Figure 44. Case Study – Prototype PatriotWeb Website – New Improved Visibility and Layout of Service Links (Student Records Page)

(4). Elimination of unnecessary links

(a). Several unnecessary services links on the Personal Information page were reduced to just one link, i.e. "**View and Update Personal Information**" link was created on the drop-down menu on the Personal Information tab and in the body of the page to replace the following separate links on the Personal Information page on the current PatriotWeb system for viewing and updating personal information:

- View Address(es) and Phone(s)
- Update Address(es) and Phone(s)
- View E-mail Address(es)
- View Emergency Contacts
- Update Emergency Contacts
- View Ethnicity and Race
- Update Ethnicity and Race

Instead of clicking on each of the above service link back and forth on current PatriotWeb system, all these services links can now be viewed and updated in one page by clicking on just a single link on the prototype PatriotWeb system, i.e. the "View and Update Personal Information" link. The elimination of unnecessary view and update services links (and transition of pages) with just one link (and one page) allowed Linda Rose to view and/or update personal information on one webpage with just one click. The improved PatriotWeb system thus helped to resolve the "**Comprehensibility**" usability issue and improve the "**Efficiency**" usability factor.

Figure 45 presents the new "View and Update Personal Information" link on the prototype PatriotWeb system.

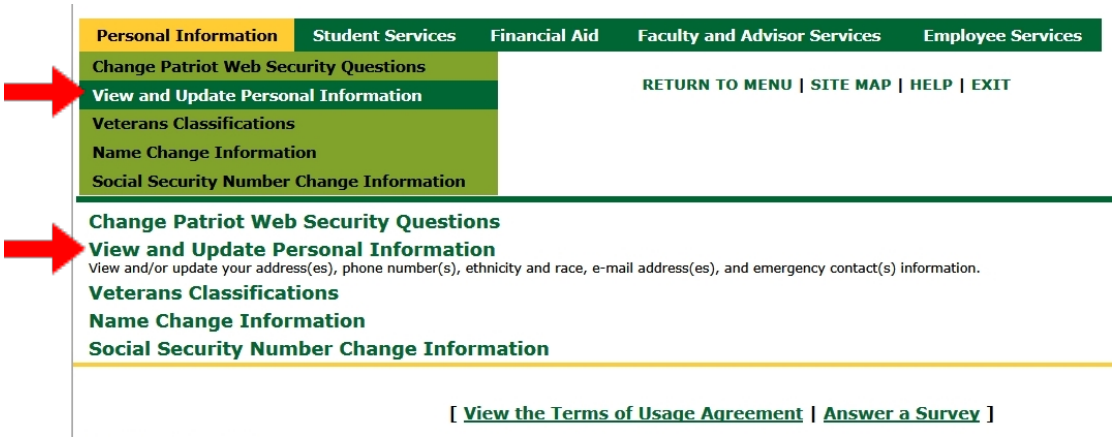


Figure 45. Case Study – Prototype PatriotWeb Website – New "View and Update Personal Information" Link (Personal Information Page)

(b). The "Select Term" link on the Registration page of the Student Services tab of the current PatriotWeb system was not necessary and was eliminated because Linda Rose was asked to select the term whenever she wants to search for classes ("Search for Classes" link) or register for classes ("Register, Add or Drop Classes" link). The removal of the "Select Term" on the Registration page of the prototype PatriotWeb system helped to resolve the "**Comprehensibility**" usability issue and improve the "**Efficiency**" usability factor. Figure 46 shows the "Select Term" link of the current PatriotWeb system.

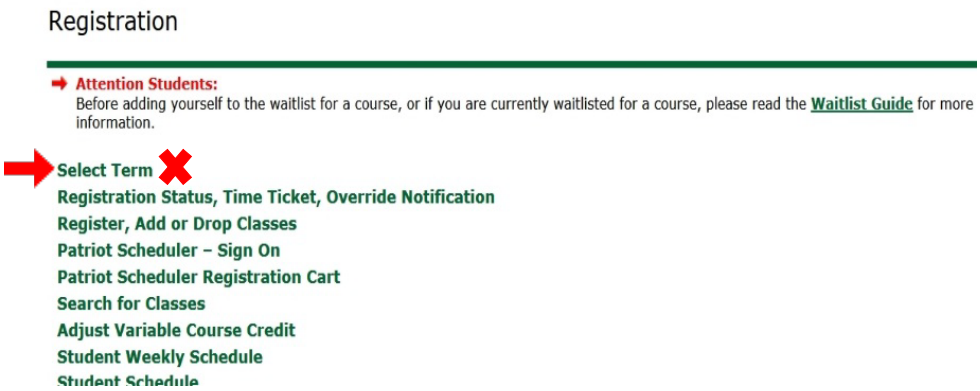


Figure 46. Case Study – Prototype PatriotWeb Website – "Select Term" Link

(5). Improved "New Search" button

The "New Search" button on the pages of the Search for Classes link of the current PatriotWeb system was not understood correctly by Linda Rose and resulted in pages that were not expected from Linda Rose. On the current PatriotWeb system, upon clicking on the "New Search" button, the page returned to the beginning page to select the term, then select the subjects and so forth, rather than returning to the courses page to select new courses (of the same subject) when Linda Rose is viewing the section(s) of that particular course or to the subjects page to select new subject (of the same term) when Linda Rose is viewing all the available courses.

For example, on the current PatriotWeb system, when Linda Rose wants to view another section(s) of the same course (of the same subject and term), the "New Search" button brought Linda Rose to the beginning to select the term again. Linda Rose has to select the same term and select the same subject in order to view other

section(s) of the courses for that particular subject. This repeated process is unnecessary and can be improved by introducing new buttons to allow Linda Rose to return to the "previous" page, i.e. the courses page or the subjects page. Hence, two new buttons "**Return to Courses Page**" and "**Return to Subjects Page**" were designed and created on the prototype PatriotWeb system.

In addition to the creation of the new buttons, short instructions were added on the top of the page to highlight what these buttons do. The introduction of the new buttons and short instructions helped to resolve and improve the "**Efficiency**", "**Comprehensibility**", "**Learnability**", and "**Accuracy**" usability issues. Figure 47 and Figure 48 demonstrate the introduction of new "Return to Courses Page" and "Return to Subjects Page" buttons and instructions respectively

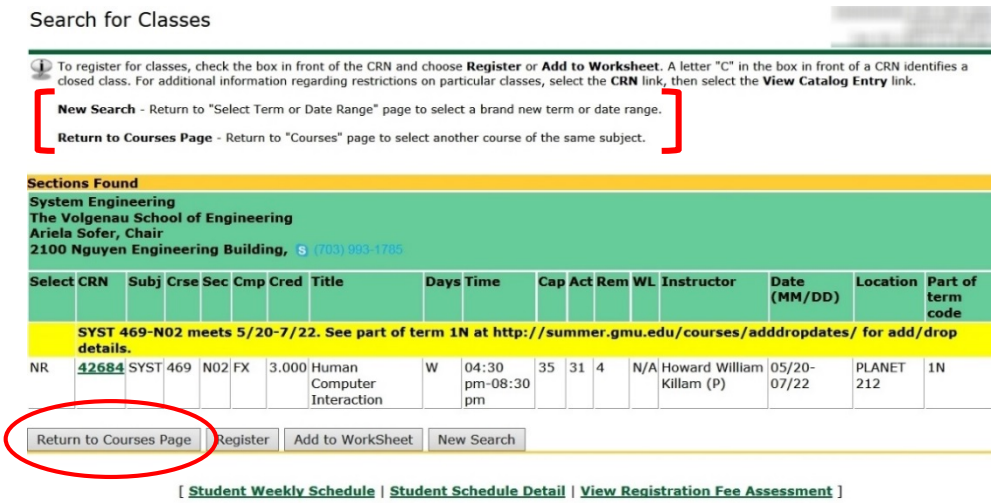


Figure 47. Case Study – Prototype PatriotWeb Website – New "Return to Courses Page" Button

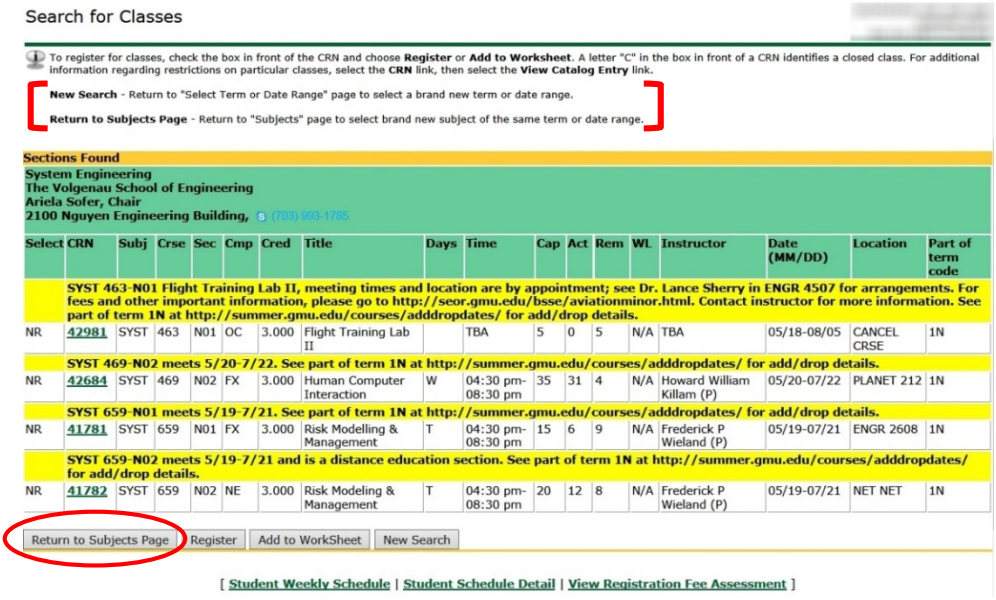


Figure 48. Case Study – Prototype PatriotWeb Website – New "Return to Subjects Page" Button

(6). Improved information presentation

(7). On the prototype PatriotWeb system, the advanced search field options page was re-designed that displayed directly below the "**Advanced Search**" button of the Search for Classes page whenever Linda Rose clicks on the button to do advanced searches. A panel slid down showing the advanced search field options information when the "Advanced Search" button was clicked and the panel can be hidden by clicking on the Advanced Search button. The advanced search field options information was displayed on the same page as the Search for Classes page. This one-page information thus eliminated an extra page transition as in the

current PatriotWeb system. The new presentation of the advanced search field options information helped to improve the "**Efficiency**" usability factor. Figure 49 presents the new advanced search field options panel.

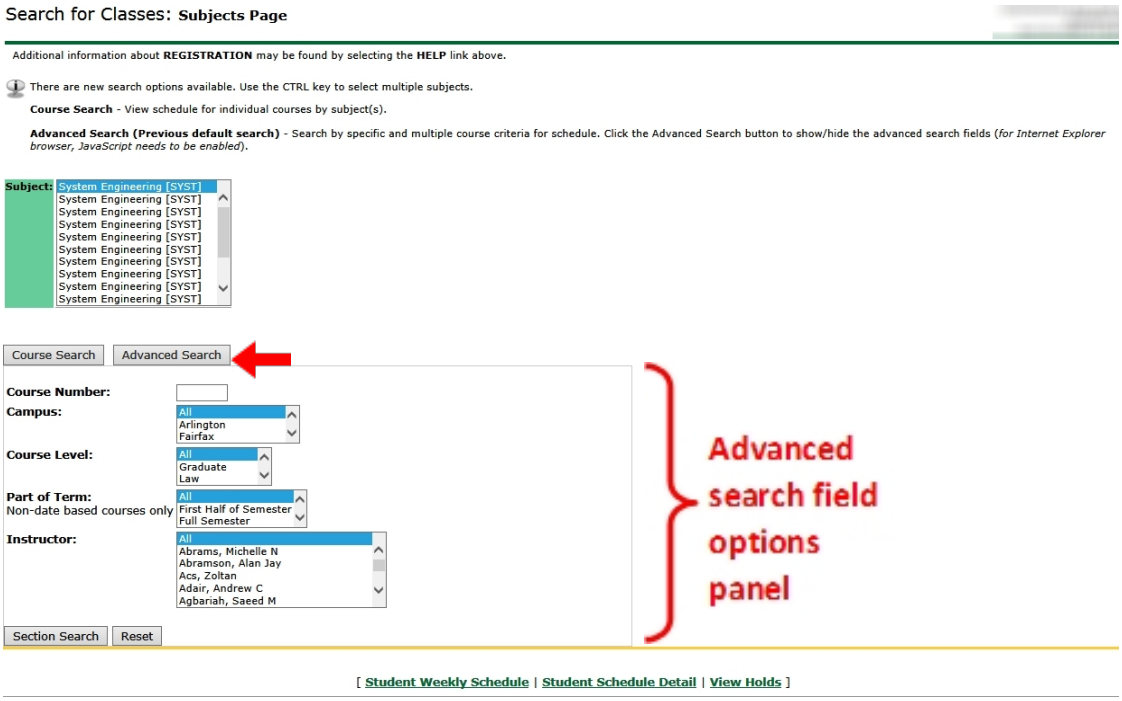


Figure 49. Case Study – Prototype PatriotWeb Website – Advanced Search Field Options Panel

CHAPTER 8: RESEARCH EVALUATION, FINDINGS, AND RESULTS

8.1 Introduction

The primary objective of this research is to investigate persona driven user requirements modeling using an ontology-based approach. The central theme of this research is focused in two areas:

(1). User requirements analysis and modeling in requirements engineering. Specifically, the application of the concept of persona and its relationship with scenarios, tasks, goals, and requirement.

(2). Ontology-based approach for defining an explicit specification of concepts, properties, and relationships between concepts in the knowledge base.

The research hypotheses (**H**) asserted in this dissertation (chapter 1) are restated below:

H1 – *The concept of persona and its relationship with scenarios, tasks, goals, and requirements help to understand the target users' needs and behaviors.*

H2 – *The concept of persona, in the context of scenarios, tasks, goals, and requirements helps to refine requirements.*

H3 – *Using an ontology-based approach to define an explicit specification of users' knowledge (i.e. personas, scenarios, tasks, goals, and requirements) helps to improve the design model quality requirements of the ontologies by providing*

checking of constraints, consistencies, and completeness of concepts in the knowledge base.

To validate the above three research hypotheses, the research evaluation was divided into two parts:

PART 1 – Evaluation on the effectiveness of applying persona concept in user requirements analysis and modeling.

PART 2 – Evaluation on the effectiveness of applying ontology-based approach in the specification of concepts, properties, and relationships between concepts in the knowledge base.

Figure 50 depicts the evaluation parts and the corresponding validation of research hypotheses.

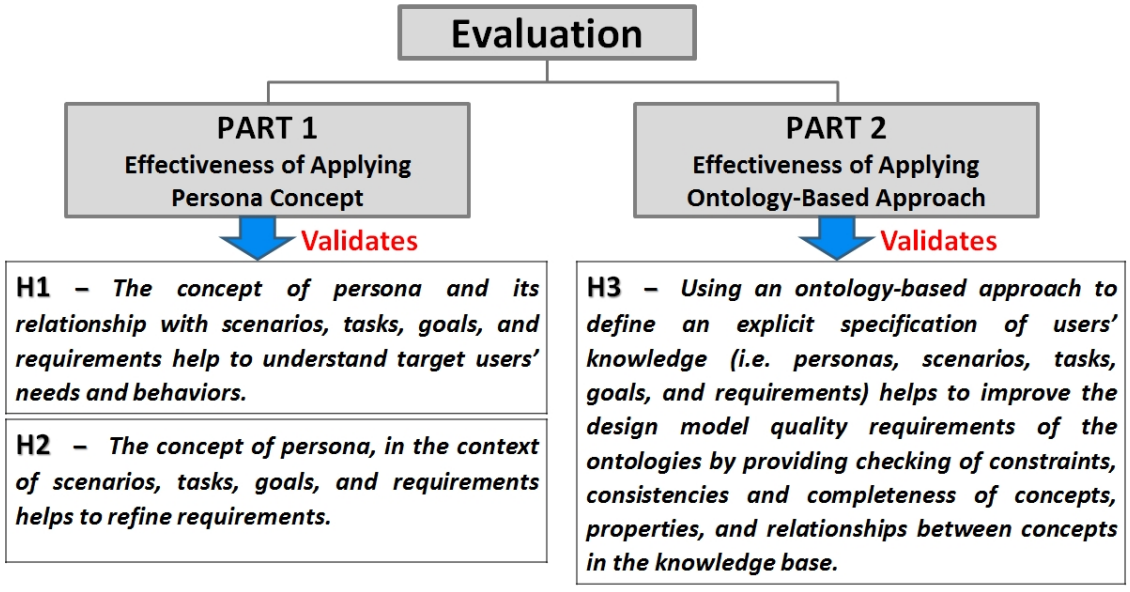


Figure 50. Evaluation and Hypotheses Validation

8.2 PART 1 Evaluation – Persona Concept

The objective of the PART 1 evaluation was to assess and measure the effectiveness of applying the concept of persona in user analysis and modeling to validate the research hypotheses, H1 and H2 (section 7.1 and also chapter 1 for the stated hypotheses). Thus an empirical evaluation was conducted to evaluate whether there was any difference in the usability goals between the current PatriotWeb and the prototype PatriotWeb (with application of persona). The evaluation sought to evaluate how the application of the persona concept can empower the development team in better understanding of the target users' needs and behaviors and the possibility of refining requirements to improve the design of the system-to-be.

As the scope of the research was focused on the application of the persona concept in user requirements analysis and modeling, and not on the identification of personas, it was thus not practical for the evaluation participants to research/identify personas and using different personas as the basis for the evaluation which would add an additional level of variability to the evaluation. Therefore, a primary persona (Linda Rose) was pre-identified and was used throughout this research, i.e. on the case study and the evaluation.

Prior to the empirical evaluation, a qualitative evaluation has been conducted on the PatriotWeb case study via the design and implementation of a prototype PatriotWeb system [PATRIOTWEB b] to reflect new and improved features as a result of applying the persona concept in the CDP model of the Persona-URM framework. The prototype PatriotWeb system used in the qualitative evaluation helped to support the empirical

evaluation in comparing with the current PatriotWeb system with respect to eight usability factors. Detailed descriptions of the PatriotWeb case study results and the new and improved features on the Prototype PatriotWeb system are provided in chapter 6 and chapter 7 respectively. In this chapter, we focus our attention on the empirical evaluation of the persona concept and the evaluation of the ontology-based approach,

8.2.1. Design of Empirical Evaluation

(1). Participants

The evaluation was conducted utilizing undergraduate students of George Mason University enrolled in two classes from the departments of Statistics and Electrical and Computer Engineering during the summer 2015 semester. Participation in the evaluation was voluntary. The total number of students participated in the evaluation were 84, with 29 students from the Electrical and Computer Engineering class and 55 students from the Statistics class.

(2). Method of Experiment

i. Based on the requirements issues and resolutions identified and analyzed during the Analysis and Evaluation process (process 4) of the CDP model conducted on the case study (chapter 6 section 6.2) with respect to the eight usability factors (Table 1 chapter 5 section 5.2.4) chosen for this research on the selected five scenarios from the viewpoint of the primary persona Linda Rose, a prototype PatriotWeb system [PATRIOTWEB b] was developed for the selected five scenarios:

- Scenario 1: View Student Services Page
- Scenario 2: View Registration Page
- Scenario 3: View Student Records Page
- Scenario 4: View Personal Information Page
- Scenario 5: Search for Classes

The five selected scenarios served as the base for the evaluation. The screenshots (webpage representations and transitions) of the step-by-step tasks of the five scenarios on both the current GMU PatriotWeb and the prototype PatriotWeb are provided in Appendix G. Representative screenshots of the "Student Services" page and the "Registration" page of the prototype PatriotWeb are shown in Figure 51 and 52 respectively.

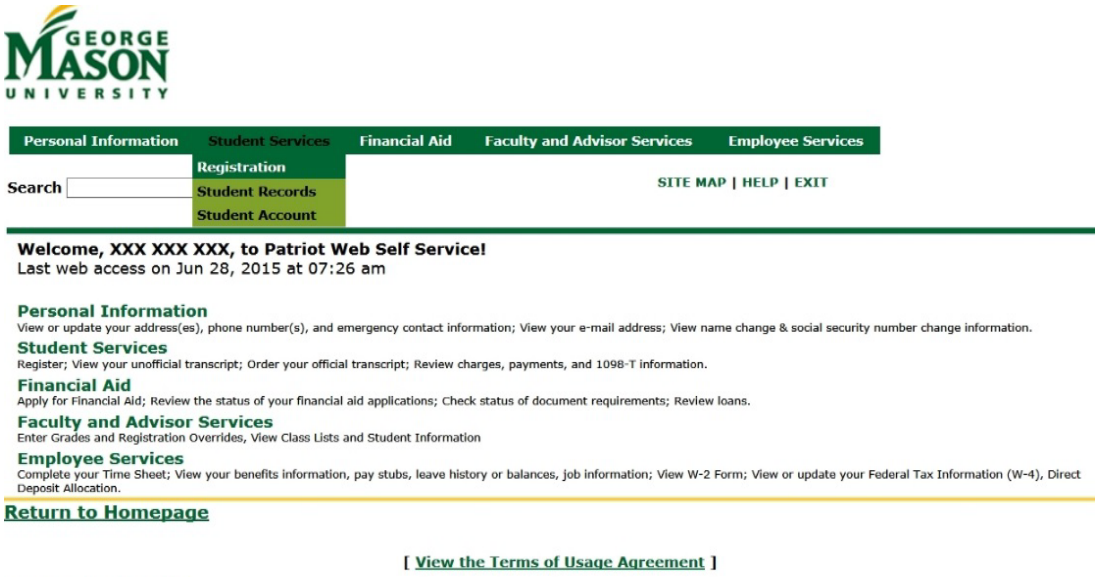


Figure 51. Evaluation – "Student Services" Page of Prototype PatriotWeb

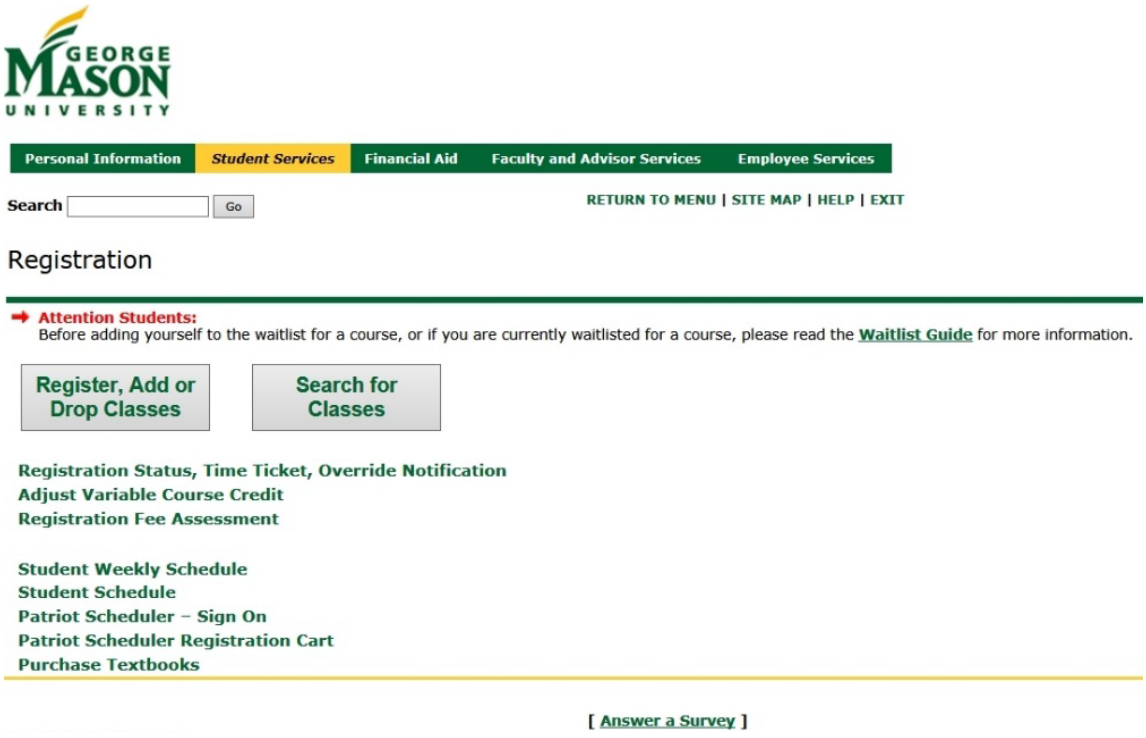


Figure 52. Evaluation – "Registration" Page of Prototype PatriotWeb

ii. All participants were given instructions prior to the start of the evaluation process. All Participants were provided with printed screenshots of step-by-step tasks of the five chosen scenarios for both the current PatriotWeb and the prototype PatriotWeb (see Appendix F). For participants who have personal computers with them, they accessed the current GMU PatriotWeb website [PATRIOTWEB a] and the prototype PatriotWeb website [PATRIOTWEB b] online and performed those tasks of the five chosen scenarios on their computers.

iii. After the participants carefully performed the step-by-step tasks of the five scenarios either following the printed screenshots or using their personal computers, the participants completed a short questionnaire consisting of questions pertaining to participant's general profile, participant's general usage of current PatriotWeb system, and the comparison of prototype PatriotWeb with the current PatriotWeb.

(3). Method of Evaluation

i. On the questionnaire, the participants rated the usability of the prototype PatriotWeb in comparison to the current PatriotWeb on the selected five scenarios based on three criteria used for the questionnaire (see actual survey questions in Appendix E): "Worse, "Same (No Improvement)", or "Better (Improvement)". To measure the scores of the ratings, the criteria were assigned numeric scores as **-2** for "Worse", **0** for "Same", and **+2** for "Better". The final score for each usability factor is then normalized to 1.0 (to nearest one decimal). The usability questions, written in non-technical terms on the questionnaire were categorized according to the eight usability factors used in this research (Table 1 chapter 5 section 5.2.4). The evaluation instructions to participants and survey questions are included in Appendix E.

8.2.2. PART 1 Empirical Evaluation Analysis, Findings, and Results

(1). Classification of Respondents

The survey conducted in PART 1 evaluation has an initial total number of 84 respondents drawn from two undergraduate level classes at George Mason University during the summer 2015 semester: 29 students from Electrical and Computer

Engineering class and 55 students from Statistics class. To evaluate the effectiveness of persona on the usability of the current PatriotWeb system and the prototype PatriotWeb system, the 84 respondents were further classified into two groups:

Group A: Respondents who resemble the primary persona Linda Rose's characteristics as closely as possible.

Group B: Respondents *not* in Group A.

The two groups were classified based on the responses from questions in Section A and Section B of the questionnare (complete survey questions are included in Appendix E). Section A contains general questions related to respondents' demographics and computer literacy; Section B contains questions related to usage profile of respondents on the current PatriotWeb system. Criteria chosen in categorizing the two groups are listed in Table 4.

Table 4. Criteria of Categorizing Survey Respondents (Group A, Group B)

Criteria		Description
1	**Demographics**	Information including education background (e.g. current school status: full-time/part-time, past education), occupations (e.g. current work status: full-time/part-time/not employed).
2	**Level of Computer and Web Literacy**	Information including computing platforms (e.g. Windows 8, 7, XP, UNIX), computing devices (e.g. desktop computer, laptop, mobile/smart phone, tablet), and web literacy (e.g. general Internet browsing, basic programming languages skills).
3	**Frequency of**	Computing devices such as desktop computer,

	Computing Devices Use	laptop, mobile/smart phone, and tablet.
4	**Environments (when using PatriotWeb system)**	Environments such as public outdoor places (on-campus, off-campus) and private (off-campus) home.
5	**Usability Preferences (when using the PatriotWeb system)**	Selective usability preferences including usability factors such as accuracy, efficiency (general), comprehensibility, clarity, ease in performing tasks, and easy navigation system. (Explanations of the usability factors are provided in Table 1, chapter 5 section 5.2.4)

The initial categorization of the two groups were based on criteria 1, i.e. Demographics which included education background (current school status: full-time/part-time) and occupation status (current work status: full-time/part-time/not employed). The education (current school status) and occupation (current work status) were chosen as the first level criteria to separate the two groups mainly because the full-time school and part-time work status represent the primary persona Linda Rose's school and work status of being a busy individual.

From the survey responses of the 84 respondents, the initial categorization of the two groups based on the criteria education and occupation is summarized as shown in Table 5a. In Table 5a, FT refers to Full-Time, PT refers to Part-Time, None refers to Not Employed, and asterisk * refers to no response(s) for the education and/or occupation fields. Recall that Group A consists of respondents who resemble the

primary persona Lina Rose's characteristics whereas Group B consists of respondents not in Group A.

Table 5a. Initial Categorization of Survey Respondents (Group A, Group B)

	Group						
Criteria	**A**	**B**					
Education	FT	FT	PT	PT	FT	PT	*
Occupation	PT	FT	FT	PT	None	None	*
Distribution of Respondents	**37**	2	4	2	34	0	5
		47					

The 37 respondents categorized as Group A after an initial categorization were further analyzed based on the remaining selected criteria (i.e. Table 4, criteria 2-5). Table 5b shows the survey responses sought in the final categorization of Group A. It is to be noted that the final categorization of Group A was formed that matched as close as possible to the primary persona Lina Rose's characteristics; it does not mean that all respondents in Group A matched the primary persona Linda Rose's characteristics completely. For example, from the 37 respondents in Group A, if no respondent matched exactly the responses sought for criteria 2-5, then if a respondent's responses matched 3 out of 4 criteria from criteria 2-5 (total 4 criteria), then that respondent was included in Group A. For criteria 5, if a respondent's responses matched 4 out of 6

attributes, then that respondent is considered to have matched criteria 5. The respondents with the highest number of matched responses were included in Group A followed by respondents with the second highest number of matched responses and so forth. Hence, with this criteria selection approach for the final categorization of Group A, the final total number of respondents in Group A resulted in **34 respondents**.

Table 5b. Final Categorization of Survey Group A

Criteria		Responses Sought
2	**Level of Computer and Web Literacy**	Comfortable, Familiar
3	**Frequency of Computing Devices Use**	Every Day
4	**Environments (when using PatriotWeb system)**	Public, Off-Campus Private Home
5	**Usability Preferences (when using the PatriotWeb system)**	Attribute 1: Efficiency – Very Important Attribute 2: Comprehensibility – Very Important Attribute 3: Clarity – Important Attribute 4: Accuracy – Very Important Attribute 5: Ease in Performing the Tasks – Important Attribute 6: Easy to Navigate the Website – Important
	Final Number of Respondents (Group A)	**34**

With the final number of respondents in Group A determined (i.e. 34 respondents), the number of respondents in Group B were thus reduced to the same number as Group A, i.e. from 47 respondents (Table 5a) to 34 respondents by randomly selecting respondents from the poll of 47 respondents in Group B. This final equal number of respondents for both groups allowed a fair comparison of survey results on both groups. Therefore, the final total number of respondents (Group A and Group B) were **68**.

(2). Survey Responses

The main results of this survey came from **Question 21** and **Question 22** of Section C of the survey form which focused on the comparison of the prototype PatriotWeb system with the current PatriotWeb system on the selected five scenarios with respect to the eight usability factors used in this research. The explanations of the eight usability factors are provided in Table 1 chapter 5 section 5.2.4. Survey Question 21 and survey responses for Question 21 are reproduced in Figure 53 and Figure 54 respectively. Complete survey questions are included in Appendix F and complete summarized data of this survey are included in Appendix H and I.

Section C:
Current PatriotWeb System and Prototype PatriotWeb System

***Question 21**, **question 22**, and **question 23** refer specifically to the tasks of the five scenarios you were asked to perform on both the current PatriotWeb System and the PatriotWeb prototype system. The five scenarios (provided to you as screenshots) are: "View Student Services Page", "View Registration Page", "View Student Records Page", "View Personal Information page", and "Search for Classes".*

You will perform the tasks by following the printed webpage transition screenshots provided to you. If you have a computer with you, you may access the current GMU PatriotWeb website and the PatriotWeb prototype website and follow the printed webpage transition screenshots to perform the tasks of the five scenarios on your computer. Both websites can be accessed at:
Current GMU PatriotWeb: https://patriotweb.gmu.edu/
PatriotWeb prototype: http://mason.gmu.edu/~wsim/Main_Menu.htm

21. **In COMPARISON to the current PatriotWeb system on the five scenarios, how would you rate the PatriotWeb prototype system according to the items listed in the first column? Feel free to refer the printed screenshots.** (*For each item, please check ✓ one checkbox that applies to you*)

Usability Item	Worse	Same (No improvement)	Better (Improvement)
[Efficiency] **Number of clicks or pages needed to reach a target destination page. (i.e. speed, task time)**	☐	☐	☐
[Efficiency] **Number of steps that are repeated before reaching a target destination page.**	☐	☐	☐
[Comprehensibility] **Visibility, consistency, or location of menus, links, or buttons.**	☐	☐	☐
[Comprehensibility] **Menus, links, or buttons are easy to understand. (i.e. easy to understand what each menu, link, or button does)**	☐	☐	☐
[Comprehensibility] **Number of redundant or unnecessary menus, links, or buttons.**	☐	☐	☐

[Accuracy] **The outcome of clicking the menus, links, or buttons produce the result correctly. (i.e. as you would expect).**	☐	☐	☐
[Learnability] **Easy to learn how to navigate the site, how to carry out a task, where to click, etc.**	☐	☐	☐
[Rememberabilty] **Able to remember how the system looks like and what the system does.**	☐	☐	☐
[Reliability] **Information is credible.**	☐	☐	☐
[Attractiveness] **Attractiveness of the web pages. (Color, layout, font style/size, etc.)**	☐	☐	☐
[Clarity] **Texts are readable.**	☐	☐	☐

Figure 53. Evaluation – Survey Question 21

Total Respondents (Final): 68 respondents						
Group A (persona applied): 34 respondents						
Group B (persona not applied): 34 respondents						

			Group A		Group B	
Q21 In comparison to the current PatriotWeb system on the five scenarios, how would you rate the PatriotWeb prototype system?	Item 1 (Efficiency)	Same	4	12%	15	44%
		Better	30	88%	19	56%
	Item 2 (Efficiency)	Same	6	18%	15	44%
		Better	28	82%	19	56%
	Item 3 (Comprehensibility)	Same	4	12%	9	26%
		Better	30	88%	25	74%
	Item 4 (Comprehensibility)	Same	10	29%	16	47%
		Better	24	71%	18	53%
	Item 5 (Comprehensibility)	Same	4	12%	12	35%
		Better	30	88%	22	65%
	Item 6 (Accuracy)	Same	12	35%	16	47%
		Better	22	65%	18	53%
	Item 7 (Learnability)	Same	11	32%	16	47%
		Better	23	68%	18	53%
	Item 8 (Rememberability)	Same	34	100%	34	100%
		Better	0	0%	0	0%
	Item 9 (Reliability)	Same	34	100%	34	100%
		Better	0	0%	0	0%
	Item 10 (Attractiveness)	Same	21	62%	24	71%
		Better	13	38%	10	29%
	Item 11 (Clarity)	Same	25	74%	27	79%
		Better	9	26%	7	21%

Figure 54. Evaluation – Survey Question 21 Responses

For **Question 21** on the rating of the prototype PatriotWeb, the survey responses (Figure 54) showed that:

a) Both groups (Group A and Group B) gave rating "Better" for usability factors of "Efficiency", "Comprehensibility", "Accuracy", and "Learnability". These four

factors correspond to items 1-7 on Question 21 on the questionnaire. In comparing the number of respondents who rated "Better" between Group A and Group B for each item 1-7, the results clearly showed that more respondents in Group A gave rating "Better" than in Group B. For examples:

- For item 1 ("Efficiency"), 30 out of 34 respondents (88%) in Group A gave rating "Better" as compared to 19 out of 34 respondents (56%) in Group B gave rating "Better".
- For item 3 ("Comprehensibility"), 30 out of 34 respondents (88%) in Group A gave rating "Better" as compared to 25 out of 34 respondents (74%) in Group B gave rating "Better".
- For item 6 ("Accuracy"), 22 out of 34 respondents (65%) in Group A gave rating "Better" as compared to 18 out of 34 respondents (53%) in Group B gave rating "Better".

The high number of responses from Group A on "Efficiency", "Comprehensibility", "Accuracy", and "Learnability" usability factors is not a surprise, as this is evident from the outcome of conducting the Scenarios-Usability Goals Evaluation (SUGE) activity on the case study (on persona Linda Rose) which highlighted issues and resolutions on "Efficiency", "Comprehensibility", "Accuracy", and "Learnability" usability factors (chapter 6, section 6.2 or Appendix D).

b) For usability factors "Rememberability" (item 8), "Reliability" (item 9), "Attractiveness" (item 10), and "Clarity" (item 11), the responses did not show much

difference between the prototype PatriotWeb system and the current PatriotWeb system for both groups. This is evident as there was no issues identified from the Scenarios-Usability Goals Evaluation (SUGE) activity conducted on the case study (chapter 6, section 6.2 or Appendix D).

Figure 55 shows the response results for **Question 22**, which asks about the overall opinion of the prototype PatriotWeb system in comparison with the current PatriotWeb system. As with Question 21, it is not surprising to see that more respondents in Group A (26 out of 34 respondents, 76%) as compared to Group B (20 out of 34, 59%) gave a rating of "Better" on the overall rating of the prototype PatriotWeb.

		Group A		Group B	
Q22 Overall how would you rate the PatriotWeb prototype system in comparison to the current PatrioWeb system, based on the five scenarios?	Same	8	24%	14	41%
	Better	26	76%	20	59%

Figure 55. Evaluation – Survey Question 22 Responses

For **Question 20** of Section B of the survey form (Figure 56) on the opinion of mobile version of the GMU PatriotWeb system, the responses from both groups showed positive results, i.e. both group feel that a mobile version of the GMU PatriotWeb is beneficial, with 85% (29 out of 34 respondents) for Group A and 71% (24 out of 34 respondents) for Group B. The high percentage result for Group A is not quite a surprise, as from the

persona Linda Rose profile (chapter 6, section 6.2, or Appendix D), Linda Rose wants to go online and quickly check for classes and other school activities mostly outside (at work lunch time). Thus a mobile version with streamlined features of GMU PatriotWeb comes naturally for Linda Rose.

		Group A		Group B	
Q20 (Section B) Do you feel that a mobile version of the GMU PatriotWeb system with streamlined most common or used features is beneficial to you as a user of the system?	Yes	29	85%	24	71%
	No	5	15%	10	29%

Figure 56. Evaluation – Survey Question 20 Responses

Some reasons as to why respondents (in both groups) did not feel that the mobile version of the GMU PatriotWeb system is beneficial were:

- Respondents preferred a full version of the GMU PatriotWeb rather than the streamlined version on the mobile device.
- Respondents preferred better interface on mobile devices such as mobile navigation system and clarity of texts.
- Respondents preferred bigger screen viewing (desktop, laptop) as compared to small screen viewing in mobile devices.

- **Survey Results Computation**

  As mentioned in section 8.2.1, the survey participants scored the usability of the prototype PatriotWeb in comparison to the current PatriotWeb on the selected five scenarios based on three criteria: “Worse, “Same (No Improvement)”, or “Better (Improvement)” (actual survey questions in Appendix F). To measure the scores, the criteria were assigned numeric scores as **-2** for “Worse”, **0** for “Same”, and **+2** for “Better”. Therefore, all responses for Question 21 and Question 22 are computed based on the numeric scores. The final score for each usability factor is then normalized to 1.0 (to nearest one decimal). As there are total 11 evaluating usability items (which correspond to eight usability factors) on the survey for Question 21, if there are more than one item belongs to a particular usability factor, then the scores are averaged. For examples:

  a) “Attractiveness” (item 10 on survey responses, see Figure 54):

  Group A: 13 respondents rated "Better" and 21 respondents rated "Same". Therefore, (13*2)+(21*0) = 26. Final normalized score = (26/68) * 1.0 = 0.4

  b) “Efficiency” (item 1 and 2 on survey responses, see Figure 54):

  There are 2 items (item 1 and 2) on Question 21 that belong to "Efficiency"

  Group A: total 58 respondents (for 2 items, 30+28) rated "Better" and total 10 respondents (for 2 items, 4+6) rated "Same". Therefore, ((58*2)+(10*0))/2 = 58. Final normalized score = (58/68) *1.0 = 0.9

The normalized scores for the eight usability factors (Question 21) and the normalized scores for the overall rating (Question 22) are presented in Figure 57 as bar charts. The charts are also included in Appendix I.

As with the survey responses (in percentages) discussed earlier, the usability factors "**Efficiency**", "**Comprehensibility**", "**Accuracy**", "**Learnability**" achieved higher normalized scores of **0.9**, **0.8**, **0.7**, and **0.7** respectively on the prototype PatriotWeb in Group A as compared to **0.6**, **0.6**, **0.5**, and **0.5** respectively in Group B. The overall rating also showed that the prototype PatriotWeb achieved higher normalized score of **0.8** in **Group A** as compared to **0.6** in **Group B**.

Figure 57. Evaluation – Normalized Scores for Survey Question 21 and Question 22

The empirical evaluation results showed that with the application of persona concept, a better understanding of the target users' needs and behaviors was achieved, requirements issues were identified and resolved, and requirements were refined early in the requirements engineering process to help improve the usability of the system-to-be. This empirical evaluation successfully validated the research

hypotheses H1 and H2 as asserted in this dissertation (refer to 8.1 of this chapter and also chapter 1 for the hypotheses).

8.3 PART 2 Evaluation – Ontology-Based Approach

The objective of the PART 2 evaluation was to assess and measure the effectiveness of using ontology approach of providing explicit specifications of concepts and properties in the knowledge base to validate the research hypotheses H3. The explicitness of the concepts, properties, and relationships between concepts specified in the ontology knowledge base allowed for reasoning rules to be performed to check for completeness and consistency, along with the ability to enforce constraints to help improve the quality requirements of the design model.

Therefore, in this PART 2 evaluation on the effectiveness of using ontology approach, we applied query and rule-based techniques to enforce constraints and check completeness and consistency of concepts, properties, and relationships between concepts specified in the ontology knowledge base based on predefined constraints and rules to assess the quality requirements of the design model.

8.3.1. Design of Evaluation (query and rule-based)

(1). Method of Experiment

i. The concepts, properties, and concepts relationships of the three ontologies (Persona Ontology, Behavioral-GST Ontology, and Requirements Ontology) of the OntoPersonaURM model were specified in the Protégé-Frames editing tool. The

concepts were instantiated with instances in Protégé-Frames using the PatriotWeb case study as demonstration.

ii. To enforce the integrity of the design model of the ontologies in the knowledge base, constraints checking was applied by writing a set of constraints via the PAL [PAL] Constraint Tab plugin to Protégé-Frames tool (chapter 4, section 4.3) to detect incomplete entry of information and to check entered information for inconsistencies. To detect any violations of completeness and consistency of concepts in the knowledge base, two sets (completeness and consistency) of Jess reasoning rules were pre-defined in natural language and to realize the completeness and consistency checks via JessTab [JESS] to Protégé-Frames tool (chapter 4, section 4.4).

(2). Method of Evaluation

The use of PAL Constraints Tab plugin to Protégé-Frames helped to express constraints expressions and maintain the integrity of the ontologies in the knowledge base. For example, Figure 58 shows a PAL constraint written to check that for all names, there must have at most one persona for each associated name in the Persona Ontology (using the PatriotWeb case study as demonstration). The constraint statement is: *Each name must have at most one persona*. A total of 76 PAL constraints were written for the three ontologies using the PatriotWeb case study as demonstration and are included in Appendix J.

```
(defrange ?name :FRAME Name)
(forall ?name (=> (and (own-slot-not-null first_name ?name)
                       (own-slot-not-null last_name ?name)
                       (own-slot-not-null middle_name ?name))
                  (and (own-slot-not-null isPersonaNameOf ?name)
                       (< (number-of-slot-values isPersonaNameOf ?name) 2))))
```

Figure 58. Evaluation – PAL Constraint Example

Jess rules were applied to check for consistency of the concepts, their properties, and relationships between concepts specified in the ontology knowledge base. The consistency rules, written in Jess via JessTab plugin to Protégé-Frames, check for conflict concepts and valid relations (between concepts) in the three ontologies using the PatriotWeb case study as demonstration. A total of 60 consistency rules were pre-defined. The consistency statements are expressed in natural language and the reasoning rules are written in Jess via JessTab plugin to Protégé-Frames. The 60 consistency rules are included in Appendix J.

For example, the following rule (Figure 59) checks for conflict between functional and non-functional requirement category. The consistency statement is: *A functional requirement category cannot also be a non-functional requirement category.*

```
(mapclass Functional)
(mapclass NonFunctional)
(defrule no_identical_requirement_category
  (object (is-a Functional) (category ?c) (id ?i))
  (object (is-a NonFunctional) (category ?c) (id ~?i))
  =>
(printout t "The category " ?c " appears in both Functional and Non-Functional requirement category and should be resolved." crlf))
```

Figure 59. Evaluation – Jess Consistency Rule Example 1

The following rule (Figure 60) checks for valid refinement relationship between requirements. The consistency statement is: *If requirement A is refined into requirement B, then there is a refined from relationship that requirement B is refined from requirement A.*

```
(mapclass Requirement)
(defrule refinesInto_refinesFrom
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (refinesInto $? ?r1 $?))
  (object (is-a Requirement) (OBJECT ?r1) (refinesFrom ?r2&:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Requirement " ?i " and refinement relationship (" (slot-get ?r1 id) ") are inconsistent." crlf))
```

Figure 60. Evaluation – Jess Consistency Rule Example 2

For completeness check, a total of 71 completeness rules were written to check the completeness of concepts, properties, and relationships between concepts of the three ontologies specified in the Protégé-Frames tool (using the PatriotWeb case study as demonstration). The completeness rules check whether important concepts (or classes) have been specified and if all required attributes and relationships have been specified in the ontology knowledge base. The completeness statements are expressed in natural language and the completeness rules are written in Jess via JessTab plugin to Protégé-Frames. The 71 Jess completeness rules are included in Appendix J.

For example, the following rule (Figure 61) checks that all required attributes of a requirement have been specified in the ontology knowledge base. The completeness statement is: *Every requirement must specify requirement id, statement, source, status, priority, validation, and verification*.

```
(mapclass Requirement)
(defrule specify_required_requirement_attribute
(or (object (is-a Requirement) (OBJECT ?obj) (id nil))
(object (is-a Requirement) (OBJECT ?obj) (statement nil))
(object (is-a Requirement) (OBJECT ?obj) (source $?s&:(= (length$ ?s) 0)))
(object (is-a Requirement) (OBJECT ?obj) (status nil))
(object (is-a Requirement) (OBJECT ?obj) (priority nil))
(object (is-a Requirement) (OBJECT ?obj) (validation nil))
(object (is-a Requirement) (OBJECT ?obj) (verification nil)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not
specified. Required attributes are id, statement, source, status, priority, validation, and verification." crlf))
```

Figure 61. Evaluation – Jess Completeness Rule Example

8.3.2. PART 2 Evaluation Findings and Results

- **Evaluation Findings**

a) PAL Constraints Check

As mentioned in section 8.3.1 and also in chapter 4, the primary purpose of PAL is to enforce the integrity of the design model in the ontology knowledge base by detecting incomplete entry of information and checking entered information for inconsistencies. In Protégé-Frames editing tool, when a required field (at instance level) is empty (not filled in), a red box appears around the field in the instance browser of Protégé-Frames tool. This serve as a warning to the engineers that the required field is missing an entry. For example, in the Name class of the Persona Ontology, the field isPersonaNameOf is a required field and it takes only a single value (i.e. cardinality is one). The isPersonaNameOf field is not entered and thus a red box appeared around the isPersonaNameOf field in the Instance Browser of the Protégé-Frames tool. (Figure 62).

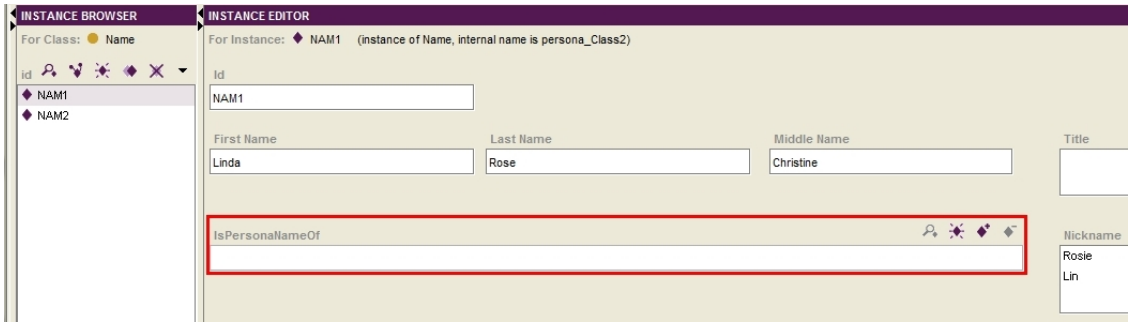


Figure 62. Evaluation – PAL Constraint Violation Example 1a

To test for constraints and check for incomplete entry of information in the knowledge base, PAL was applied to test the constraints by executing constraint statements written via PAL Constraints Tab plugin to Protégé-Frames. A total of 76 PAL constraint statements were written for the three ontologies developed in Protégé-Frames tool (using the PatriotWeb case study as demonstration). The 76 PAL constraint statements are included in Appendix J. One of the PAL constraints to test the missing entry and cardinality constraint as described earlier, i.e. the missing entry of the isPersonaNameOf field of the Name class (Figure 62) has been written (see section 8.3.1) and is reproduced here (Figure 58). The constraint statement is: *Each name must have at most one persona*.

```
(defrange ?name :FRAME Name)
(forall ?name (=> (and (own-slot-not-null first_name ?name)
                       (own-slot-not-null last_name ?name)
                       (own-slot-not-null middle_name ?name))
                  (and (own-slot-not-null isPersonaNameOf ?name)
                       (< (number-of-slot-values isPersonaNameOf ?name) 2))))
```

Figure 58. Evaluation – PAL Constraint Example

When the above constraint statement is executed via PAL Constraints Tab, a red circle is displayed next to the constraint statement and the instance that violated this constraint is displayed on the right of the PAL Constraints window (Figure 63).

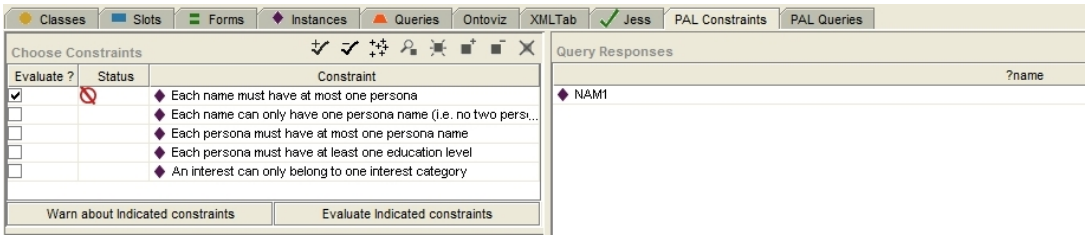


Figure 63. Evaluation – PAL Constraint Violation Example 1b

b) Jess Consistency and Completeness Check

To check for consistency of the concepts, their properties, and relationships specified in the ontology knowledge base, Jess rules were written via the JessTab plugin to Protégé-Frames. The consistency rules checked for conflict concepts and valid relations (between concepts) in the three ontologies (using the PatriotWeb case study as demonstration). A total of 60 Jess consistency rules were written. The consistency statements are expressed in natural language and the reasoning rules are written in Jess. The 60 Jess consistency rules are included in Appendix J.

One of the consistency rules checks for conflict between functional and non-functional requirement category (section 8.3.1). The consistency statement is: *A functional requirement category cannot also be a non-functional requirement category*. The Jess rule for this consistency check has been written (section 8.3.1) and is reproduced here (Figure 59).

```
(mapclass Functional)
(mapclass NonFunctional)
(defrule no_identical_requirement_category
  (object (is-a Functional) (category ?c) (id ?i))
  (object (is-a NonFunctional) (category ?c) (id ~?i))
  =>
(printout t "The category " ?c " appears in both Functional and Non-Functional requirement category and should be resolved." crlf))
```

Figure 59. Evaluation – Jess Consistency Rule Example 1

When requirement category "Design" and requirement category "Interface" were mistakenly included in both the functional and non-functional sub-classes of the requirement category class (Figure 64), the above Jess rule detected this inconsistency and returned the message indicating that both "Design" and "Interface" requirement categories need to be resolved (Figure 65).

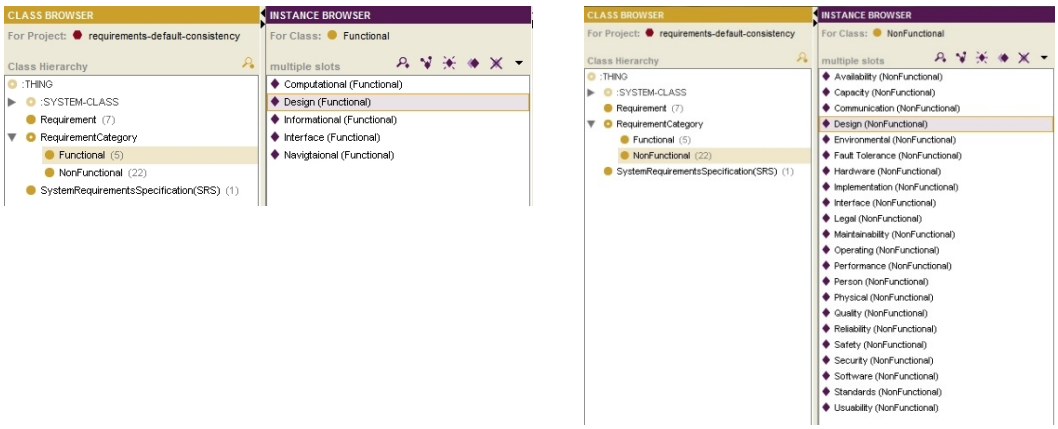


Figure 64. Evaluation – Inconsistent Class Instances

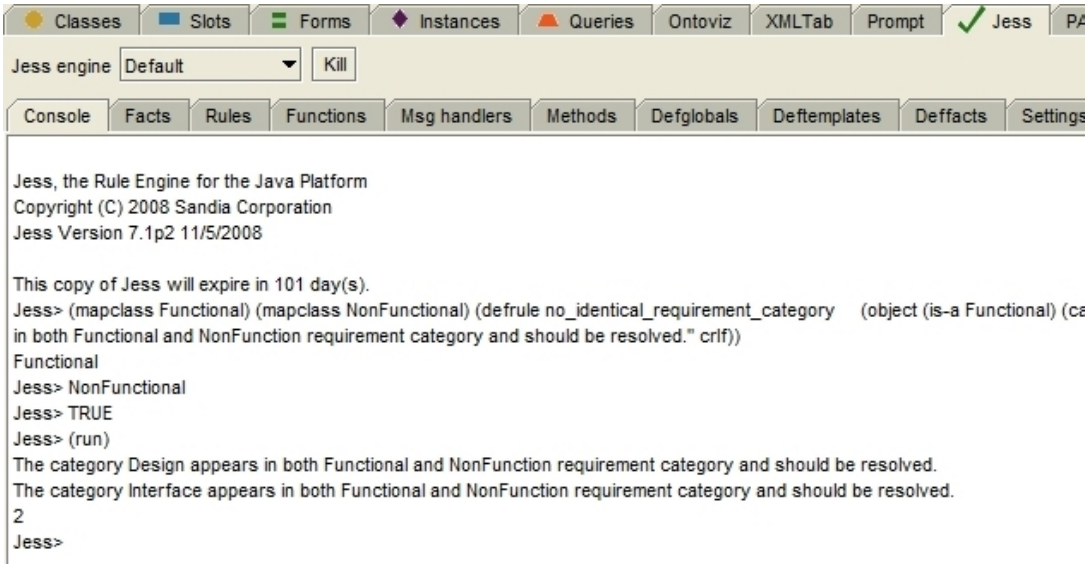


Figure 65. Evaluation – Jess Rule Execution Result for Inconsistency

For completeness check, a total of 71 completeness rules were written to check the completeness of concepts, properties, and relationships of concepts of the three ontologies specified in the Protégé-Frames tool (using the PatriotWeb case study as demonstration). The completeness rules check whether important concepts (or classes) have been specified and if all required attributes and relationships have been specified in the ontology knowledge base. The 71 Jess completeness rules are included in Appendix J.

One of the completeness rules discussed in earlier section (section 8.3.1) checks that all required attributes of a requirement have been specified in the ontology knowledge base. The completeness statement is: *Every requirement must specify requirement id,*

statement, source, status, priority, validation, and verification. The Jess rule for this completeness check has been written (section 8.3.1) and is reproduced here (Figure 61).

```
(mapclass Requirement)
(defrule specify_required_requirement_attribute
(or (object (is-a Requirement) (OBJECT ?obj) (id nil))
(object (is-a Requirement) (OBJECT ?obj) (statement nil))
(object (is-a Requirement) (OBJECT ?obj) (source $?s&:(= (length$ ?s) 0)))
(object (is-a Requirement) (OBJECT ?obj) (status nil))
(object (is-a Requirement) (OBJECT ?obj) (priority nil))
(object (is-a Requirement) (OBJECT ?obj) (validation nil))
(object (is-a Requirement) (OBJECT ?obj) (verification nil)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not
specified. Required attributes are id, statement, source, status, priority, validation, and verification." crlf))
```

Figure 61. Evaluation – Jess Completeness Rule Example

In the Requirement class of the Requirements Ontology, a requirement, *REQ1.2 "The system shall display message if student's request does not succeed.*" (Protégé-Frames' internal name is requirements_Class64) did not specify its priority value (Figure 66). The above Jess rule detected this incompleteness and returned the error message (Figure 67).

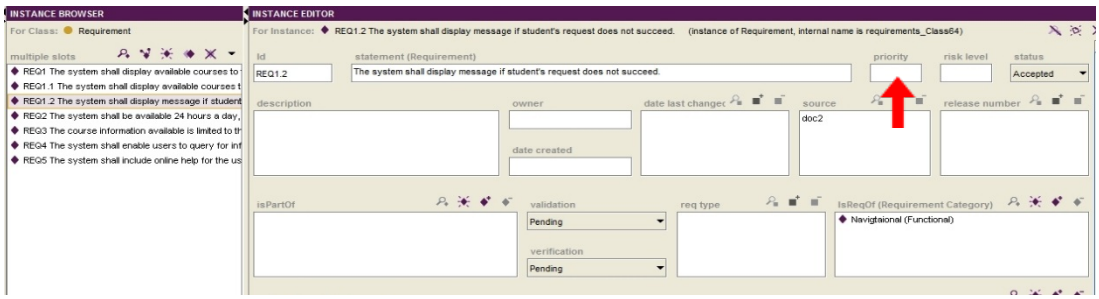


Figure 66. Evaluation – Incomplete Class Attribute Instance

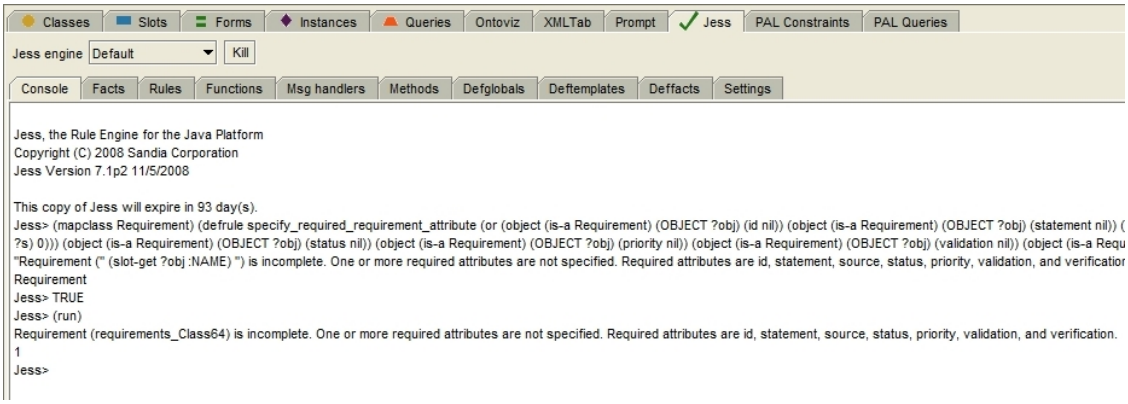


Figure 67. Evaluation – Jess Rule Execution Result for Incompleteness

- **Evaluation Results**

  The goal of PART 2 evaluation was to assess and measure the effectiveness of using ontology approach to validate the research hypotheses H3. The explicitness of the concepts, properties, and relationships specified in the ontology knowledge base enabled reasoning rules and constraints to be performed to check for consistency and

completeness, along with the ability to enforce constraints to help improve the quality requirements of the design model of the ontologies.

The PART 2 evaluation criteria on the ontologies developed in this research were:

(1). To enforce the integrity of the design model in the ontology knowledge base by applying PAL constraints check – **constraint check**.

(2). To check for conflict concepts and valid relations (between concepts) in the ontologies – **consistency check**.

(3). To check whether important concepts (or classes) have been specified and if all required attributes and relationships have been specified in the ontology knowledge base – **completeness check**.

A first round of tests were conducted on the 76 PAL constraints, 60 Jess consistency rules, and 71 Jess completeness rules written for the three ontologies developed in Protégé-Frames editing tool. Table 6 presents the results of the tests. The "No. of Success" column in Table 6 means that the number of rules/constraints that have no violation after executing the rules/constraints. After the first round of tests, necessary corrections on the failed areas were performed on the affected ontologies. For example, the missing entry of information on the priority value of REQ1.2 was added in the Requirement class of the Requirements Ontology. A second round of tests showed a much improved results (Table 7) which contributed to an improved quality of the ontologies. Through an iterative testing, constraints and rules violations will be reduced and subsequently produced a higher quality of design model of the ontologies.

It is important to note that the evaluation was based on the number of *pre-defined* constraints and rules, i.e. 76 pre-defined PAL constraints, 60 pre-defined Jess consistency rules, and 71 pre-defined Jess completeness rules. These constraints and rules are by no means covered the complete set of constrains and rules. There could have more constraints, consistency, and/or completeness rules that could be written and tested on the ontologies or existing constraints and/or rules that could be refined. Nevertheless, the main focus of the PART 2 evaluation (i.e. hypothesis, H3) is to determine whether the quality requirements of the design model of the ontologies in the knowledge base can be improved by detecting constraint violations, inconsistencies and incompleteness of the concepts, properties, and relationships between concepts in the knowledge base through explicit specifications of concepts, properties, and relationships between concepts in the knowledge base, not on the number of constraints, consistency, and completeness rules to test and not to achieve 100% consistency and completeness.

The results of the tests showed that with the pre-defined constraints, consistency, and completeness rules, constraint violations, inconsistencies, and incompleteness were able to be detected, and subsequently errors corrected to improve the quality of the ontologies in the knowledge base. Also, the errors (i.e. detected constraint/rules violations) committed were partly due to human errors, e.g. missed or overlooked entering the required information. Availability of information could also have contributed to the errors, e.g. unknown requirement status, validation, or verification status. Thus an iterative testing of the rules on the ontologies is necessary to resolve

unknown information and correct errors to bring the quality requirements of the design model of the ontologies to a higher level. Table 6 and Table 7 are included in Appendix K.

Table 6. PART 2 Evaluation Tests Results (First Round)

Criteria	No. of Rules	No. of Success	% of Success
PAL Constraints Check	76	56	**74%**
Jess Consistency Check	60	45	**75%**
Jess Completeness Check	71	54	**76%**
TOTAL	207	155	**75%**

Table 7. PART 2 Evaluation Tests Results (Second Round)

Criteria	No. of Rules	No. of Success	% of Success
PAL Constraints Check	76	66	**87%**
Jess Consistency Check	60	51	**85%**
Jess Completeness Check	71	58	**82%**
TOTAL	207	175	**85%**

The results showed that the quality requirements of design model of the ontologies in the knowledge base was improved by using an ontology-based approach through explicit specifications of concepts, properties, and relationships between concepts that allowed for constraint, consistency, and completeness checking in iterative refinement steps. This evaluation successfully validated the research hypothesis H3 as asserted in this dissertation (refer to 8.1 of this chapter and also chapter 1 for hypothesis H3).

CHAPTER 9: CONCLUSIONS AND FUTURE RESEARCH

9.1. Conclusions

The goal of this dissertation is to bring contributions to user requirements analysis and modeling activity in requirements engineering by:

(1). Investigating the application of the concept of persona in requirements engineering activities; specifically, user requirements analysis and modeling to help provide an improvement in the understanding of target users' needs and behaviors, and in refining requirements.

(2). Applying an ontology-based approach to define explicitly the concepts of persona, scenario, task, goals, requirements, and other relevant concepts to help improve the quality requirements of design model of ontologies through utilizing the capability of ontology in checking constraints, completeness, and consistency of concepts in the knowledge base.

In order to achieve the goal, I have investigated the following three research questions (**RQ**):

RQ1 – *How can personas, in the context of scenarios, tasks, goals, and requirements help to understand the target users' needs and behaviors?*

RQ2 – *How can personas, in the context of scenarios, tasks, goals, and requirements, help to refine requirements?*

RQ3 – *How can ontology-based approach of defining an explicit specification of users' knowledge helps to enhance or improve the quality requirements of design model of the ontologies by providing constraints, consistencies, and completeness check?*

In chapter 1, I provided the motivation of this dissertation research, stated the research problem statement, identified the significance of the research problem, and asserted the following three research hypotheses (**H**):

H1 – *The concept of persona and its relationship with scenarios, tasks, goals, and requirements help to understand target users' needs and behaviors.*

H2 – *The concept of persona, in the context of scenarios, tasks, goals, and requirements helps to refine requirements.*

H3 – *Using an ontology-based approach to define an explicit specification of users' knowledge (i.e. personas, scenarios, tasks, goals, and requirements) helps to improve the quality requirements of design model of the ontologies by providing checking of constraints, consistencies, and completeness of concepts, properties, and relationships between concepts in the knowledge base.*

In Chapter 2, I conducted a literature review on Requirements Engineering, Persona-Based, Scenario-Based, Goal-Based, Task-Based, and Ontology-Based Methods explored by other researchers. The literature was reviewed to address the different approaches, gaps and deficiencies in applying persona concept and ontology-based approach in requirements engineering.

In chapter 3 and chapter 4, I described the persona concept and ontologies respectively. I highlighted the benefits of using persona concepts and ontologies and addressed approaches to improve the design model quality requirements of ontology through checking of constraints, completeness, and consistency of concepts specified in the knowledge base.

In chapter 5, I introduced and described the Persona-Driven User Requirements Modeling (Persona-URM) methodological framework proposed in this dissertation research. Within the Persona-URM framework, I introduced and developed two models: (1) Concept Development Process (CDP) model to help guide engineers and developers in the development of the persona concept and the integration of the concept into the requirements engineering process, (2) Ontology-Based Persona-Driven User Requirements (OntoPersonaURM) model with an ontology development process to provide insights and help guide engineers and developers into the construction of ontologies for explicit specifications of the concept of persona in representing users' characteristics, the concepts of goal, scenario, task, requirement, and other relevant concepts.

In chapter 6, I examined the application of the persona concept and ontology approach through a case study on George Mason's PatriotWeb website.

In chapter 7, I presented the design and implementation of the prototype PatriotWeb system which demonstrated the new features and improvements of the current PatriotWeb system on the five chosen scenarios with respect to the eight usability factors used in this research from the viewpoint of the primary persona Linda Rose.

In chapter 8, I validated the three research hypotheses stated in chapter 1 through (1) designing and implementing a prototype PatriotWeb system for the selected five scenarios and applying usability goal measures to evaluate the effectiveness of applying the persona concept, and (2) applying PAL constraints to check for constraint violations, and Jess rules to check for consistency and completeness of concepts, properties, and relationships between concepts in the knowledge base.

9.1.1. Research Contributions

As the concept of persona is a relatively new paradigm in software requirements engineering, in particular, in user requirements analysis and modeling, limited works have been conducted on proposing techniques to investigate persona relationships with goals, viewpoints, scenarios, task, and requirements in requirements engineering. Additionally, as of this writing, there has been no similar work using an ontology approach to represent and integrate the concepts of persona, viewpoint, scenario, task, goal, and requirement in requirements engineering.

This research distinguishes from other previous works conducted by other researchers in two main areas:

(1). It investigates and integrates the relationships of the concepts of persona, viewpoint, scenario, task, goal, and requirement in a unified environment to help engineers and developers achieve a better understanding of the target users' needs and behaviors and helps to identify requirements issues and refine requirements early in the requirements engineering process.

(2). It represents and specifies explicitly the concepts using an ontology-based approach; specifically, the design and construction of three domain independent ontologies: Persona Ontology, Behavioral Goal-Scenario-Task (GST) Ontology, and Requirements Ontology, and applies constraints and inference rules to check for constraints, consistency, and completeness of the concepts, properties, and relationships between concepts in the knowledge base to help improve the design model quality requirements of the ontologies.

The research conducted and the development of the Persona-Driven User Requirements Modeling (Persona-URM) methodological framework offer unique contributions to knowledge in the areas of persona research, user modeling, requirements, and knowledge engineering. This research provided two major contributions stated formally in chapter 1 which are reproduced here:

Contribution 1 – *An improvement in user requirements analysis and modeling by applying the concept of persona, in the context of scenarios, tasks, goals, and requirements, to achieve a better understanding of users' needs and behaviors and help in refining requirements.*

Contribution 2 – *An ontology-based approach to user requirements modeling by building ontologies to define explicit ontological specifications of user's knowledge that will enhance shared understanding of common knowledge and enable for constraints, consistencies, and completeness validation to improve the design model quality requirements of the ontologies in the knowledge base.*

The two major contributions are broken down into several sub-contributions as detailed below:

(1). The development of **Persona-Driven User Requirements Modeling (Persona-URM) methodological framework** (chapter 5) to support the research hypotheses stated in chapter 1.

i). Within the Persona-URM framework, I developed a **Concept Development Process (CDP) model** to help guide developers in the development of the persona concept and the integration of the persona concept into the requirements engineering process. The CDP model consisted of four main processes: 1 – Personas Construction 2 – Viewpoints Identification and Construction 3 – Concepts Modeling 4 – Analysis and Evaluation. The CDP model differentiated from other models developed by other researchers [AOYAMA 07, CASTRO 08, GRUDIN 02, PRUITT 06] in the past in that it integrated the relationships of a number of core concepts including persona, viewpoint, goal, scenario, task, and requirement in a unified environment that helped developers and engineers in better understanding the target users' needs and behaviors.

- In the Personas Construction process, I created a **Persona Definition Document (PDD)** which distinguished from other similar documents created by other researchers [AOYAMA 07, CONSTANTINE 99, MULDER 07, PRUITT 06] in the past in that it encompassed an extensive, if not a complete, set of attributes in a template that defined the persona based on the

information synthesized from the persona's profile. I created the following new attributes in the PDD;

  - ENVIRONMENTS containing specific sub-attributes of Location, Time of Day, Duration, Frequency, and Attributes.
  - ABILITIES with separate Physical and Cognitive sub-attributes.
  - TOOLS, and SOCIAL LEVEL

- In the Viewpoints Identification and Construction process, I created a **Viewpoint Document (VPD)** which distinguished from other similar viewpoint documents developed by other researchers [CHARREL 03, FINKELSETIN 92, KOTONYA 92, IEEE 11 b, PU 09, SOMMERVILLE 97] in the past in that it incorporated a set of core attributes in a template that defined the viewpoint of the persona. I included the following new core attributes in the VPD template: VIEWPOINT NAME, ENVIRONMENT, ROLE, and GOALS. Each VPD was identified uniquely by VIEWPOINT NAME which is composed of three core attributes: PERSONA, ROLE, and ENVIRONMENT.
- In the Analysis and Evaluation process, I developed the following modified matrices (chapter 5 for descriptions of the matrices):
  - **Personas-Viewpoints-Requirements Matrix** was a modification of the Persona-Weighted Features Matrix [PRUITT 06] by including personas' viewpoints in the matrix and modifying the rating scale as: 5 – Critical, 4

– Very Important, 3 – Important 2 – Somewhat important, 1 – Unimportant.

- **Personas-Viewpoints-Scenarios Matrix** was an extended version in [PRUITT 06, MIKKELSON 00] by incorporating viewpoints in the matrix.
- **Scenarios-Usability Goals Matrix** was a slight modification of the Scenario-Goal-Issue Matrix approach in Aoyama's Persona-Scenario-Goal (PSG) methodology [AOYAMA 07] by modifying the usability factors used in [AYOAMA 07] such as removed "Responsiveness" and "Usefulness" factors, added "Accuracy" and "Efficiency" factors with revised descriptions and metrics, and edited the names and descriptions of some of the usability factors in [AOYAMA 07].

ii). Within the Concepts Modeling process (process 3) of the CDP model in the Persona-URM framework, I developed an **Ontology-Based Persona-Driven User Requirements Modeling (OntoPersonaURM) model** that incorporated a five-step iterative ontology development process to help guide developers in the construction of three generic domain independent ontologies: Persona Ontology, Behavioral Goal-Scenario-Task (GST) Ontology, and Requirements Ontology. The OntoPersonaURM model distinguished from other ontological models developed by other researchers [DARDENNE 93, KAIYA 05, KAIYA 06, KOAY 09, YU 93, YU 11a, YU 11b] in that:

a) It provided insights and helped guide ontology engineers and developers in the construction of ontologies for explicit specifications of the concept of persona in representing users' characteristics and its relationships with the concepts of viewpoint, goal, scenario, task, requirement, and other related concepts integrated in a unified ontological environment.

b) The ontologies constructed in the OntoPersonaURM model were developed to be as general as possible yet encompassed a broad set of concepts that were not domain dependent and thus can be applied, modified, extended to other applicable domains, or used as a referenced model.

In the OntoPersonaURM model, I developed the following ontologies, concepts, and process that contributed to this research work:

- The construction and integration of three domain independent ontologies in a unified ontological environment with explicit specifications of concepts and relationships. In particular:
  - The **Persona Ontology** I developed differed from other persona/person ontologies created by other researchers [GOLEMATI 07, KATIFORI 08] in the past in that it encompassed an extensive, if not a complete, set of concepts pertaining to person characteristics and environment. The concepts in the Persona Ontology not only captured the basic characteristics and preferences of a person such as age, gender, name, education, occupation, abilities, expertise, interests and so forth, but also the relationships to the environment in which the person engages in when

using the system. New concepts that I introduced in the Persona Ontology thus included the ENVIRONMENT and its USABILITY PREFERENCE pertaining to the persona.

  - The **Behavioral-GST Ontology** I developed differed from other similar behavioral/goal/scenario models [ALEXANDER 04, ANTON 96, DARDENNE 93, VAN LAMSWEERDE 09] in that it offered unique perspectives in capturing the needs and behaviors of the persona and the system-to-be through the integration of an extensive set of concepts with a new inclusion of the VIEWPOINT concept that played an important role in relating to the Persona Ontology.

- A **revision of the ontology development process** that resulted in a five-step iterative process with an addition of the process of combining the ontologies in the Protégé-Frames ontology environment (Step 5: Combine Ontologies) as one of the five steps in the ontology development process.

(2). A **GMU PatriotWeb case study** (see chapter 6) was studied and presented that demonstrated:

i). The application of persona concept to help engineers and developers gain a better understanding of the target users' needs and behaviors, identify requirements issues, and refine requirements early in the requirements engineering process. Through applying the persona concept via analyzing the Persona Profile Document (PPD) and Persona Definition Document (PDD), and conducting analysis using Personas-Viewpoints-Requirements Matrix and Scenarios-

Usability Goals Matrix, requirements on the current PatriotWeb system with respect to the five selected scenarios were highlighted and refined (chapter 6).

ii). The ontology-based approach for explicit specifications of concepts, their properties, and relationships to help serve as a knowledge repository and foster common understanding of users' needs and behaviors among developers and ontology engineers during the requirements analysis and modeling activity.

(3). The design and implementation of a **prototype of the PatriotWeb system** to support the empirical evaluation in validating the research hypotheses H1 and H2 stated in chapter 1.

I designed and implemented the prototype [PATRIOTWEB b] of the current PatriotWeb system which demonstrated new and improved features based on the chosen five scenarios and the analysis results from conducting the Scenarios-Usability Goals Matrix (chapter 5 and chapter 6) with respect to the eight usability factors chosen for this research from the viewpoint of the primary persona Linda Rose. Descriptions of the new and improved features created in the prototype PatriotWeb system are described in detail in chapter 7. Screenshots of the new and improved features are included in Appendix E.

(4). The **validation of the research hypotheses** (stated in chapter 1).

i). I conducted a **qualitative evaluation** on the PatriotWeb case study (see (2) i) above and also chapter 6) via the design and implementation of a prototype PatriotWeb system [PATRIOTWEB b] to reflect new and improved features (see (3) above and also chapter 7) as a result of applying the persona concept in the

CDP model of the Persona-URM framework. The prototype PatriotWeb system used in the qualitative evaluation helped to support the empirical evaluation (see ii) below) in comparing with the current PatriotWeb system with respect to eight usability factors.

ii). I conducted an **empirical evaluation** (chapter 8) to assess and measure the effectiveness of applying the concept of persona by having participants (i) rated the prototype PatriotWeb system against the current PatriotWeb system on the selected five scenarios with respect to the eight usability factors adopted in this research, and (ii) completed post experiment questionnaire.

The empirical evaluation results showed that with the application of persona concept, a better understanding of the target users' needs and behaviors was achieved, requirements issues were identified and resolved, and requirements were refined early in the requirements engineering process that helped to improve the usability of the system-to-be (PatriotWeb system).

iii). I conducted a **query and rule-based evaluation** (chapter 8) to assess and measure the effectiveness of using ontology-based approach in improving the quality requirements of the design model of the developed ontologies via performing tests on constraints, consistency, and completeness of concepts in the knowledge based. I wrote 76 PAL constraints, 60 Jess consistency rules, and 71 Jess completeness rules. I conducted two tests via executing these constraints and rules on the Protégé-Frames ontology editing tool via PAL Constraint Tab and

Jess plugins to check for any violations of constraints, consistency, and completeness of concepts in the knowledge based of the three ontologies.

The evaluation results showed that the quality requirements of design model of the three ontologies in the knowledge base was improved by using an ontology-based approach through explicit specifications of concepts, properties, and relationships between concepts that allowed for checking of constraints, consistency, and completeness in an iterative refinement steps.

9.1.2. Limitations/Challenges

This research was conducted with some limitations and challenges; in particular, the evaluation findings and results (chapter 7) were limited due to the fact that:

(1). The empirical evaluation (on the effectiveness of the persona concept) and the survey responses were based on a total of 68 participants from two undergraduate classes in engineering and science disciplines respectively. Although the survey results (chapter 8) showed positively that using persona concept produced high scores on the usability goals on the prototype PatriotWeb, more participants from different academic disciplines and/or class levels may be included in future evaluations to allow more responses for assessing and measuring the effectiveness of applying the concept of persona. The challenges will be the commitment and support from a large pool of participants from varied academic disciplines.

(2). The research evaluations to validate the three stated hypotheses (chapter 1) were assessed based on one case study (chapter 6), i.e. George Mason University

PatriotWeb system. Furthermore, only selected five scenarios of the case study were chosen to evaluate the effectiveness of persona concept to validate research hypotheses, H1 and H2 (chapter 1). A large-scale case study involving more scenarios may be performed in the future to achieve further analysis and results. In addition, more case studies in other domains applicable to the research may be investigated. The challenges of conducting a large-scale case study and/or multiple case studies in different domains may involve the availability of resources and time.

(3). The PART 2 evaluation on the effectiveness of the ontology approach to validate research hypothesis H3 (chapter 1) was based on the number of *pre-defined* constraints and rules, i.e. a total of 76 pre-defined PAL constraints, 60 pre-defined Jess consistency rules, and 71 pre-defined Jess completeness rules. Although the evaluation tests results showed positively that the application of constraints, consistency, and completeness tests were able to detect constraint violations, inconsistencies, and incompleteness of the concepts, properties, and relationships between concepts specified in the ontologies in the knowledge base and consequently were able to help improve the quality requirements of design model of the ontologies through an iterative process of testing of the constraints and rules, more PAL constraints and Jess rules for consistency and completeness tests may be written in the future and tested to cover more concepts, properties, and relationships in the ontologies.

9.2. Future Research

The development of the Persona-URM methodological framework; specifically, the Concept Development Process (CDP) model of applying and integrating the concept of persona and the Ontology-Based Persona-Driven User Requirements Modeling (OntoPersonaURM) model of constructing the ontologies using an ontology-based approach have been demonstrated through the GMU PatriotWeb case study (chapter 6) and the implementation of the prototype PatriotWeb system (chapter 7) to bring contributions to user modeling and analysis in requirements engineering. The effectiveness of the persona concept and the ontology-based approach have been evaluated (in chapter 8) to validate the stated three hypotheses (chapter 1). However, beyond the three validated hypotheses in this dissertation, there are other hypotheses that can be hypothesized and validated with respect to this research. There are also many other aspects that can be or should be improved and need further research and investigations. These further improvements are highlighted below:

(1). Further Refinement of the Persona-URM Framework

i). CDP Model

The objective of the CDP model is to help engineers and developers in the construction and application of personas, in the context of the concepts of scenario, task, goal, requirement, and other pertaining concepts. The CDP model also helped to enrich the requirements engineering activities by incorporating the proposed CDP processes into the requirements engineering activities: requirements

elicitation, requirements analysis, requirements specification, and requirements validation. By empowering the concept of persona into the requirements engineering activities, a better understanding of the target users' needs and behaviors can be realized early in the requirements engineering process, thus allowing engineers to identify any requirements issues. Future work on the CDP model may include implementing software tool support for the development of personas and viewpoints documents, e.g. web forms with backend database for creating and updating of Persona Profile Document (PPD), Persona Definition Document (PDD), and Viewpoint Document (VPD), and relating information from the personas documents to information of scenarios, tasks, goals, requirements, and other relevant concepts.

ii). OntoPersonaURM Model

The objective of the OntoPersonaURM model is to help engineers and developers in the development of domain independent generic ontologies that can be applied to other applicable domains, via the five-step ontology development process employed in the model. The OntoPersonaURM model is continuously developing and refining. Future work on the OntoPersonaURM model may include further refinements in the specifications of the concepts and relationships in the Protégé editing environment.

(2). Extending the Case Study

The research evaluations conducted in this dissertation were based on the study of George Mason University PatriotWeb system (chapter 6) and the implementation of

the prototype PatriotWeb system (chapter 7) based on the five selected scenarios. More scenarios of the PatriotWeb case study may be conducted in the future to further evaluate the research hypotheses to achieve more thorough analysis and results. Furthermore, the Persona-URM framework (the CDP model and the OntoPersonaURM model) may be applied or extended to other applicable domains and thus more case studies may be conducted and studied with respect to the Persona-URM framework.

(3). Further improvement on Prototype PatriotWeb System

The prototype PatriotWeb system implemented to support the empirical evaluation in validating the research hypotheses H1 and H2 (see hypotheses in chapter 1) was designed based on the five scenarios chosen for the evaluation. New and improved features (chapter 7) on the prototype PatriotWeb system were thus limited to these five scenarios. More scenarios may be needed to be analyzed and evaluated in the future to further improve on the prototype PatriotWeb system.

(4). Further Evaluation of the Persona-URM Framework

i). Effectiveness of Persona Concept

In this dissertation, an empirical evaluation was conducted from a total of 68 participants from two undergraduate classes in the engineering and science disciplines respectively at George Mason University. The objective of the empirical evaluation was to assess and measure the effectiveness of applying the concept of persona in validating research hypotheses, H1 and H2 (chapter 1) via five selected scenarios from the case study from the viewpoint of a primary

persona Linda Rose. A large-scale case study and a more extensive evaluation with respect to the number of participants, academic disciplines and/or class levels of the participants and additional personas (primary and/or secondary) will be necessary in order to obtain more data for a more in-depth analysis to further evaluate the effectiveness of applying the persona concept. Furthermore, the effectiveness of the entire framework in requirements engineering process may be evaluated by carrying out more extensive evaluation procedures.

ii). Effectiveness of Ontology-Based Approach

In this dissertation, the developed ontologies were tested using the GMU PatriotWeb case study. The research hypothesis, H3 (chapter 1) was validated (chapter 8) through conducting tests by:

a) Writing and executing PAL constraints using PAL plugin to Protégé-Frames modeling environment to check constraint violations in the knowledge base.

b) Writing and executing Jess rules to check for consistency and completeness of concepts, properties, and relationships between concepts specified in the knowledge base.

A more extensive checking of constraints, consistency, and completeness may be conducted on the developed ontologies to encompass more PAL constraints and Jess rules to cover more concepts, properties, and relationships between concepts in the knowledge base.

APPENDICES

APPENDIX A: LIST OF RESEARCH PUBLICATIONS

Sim, W. W., Brouse, S. P. (2014). Empowering Requirements Engineering Activities with Personas. In: *12th Annual Conference on Systems Engineering Research (CSER 2014)*, *Procedia Computer Science*, *28*, 237–246. http://dx.doi.org/10.1016/j.procs.2014.03.030

Sim, W. W., Brouse, S. P. (2014). Towards an Ontology-Based Persona-Driven Requirements and Knowledge Engineering. In: *Conference on Complex Adaptive Systems (2014)*, *Procedia Computer Science*, *36*, 314–321. http://dx.doi.org/10.1016/j.procs.2014.09.099

Sim, W. W., Brouse, S. P. (2015). Developing Ontologies and Persona to Support and Enhance Requirements Engineering Activities. In: *13th Annual Conference on Systems Engineering Research (CSER 2015), Procedia Computer Science, 44,* 275-284. http://dx.doi.org/10.1016/j.procs.2015.03.060

Sim, W. W., Brouse, S. P. (2015). Persona-Driven User Requirements Modeling (Persona-URM) Methodology to Support Requirements and Knowledge Engineering. In: *Conference on Complex Adaptive Systems (2015)* - <TBD>

APPENDIX B: CONCEPT DEVELOPMENT PROCESS (CDP) TEMPLATES

{persona name} {title, tagline}	{Photo}	{persona type: primary, secondary}	PRIORITY

DEMOGRAPHIC PROFILE

NAME				GENDER	AGE	MARITAL STATUS	HIGHEST EDUCATION LEVEL	OCCUPATIONS				SOCIAL LEVEL
First	Last	Nickname	Title					Job Title	Work Status	Length of Employment	Salary	

LANGUAGE PROFICIENCY			INTERESTS	ABILITIES {average, medium, high}									
				Physical				Cognitive					
Listening	Speaking	Writing		Motor	Sensory-Hearing	Sensory-Speech	Sensory-Visual	Attention	Memory	Problem-Solving	Reading-Comp.	Visual-Comp.	Spatial

KNOWLEDGE & SKILLS PROFILE

COMPUTER
WEB
DOMAIN

INTERACTION PROFILE

ROLES	GOALS	CONCERNS

ENVIRONMENTS				
Location	Time of Day	Duration	Frequency (low: 0-5 times/wk, medium: 6-10 times/wk, high: >10 times/wk)	Attitudes (eager, skeptical, cautious, indifferent, impatient, resentful, curious, trusting, others)

USAGE PATTERNS (scenarios titles)	TOOLS	USABILITY PREFERENCES (accuracy, attractiveness, efficiency, learnability, reliability, comprehensibility, clarity, rememberability)

{<Persona, Role, Environment)>}	VIEWPOINT (VP) NAME

SOURCES	STAKEHODLERS	HISTORY (VP-version, date, author, changes)	ROLE

ENVIRONMENT (location, time of day, duration, tools, etc)	

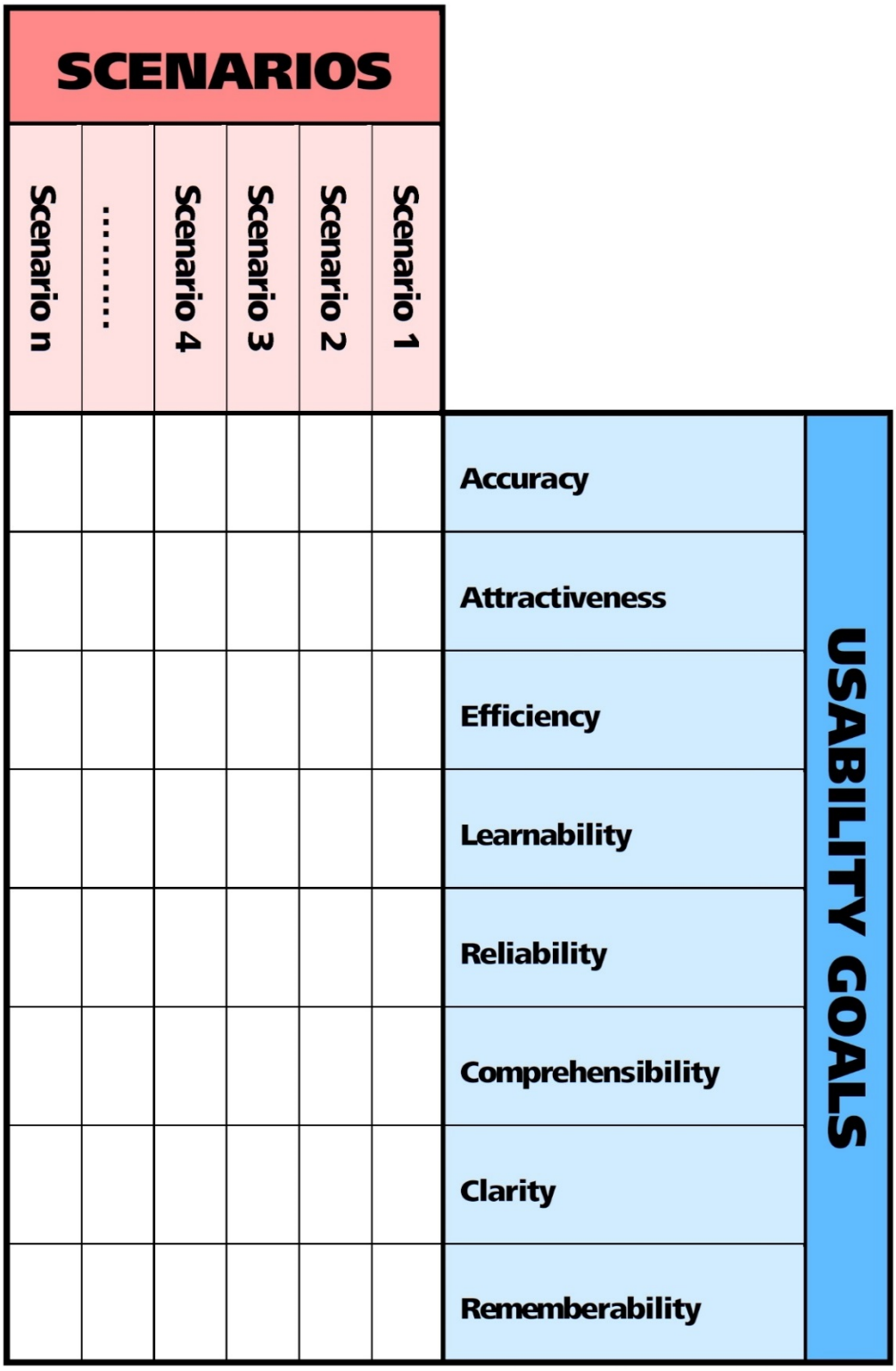
GOALS	SCENARIOS, TASKS	
	Scenarios (titles)	Tasks (titles)
CONCERNS		

REQUIREMENTS	
Functional	
Non-Functional	

MODEDLING		
Modeling Techniques	Concepts Representation	Models, Ontologies

Rating scale: 5 – Critical 4 – Very Important 3 – Important 2 – Somewhat important 1 – Unimportant		PERSONAS	Persona 1			Persona ..			Persona n			WEIGHTED SUM
		WEIGHTS										
		VIEWPOINTS (VPB)	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	
REQUIREMENTS	REQ 1											
	REQ 2											
	REQ 3											
	REQ ..											
	REQ ..											
	REQ n											

PERSONAS		Persona 1			Persona ...			Persona n			SCENARIO FREQUENCY COUNT
VIEWPOINTS (VPB)		VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	VPB 1: <Persona, Role, Environment>	VPB ...: <Persona, Role, Environment>	VPB n: <Persona, Role, Environment>	
SCENARIOS (titles)	Scen 1 title 1										
	Scen 2 title 2										
	Scen 3 title 3										
	Scen .. title ..										
	Scen .. Title ..										
	Scen .. title ..										
	Scen n title n										
SCENARIO COUNT (per Persona-Viewpoint)											


SCENARIOS
Scenario 1
Scenario 2
Scenario 3
Scenario 4
.....
Scenario n
USABILITY GOALS
Accuracy
Attractiveness
Efficiency
Learnability
Reliability
Comprehensibility
Clarity
Rememberability

		TASKS	REQUIREMENTS			
			REQ 1	RRQ 2	.....	REQ n
SCENARIOS	Scenario 1	Task 1				
		Task 2				
		.....				
		Task n				
	Scenario 2	Task 1				
		Task 2				
		.....				
		Task n				
	.....	Task 1				
		Task 2				
		.....				
		Task n				
	Scenario n	Task 1				
		Task 2				
		.....				
		Task n				

APPENDIX C: PROTÉGÉ-FRAMES ONTOLOGY CLASS DESCRIPTIONS

C.1. Persona Ontology Class Descriptions

Persona Ontology Class Hierarchy

Class: Ability

The abilities (and/or disabilities) of a person, such as cognitive and physical.

Super-classes: :THING

Sub-classes: Cognitive
Physical

In domain of: isAbilityOf

In range of: hasAbilityOf

Slot Name	Type	Cardinality
ability_level	{Low, Average, High}	0:1
id	String	1:1
isAbilityOf	Persona	0:*

Class: Cognitive

The cognitive abilities (and/or disabilities) of a person. Types of cognitive abilities include attention, memory, problem-solving, reading comprehension, visual comprehension, and spatial.

Super-classes: Ability

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
ability_level	{Low, Average, High}	0:1
cognitive_type	{Attention, Memory, Problem-Solving, Reading-Comprehension, Visual-Comprehension, Spatial, Other}	0:1
cognitive_type_other	String	0:1
id	String	1:1
isAbilityOf	Persona	0:*

Class: Computer

The computer knowledge level of a person. The computer knowledge includes general computer knowledge such as general operating of desktop/laptop computer.

Super-classes: Knowledge

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
computer_knowledge	String	0:1
id	String	1:1
isKnowledgeOf	Persona	0:*
knowledge_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1

Class: Concern

Captures the interests or concerns of a persona in the context of a particular environment the persona is engaging in. Concerns reflect organizational goals, objectives, and priorities that are important to the stakeholders and addressed by the viewpoints.

Super-classes: :THING

Sub-classes: None

In domain of: isConcernOf, relatesTo, addressedBy, isImportantTo

In range of: hasConcern, presents, addresses, raises

Slot Name	Type	Cardinality
addressedBy	Viewpoint	1:*
description	String	0:1
id	String	1:1
isConcernOf	Persona	0:*
isImportantTo	Stakeholder	0:*
name	String	1:1

rating	{Very-Important, Important, Somewhat-Important, Not-At-All-Important, Not-Sure-Not-Applicable}	0:1
relatesTo	Environment	0:*

Class: Domain

The domain knowledge level of a person. The domain knowledge refers to application-specific knowledge.

Super-classes: Knowledge

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
domain_knowledge	String	0:1
id	String	1:1
isKnowledgeOf	Persona	0:*
knowledge_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1

Class: Education

The highest educational level of a person.

Super-classes: :THING

Sub-classes: None

In domain of: isHighestEducationOf

In range of: hasHighestEducation

Slot Name	Type	Cardinality
degree_title	String	0:1

degree_year	String	0:1
highest_education_level	{Vocational, GED, Associate, Bachelor's, Master's, Doctoral, Professional}	1:1
id	String	1:1
isHighestEducationOf	Persona	1:*

Class: Environment

Captures information of the environment the persona is engaged in while using the system, e.g. location, time of day, frequency. The environment concept relates to the persona's viewpoint which specifies the concerns of the persona, the scenarios the persona have and the tasks the persona performed to achieve the goals of the persona interacting in a particular environment. The Environment class thus captures the (partial) dynamic aspect of a persona, such as the location, time, and frequency the persona is interacting in a particular environment.

A persona engages in two different environments to perform some task may have different scenarios, concerns, goals, requirements, and so forth. For example, a public outdoor place such as Starbucks during work lunch hour versus a private indoor place such as home during flexible time may have differences in terms of availability of internet connection, privacy, security concerns, time constraint, and so forth. Some of the main attributes and their specifications are:

Super-classes: :THING

Sub-classes: None

In domain of: Presents, isEnvironmentOf, engagedBy, hasUsabilityPref, particpatedBy

In range of: relatesTo, hasEnvironment, engagesIn, isUsabilityPrefOf, participatesIn

Slot Name	Type	Cardinality
activity_level	{Very-Active, Active, Passive, Inactive}	0:1
description	String	0:1
engagedBy	Persona	1:*
frequency	{Low, Medium, High} Low (0-5 times/week), Medium (6-10 times/week), High (>10 times/week)	0:1
hasUsabilityPref	UsabilityPref	0:*
id	String	1:1

isEnvironmentOf	Viewpoint	1:*
location	{Home, Work-Office, Work-Public, Public, Other}	0:1
name	String	1:1
participatedBy	Role	0:*
persona_attitude	{Eager, Skeptical, Cautious, Indifferent, Impatient, Resentful, Curious, Trusting, Varies}	0:*
presents	Concern	0:*
time_of_day	{Morning, Afternoon, Evening, Late-Night, Early-Morning, Afternoon, Mid-Day}	0:1
time_spent	String	0:1

Class: Interest

A person's hobbies, both work and non-work interests.

The Interest class is associated with the InterestCategory class via aggregation, i.e. an InterestCategory class is an aggregation of an Interest class. As it is common that there are various interest names that may belong to a same interest category, we chose to place a one-to-many relationship between InterestCategory and Interest classes, i.e. an interest category hasInterestPart one or more interests. For example, an interest category "*Entertainment*" hasInterestPart "*Listening music*", "*Playing guitar*". For Simplicity, we chose to specify that an interest is part of one and only one interest category. For example, an interest "*Listening music*" isInterestPartOf "*Entertainment*". The cardinalities between Interest and InterestCategory classes (and vice versa) is a design choice and thus may be modified by the ontology designer.

Super-classes: :THING

Sub-classes: None

In domain of: isInterestOf, isInterestPartOf

In range of: hasInterest, hasInterestPart

Slot Name	Type	Cardinality
id	String	1:1
isInterestOf	Persona	0:*
isInterestPartOf	InterestCategory	0:1
name	String	1:1

Class: InterestCategory

The categorization of a person's interests.

The InterestCategory class has a simple String type, for examples, "*Internet*", "*Entertainment*", "*Business & Finance*", "*Books & Literature*", "*Computers*", and so forth. An interest category may be part of another interest category, for example, an interest category "*TV & Movies*" may be considered part of an interest category "Entertainment". Thus, the InterestCategory class is related to itself via aggregation relationship. An interest category hasSubCategory zero or more interest category; inversely, an interest category isSubCategoryOf one and only one interest category. For simplicity, an interest category is part of one and only one interest category.

Super-classes: :THING

Sub-classes: None

In domain of: hasInterestPart, isSubCategoryOf

In range of: isInterestPartOf, hasSubCategory

Slot Name	Type	Cardinality
category	String	0:1
hasInterestPart	Interest	0:*
hasSubCategory	InterestCategory	0:*
id	String	1:1
isSubCategoryOf	InterestCategory	0:1

Class: Knowledge

The knowledge of a person. The Knowledge class is categorized into three subclasses: Computer, Domain, and Web. The computer knowledge includes general computer knowledge such as general operating of desktop/laptop computer. The web knowledge refers to general web browsing knowledge such as ability to recognize and use URL addresses and hyperlinks, ability to perform searches using common search engines, etc. The domain knowledge refers to application-specific knowledge.

Super-classes: :THING

Sub-classes: Computer
Domain
Web

In domain of: isKnowledgeOf

In range of: hasKnowledge

Slot Name	Type	Cardinality
id	String	1:1
isKnowledgeOf	Persona	0:*
knowledge_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1

Class: LanguageProficiency

The proficiency level of the language(s) a person possesses.

Super-classes: :THING

Sub-classes: Listening
Speaking
Writing

In domain of: isLanguageProficiencyOf

In range of: hasLanguageProficiency

Slot Name	Type	Cardinality
id	String	1:1
isLanguageProficiencyOf	Persona	0:*
proficiency_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1

Class: Listening

The language proficiency level of a person with respect to listening proficiency.

Super-classes: LanguageProficiency

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
isLanguageProficiencyOf	Persona	0:*
listening_language	String	0:1
proficiency_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1

Class: Name

The name of a person.

Super-classes: :THING

Sub-classes: None

In domain of: isPersonaNameOf

In range of: hasPersonaName

Slot Name	Type	Cardinality
first_name	String	0:1
id	String	1:1
isPersonaNameOf	Persona	1:1
last_name	String	0:1
middle_name	String	0:1
nickname	String	0:*
title	String	0:*

Class: Occupation

The occupation of a person.

Super-classes: :THING

Sub-classes: None

In domain of: isOccupationOf

In range of: hasOccupation

Slot Name	Type	Cardinality
id	String	1:1
isOccupationOf	Persona	0:*
length_of_employment	{<1year, 1-3years, 3-6years, 6-10years, >10years}	0:1
occupation_group	{Student, Teaching, Management, Administrative, Technical, Non-technical, None}	0:1
position	String	0:1
salary_range	{<$9,999, $10,000-$29,999, $30,00-$49,999, $50,000-$79,999, $80,00-$99,999, >$100,000,}	0:1
work_hours	Integer	0:1
work_status	{Full-Time, Part-Time, Self-Employed}	0:1

Class: Persona

The general characteristics of a person. A simple type attribute, personal_title, is an identification of a persona of the form {*name*, *role or job title*}. For example: *John Hurt, The Casual Buyer*.

Super-classes: :THING

Sub-classes: None

In domain of: hasPersonaName, hasEducation, hasOccupation, hasLanguageProficiency, hasAbility, hasInterest, hasKnowledge, personifies, engagesIn, hasConcern, isPersonaOf

In range of: isPersonaNameOf, isHighestEducationOf, isOccupationOf, isLanguageProficiencyOf, isAbilityOf, isInterestOf, isKnowledgeOf, personaifiedBy, engagedBy, isConcernOf, hasPersona

Slot Name	Type	Cardinality
age	{Young:18-24, Young:25-34, Middle:35-44, Middle:45-54, Older:55-64, Elderly:>65}	0:1
engagesIn	Environment	1:*
gender	{Male, Female}	0:1
has Ability	Ability	0:*

hasConcern	Concern	0:*
hasHighestEducation	Education	0:1
hasInterest	Interest	0:*
hasKnowledge	Knowledge	0:*
hasLanguageProficiency	LanguageProficiency	0:*
hasOccupation	Occupation	0:*
hasPersonaName	Name	1:1
id	String	1:1
isPersonaOf	Viewpoint	1:*
marital_status	{Single, Married, Divorced, Separated, Widowed}	0:1
persona_title	String	0:1
persona_type	{Primary, Secondary, Served, Customer}	0:1
personifies	Role	1:*
social_level	{Upper-Class, Middle-Class, Working-Class}	0:1

Class: Physical

The physical abilities (and/or disabilities) of a person. Types of physical abilities include motor and sensory such as visual, hearing, speech.

Super-classes: Ability

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
ability_level	{Low, Average, High}	0:1
id	String	1:1
isAbilityOf	Persona	0:*
physical_type	{Motor, Sensory-Visual, Sensory-Hearing, Sensory-Speech, Other}	0:1
physical_type_other	String	0:1

Class: Role

A role played by a person when interacting in a particular environment at a particular time. Examples of roles in an online self-service web application for a university may be: student, faculty, staff.

A role is personified by one or more Personas. A Persona personifies more or more roles. A role is played by zero or more actor, e.g. a device, system, an organization, or a person. A device, system, or an organization does not have persona, but is an actor that plays a particular role. This role is not a persona, and thus a 0 cardinality is assigned.

Super-classes: :THING

Sub-classes: None

In domain of: personifiedBy, participatesIn, isRoleOf, playedBy

In range of: Personifies, participatedBy, hasRole, plays

Slot Name	Type	Cardinality
id	String	1:1
isRoleOf	Viewpoint	1:*
name	String	1:1
participatesIn	Environment	0:*
playedBy	Actor	0:*
personifiedBy	Persona	1:*
role_type	{Primary, Secondary}	0:1

Class: Speaking

The language proficiency level of a person with respect to speaking proficiency.

Super-classes: LanguageProficiency

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1

isLanguageProficiencyOf	Persona	0:*
proficiency_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1
speaking_language	String	0:1

Class: UsabilityPref

Usability is concerned with the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments [ISO 06]. Thus, usability relates to the ease with which a person can achieve his or her goals while interacting with a system in a particular environment. Usability factors include accuracy, attractiveness, efficiency, learnability, reliability, comprehensibility, clarity, rememberability, and so forth.

Super-classes: :THING

Sub-classes: None

In domain of: isUsabilityPrefOf

In range of: hasUsabilityPref

Slot Name	Type	Cardinality
description	String	0:1
id	String	1:1
isUsabilityOf	Environment	0:*
name	String	1:1
rating	{Critical, Very-Important, important, Somewhat-Important, Not-At-All-Important, Not-Sure-Not-Applicable}	0:1
usability_type	{Accuracy, Attractiveness, Clarity, Comprehensibility, Efficiency, Learnability, Reliability, Rememberability, Other}	0:1

Class: Web

The web knowledge level of a person. The web knowledge refers to general web browsing knowledge such as ability to recognize and use URL addresses and hyperlinks,

ability to perform searches using common search engines, and so forth.

Super-classes: Knowledge

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
isKnowledgeOf	Persona	0:*
knowledge_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1
web_knowledge	String	0:1

Class: Writing

The language proficiency level of a person with respect to writing proficiency.

Super-classes: LanguageProficiency

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
isLanguageProficiencyOf	Persona	0:*
proficiency_level	{Poor, Fair, Good, Excellent, Not-Sure-Not-Applicable}	0:1
writing_language	String	0:1

C.2. Behavioral-GST Ontology Class Descriptions

Behavioral-GST Ontology Class Hierarchy

Class: Action

Defines operations for the execution of certain tasks in a particular situation. An action may be an atomic action or flows of actions.

Super-classes: :THING

Sub-classes: None

In domain of: performedBy, isActionOf

In range of: performes, hasAction

Slot Name	Type	Cardinality
description	String	0:1
id	String	1:1
isActionOf	Task	1:*
name	String	1:1
performedBy	Actor	1:*

Class: Actor

Defines the role played by a person, an organization, a system, or a device that interacts with the system-to-be.

Super-classes: :THING

Sub-classes: Device
Organization
Person
System

In domain of: performes, uses, plays, represents

In range of: performedBy, usedBy, playedBy, representedBy

Slot Name	Type	Cardinality
id	String	1:1
name	String	1:1
plays	Role	0:*
performs	Action	1:*
represents	Stakeholder	0:1

uses	Tool	0:*

Class: BusinessGoal

The goal(s) of a business. Generally, a business goal is an unambiguous definition of the organizational objectives or targets that need to be achieved. Examples of business goals: "*Increase profit by 10% by next quarter*.", "*Reduce staff turnover by 15%.*"

Super-classes: GoalType

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
isGoalTypeOf	Goal	1:*

Class: Constraint

Defines requirements (non-functional) that must be met in order to achieve a goal. A constraint imposes certain conditions on the achievement of a goal: a pre-condition defines what conditions imposed on input variables must be met before execution of an operation, a post-condition defines what condition imposed on output variables or what the expected outcome is after execution of an operation, and an invariant defines mathematical operators, formulae, criterion, or values (as in attributes) that must always hold, i.e. cannot be changed.

A constraint may be categorized as one of the constraint types: Business constraint, Design constraint, Environmental constraint, System constraint, or other constraints. Examples of common business constrains may include time, financial concerns, management and policies. An example of design constraint may include the design of an existing database that the system-to-be needs to depend on. Examples of common environmental constraints are any limitations on strategy options due to political, external, competition, social requirements and expectations, cultural or economic factors, technological or legal requirements. An example of system constraint may include software or hardware limitation.

Super-classes: :THING

Sub-classes: None

In domain of: isConstraintOf

In range of: hasConstraint

Slot Name	Type	Cardinality
constraint_type	{Business, Design, Environmental, System, Other}	0:*
constraint_type_other	String	0:1
critical_level	Float	0:1
description	String	0:1
id	String	1:1
isConstraintOf	Scenario, Task, Goal, Viewpoint	0:*
name	String	1:1

Class: Device

A tool, an equipment, or a technique used to perform a task.

Super-classes: Actor

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
name	String	1:1
plays	Role	0:*
performs	Action	1:*
represents	Stakeholder	0:1
uses	Tool	0:*

Class: ExceptionScenario

A scenario that describes the flow of an abnormal situation or undesired situation which

interferes with progress towards accomplishing an objective. In this context, an exception scenario includes abnormal event that may not fulfill its objective or alternative event that may fulfil its objective.

Super-classes: Scenario

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
author	String	0:1
date_created	String	0:1
description	String	0:1
extension_condition	String	0:*
hasConstraint	Constraint	0:*
hasEpisode	Scenario	0:*
hasTask	Task	1:*
id	String	1:1
isEpisodeOf	Scenario	0:*
isScenarioOf	Viewpoint	1:*
name	String	1:1
operationalizes	Requirement	1:*
postconditions	String	0:*
preconditions	String	0:*
release_number	String	0:*
status	{Active, In-Active, Terminated}	0:1
trigger	String	0:1

Class: Goal

A goal is a high level objective of the business organization, system, or person. The Goal class provides a general classification of the properties of a goal and defines attributes that describe the properties of a generic goal. A goal may belong to a particular goal type such as a person goal, a system goal, or a business goal. The relationship between a goal and its type is captured in the GoalType class.

Goals are composite objects, i.e. a goal may relate with other goals. For examples, a goal may decompose into sub-goals through goal refinements, i.e. an AND-refinesInto

(inversely, AND-refinesFrom) and an OR-refinesInto (inversely, OR-refinesFrom). Both types of refinements are associated with other goals via an aggregation relationship, i.e. a goal is an aggregation of other goals via AND-refinesInto or OR-refinesInto relationship.

An AND-refinesInto relationship describes an "AND" decomposition (or refinement) of a goal (parent) into a set of sub-goals (child). If a goal is AND-refinesInto a set of sub-goals, then satisfying ALL sub-goals in the decomposition (or refinement) is sufficient for satisfying the parent goal.

The OR-refinesInto relationship describes an "OR" decomposition (or refinement) of a goal (parent) into a set of sub-goals (child), i.e. alternative goal decomposition (or refinement). If a goal is OR-refinesInto a set of sub-goals, then satisfying ANY of the decompositions (or refinements) is sufficient for satisfying the parent goal.

A goal may be categorized as functional goal or non-functional goal. Functional goals are goals pertaining to services that the system is expected to deliver whereas non-functional goals are goals related to expected system qualities such as performance, usability, security, safety, and so forth.

The priority value of a goal represents a quantitative level of the priority of a goal in comparison with other goals. In the Behavioral-GST Ontology, the priority value is defined as a Float type which takes real values between 0 and 1, with up to 2 decimal places for precision. For example, a goal of 0.80 priority value is viewed more important compared to another goal of 0.45 priority value. In situation where a goal is not met, e.g. conflict with another goal, then the priority value may be used to determine which goal is more important. The priority value of a goal may also be defined as an integer type which takes integer value from 1 to 10 or as an enumerate type such as high, medium, or low. The ontology engineer has the freedom to choose which definition is appropriate based on the application domain. The priority value plays an important role when a goal is refined into sub-goals of multiple branches. In this case, the priority of each branch can be computed by multiplying the priority value of each sub-goal and their parent goal.

Super-classes:	:THING
Sub-classes:	None
In domain of:	AND-refinesInto, OR-refinesInto, conflictsWith, supports, hasObstacle, hasConstraint, isGoalOf, hasGoalType, hasStakholder, metBy
In range of:	AND-refinesFrom, OR-refinesFrom, conflictsWith, supportedBy, isObstacleOf, isConstraintOf, hasGoal, isGoaltypeOf, isStakeholderOf, meets

Slot Name	Type	Cardinality
AND-refinesFrom	Goal	0:*
AND-refinesInto	Goal	1:*
conflictsWith	Goal	0:*
date_created	String	0:1
description	String	0:1
goal_category	{Functional, Non- Functional}	0:1
hasConstraint	Constraint	0:*
hasGoalType	GoalType	1:*
hasObstacle	Obstacle	0:*
hasStakeholder	Stakeholder	1:*
id	String	1:1
isGoalOf	Viewpoint	1:*
metBy	Requirement	1:*
name	String	1:1
OR-refinesFrom	Goal	0:*
OR-refinesInto	Goal	0:*
ownership	String	0:*
priority	Float	0:1
status	{Met, Not- Met, Partially- Met, Withdrawn, Unknown}	0:1
supportedBy	Goal	0:*
supports	Goal	0:*

Class: GoalType

Defines the types of goal: person goal, business goal, and system goal. Generally, a person goal is an individual person's objectives or interests in using the system, for example: "*Find the best class schedules that do not conflict with my work hours*." A business goal is an unambiguous definition of the organizational objectives or targets that need to be achieved, for example: "*Increase profit by 10% by next quarter*." A system goal is a goal of the system as a whole, i.e. the requirements on the services that the system provides, for example: "*Ensure that system does not need more than three hours of weekly maintenance*."

Super-classes: :THING

Sub-classes: BusinessGoal
PersonGoal

	SystemGoal
In domain of:	isGoalTypeOf
In range of:	hasGoalType

Slot Name	Type	Cardinality
id	String	1:1
isGoalTypeOf	Goal	1:*

Class: NormalScenario

The basic flow of events to accomplish an objective. It captures a sequence of interaction where everything proceeds as normally expected.

Super-classes:	Scenario
Sub-classes:	None
In domain of:	None
In range of:	None

Slot Name	Type	Cardinality
author	String	0:1
date_created	String	0:1
description	String	0:1
hasConstraint	Constraint	0:*
hasEpisode	Scenario	0:*
hasTask	Task	1:*
id	String	1:1
isEpisodeOf	Scenario	0:*
isScenarioOf	Viewpoint	1:*
name	String	1:1
operationalizes	Requirement	1:*
postconditions	String	0:*
preconditions	String	0:*
release_number	String	0:*
status	{Active, In-Active, Terminated}	0:1
trigger	String	0:1

Class: Obstacle

Defines system behaviors or other goals that prevent the achievement of a given goal with respect to a particular environment.

Super-classes: :THING

Sub-classes: None

In domain of: isObstacleOf

In range of: hasObstacle

Slot Name	Type	Cardinality
critical_level	Float	0:1
description	String	0:1
id	String	1:1
isObstacleOf	Viewpoint, Goal	0:*
name	String	1:1
obstacle_type	{Hazard, Threat, Misinformation, Inaccuracy, Dissatisfaction, Unusability, Other}	0:*
obstacle_type_other	String	0:1

Class: Organization

A group of people who work together in a structured way with a shared purpose to meet a need or to pursue collective goals. Examples of an organization: a union, a corporation, a neighborhood association, or a charity.

Super-classes: Actor

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
name	String	1:1
plays	Role	0:*

performs	Action	1:*
represents	Stakeholder	0:1
uses	Tool	0:*

Class: Person

A person represents an individual, as a human being.

Super-classes: Actor

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
name	String	1:1
plays	Role	0:*
performs	Action	1:*
represents	Stakeholder	0:1
uses	Tool	0:*

Class: PersonGoal

The goal(s) of a person. A person goal is an individual person's objectives or interests in using the system. Example of a person goal: "*Find the best class schedules that do not conflict with my work hours*."

Super-classes: GoalType

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
isGoalTypeOf	Goal	1:*

Class: Postrequisite

Defines the post-requisite information after a task is completed; information such as the output information and the conditions (i.e. post-condition) that must hold after a task is completed.

Super-classes: :THING

Sub-classes: None

In domain of: isPostrequisiteOf

In range of: hasPostrequisite

Slot Name	Type	Cardinality
id	String	1:1
isPostRequisiteOf	Task	0:*
name	String	1:1
output	String	0:*
postconditions	String	0:*

Class: Prerequisite

Defines the pre-requisite information required before performing a task; information such as the needed input information and the conditions (i.e. pre-condition) that must hold before execution of a task.

Super-classes: :THING

Sub-classes: None

In domain of: isPrerequisiteOf

In range of: hasPrequisite

Slot Name	Type	Cardinality
id	String	1:1
input	String	0:*
isPrequisiteOf	Task	0:*
name	String	1:1
preconditions	String	0:*

Class: Scenario

A behavioral description of situations in the system and its relation with its environment. It has an initial state that defines pre-condition and a final state that defines post-condition, with possible normal scenario (success scenario) and an exceptional scenario (unsuccessful or alternative scenario). The normal and exceptional scenarios are categorized as sub-classes of the Scenario class, namely NormalScenario sub-class and ExceptionScenario sub-class. A scenario captures various viewpoints and connects with tasks and actions performed with respect to the viewpoints.

A scenario may decompose (i.e. refine) into sub-scenarios called episodes through an aggregation relationship, i.e. a scenario hasEpisode zero or more sub-scenarios (or episodes). Inversely, a sub-scenario (or episode) isEpisodeOf zero or more scenarios which also means that a sub-scenario (or episode) may be shared among several scenarios. The operational nature of scenarios helps to drive the definition of requirements and consequently operationalize the requirements to meet some goals. A scenario operationalizes one or more requirements; inversely, a requirement is operationalizedBy one of more scenarios.

Super-classes: :THING

Sub-classes: ExceptionScenario
NormalScenario

In domain of: isScenarioOf, hasTask, isEpisodeOf, hasConstraint, operationalizes,

In range of: hasScenario, isTaskOf, hasEpisodeOf, isConstraintOf, operationalizedBy,

Slot Name	Type	Cardinality
author	String	0:1
date_created	String	0:1
description	String	0:1
hasConstraint	Constraint	0:*
hasEpisode	Scenario	0:*
hasTask	Task	1:*
id	String	1:1
isEpisodeOf	Scenario	0:*
isScenarioOf	Viewpoint	1:*
name	String	1:1
operationalizes	Requirement	1:*
postconditions	String	0:*
preconditions	String	0:*
release_number	String	0:*

status	{Active, In-Active, Terminated}	0:1
trigger	String	0:1

Class: Stakeholder

A stakeholder represents the role(s) played by a person, an organization, a group, or a system that has an interest in (or benefit from), an opinion for, some responsibility for, that may influence or be influenced by, or may affects or be affected by the result of the system-to-be. Examples of stakeholders: shareholders, investors, sponsors, managers, administrators, system buyers, system users, system engineers, system developers, end-users, government, and regulatory bodies.

An actor is always a potential stakeholder but a stakeholder may not necessary be an actor, i.e. a stakeholder has an interest in the success of the system-to-be but does not interact with the system-to-be.

Super-classes: :THING

Sub-classes: None

In domain of: isStakeholderOf, representedBy, raises

In range of: hasStakeholder, represents, isImportantTo

Slot Name	Type	Cardinality
id	String	1:1
isStakeholderOf	Goal, Viewpoint	1:*
name	String	1:1
priority	Float	0:1
raises	Concern	0:*
representedBy	Actor	1:*

Class: System

A collection of interacting, interrelated, or interdependent elements that forms an integrated whole that functions together as a unit to achieve one or more goals. The elements may include people, hardware, software, facilities, policies, and documents and all things required to produce systems-level results. Examples of a system: computer system, transmission system.

Super-classes: Actor

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
name	String	1:1
plays	Role	0:*
performs	Action	1:*
represents	Stakeholder	0:1
uses	Tool	0:*

Class: SystemGoal

The goal(s) of a system. Generally, a system goal is a goal of the system as a whole, i.e. the requirements on the services that the system provides. Example of a system goal: "*Ensure that system does not need more than three hours of weekly maintenance.*"

Super-classes: GoalType

Sub-classes: None

In domain of: None

In range of: None

Slot Name	Type	Cardinality
id	String	1:1
isGoalTypeOf	Goal	1:*

Class: Task

An activity performed by an actor engaging in a particular environment associated with a particular scenario to accomplish a particular goal. A task may be decomposed (i.e. refined) into sub-tasks and eventually into actions (or operations).

Super-classes: :THING

Sub-classes: None

In domain of: AND-refinesInto, OR-refinesInto, hasPostrequisite, hasPrerequisite, isTaskOf, hasTool, hasAction, hasConstraint

In range of: AND-refinesFrom, OR-refinesInto, isPostrequisiteOf, isPrerequisiteOf, hasTask, isToolOf, isAction, isConstraintOf

Slot Name	Type	Cardinality
AND-refinesFrom	Task	0:*
AND-refinesInto	Task	1:*
description	String	0:1
duration	{<01:00mins, 01.01-to-05:00mins, 05:01-to-10:00mins, 10:01-to-15:00mins, 15:01-to- 30:00mins, 30:01-to-60:00mins, >60:00mins}	0:1
frequency	{Low, Medium, High}	0:1
hasAction	Action	1:*
hasConstraint	Constraint	0:*
hasPostrequisite	Postrequisite	0:*
hasPrerequiste	Prerequisite	0:*
hasTool	Tool	0:*
id	String	1:1
isTaskOf	Viewpoint, Scenario	1:*
name	String	1:1
OR-refinesFrom	Task	0:*
OR-refinesInto	Task	0:*
priority	Flaot	0:1
task_type	{Simple-Atomic, Simple-Composite, Complex}	0:1

Class: Tool

A physical object that is used by an actor to support the system for the execution of the tasks. For examples, a printer, a laptop.

Super-classes: :THING

Sub-classes: None

In domain of:	isToolOf, usedBy
In range of:	hasTool, uses

Slot Name	Type	Cardinality
id	String	1:1
isToolOf	Task, Viewpoint	0:*
name	String	1:1
usedBy	Actor	0:*

Class: Viewpoint

Defines the role, environment, goals, concerns, scenarios, tasks, requirements, and modeling methods according to the characteristics of the persona with respect to a particular environment the persona is engaging in. A viewpoint provides a means to define views at a certain level of detail of design concepts and addresses certain design concerns by the persona playing a particular role in a particular environment. Each viewpoint addresses the persona's concerns that are important to the stakeholders in the context of a system-to-be. A persona possesses one or more viewpoints depending on the role the persona plays and the environment the persona engages in at a particular moment in time. For examples, an expert developer and a novice developer may have varied viewpoints (in terms of scenarios and tasks) in performing a unit test, a student may have different viewpoints on using an online application in a home environment versus a public environment. The modeling methods are methods or languages to be used in constructing the models of a view, e.g. UML Class Diagrams, UML Instance Diagram, Ontology Protégé-Frames, PAL, and so forth.

The Viewpoint class plays an important class in the Behavioral-GST Ontology as it relates with several classes in the Behavioral-GST Ontology, the Persona Ontology, and the Requirements Ontology. For examples, in Behavioral-GST Ontology, a viewpoint has goals, stakeholders, tasks, and scenarios; in Persona Ontology, a viewpoint addresses concerns, a viewpoint has (is defined uniquely by) one persona, one role, and one environment; in Requirements Ontology, a viewpoint has requirements. The Viewpoint class captures information synthesized from the persona documents (Persona Profile Document and Persona Definition Document) and specifies the information explicitly in Protégé-Frames.

Super-classes:	:THING
Sub-classes:	None

In domain of:	hasPersona, hasEnvironment, hasRole, addresses, hasStakeholder, hasGoal, hasObstacle, hasTool, hasTask, hasScenario, hasRequirements, hasConstraint
In range of:	isPersonaOf, isEnvironmentOf, isRoleOf, addressedBy, isStakeholderOf, isGoalOf, isObstacleOf, isToolOf, isTaskOf, isScenarioOf, isRequirementsOf, isConstraintOf

Slot Name	Type	Cardinality
addresses	Concern	1:*
author	String	0:1
date_created	String	0:1
description	String	0:1
hasConstraint	Constraint	0:*
hasEnvironment	Environment	1:1
hasGoal	Goal	1:*
hasObstacle	Obstacle	0:*
hasPersona	Persona	1:1
hasRole	Role	1:1
hasScenario	Scenario	1:*
hasStakeholder	Stakeholder	1:*
hasTask	Task	1:*
hasTool	Tool	0:*
history	String	0:*
id	String	1:1
modeling_techniques	String	0:*
models	String	0:*
name	String	1:1
source	String	0:*

C.3. Requirements Ontology Class Descriptions

Requirements Ontology Class Herarchy

Class: Functional

Defines the desired functionality of a system, i.e. what a system is supposed to do. Functional requirements are the typical user requirements for a system. An example of a functional requirement: "*The system shall provide a list of all courses of a subject based on the selected term and subject*."

Super-classes: RequirementCategory

Sub-classes: None

In domain of: hasSubReqCategory

In range of: isSubReqCategoryOf

Slot Name	Type	Cardinality
category	String	0:1
hasReq	Requirement	1:*
hasSubReqCategory	RequirementCategory	0:*
id	String	1:1
isSubReqCategoryOf	RequirementCategory	0:1

Class: NonFunctional

Defines criteria that judge the qualities or constraints of the system, i.e. how a system is supposed to be. Criteria to judge the qualities or constraints of a system include:

reliability, *performance*, *security*, *availability*, *maintainability*, *usability*, and so forth.

Super-classes: RequirementCategory

Sub-classes: None

In domain of: hasSubReqCategory

In range of: isSubReqCategoryOf

Slot Name	Type	Cardinality
category	String	0:1
hasReq	Requirement	1:*
hasSubReqCategory	RequirementCategory	0:*
id	String	1:1
isSubReqCategoryOf	RequirementCategory	0:1

Class: Requirement

A generic requirement describing the properties of a requirement.

Requirements are composite objects, i.e. a requirement may relate with other requirements. For example, a requirement may derive into (or decompose into, refine into) sub-requirements through requirement refinements via. refinesInto (inversely, refinesFrom) relationship. Both types of refinements are associated with other requirements via an aggregation relationship, i.e. a requirement is an aggregation of other requirements via refinesInto relationship.

Super-classes: :THING

Sub-classes: None

In domain of: refinesInto, derivesInto, requires, constraints, conflictsWith, meets, operationalizedBy, isReqOf, hasPart

In range of: refinesFrom, derivesFrom, requiredBy, constrainedBy, conflictsWith, metby, operationalizes, hasReq, isPartOf

Slot Name	Type	Cardinality
conflictsWith	Requirement	0:*
constrainedBy	Requirement	0:*
constraints	Requirement	0:*

date_created	String	0:1
date_last_changed	String	0:*
description	String	0:1
id	String	1:1
isPartOf	SystemRequirementsSpecificationSRS	1:*
isReqOf	RequirementCategory	1:*
isRequirementsOf	Viewpoint	1:*
meets	Goal	1:*
owner	String	0:1
priority	Float	0:1
refinesFrom	Requirement	0:1
refinesInto	Requirement	0:*
release_number	String	0:*
req_type	{PersonaReq, BusinessReq, SystemReq}	0:*
requiredBy	Requirement	0:*
requires	Requirement	0:*
risk_level	Float	0:1
source	String	0:*
statement	String	0:1
status	{Proposed, Pending, Accepted, Rejected, Replaced, Implemented}	0:1
validation	{Validated, Not-Validated, Pending}	0:1
verification	{Verified, Not-Verified, Pending}	0:1

Class: RequirementCategory

Defines the various categories of a requirement. It has a category attribute of String type which specifies the category of a requirement. It is specialized into two sub-classes: Functional, NonFunctional. Each sub-class has various categories of requirement. For examples: “Computational”, “Informational”, and “Navigational” are categories of function requirement in the Functional sub-class; “Availability”, “Performance”, “Usability”, “Quality”, Reliability”, “Safety” are categories of non-functional requirement in the NonFunctional sub-class.

A requirement category may be a sub-requirement category of zero or more requirement categories. For example, a non-functional requirement category “Quality” may have several sub-requirement categories such as “Availability”, “Capacity”, “Performance”, “Usability”, “Reliability”, “Maintainability”, ‘Fault Tolerance”, “Safety”, and “Security”. Each sub-requirement category is also a requirement category.

The decomposition of a requirement category is accomplished via an aggregation

relationship, i.e. a requirement category hasSubReqCategory zero or more requirement categories. Inversely, for simplicity, a requirement category isSubReqCategoryOf one and only one requirement category. For example, a non-functional requirement category "Quality" hasSubReqCategory "Availability", "Capacity", "Performance", "Usability", "Reliability", "Maintainability", 'Fault Tolerance", "Safety", and "Security".

Super-classes: :THING

Sub-classes: Functional
NonFunctional

In domain of: hasSubReqCategory, hasReq

In range of: isSubReqCatehoryOf, isReqOf

Slot Name	Type	Cardinality
category	String	0:1
hasReq	Requirement	1:*
hasSubReqCategory	RequirementCategory	0:*
id	String	1:1
isSubReqCategoryOf	RequirementCategory	0:1

Class: SystemRequirementsSpecificationSRS

Formal definitions of the system functional and operational requirements.

Super-classes: :THING

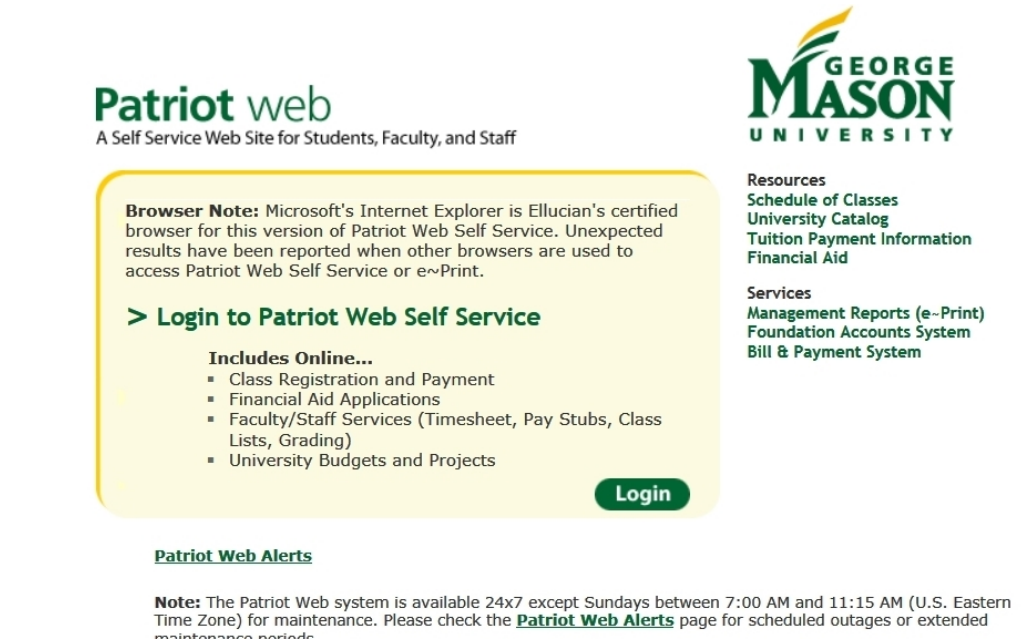
Sub-classes: None

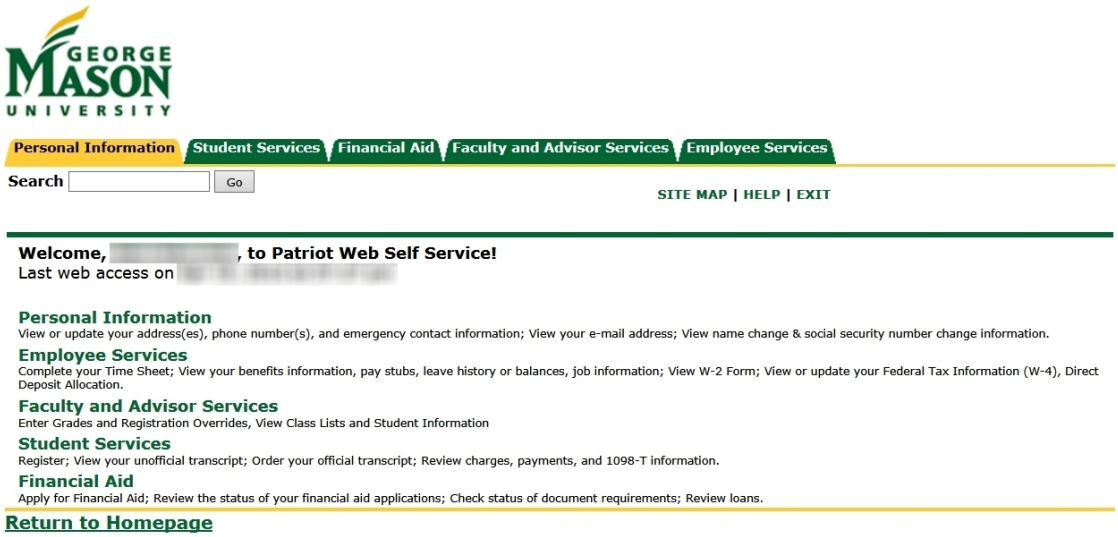
In domain of: hasPart

In range of: isPartOf

Slot Name	Type	Cardinality
author	String	0:1
date_created	String	0:1
description	String	0:1
hasPart	Requirement	1:*
id	String	1:1
version	String	0:1

APPENDIX D: CASE STUDY SCREENSHOTS





Linda Rose,

the busy undergraduate student and web developer

Version 1.1

Linda Rose is 26 years old and is currently a full-time undergraduate student pursuing a B.S. Degree in Information Technology at the Department of Information Sciences and Technology at George Mason University. Currently, Linda is currently in her senior year in the Information Technology B.S. program. Linda has an Associate degree in Information Systems at Virginia Community College and a high school diploma at Fairfax High School in Virginia.

Besides being a full-time undergraduate student, Linda is also currently working part-time as a web developer at a local web development firm in Fairfax, Virginia. She has been working at her current job for 2 years. Linda has 3 years of technical experience working with website coding such as HTML, CSS, Javascript, and programming languages such as Java, C++, and Visual Basic. Linda uses computer in a daily basis, at work and at home. Linda considers her computer and technical knowledge proficiency as intermediate. She is very comfortable using computer in various platform environments, including UNIX and WINDOWS (8, 7, XP). Linda uses a desktop computer at home and carries her laptop and smartphone to school and work.

In her spare time, Linda likes to watch fantasy and sci-fi movies. She is a fan of Harry Potter and Alien movies. She feels that watching fantasy and sci-fi movies bring a lot of inspiration and creative thinking in her work. Linda speaks English and Spanish fluently. However, Linda is not good in writing in Spanish.

Linda exemplifies a real-world person who wants to go online, quickly browse for classes that interest her, enroll for classes, and make online credit card tuition payment. Linda usually check her school activities mostly during lunch time at work. She is busy in her work and study and so doesn't like to spend a lot of time figuring out how to use the system, and when she needs to use the system the next time, she doesn't want to restart learning how to use the system. Linda does not care much about how attractive the website is. She represents a user group that wants simple uncomplicated system so that she doesn't make a lot of mistakes while using it, as well as a pleasing, uncluttered, and professional looking system.

Linda Rose,

the busy undergraduate student and web developer

Primary Persona

PRIORITY

DEMOGRAPHIC PROFILE

NAME				GENDER	AGE	MARITAL STATUS	HIGHEST EDUCATION LEVEL	OCCUPATIONS				SOCIAL LEVEL
First	Last	Nickname	Title					Job Title	Work Status	Length of Employment	Salary	
Linda	Rose	Rosie		Female	25-34	Single	Associate degree	Web developer	Part-time	3 years	$30K - $50K	Middle-class

LANGUAGE PROFICIENCY			INTERESTS	ABILITIES									
				Physical				Cognitive					
Listening	Speaking	Writing		Motor	Sensory-Hearing	Sensory-Speech	Sensory-Visual	Attention	Memory	Problem-Solving	Reading-Comp.	Visual-Comp.	Spatial
• English – excellent • Spanish – good	• English – excellent • Spanish – good	• English - good • Spanish – fair	• Listen music • Play guitar • Surf the web • Shop online	Average	Average	Average	Average	High	Average	High	Average	Average	Average

KNOWLEDGE & SKILLS PROFILE

COMPUTER

- Software (office applications) – excellent
- Hardware – good
- Programming (Java, C++, Visual Basic) – good
- Web development (HTML, CSS, JavaScript) – good
- Operating systems (Windows 8, 7, XP, UNIX) – good

WEB

- Internet – good
- Social media – good

DOMAIN

- UML – good
- Requirements engineering – good

INTERACTION PROFILE

ROLES	GOALS	CONCERNS
Full-time undergraduate student	• Browse courses with ease • Browse courses with fast response • Enroll courses with ease • Check student records with ease • Make online tuition payment with ease	• Will the website interface and information be properly presented on cell phone, laptop, and desktop computer? • Will the website be easy to navigate that does not take too much? • Will the website content be properly presented without much confusion? • Will the system be available for service when needed? • Will the public WiFi connection be available when needed? • Will the public WiFi connection be secured? • Will the online payment function properly?

ENVIRONMENTS

Location	Time of Day	Duration	Frequency (low: 0-5 times/wk, medium: 6-10 times/wk, high: >10 times/wk)	Attitudes (eager, skeptical, cautious, indifferent, impatient, resentful, curious, trusting, others)
Starbucks	Mid-work day	15 mins	Medium	Eager, skeptical, cautious, impatient
Home	Evening	40 mins	Medium	Eager, trusting

USAGE PATTERNS (scenarios titles)	TOOLS	USABILITY PREFERENCES (accuracy, attractiveness, efficiency, learnability, reliability, comprehensibility, clarity, rememberability)
• Browse courses • Register courses • Check student schedule • Review student records • Review financial aid • Make online payment	• Laptop • Desktop • Cell phone • Tablet	• Accuracy – very important • Attractiveness – not at all important • Efficiency – very important • Learnability – important • Comprehensibility – very important • Clarity – important • Rememberability – somewhat important

<Linda Rose, Full-Time Undergraduate Student, Starbucks (mid-work day)>	VIEWPOINT (VP) NAME

SOURCES	STAKEHODLERS	HISTORY (VP-version, date, author, changes)	ROLE
• Persona Profile Document (PPD) • Persona Definition Document (PDD)	• Administrators • Billing system • Instructors	VP-v1.0, 10-27-2013, John Deer	Full-time undergraduate student

ENVIRONMENT (location, time of day, duration, tools, etc)	• Starbucks coffee shop • Mid-day (lunch work time) • Limited available time (~15 mins) • Cell phone (internet WiFi capable) • Heavy human traffic • Rely on WiFi connection • Security and privacy issues

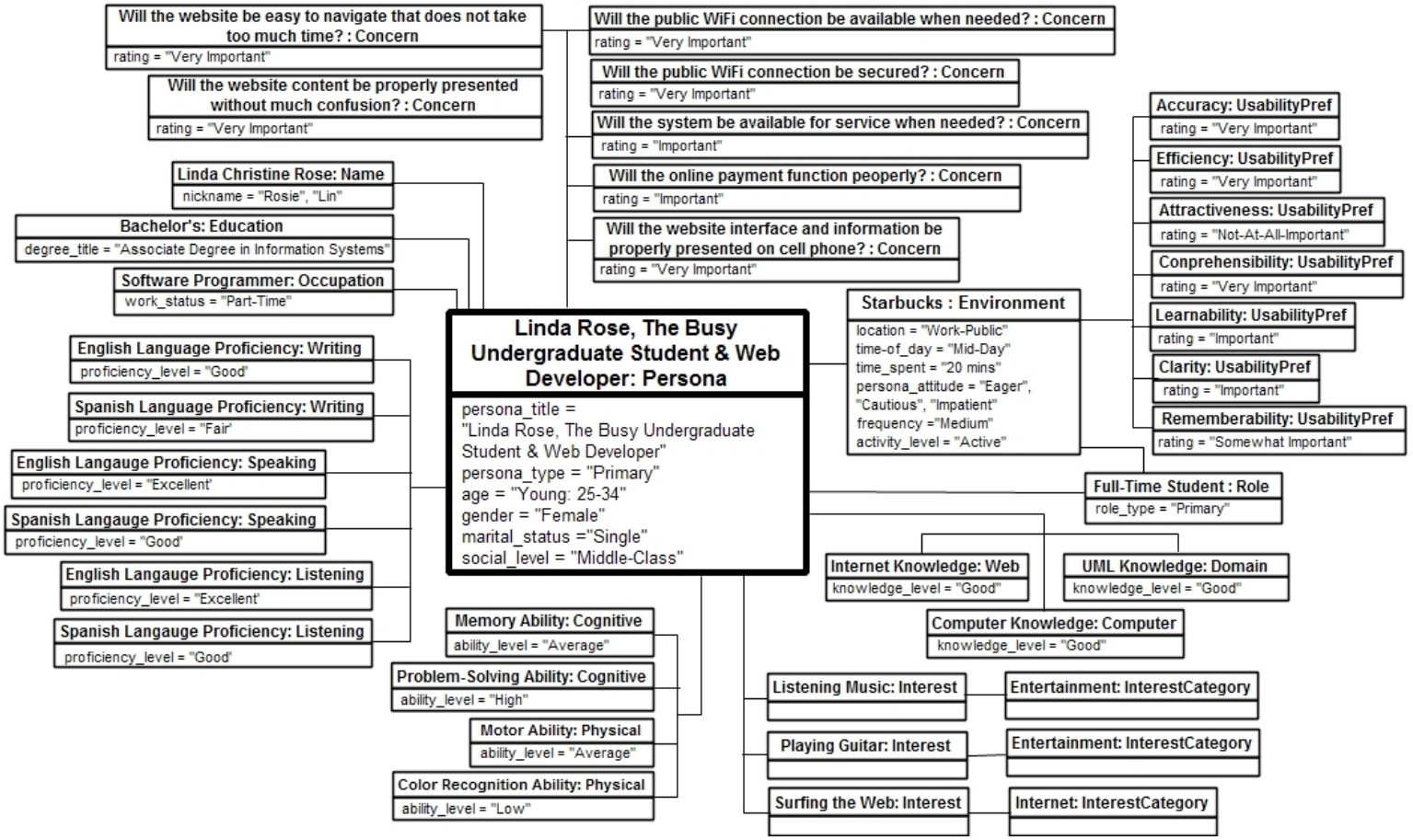
GOALS	SCENARIOS, TASKS	
	Scenarios (titles)	Tasks (titles)
• Browse courses with ease • Enroll courses with ease • Check student records with ease	• Search for classes	•Find a specific course • Browse course registration catalog
CONCERNS	• Register for classes	•Find a specific course •Browse course registration catalog •Add course to schedule • Select specific course
• The presentation of website interface and information on cell phone, laptop, and desktop computer • The navigational system and the time to complete the tasks • The accessibility of internet (WiFi) connection • The security and privacy of public internet (WiFi) connection • The availability of the system when needed	• View personal Information	....
	• View student records	....

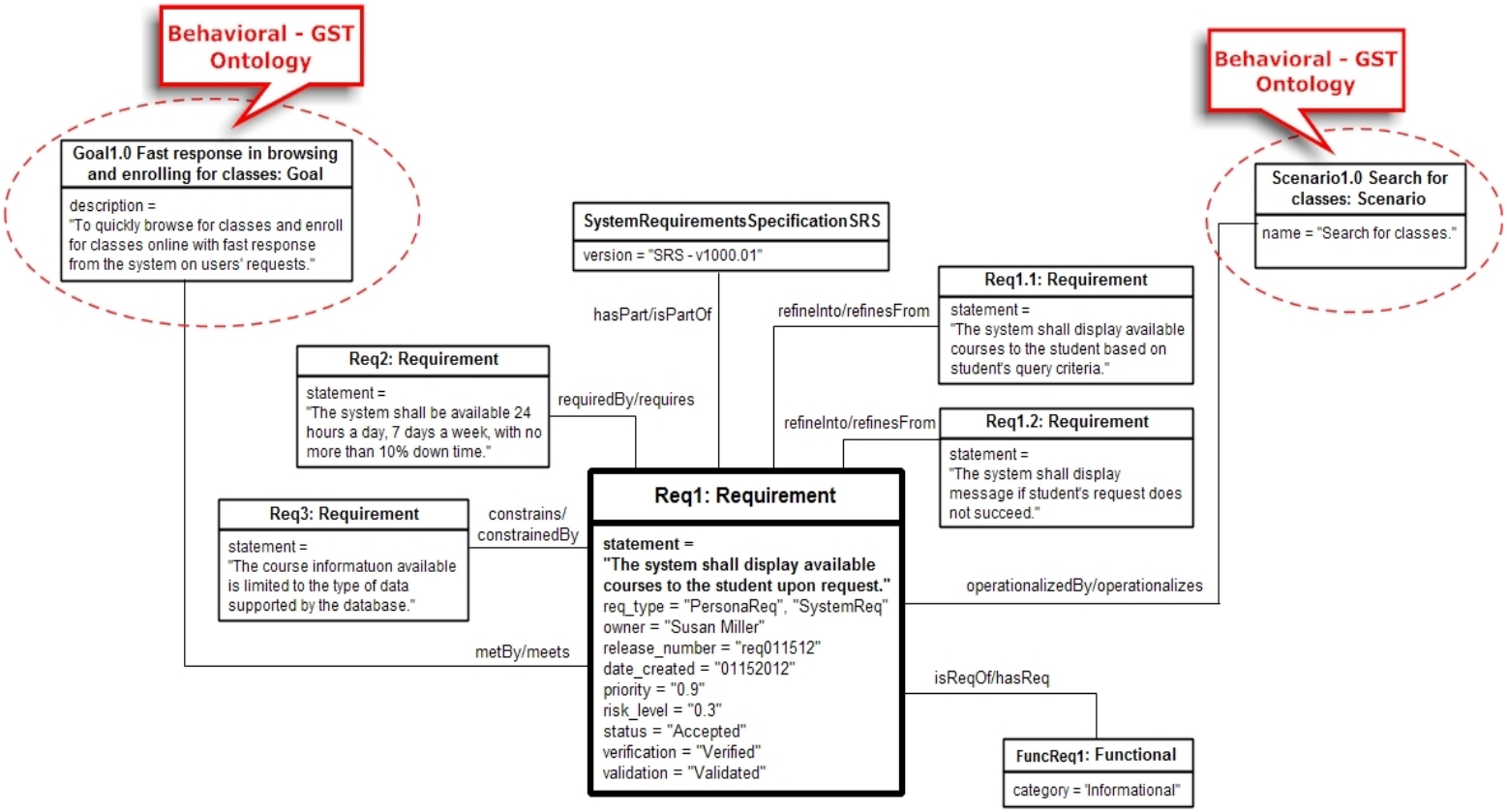
REQUIREMENTS	
Functional	• The system shall display available courses to the student based on student's query criteria on courses • The system shall accept student's course registrations • The system shall validate based upon course availability • The system shall enable student to view her school records • The system shall display message if student's course registration does not succeed
Non-Functional	• The system shall provide uncomplicated navigational system that allows users to find available services options with ease and without spending too much time. • The system shall ensure that content does not clutter on a page.• The system shall display page layout and information appropriately for smaller screen size devices such as mobile phones and tablets • The system shall ensure that all links, menus, or buttons produce correct pages upon clicking. • The system shall ensure that all confidential information be encrypted to maintain security

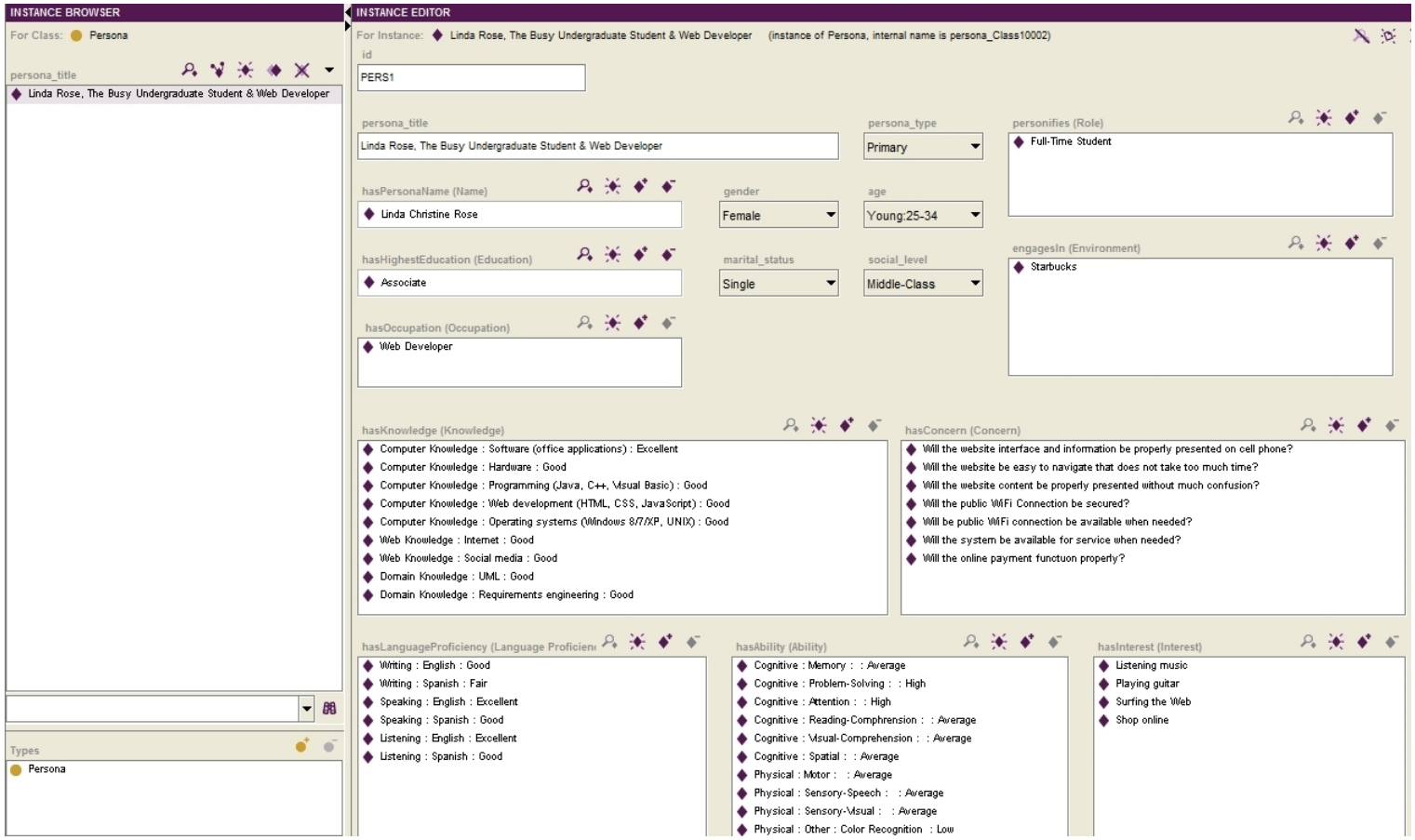
MODEDLING		
Modeling Techniques	Concepts Representation	Models, Ontologies
• UML Class Diagram • UML Instance Diagram	• Ontology tool (Protégé-Frames)	• Persona Class Diagram, Persona Instance Diagram, Behavioral-GoalScenarioTask Class Diagram, Behavioral-GoalScenarioTask Instance Diagram, Requirements Class diagram, Requirements Instance Diagram • Persona Ontology, Behavioral-GoalScenarioTask Ontology, Requirements Ontology

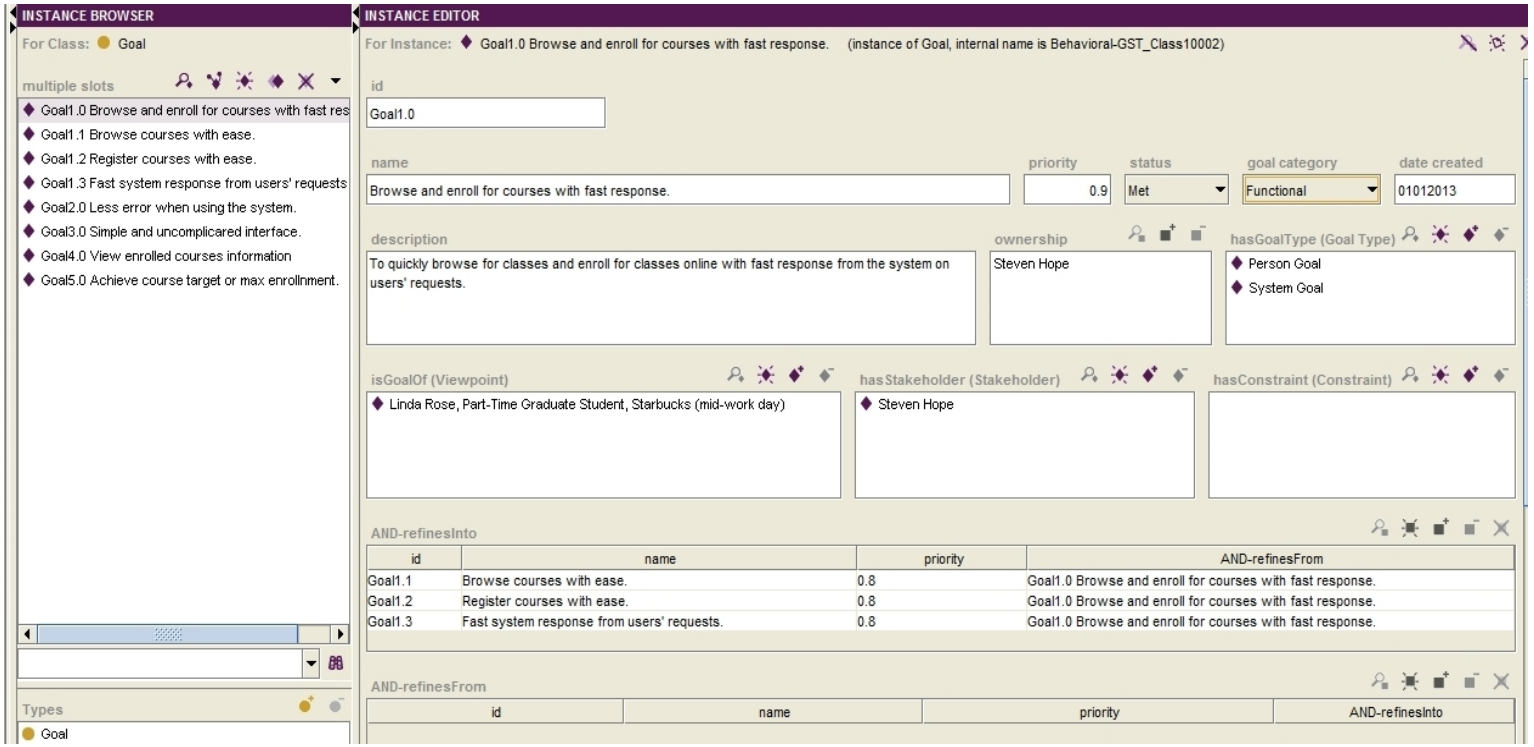
Rating scale: 5 – Critical 4 – Very Important 3 – Important 2 – Somewhat important 1 – Unimportant		PERSONAS	Linda Rose, *the busy undergraduate student and web developer* (*primary persona*)		WEIGHTED SUM
		WEIGHTS	50		
		VIEWPOINTS (VPB)	<Linda Rose, Full-Time Undergraduate Student, Starbucks (mid-work day)>	<Linda Rose, Full-Time Undergraduate Student, Home (evening)>	
REQUIREMENTS	REQ1	The system shall provide uncomplicated navigational system that allows users to find available services options with ease and without spending too much time.	4	3	350
	REQ2	The system shall ensure that content does not clutter on a page.	4	3	350
	REQ3	The system shall ensure that all links, menus, or buttons produce correct pages upon clicking.	4	3	350
	REQ4	The system shall display message if student's course registration does not succeed.	3	3	300
	REQ5	The system shall display available courses to the student based on student's query criteria on courses.	4	3	350
	REQ6	The system shall display page layout and information appropriately for smaller screen size devices such as mobile phones and tablets.	4	2	300
	REQ7	The system shall ensure that all confidential information be encrypted to maintain security.	4	4	400

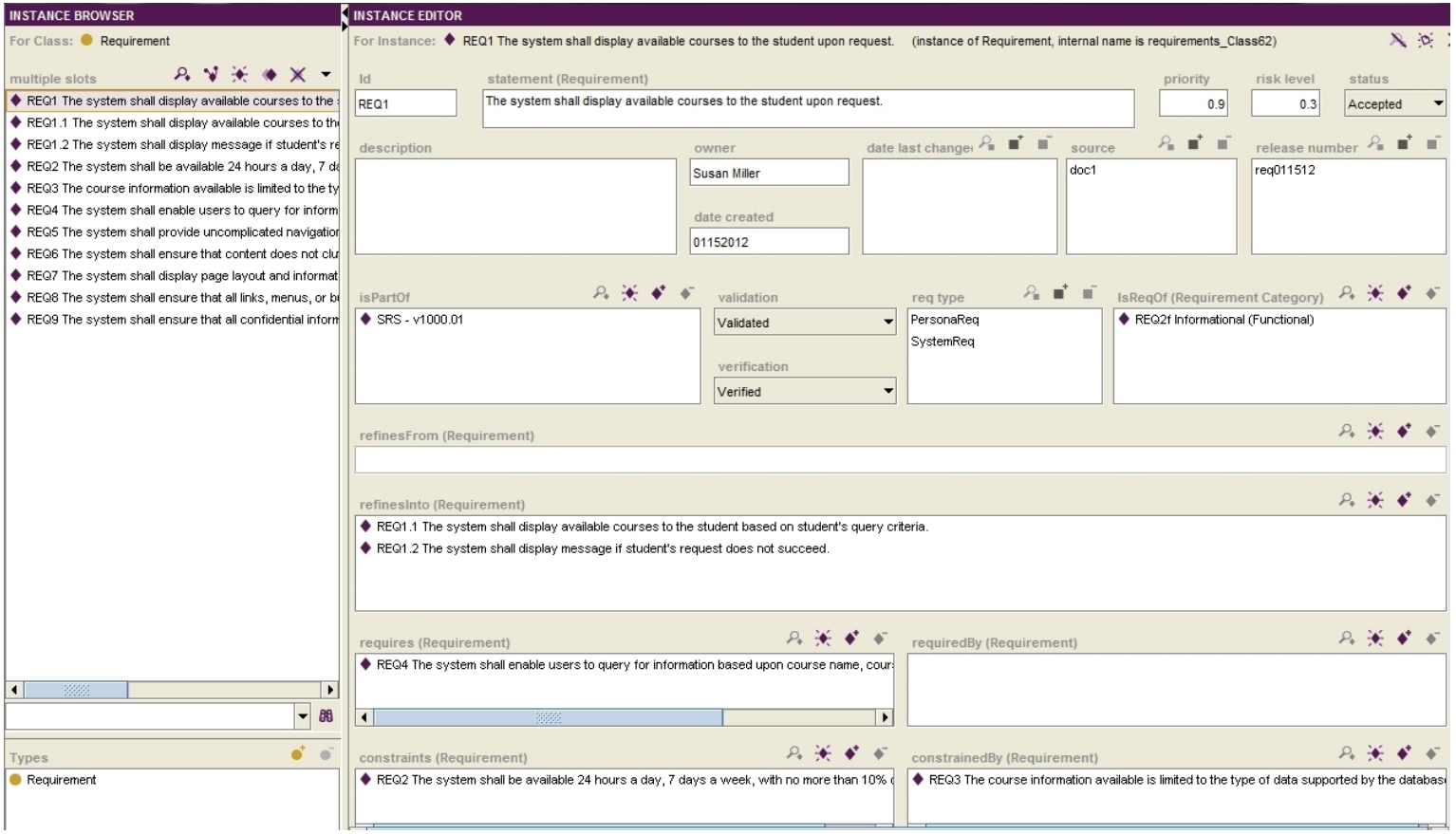
		USABILITY GOALS							
		Accuracy	Attractiveness	Efficiency	Learnability	Reliability	Comprehensibility	Clarity	Rmember-ability
SCENARIOS	SCEN 1: View Student Service Page			Extra click and page transition: needs to click on Student Services tab to go to next page, then clicks to select services, i.e. Registration, Student Records, Student Account – *it is better to have an effective main navigational system, e.g. a drop down menu for each menu tab.*	Navigational system not properly designed: needs to click on main menu tab in order to view other services for that menu tab – *it is better to incorporate a drop-down menu for each menu tab to allow viewing of available services for each menu tab.*		On the Main Menu page, services links on body content are a bit messy, order not in sequence with the order of the menu tabs – *it is better to reorder the services links on body content to be in same order as the menu tabs.*		
	SCEN 2: View Registration Page			(1) Same issue as in SCEN 1			(1) Services links on body content are in random order – *it is better to re-arrange the services links on body content, group relevant services links in same group.* (2) Visibility and location of commonly used links issue: two commonly used functions: "Search for Classes" and "Register, Add or Drop Classes" are tucked in the body content with other links – *it is better to be visible and easily accessible, e.g. big buttons.* (3) "Select Term" link on Registration page is redundant: needs to select the term again when search for classes, etc) – *it is better to remove the "Select Term" link.*		
	SCEN 3: View Student Records Page			(1) Same issue as in SCEN 1			(1) Services links on body content are in random order – *it is better to re-arrange the services links on body content, group relevant services links in same group.* (2) Visibility and location of commonly used links issue: two commonly used functions: "View Official Transcript" and "View Final Grades" are tucked in the body content with other links – *it is better to be visible and easily accessible, e.g. big buttons.*		
	SCEN 4: View Personal Information Page			(1) Same issue as in SCEN 1 (2) The unnecessary individual view and update services links results in unnecessary clicks and back and forth pages issues (affect task time) – *it is better to have one single link to view/update information on a single page.*			Cluttered content: too many unnecessary links to view and update personal information which cluttered the page. Needs to click the view addresses link to view addresses, return to Personal Information page and selects other services to view/update persona information – *it is better to combine all these individual view and update links into one link and view/update all in one same page.*		
	SCEN 5: Search for Classes	"New Search" button not accurately returns page that reflects what would be expected, i.e. return to previous page for new course search, rather than returning to the beginning for a brand new term search – *it is better to have another button to return to previous courses page and short instruction on top of the page body.*		(1) Same issue as in SCEN 1 (2) Repeated task: "New Search" button returns to the "Select Term" page. Needs to repeat the process again to select the term again – *it is better to have a back to previous page button.* (3) Advanced search field options are displayed in a separate page (i.e. extra page transition) – *it is better to display the field options on same page.*	(1) Same issue as in SCEN 1-4 navigational system issue. (2) "New Search" button issue: needs to know and learn that clicking the new search button will start from beginning, i.e. select new term. It does not mean to search new courses or subjects (of the same term) – *it is better to have short instruction highlighting what the 'New Search" button does.*		"New Search" button can be misleading and not understood correctly – this button original intention is to do a brand new search, starting at the Term selection. When searching for new courses of the same term and/or same subject, this button can thus be inconvenient – *it is better to have another button to return to previous course search page, rather than repeating the process starting at the Term selection. Also, short instruction on top of the page body to indicate what the buttons do.*		

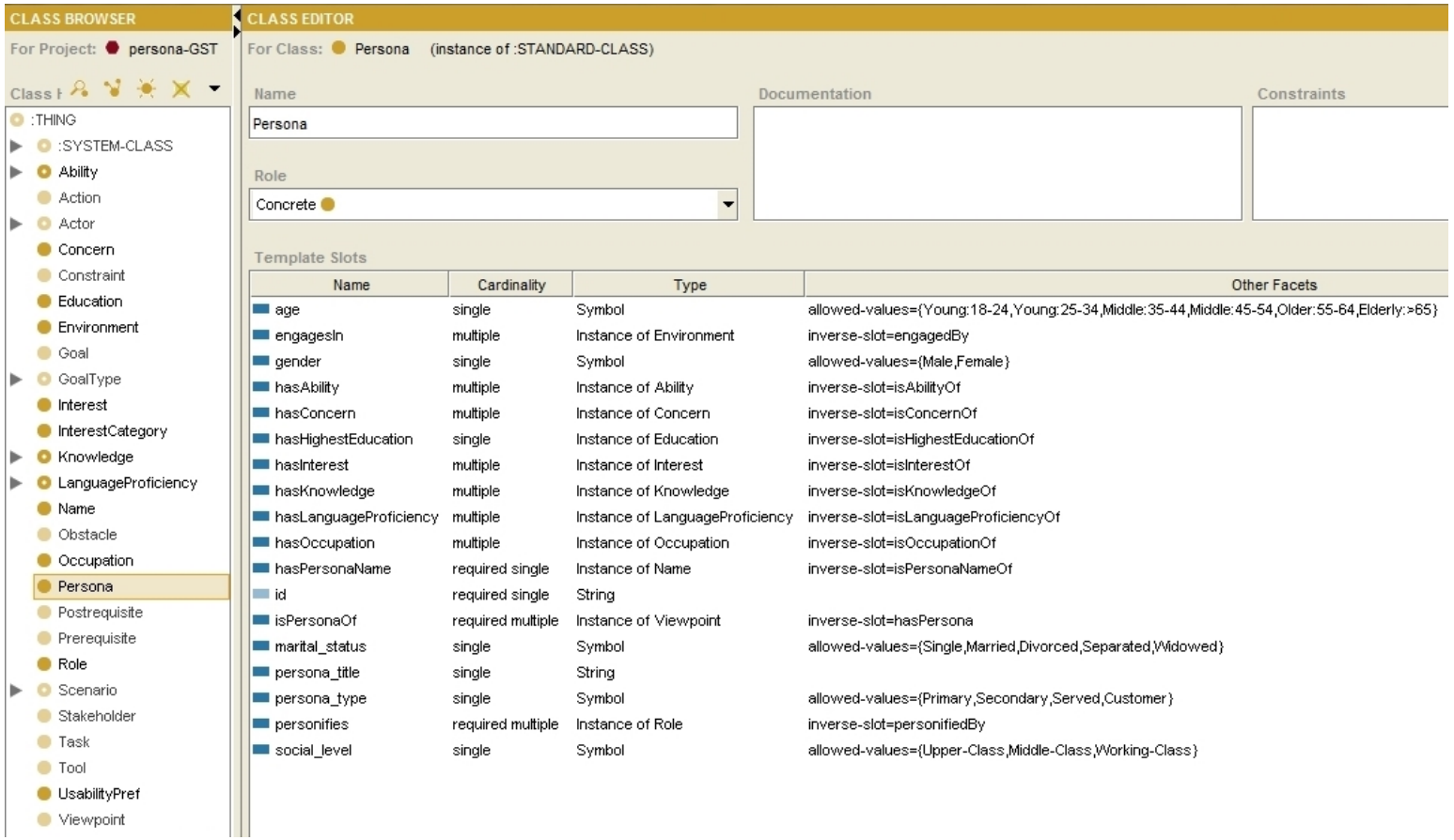

Will the website be easy to navigate that does not take too much time? : Concern
rating = "Very Important"
Will the website content be properly presented without much confusion? : Concern
rating = "Very Important"
Will the public WiFi connection be available when needed? : Concern
rating = "Very Important"
Will the public WiFi connection be secured? : Concern
rating = "Very Important"
Will the system be available for service when needed? : Concern
rating = "Important"
Will the online payment function peoperly? : Concern
rating = "Important"
Will the website interface and information be properly presented on cell phone? : Concern
rating = "Very Important"
Linda Christine Rose: Name
nickname = "Rosie", "Lin"
Bachelor's: Education
degree_title = "Associate Degree in Information Systems"
Software Programmer: Occupation
work_status = "Part-Time"
English Language Proficiency: Writing
proficiency_level = "Good"
Spanish Language Proficiency: Writing
proficiency_level = "Fair"
English Langauge Proficiency: Speaking
proficiency_level = "Excellent"
Spanish Langauge Proficiency: Speaking
proficiency_level = "Good"
English Langauge Proficiency: Listening
proficiency_level = "Excellent"
Spanish Langauge Proficiency: Listening
proficiency_level = "Good"
Linda Rose, The Busy Undergraduate Student & Web Developer: Persona
persona_title = "Linda Rose, The Busy Undergraduate Student & Web Developer"
persona_type = "Primary"
age = "Young: 25-34"
gender = "Female"
marital_status ="Single"
social_level = "Middle-Class"
Starbucks : Environment
location = "Work-Public"
time-of_day = "Mid-Day"
time_spent = "20 mins"
persona_attitude = "Eager", "Cautious", "Impatient"
frequency ="Medium"
activity_level = "Active"
Accuracy: UsabilityPref
rating = "Very Important"
Efficiency: UsabilityPref
rating = "Very Important"
Attractiveness: UsabilityPref
rating = "Not-At-All-Important"
Conprehensibility: UsabilityPref
rating = "Very Important"
Learnability: UsabilityPref
rating = "Important"
Clarity: UsabilityPref
rating = "Important"
Rememberability: UsabilityPref
rating = "Somewhat Important"
Full-Time Student : Role
role_type = "Primary"
Internet Knowledge: Web
knowledge_level = "Good"
UML Knowledge: Domain
knowledge_level = "Good"
Computer Knowledge: Computer
knowledge_level = "Good"
Memory Ability: Cognitive
ability_level = "Average"
Problem-Solving Ability: Cognitive
ability_level = "High"
Motor Ability: Physical
ability_level = "Average"
Color Recognition Ability: Physical
ability_level = "Low"
Listening Music: Interest
Entertainment: InterestCategory
Playing Guitar: Interest
Entertainment: InterestCategory
Surfing the Web: Interest
Internet: InterestCategory

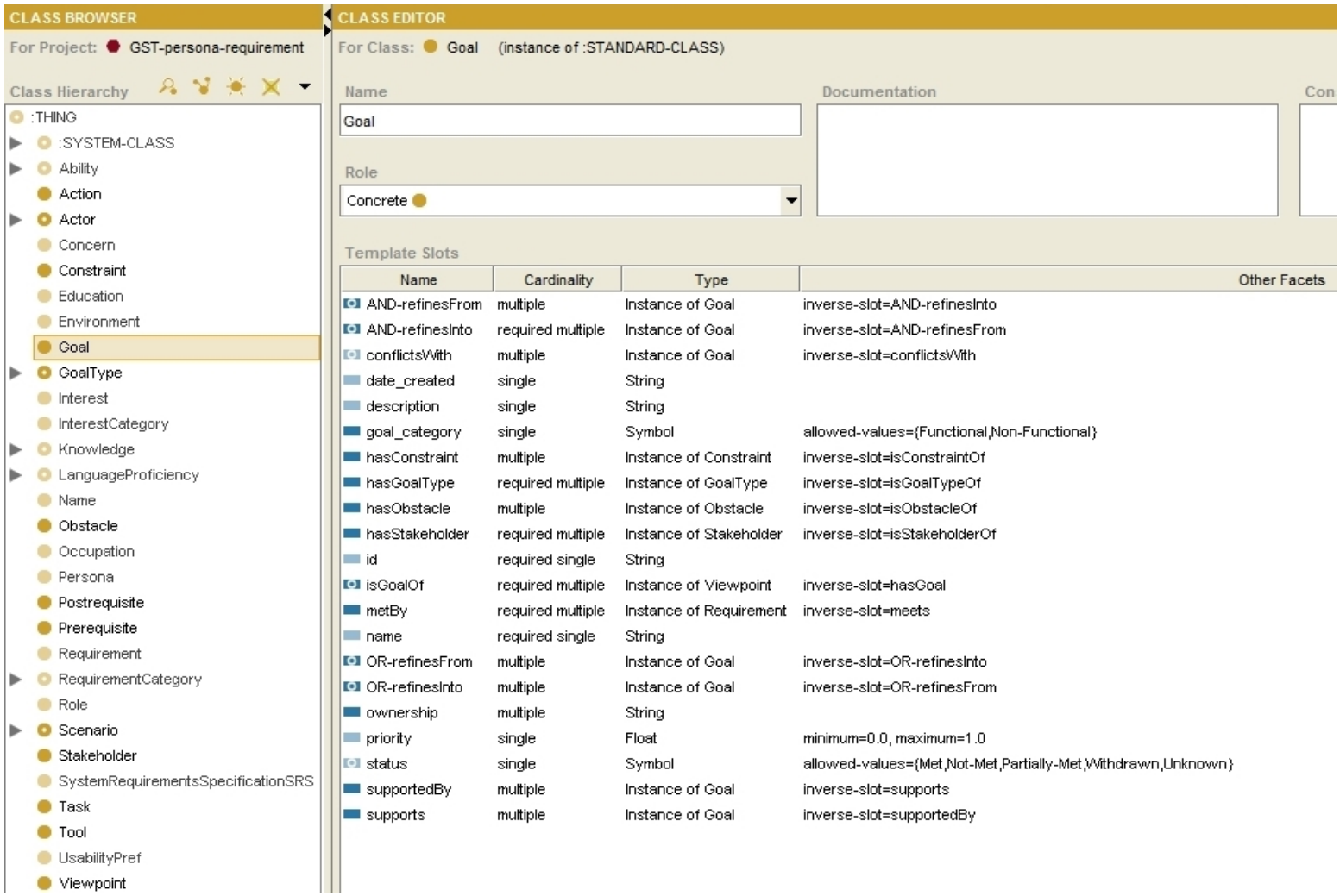

Behavioral - GST Ontology
Goal1.0 Fast response in browsing and enrolling for classes: Goal
description = "To quickly browse for classes and enroll for classes online with fast response from the system on users' requests."
SystemRequirementsSpecificationSRS
version = "SRS - v1000.01"
hasPart/isPartOf
Req2: Requirement
statement = "The system shall be available 24 hours a day, 7 days a week, with no more than 10% down time."
requiredBy/requires
Req3: Requirement
statement = "The course informatuon available is limited to the type of data supported by the database."
constrains/ constrainedBy
metBy/meets
Req1: Requirement
statement = "The system shall display available courses to the student upon request."
req_type = "PersonaReq", "SystemReq"
owner = "Susan Miller"
release_number = "req011512"
date_created = "01152012"
priority = "0.9"
risk_level = "0.3"
status = "Accepted"
verification = "Verified"
validation = "Validated"
refineInto/refinesFrom
Req1.1: Requirement
statement = "The system shall display available courses to the student based on student's query criteria."
refineInto/refinesFrom
Req1.2: Requirement
statement = "The system shall display message if student's request does not succeed."
operationalizedBy/operationalizes
isReqOf/hasReq
FuncReq1: Functional
category = 'Informational"
Behavioral - GST Ontology
Scenario1.0 Search for classes: Scenario
name = "Search for classes."

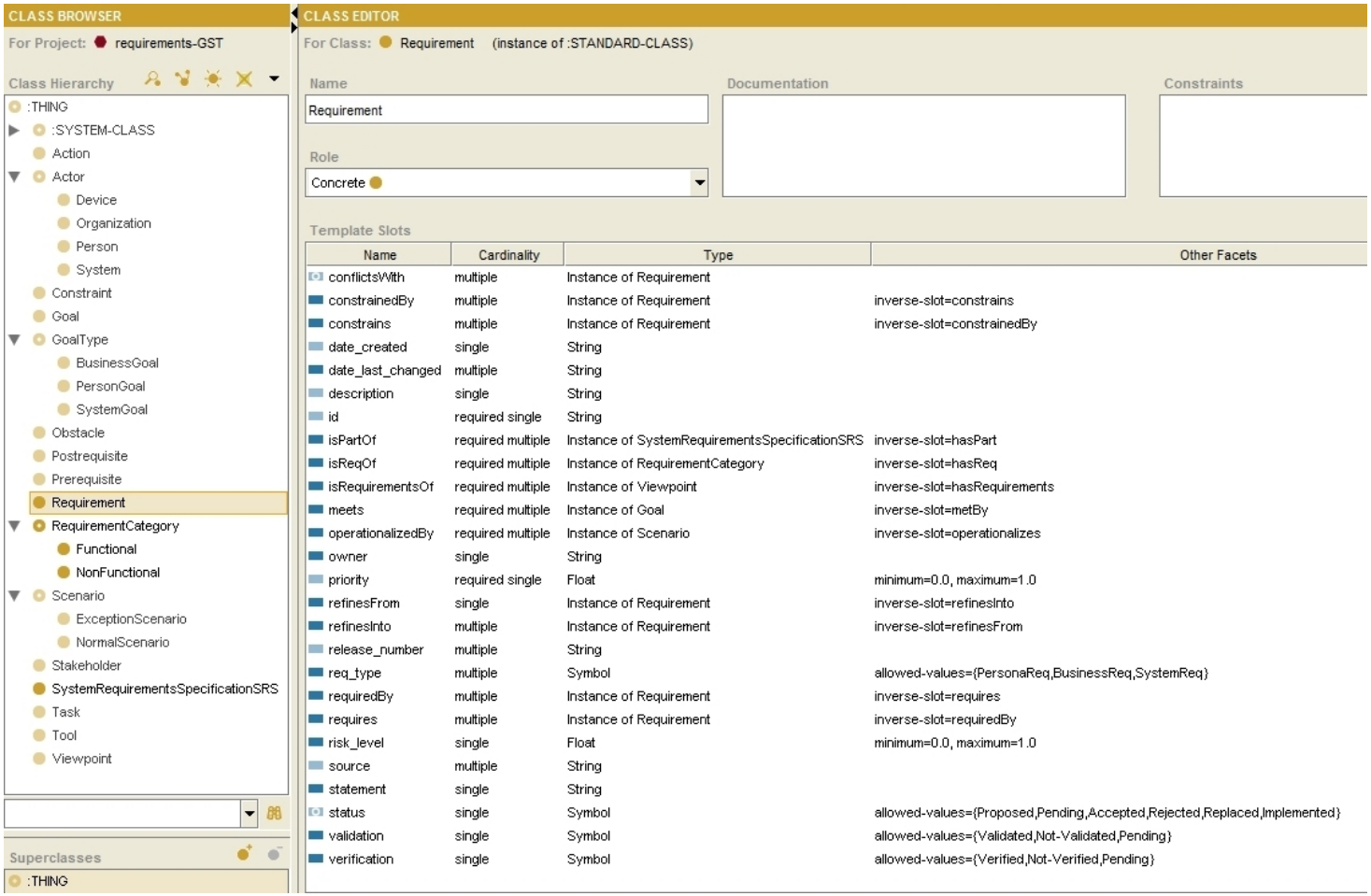

INSTANCE BROWSER
For Class: Persona
persona_title
Linda Rose, The Busy Undergraduate Student & Web Developer
Types
Persona
INSTANCE EDITOR
For Instance: Linda Rose, The Busy Undergraduate Student & Web Developer (instance of Persona, internal name is persona_Class10002)
id
PERS1
persona_title
Linda Rose, The Busy Undergraduate Student & Web Developer
persona_type
Primary
personifies (Role)
Full-Time Student
hasPersonaName (Name)
Linda Christine Rose
gender
Female
age
Young:25-34
hasHighestEducation (Education)
Associate
marital_status
Single
social_level
Middle-Class
engagesIn (Environment)
Starbucks
hasOccupation (Occupation)
Web Developer
hasKnowledge (Knowledge)
Computer Knowledge : Software (office applications) : Excellent
Computer Knowledge : Hardware : Good
Computer Knowledge : Programming (Java, C++, Visual Basic) : Good
Computer Knowledge : Web development (HTML, CSS, JavaScript) : Good
Computer Knowledge : Operating systems (Windows 8/7/XP, UNIX) : Good
Web Knowledge : Internet : Good
Web Knowledge : Social media : Good
Domain Knowledge : UML : Good
Domain Knowledge : Requirements engineering : Good
hasConcern (Concern)
Will the website interface and information be properly presented on cell phone?
Will the website be easy to navigate that does not take too much time?
Will the website content be properly presented without much confusion?
Will the public WiFi Connection be secured?
Will be public WiFi connection be available when needed?
Will the system be available for service when needed?
Will the online payment functuon properly?
hasLanguageProficiency (Language Proficien
Writing : English : Good
Writing : Spanish : Fair
Speaking : English : Excellent
Speaking : Spanish : Good
Listening : English : Excellent
Listening : Spanish : Good
hasAbility (Ability)
Cognitive : Memory : : Average
Cognitive : Problem-Solving : : High
Cognitive : Attention : : High
Cognitive : Reading-Comphrension : : Average
Cognitive : Visual-Comprehension : : Average
Cognitive : Spatial : : Average
Physical : Motor : : Average
Physical : Sensory-Speech : : Average
Physical : Sensory-Visual : : Average
Physical : Other : Color Recognition : Low
hasInterest (Interest)
Listening music
Playing guitar
Surfing the Web
Shop online


INSTANCE BROWSER
For Class: Goal
multiple slots
Goal1.0 Browse and enroll for courses with fast res
Goal1.1 Browse courses with ease.
Goal1.2 Register courses with ease.
Goal1.3 Fast system response from users' requests
Goal2.0 Less error when using the system.
Goal3.0 Simple and uncomplicared interface.
Goal4.0 View enrolled courses information
Goal5.0 Achieve course target or max enrollnment.
Types
Goal
INSTANCE EDITOR
For Instance: Goal1.0 Browse and enroll for courses with fast response. (instance of Goal, internal name is Behavioral-GST_Class10002)
id
Goal1.0
name
Browse and enroll for courses with fast response.
priority
0.9
status
Met
goal category
Functional
date created
01012013
description
To quickly browse for classes and enroll for classes online with fast response from the system on users' requests.
ownership
Steven Hope
hasGoalType (Goal Type)
Person Goal
System Goal
isGoalOf (Viewpoint)
Linda Rose, Part-Time Graduate Student, Starbucks (mid-work day)
hasStakeholder (Stakeholder)
Steven Hope
hasConstraint (Constraint)
AND-refinesInto
id
name
priority
AND-refinesFrom
Goal1.1
Browse courses with ease.
0.8
Goal1.0 Browse and enroll for courses with fast response.
Goal1.2
Register courses with ease.
0.8
Goal1.0 Browse and enroll for courses with fast response.
Goal1.3
Fast system response from users' requests.
0.8
Goal1.0 Browse and enroll for courses with fast response.
AND-refinesFrom
id
name
priority
AND-refinesInto


INSTANCE BROWSER
For Class: Requirement
multiple slots
REQ1 The system shall display available courses to the
REQ1.1 The system shall display available courses to th
REQ1.2 The system shall display message if student's re
REQ2 The system shall be available 24 hours a day, 7 d
REQ3 The course information available is limited to the ty
REQ4 The system shall enable users to query for inform
REQ5 The system shall provide uncomplicated navigatio
REQ6 The system shall ensure that content does not clu
REQ7 The system shall display page layout and informat
REQ8 The system shall ensure that all links, menus, or b
REQ9 The system shall ensure that all confidential inform
Types
Requirement
INSTANCE EDITOR
For Instance: REQ1 The system shall display available courses to the student upon request. (instance of Requirement, internal name is requirements_Class62)
Id
REQ1
statement (Requirement)
The system shall display available courses to the student upon request.
priority
0.9
risk level
0.3
status
Accepted
description
owner
Susan Miller
date created
01152012
date last change
source
doc1
release number
req011512
isPartOf
SRS - v1000.01
validation
Validated
verification
Verified
req type
PersonaReq
SystemReq
IsReqOf (Requirement Category)
REQ2f Informational (Functional)
refinesFrom (Requirement)
refinesInto (Requirement)
REQ1.1 The system shall display available courses to the student based on student's query criteria.
REQ1.2 The system shall display message if student's request does not succeed.
requires (Requirement)
REQ4 The system shall enable users to query for information based upon course name, cour
requiredBy (Requirement)
constraints (Requirement)
REQ2 The system shall be available 24 hours a day, 7 days a week, with no more than 10%
constrainedBy (Requirement)
REQ3 The course information available is limited to the type of data supported by the databas

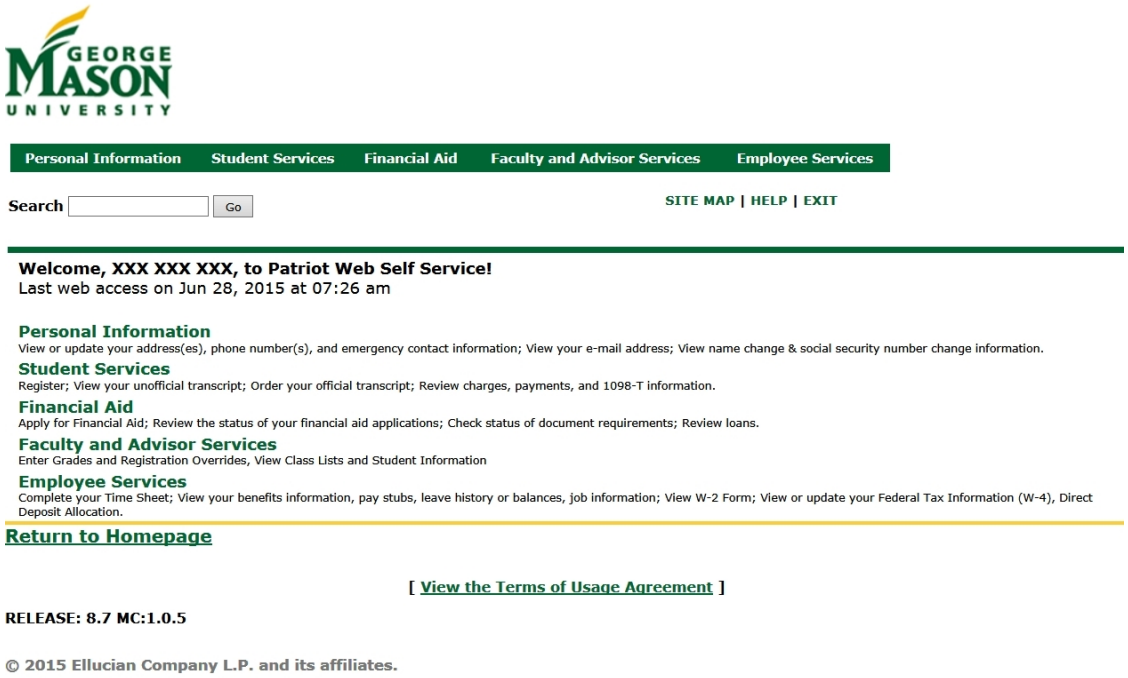

CLASS BROWSER
For Project: persona-GST
Class H
:THING
:SYSTEM-CLASS
Ability
Action
Actor
Concern
Constraint
Education
Environment
Goal
GoalType
Interest
InterestCategory
Knowledge
LanguageProficiency
Name
Obstacle
Occupation
Persona
Postrequisite
Prerequisite
Role
Scenario
Stakeholder
Task
Tool
UsabilityPref
Viewpoint
CLASS EDITOR
For Class: Persona (instance of :STANDARD-CLASS)
Name
Persona
Documentation
Constraints
Role
Concrete
Template Slots
Name
Cardinality
Type
Other Facets
age
single
Symbol
allowed-values={Young:18-24,Young:25-34,Middle:35-44,Middle:45-54,Older:55-64,Elderly:>65}
engagesIn
multiple
Instance of Environment
inverse-slot=engagedBy
gender
single
Symbol
allowed-values={Male,Female}
hasAbility
multiple
Instance of Ability
inverse-slot=isAbilityOf
hasConcern
multiple
Instance of Concern
inverse-slot=isConcernOf
hasHighestEducation
single
Instance of Education
inverse-slot=isHighestEducationOf
hasInterest
multiple
Instance of Interest
inverse-slot=isInterestOf
hasKnowledge
multiple
Instance of Knowledge
inverse-slot=isKnowledgeOf
hasLanguageProficiency
multiple
Instance of LanguageProficiency
inverse-slot=isLanguageProficiencyOf
hasOccupation
multiple
Instance of Occupation
inverse-slot=isOccupationOf
hasPersonaName
required single
Instance of Name
inverse-slot=isPersonaNameOf
id
required single
String
isPersonaOf
required multiple
Instance of Viewpoint
inverse-slot=hasPersona
marital_status
single
Symbol
allowed-values={Single,Married,Divorced,Separated,Widowed}
persona_title
single
String
persona_type
single
Symbol
allowed-values={Primary,Secondary,Served,Customer}
personifies
required multiple
Instance of Role
inverse-slot=personifiedBy
social_level
single
Symbol
allowed-values={Upper-Class,Middle-Class,Working-Class}


CLASS BROWSER
For Project: GST-persona-requirement
Class Hierarchy
:THING
:SYSTEM-CLASS
Ability
Action
Actor
Concern
Constraint
Education
Environment
Goal
GoalType
Interest
InterestCategory
Knowledge
LanguageProficiency
Name
Obstacle
Occupation
Persona
Postrequisite
Prerequisite
Requirement
RequirementCategory
Role
Scenario
Stakeholder
SystemRequirementsSpecificationSRS
Task
Tool
UsabilityPref
Viewpoint
CLASS EDITOR
For Class: Goal (instance of :STANDARD-CLASS)
Name
Goal
Documentation
Role
Concrete
Template Slots
Name | Cardinality | Type | Other Facets
AND-refinesFrom | multiple | Instance of Goal | inverse-slot=AND-refinesInto
AND-refinesInto | required multiple | Instance of Goal | inverse-slot=AND-refinesFrom
conflictsWith | multiple | Instance of Goal | inverse-slot=conflictsWith
date_created | single | String |
description | single | String |
goal_category | single | Symbol | allowed-values={Functional,Non-Functional}
hasConstraint | multiple | Instance of Constraint | inverse-slot=isConstraintOf
hasGoalType | required multiple | Instance of GoalType | inverse-slot=isGoalTypeOf
hasObstacle | multiple | Instance of Obstacle | inverse-slot=isObstacleOf
hasStakeholder | required multiple | Instance of Stakeholder | inverse-slot=isStakeholderOf
id | required single | String |
isGoalOf | required multiple | Instance of Viewpoint | inverse-slot=hasGoal
metBy | required multiple | Instance of Requirement | inverse-slot=meets
name | required single | String |
OR-refinesFrom | multiple | Instance of Goal | inverse-slot=OR-refinesInto
OR-refinesInto | multiple | Instance of Goal | inverse-slot=OR-refinesFrom
ownership | multiple | String |
priority | single | Float | minimum=0.0, maximum=1.0
status | single | Symbol | allowed-values={Met,Not-Met,Partially-Met,Withdrawn,Unknown}
supportedBy | multiple | Instance of Goal | inverse-slot=supports
supports | multiple | Instance of Goal | inverse-slot=supportedBy

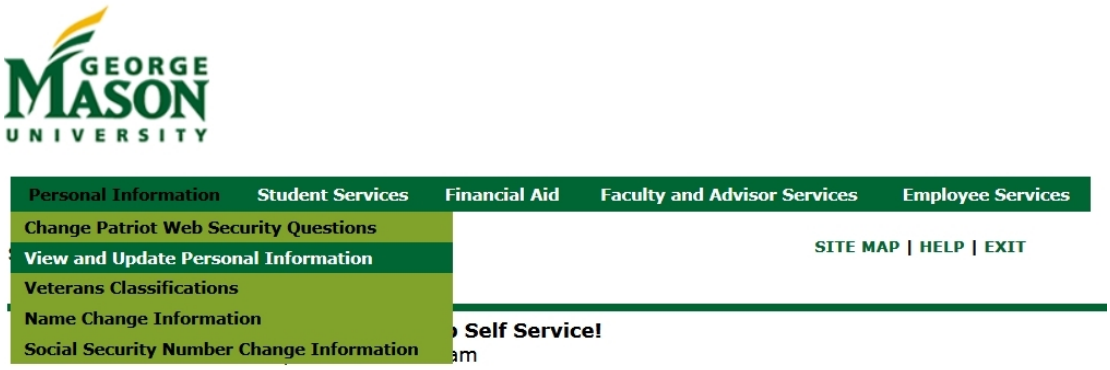

CLASS BROWSER
For Project: requirements-GST
Class Hierarchy
:THING
:SYSTEM-CLASS
Action
Actor
Device
Organization
Person
System
Constraint
Goal
GoalType
BusinessGoal
PersonGoal
SystemGoal
Obstacle
Postrequisite
Prerequisite
Requirement
RequirementCategory
Functional
NonFunctional
Scenario
ExceptionScenario
NormalScenario
Stakeholder
SystemRequirementsSpecificationSRS
Task
Tool
Viewpoint
Superclasses
:THING
CLASS EDITOR
For Class: Requirement (instance of :STANDARD-CLASS)
Name
Requirement
Documentation
Constraints
Role
Concrete
Template Slots
Name | Cardinality | Type | Other Facets
conflictsWith | multiple | Instance of Requirement
constrainedBy | multiple | Instance of Requirement | inverse-slot=constrains
constrains | multiple | Instance of Requirement | inverse-slot=constrainedBy
date_created | single | String
date_last_changed | multiple | String
description | single | String
id | required single | String
isPartOf | required multiple | Instance of SystemRequirementsSpecificationSRS | inverse-slot=hasPart
isReqOf | required multiple | Instance of RequirementCategory | inverse-slot=hasReq
isRequirementsOf | required multiple | Instance of Viewpoint | inverse-slot=hasRequirements
meets | required multiple | Instance of Goal | inverse-slot=metBy
operationalizedBy | required multiple | Instance of Scenario | inverse-slot=operationalizes
owner | single | String
priority | required single | Float | minimum=0.0, maximum=1.0
refinesFrom | single | Instance of Requirement | inverse-slot=refinesInto
refinesInto | multiple | Instance of Requirement | inverse-slot=refinesFrom
release_number | multiple | String
req_type | multiple | Symbol | allowed-values={PersonaReq,BusinessReq,SystemReq}
requiredBy | multiple | Instance of Requirement | inverse-slot=requires
requires | multiple | Instance of Requirement | inverse-slot=requiredBy
risk_level | single | Float | minimum=0.0, maximum=1.0
source | multiple | String
statement | single | String
status | single | Symbol | allowed-values={Proposed,Pending,Accepted,Rejected,Replaced,Implemented}
validation | single | Symbol | allowed-values={Validated,Not-Validated,Pending}
verification | single | Symbol | allowed-values={Verified,Not-Verified,Pending}

APPENDIX E: PROTOTYPE PATRIOTWEB WEBSITE NEW FEATURES AND IMPROVEMENTS SCREENSHOTS

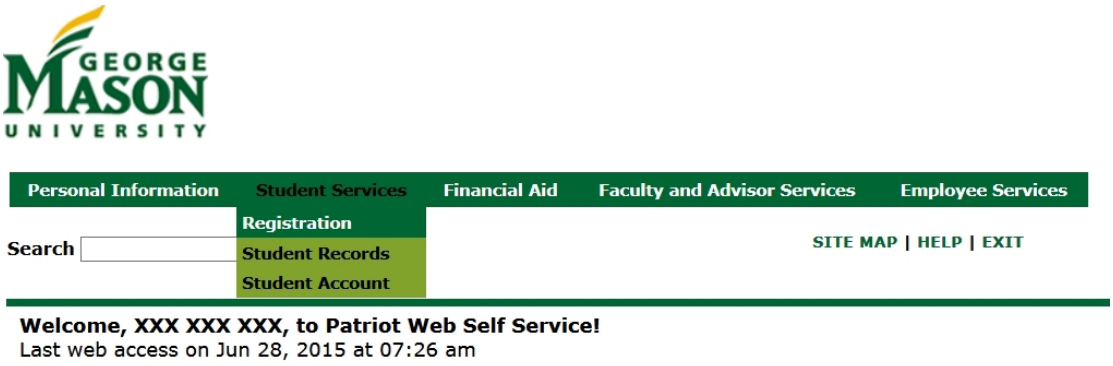
Main Menu page



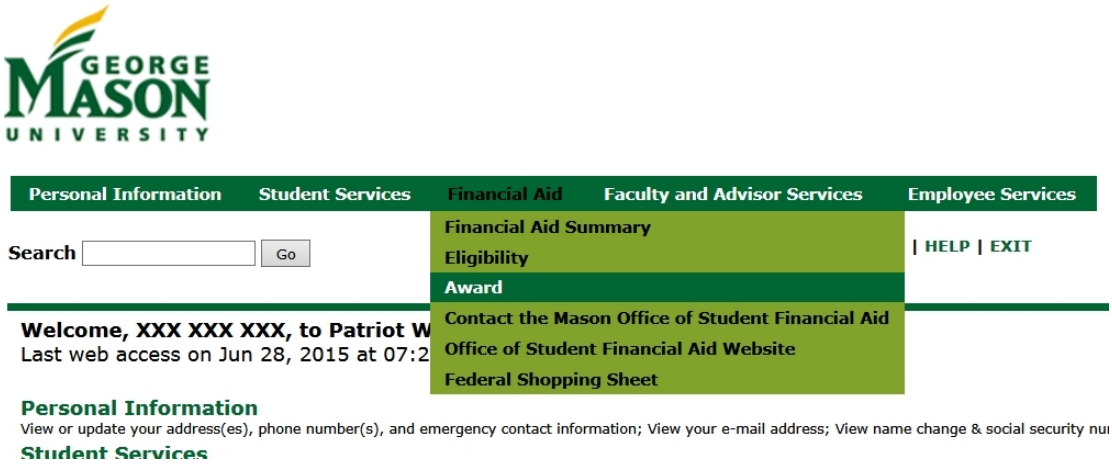
Persona Information Tab – New Drop-Down Menu



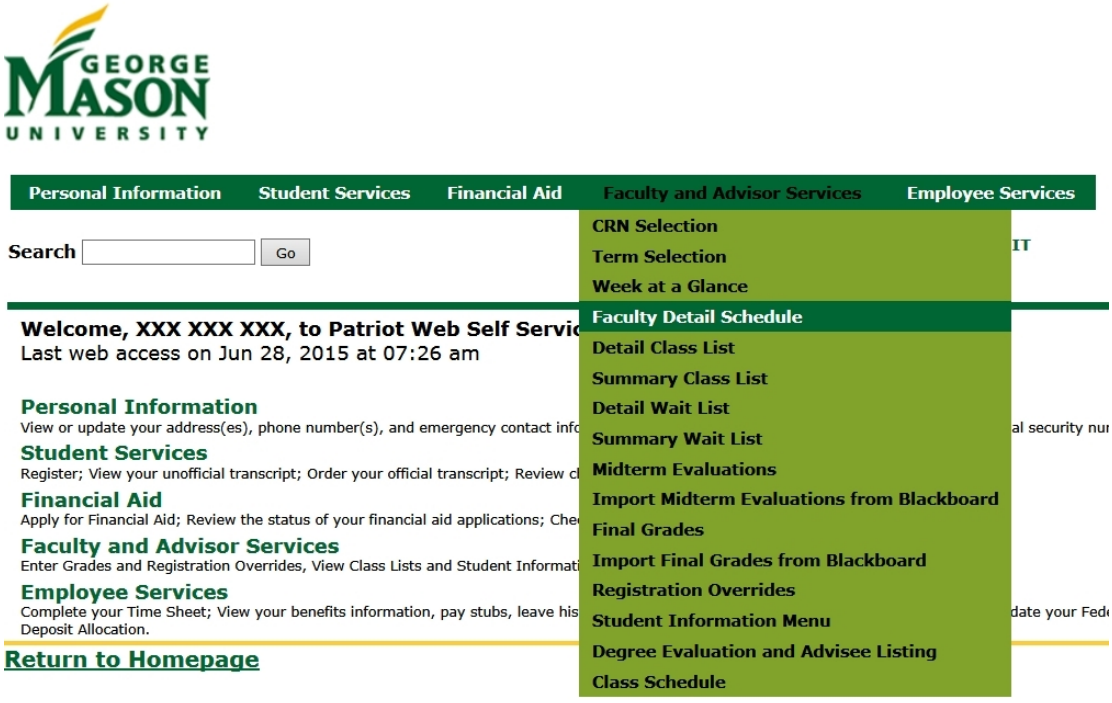
Student Services Tab – New Drop-Down Menu



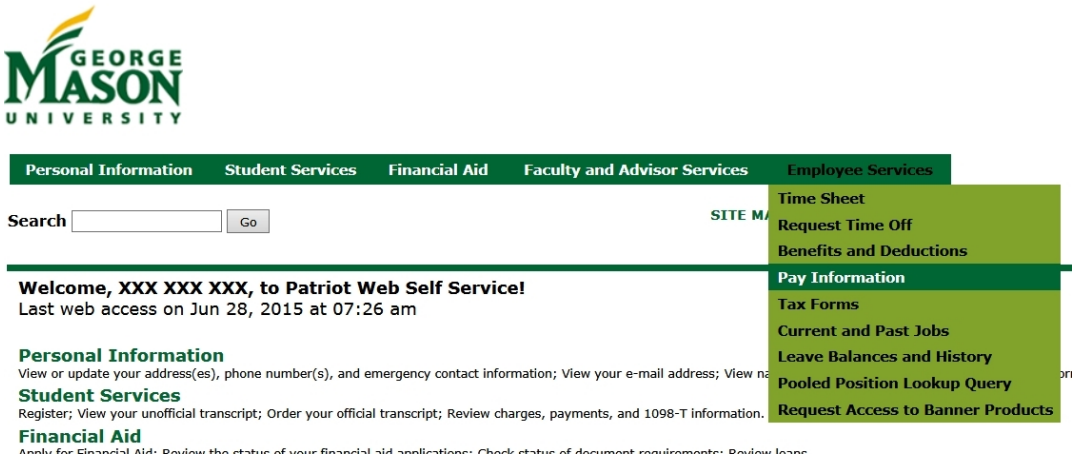
Financial Aid Tab – New Drop-Down Menu



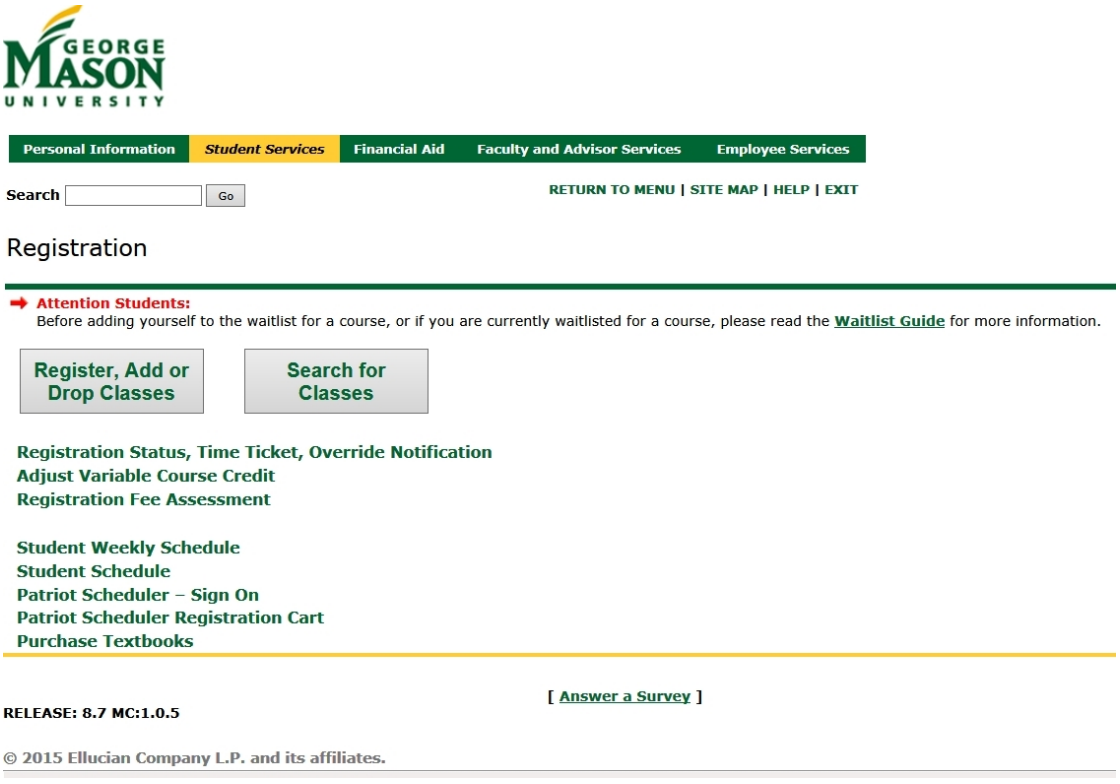
Faculty and Advisor Service Tab – New Drop-Down Menu



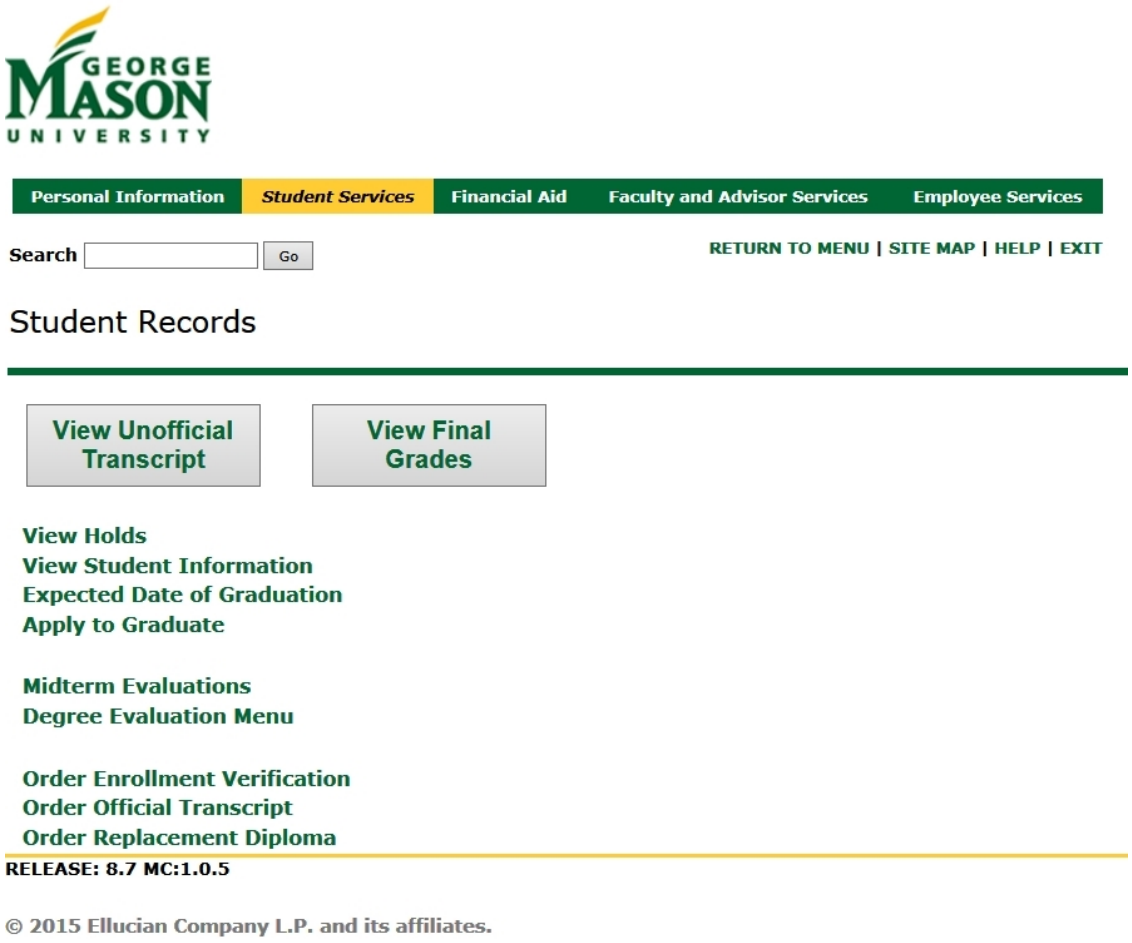
Employee Services Tab – New Drop-Down Menu



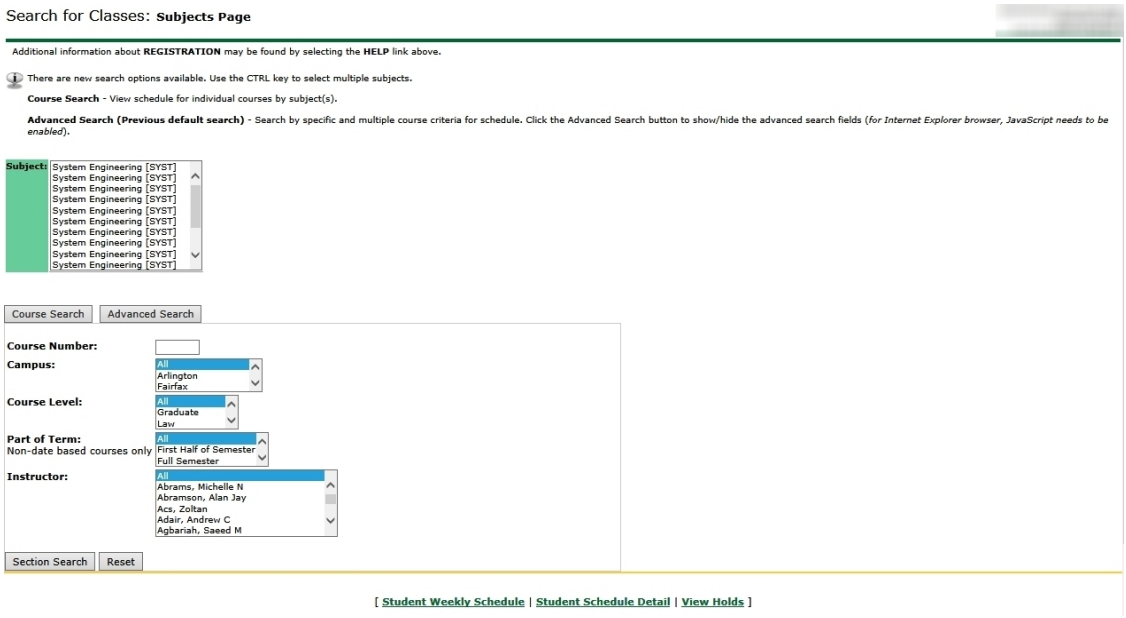
Registration Page – Improved "Register, Add or Drop Classes" and "Search for Classes" Buttons



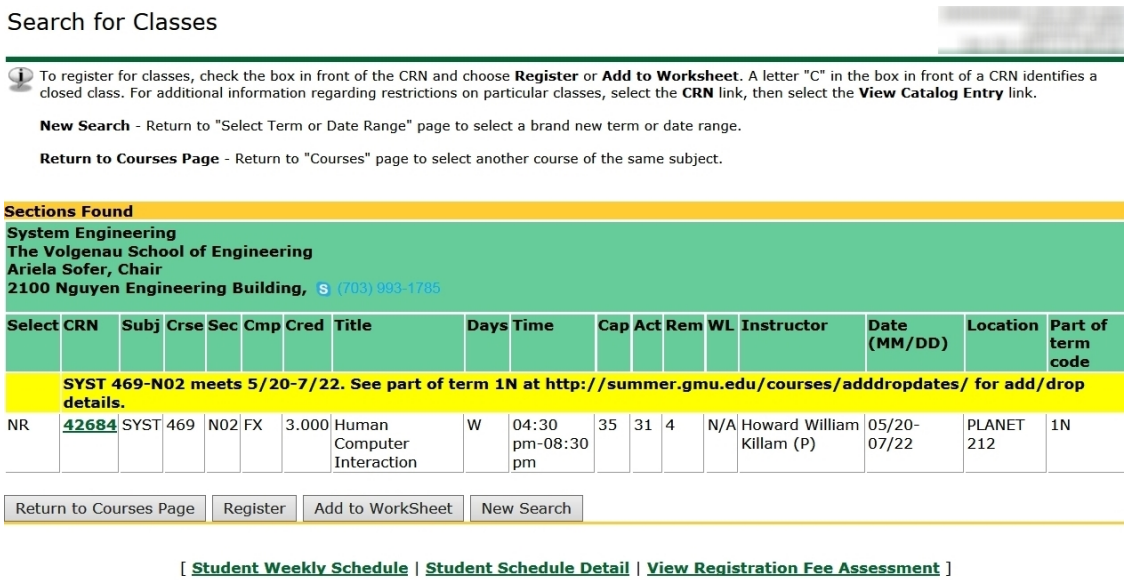
Student Records Page – Improved "View Unofficial Transcript" and "View Final Grades" Buttons



Search for Classes – New Advanced Search Field Options Panel



Search for Classes – New "Return to Courses Page" Button



Search for Classes – New "Return to Subjects Page" Button

Search for Classes

To register for classes, check the box in front of the CRN and choose **Register** or **Add to Worksheet**. A letter "C" in the box in front of a CRN identifies a closed class. For additional information regarding restrictions on particular classes, select the **CRN** link, then select the **View Catalog Entry** link.

New Search - Return to "Select Term or Date Range" page to select a brand new term or date range.

Return to Subjects Page - Return to "Subjects" page to select brand new subject of the same term or date range.

Sections Found

System Engineering
The Volgenau School of Engineering
Ariela Sofer, Chair
2100 Nguyen Engineering Building, (703) 993-1785

Select	CRN	Subj	Crse	Sec	Cmp	Cred	Title	Days	Time	Cap	Act	Rem	WL	Instructor	Date (MM/DD)	Location	Part of term code
	SYST 463-N01 Flight Training Lab II, meeting times and location are by appointment; see Dr. Lance Sherry in ENGR 4507 for arrangements. For fees and other important information, please go to http://seor.gmu.edu/bsse/aviationminor.html. Contact instructor for more information. See part of term 1N at http://summer.gmu.edu/courses/adddropdates/ for add/drop details.																
NR	42981	SYST	463	N01	OC	3.000	Flight Training Lab II		TBA	5	0	5	N/A	TBA	05/18-08/05	CANCEL CRSE	1N
	SYST 469-N02 meets 5/20-7/22. See part of term 1N at http://summer.gmu.edu/courses/adddropdates/ for add/drop details.																
NR	42684	SYST	469	N02	FX	3.000	Human Computer Interaction	W	04:30 pm-08:30 pm	35	31	4	N/A	Howard William Killam (P)	05/20-07/22	PLANET 212	1N
	SYST 659-N01 meets 5/19-7/21. See part of term 1N at http://summer.gmu.edu/courses/adddropdates/ for add/drop details.																
NR	41781	SYST	659	N01	FX	3.000	Risk Modelling & Management	T	04:30 pm-08:30 pm	15	6	9	N/A	Frederick P Wieland (P)	05/19-07/21	ENGR 2608	1N
	SYST 659-N02 meets 5/19-7/21 and is a distance education section. See part of term 1N at http://summer.gmu.edu/courses/adddropdates/ for add/drop details.																
NR	41782	SYST	659	N02	NE	3.000	Risk Modeling & Management	T	04:30 pm-08:30 pm	20	12	8	N/A	Frederick P Wieland (P)	05/19-07/21	NET NET	1N

Return to Subjects Page | Register | Add to WorkSheet | New Search

[Student Weekly Schedule | Student Schedule Detail | View Registration Fee Assessment]

APPENDIX F: EVALUATION – PART 1 SURVEY

EFFECTIVENESS OF PERSONA IN USER REQUIREMENTS ENGINEERING

INFORMED CONSENT FORM

RESEARCH PROCEDURES
This research is being conducted to help test the hypotheses of my Ph.D. research in evaluating the effectiveness of the concept of persona in user requirements modeling and analysis. If you agree to participate, you will be asked to perform some tasks by following through the printed screenshots which will be provided to you, or on a computer if you have accessed to a computer (laptop or desktop), After performing the tasks, you will complete a survey. The entire process should take approx. 15 to 20 minutes.

RISKS
There are no foreseeable risks for participating in this research.

BENEFITS
There are no benefits to you as a participant other than to further research in the application of persona in user requirements modeling.

CONFIDENTIALITY
The data in this study will be confidential. Your name, date of birth, SSN, home address, and other sensitive information identifiable with you will not be asked in the survey. Your replies will be completely anonymous, so do not put your name anywhere on the survey form. All information collected in this survey will be used only for my Ph.D. research and will be kept confidential. There will be no connection to you specifically in the results or in future publication of the results.

PARTICIPATION
Your participation is voluntary, and you may withdraw from the study at any time and for any reason. If you decide not to participate or if you withdraw from the study, there is no penalty or loss of benefits to which you are otherwise entitled. There are no costs to you or any other party. You may choose to not answer any survey question by simply leaving it blank. If you choose to not participate in this survey, you may either return the blank survey form or you may discard the survey form.

CONTACT
This research is being conducted by Wee Wee Sim and Dr. Peggy Brouse at George Mason University. Mr. Sim may be reached at (202) 737-4640 for questions or to report a research-related problem. Dr. Brouse may be reached at (703) 993-1502. You may contact the George Mason University Office of Research Integrity & Assurance at 703-993-4121 if you have questions or comments regarding your rights as a participant in the research.

This research has been reviewed according to George Mason University procedures governing your participation in this research.

CONSENT
I have read this form, all of my questions have been answered by the research staff, and I agree to participate in this study. If you do not wish to participate, feel free to discard this form.

_____
Name

_____
Date of Signature

Instructions for the participants

1. **Carefully go through the printed screenshots provided to you.**
   The screenshots are webpage transitions pertaining to the steps taken to perform five scenarios on the current PatriotWeb system and the PatriotWeb prototype system (*modified version of the* current PatriotWeb system). The five scenarios (provided to you as screenshots) are:
   - "View Student Services Page"
   - "View Registration Page"
   - "View Student Records Page"
   - "View Personal Information Page"
   - "Search for Classes".

2. **If you have a computer with you, you may perform the steps of the five scenarios using your computer by accessing the current PartiotWeb website and the PatriotWeb prototype website.**
   Current PatriotWeb website: https://patriotweb.gmu.edu/
   PatriotWeb prototype website: http://mason.gmu.edu/~wsim/Main_Menu.htm
   (*For the PatriotWeb prototype website, you do not need to login*)

3. **After you have followed the step-by-step tasks of the five scenarios via the printed screenshots or on your computer, complete the survey questions.**
   There are three sections on the survey:
   Section A contains general questions related to the participant who is completing this survey.
   Section B contains general questions related to how the participant who is completing this survey uses the PatriotWeb system.
   Section C contains specific questions related to the comparison of the current PatriotWeb system and the PatriotWeb prototype system.

4. **Throughout this survey, if you have any questions, please do not hesitate to ask me.**

Thank you for agreeing to participate in this survey in assisting me test the hypotheses of my Ph.D. research.

Section A:
Participant General Questions

1. What is your age group? (*Please check ✓ one checkbox that applies to you*)

☐ 18-24 ☐ 25-34 ☐ 35-44 ☐ 45-54 ☐ 55-64 ☐ >65

2. What is your gender? (*Please check ✓ one checkbox that applies to you*)

☐ Male ☐ Female

3. What is your marital status? (*Please check ✓ one checkbox that applies to you*)

☐ Married ☐ Single

4. What is your current school status? (*Please check ✓ one checkbox that applies to you*)

☐ Full-Time ☐ Part-Time

5. What is your current GMU degree program? _____

6. What is your education background? (*Please check ✓ the relevant checkboxes that apply to you*)

☐ High School Diploma or Equivalent (GED)
☐ Vocational
☐ Associate
☐ Bachelor's
☐ Master's
☐ Doctoral
☐ Professional

7. **Please provide information about your CURRENT and/or PAST employment(s). Please provide minimum one to maximum two employments.** *(Please skip this question if you do not have any employments)*

Job Title	Job #1:	Job #2:
Current Job? *(Please check ✓ one checkbox that applies to you)*	☐	☐
Job Status *(Please check ✓ one checkbox that applies to you)*	☐ Full-Time ☐ Part-Time ☐ Freelance/ Self-Employed	☐ Full-Time ☐ Part-Time ☐ Freelance/ Self-Employed
Years of Employment *(Please check ✓ one checkbox that applies to you)*	☐ <1year ☐ 1-3years ☐ 3-6years ☐ 6-10years ☐ >10years	☐ <1year ☐ 1-3years ☐ 3-6years ☐ 6-10years ☐ >10years

8. **What device(s) do you currently own?** *(Please check ✓ the relevant checkboxes that apply to you)*

☐ Desktop Computer

☐ Laptop

☐ Tablet

☐ Mobile/Smart Phone (Internet-capable)

☐ None of the above

9. **In an average week, how often do you use the following devices (listed in first column in below table)?** *(For each device, please check ✓ one checkbox that BEST applies to you)*

Device	Never	1-3 Times A Week	4-6 Times A Week	Every Day	Several Times A Day, Every Day
Desktop Computer	☐	☐	☐	☐	☐
Laptop	☐	☐	☐	☐	☐
Mobile/Smart Phone	☐	☐	☐	☐	☐
Tablet	☐	☐	☐	☐	☐

10. **How comfortable are you with using the following devices (listed in first column in below table) for basic internet browsing?** (*For each device, please check ✓ one checkbox that BEST applies to you*)

Device	Not At All Comfortable	Slightly Comfortable	Somewhat Comfortable	Comfortable	Very Comfortable
Desktop Computer	☐	☐	☐	☐	☐
Laptop	☐	☐	☐	☐	☐
Mobile/Smart Phone	☐	☐	☐	☐	☐
Tablet	☐	☐	☐	☐	☐

11. **How comfortable are you with using the following computing platforms (listed in first column in below table)?** (*For each platform, please check ✓ one checkbox that BEST applies to you*)

Platform	Not At All Comfortable	Slightly Comfortable	Somewhat Comfortable	Comfortable	Very Comfortable
UNIX	☐	☐	☐	☐	☐
LINUX	☐	☐	☐	☐	☐
WIDNOWS (8, 7, XP)	☐	☐	☐	☐	☐

12. **How familiar (in general) are you with the following web literacy items?** (*For each web literacy item, please check ✓ one checkbox that BEST applies to you*)

Web Literacy	Not At All Familiar	Slightly Familiar	Somewhat Familiar	Familiar	Very Familiar
Basic Internet searching (e.g. using Google, Yahoo, or other search engines.)	☐	☐	☐	☐	☐
Downloading or uploading files (pdfs, word files, music, videos, graphics, etc.) on the Internet.	☐	☐	☐	☐	☐
Basic HTML, CSS.	☐	☐	☐	☐	☐
Basic programming languages (e.g. Java, Visual Basic, C++, C)	☐	☐	☐	☐	☐
Adjusting your browser settings.	☐	☐	☐	☐	☐

Section B:
Current PatriotWeb System

13. **In a weekly basis, how often (on average) do you use the GMU PatriotWeb system?** (*Please check one checkbox that applies to you*)

☐ 0-5 times per week ☐ 6-10 times per week ☐ >10 times per week

14. **What device do you use <u>most often</u> to access the GMU PatriotWeb system?** (*Please check ✓ one checkbox that applies to you*)

☐ Desktop Computer ☐ Laptop ☐ Tablet ☐ Mobile/Smart Phone

15. **How do you access GMU PatriotWeb system when you are in a public (outdoor) place where desktop computers are not available?** (*Please list the device(s), e.g. laptop, mobile phone, etc.*)

_____

_____

16. **In what environment do you use the PatriotWeb system <u>most often</u>?**

(*Please check ✓ one checkbox that applies to you*)

☐ Off-Campus Private Home

☐ Campus Dormitory

☐ Public Places (*Any outdoor places, including on-campus facilities*, but excluding campus dormitory*)

**on campus facilities include library, student union, classroom buildings, etc.*

17. **In what part of the day do you usually access the GMU PatriotWeb system?** (*Please check ✓ one checkbox that <u>BEST</u> applies to you*)

☐ Morning ☐ Lunch Hour (12pm-1pm) ☐ Afternoon ☐ Evening ☐ Night

18. **List up to three services you usually use when using the PatriotWeb system? Examples of services: search for classes, register for classes, check financial aid, check grades, etc.**

Service 1	Service 2	Service 3

19. **In your opinion, how important to you are the following factors when using the PatriotWeb system in general (examples of general usages: search for classes, register for classes)?** *(For each factor, please check ✓ one checkbox that applies to you)*

Factor	Not At All Important	Low Importance	Somewhat Important	Important	Very Important
Ease in performing the tasks.	☐	☐	☐	☐	☐
Easy to navigate the website without too much learning. (i.e. navigation menus are not confusing)	☐	☐	☐	☐	☐
Number of mouse clicks or web pages needed to reach the destination page. (i.e. speed, task time)	☐	☐	☐	☐	☐
Visibility, consistency, or location of menus, links, or buttons.	☐	☐	☐	☐	☐
Menus, links, or buttons are easy to understand. (i.e. easy to understand what each menu, link, or button does)	☐	☐	☐	☐	☐
The outcome of clicking the menus, links, or buttons produce the result correctly. (i.e. as you would expect)	☐	☐	☐	☐	☐
Attractiveness of the web pages. (i.e. color, layout, font style, font size, etc.)	☐	☐	☐	☐	☐
Able to read the texts clearly.	☐	☐	☐	☐	☐

Able to remember how the system looks like and what the system does when you visit the system the next time (e.g. a week later).	☐	☐	☐	☐	☐

20. **In your opinion, do you feel that a <u>mobile version</u> of the GMU PatriotWeb system with streamlined most common or most used features is beneficial to you as a user of the system? If YES, please explain briefly. If NO, please also explain briefly.**

Section C:
Current PatriotWeb System and Prototype PatriotWeb System

***Question 21**, **question 22**, and **question 23** refer specifically to the tasks of the five scenarios you were asked to perform on both the current PatriotWeb System and the PatriotWeb prototype system. The five scenarios (provided to you as screenshots) are: "View Student Services Page", "View Registration Page", "View Student Records Page", "View Personal Information page", and "Search for Classes".*

You will perform the tasks by following the printed webpage transition screenshots provided to you. If you have a computer with you, you may access the current GMU PatriotWeb website and the PatriotWeb prototype website and follow the printed webpage transition screenshots to perform the tasks of the five scenarios on your computer. Both websites can be accessed at:
Current GMU PatriotWeb: https://patriotweb.gmu.edu/
PatriotWeb prototype: http://mason.gmu.edu/~wsim/Main_Menu.htm

21. **In COMPARISON to the current PatriotWeb system on the five scenarios, how would you rate the PatriotWeb prototype system according to the items listed in the first column? Feel free to refer the printed screenshots.** (*For each item, please check ✓one checkbox that applies to you*)

Usability Item	Worse	Same (No improvement)	Better (Improvement)
[Efficiency] **Number of clicks or pages needed to reach a target destination page. (i.e. speed, task time)**	☐	☐	☐
[Efficiency] **Number of steps that are repeated before reaching a target destination page.**	☐	☐	☐
[Comprehensibility] **Visibility, consistency, or location of menus, links, or buttons.**	☐	☐	☐
[Comprehensibility] **Menus, links, or buttons are easy to understand. (i.e. easy to understand what each menu, link, or button does)**	☐	☐	☐
[Comprehensibility] **Number of redundant or unnecessary menus, links, or buttons.**	☐	☐	☐

[Accuracy] **The outcome of clicking the menus, links, or buttons produce the result correctly. (i.e. as you would expect).**	☐	☐	☐
[Learnability] **Easy to learn how to navigate the site, how to carry out a task, where to click, etc.**	☐	☐	☐
[Rememberabilty] **Able to remember how the system looks like and what the system does.**	☐	☐	☐
[Reliability] **Information is credible.**	☐	☐	☐
[Attractiveness] **Attractiveness of the web pages. (Color, layout, font style/size, etc.)**	☐	☐	☐
[Clarity] **Texts are readable.**	☐	☐	☐

22. **Overall, as a user of the PatriotWeb system, how would you rate the PatriotWeb prototype system in COMPARISON to the current PatriotWeb system, based on the five scenarios provided to you as printed screenshots?** (*Please check ✓ one checkbox that applies to you*)

Worse	Same (No Improvement)	Better (Improvement)
☐	☐	☐

23. **Through performing the tasks on the five scenarios on both the current PatriotWeb system and the PatriotWeb prototype system, in your opinion, what factors are important to you as a user of the current PatriotWeb system? Please list one to three factors.**
(*Factors may include: speed, accuracy, no confusing links/menus, appearance, etc.*)

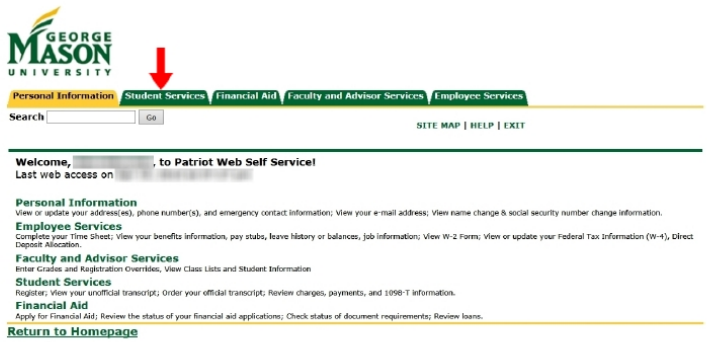
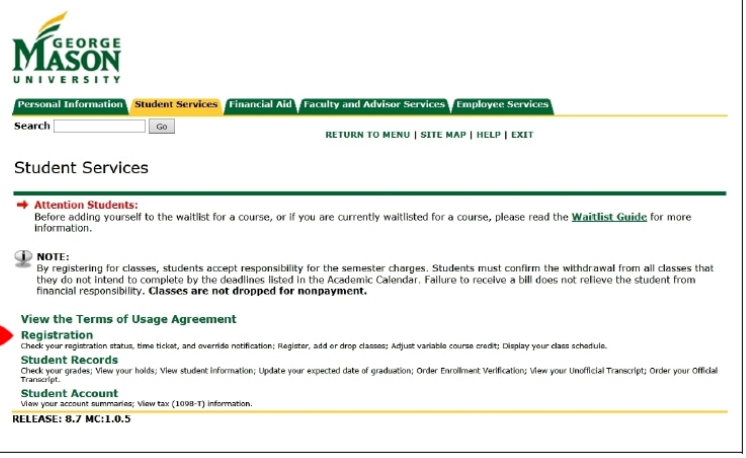
Factor 1	Factor 2	Factor 3

APPENDIX G: EVALUATION – PART 1 SURVEY SCREENSHOTS

Scenario 1: View Student Services Page

Current PatriotWeb System

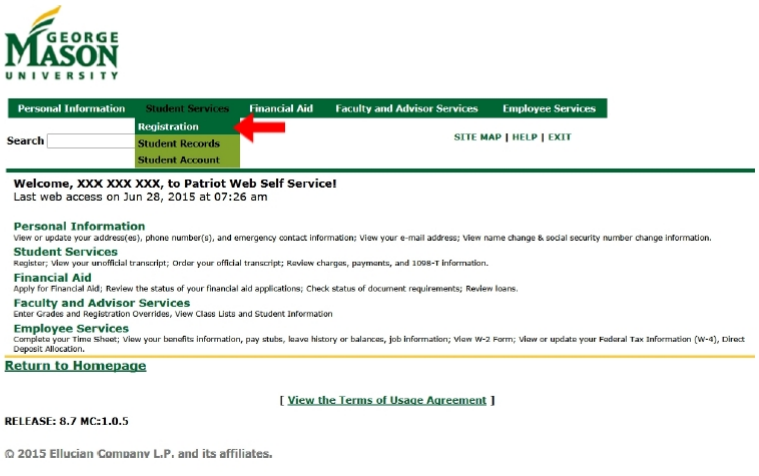
(*NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.*)

1	On Welcome page, click "**Student Services**" tab.	
2	On Student Services page, click "**Registration**" link.	

Scenario 1: View Student Services Page

Prototype PatriotWeb System

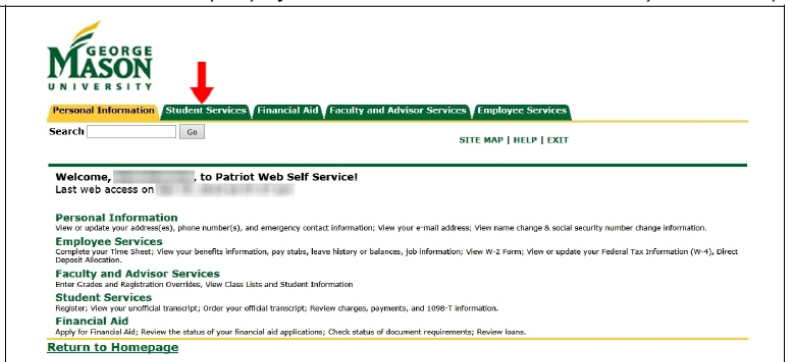
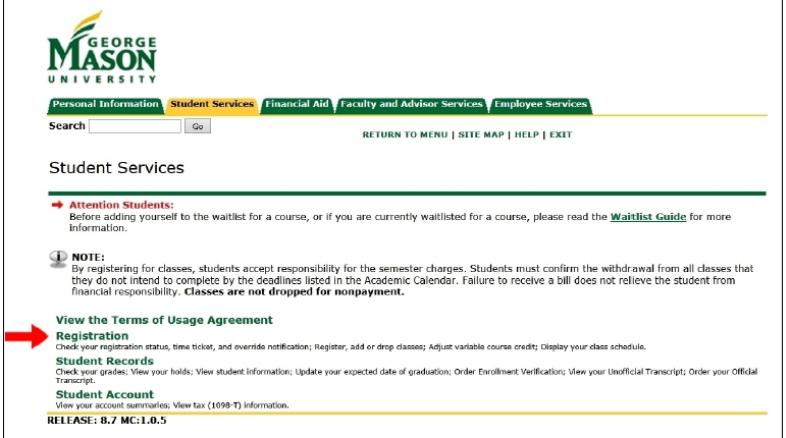
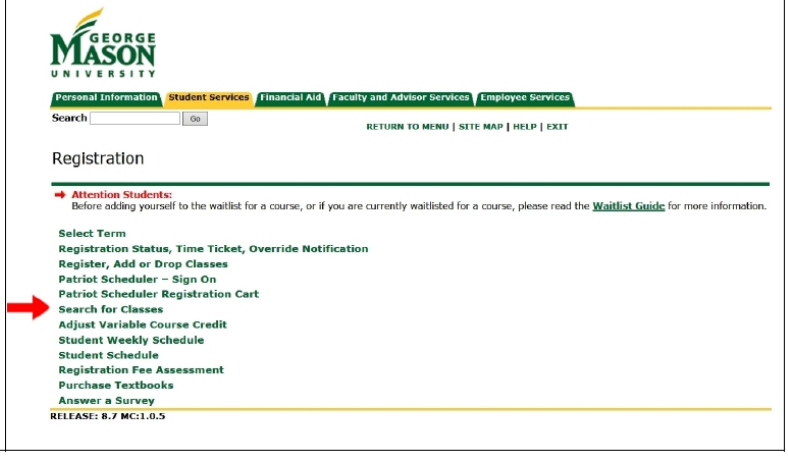
(*NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.*)

1	On Welcome page, mouse over "**Student Services**" tab. Select "Registration"link from the drop down menu.	

Scenario 2: View Registration Page

Current PatriotWeb System

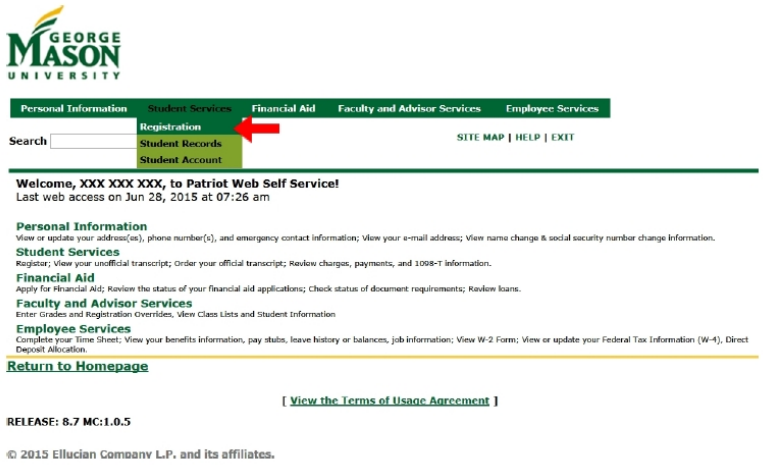
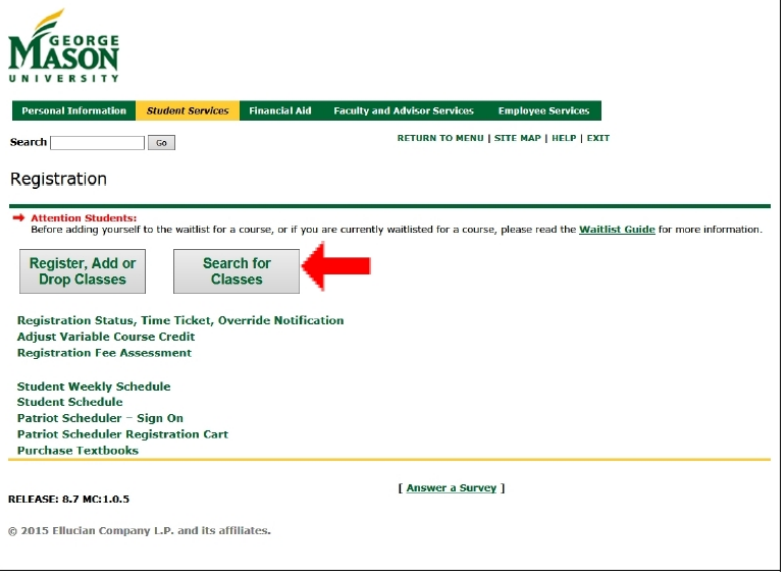
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

1	On Welcome page, click "**Student Services**" tab.	
2	On Student Services page, click "**Registration**" link.	
3	On Registration page, click "**Search for Classes**" link.	

Scenario 2: View Registration Page

Prototype PatriotWeb System

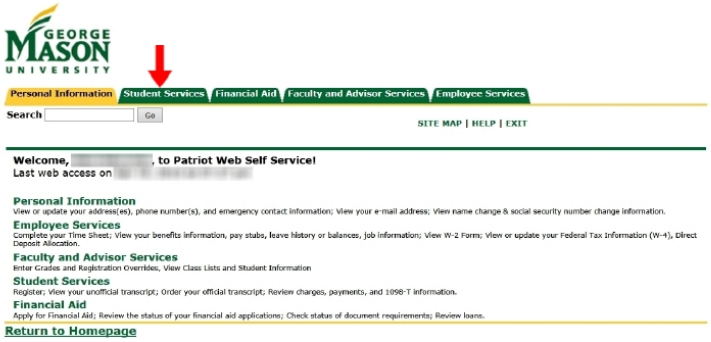
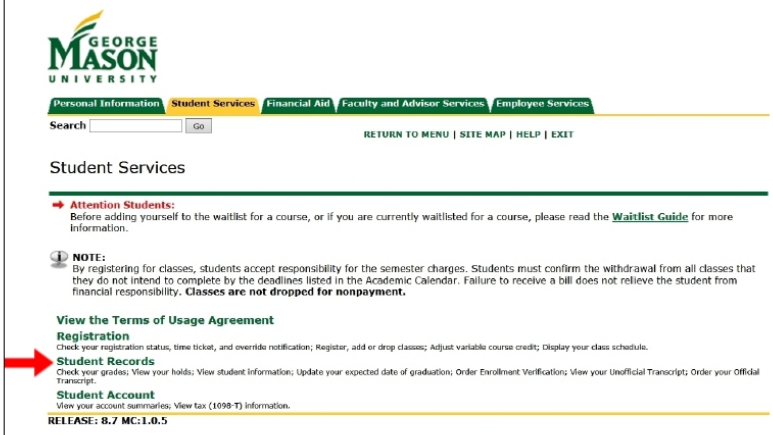
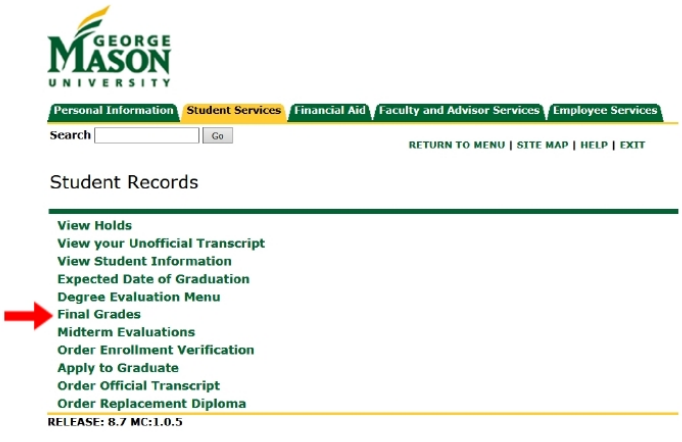
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

1	On Welcome page, mouse over "**Student Services**" tab. Select "**Registration**" link from the drop down menu list.	
2	On Registration page, click "**Search for Classes**" button.	

Scenario 3: View Student Records Page

Current PatriotWeb System

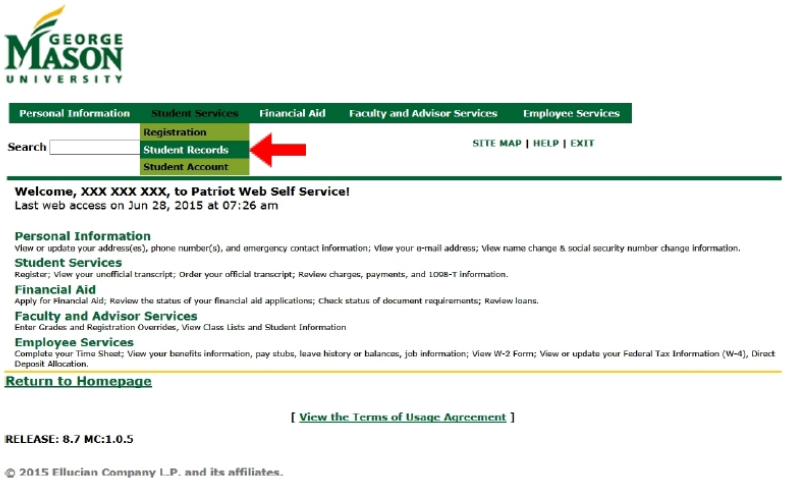
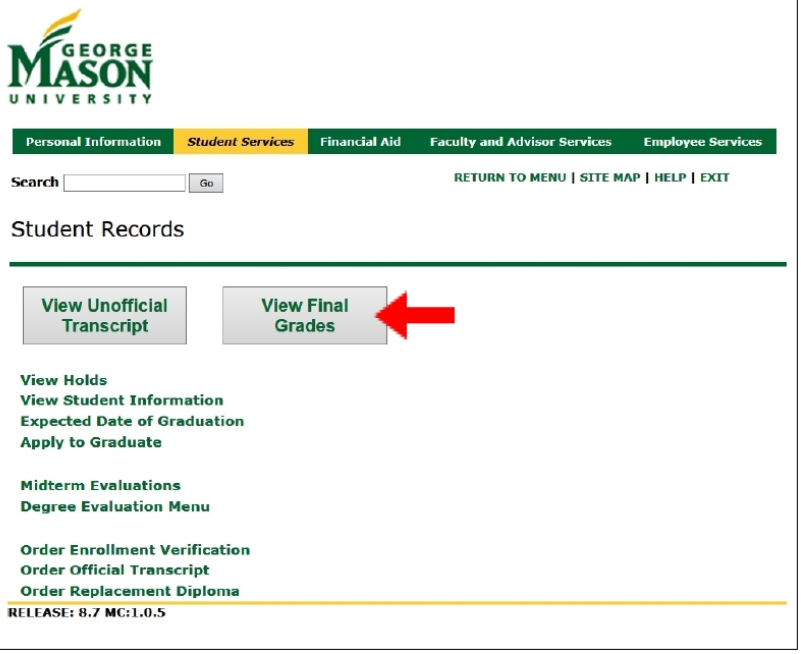
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

1	On Welcome page, click "**Student Services**" tab.	
2	On Student Services page, click "**Student Records**" link.	
3	On Student Records page, click "**Final Grades**" link.	

Scenario 3: View Student Records Page

Prototype PatriotWeb System

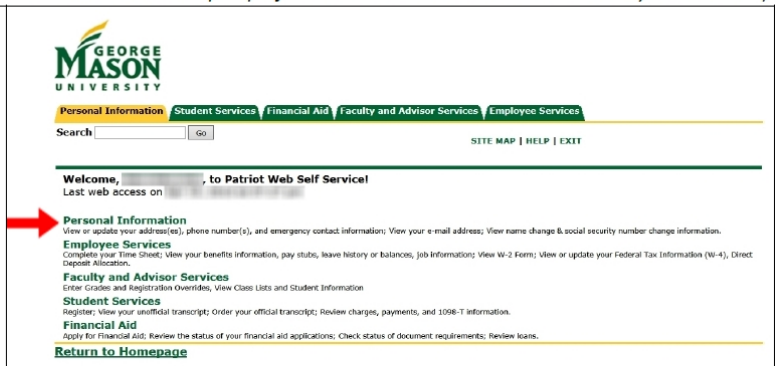
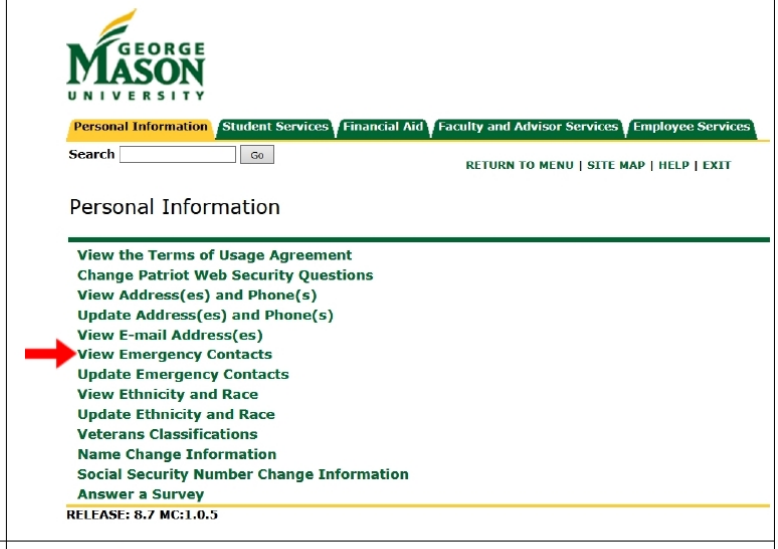
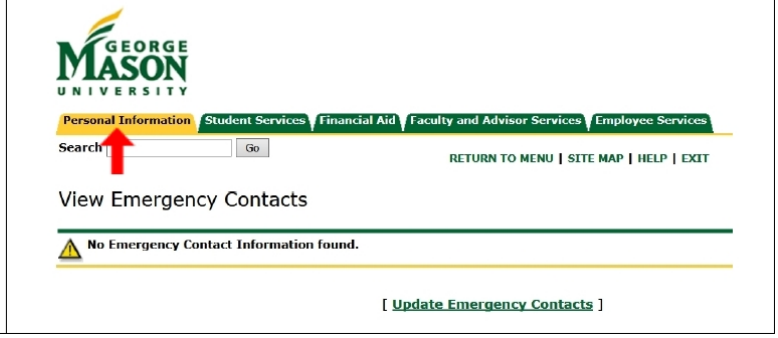
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

1	On Welcome page, mouse over "**Student Services**" tab. Select "**Student Records**" link from the drop down menu list.	
2	On Student Records page, click "**View Final Grades**" button.	

Scenario 4: View Personal Information Page

Current PatriotWeb System

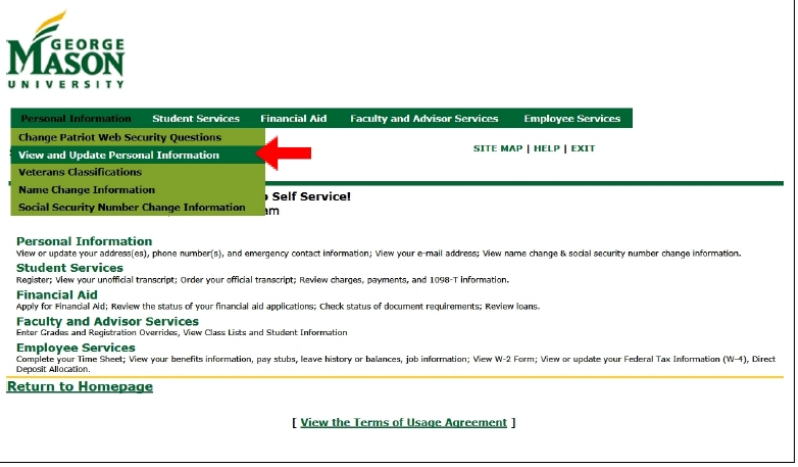
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

1	On Welcome page, click "**Personal Information**" link.	
2	On Personal Information page, click "**View Emergency Contacts**" link.	
3	On View Emergency Contacts page, to view another information such as E-mail address(es), click Personal Information tab on the main menu. This will bring you to the Personal Information page (screenshot 2 above).	

Scenario 4: View Personal Information Page

Prototype PatriotWeb System

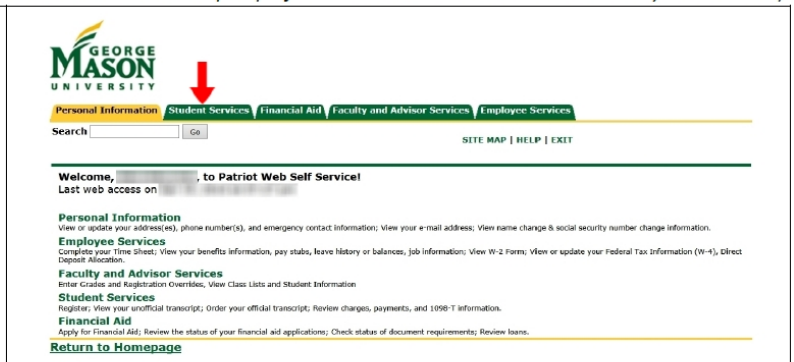
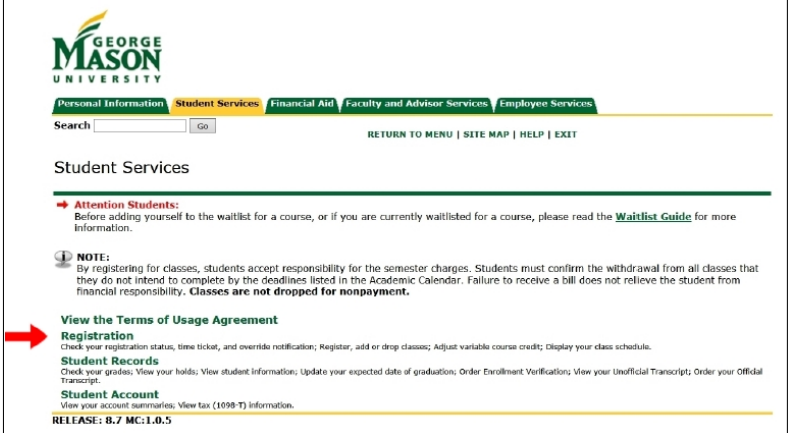
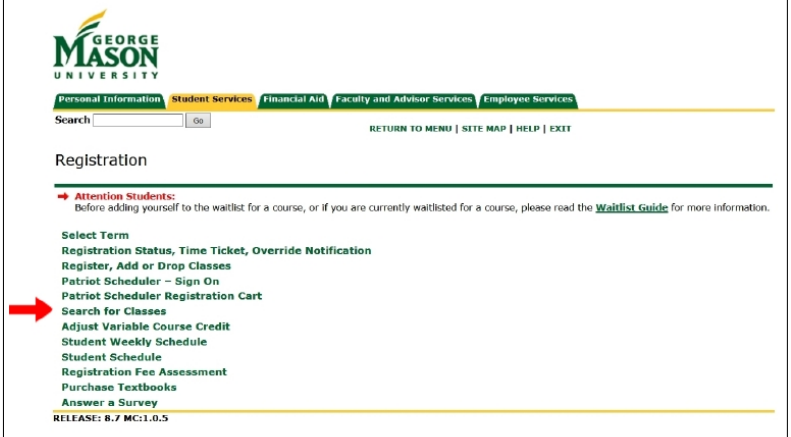
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

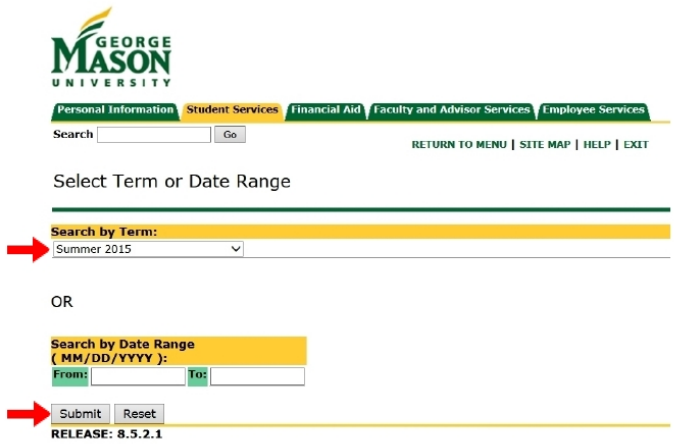
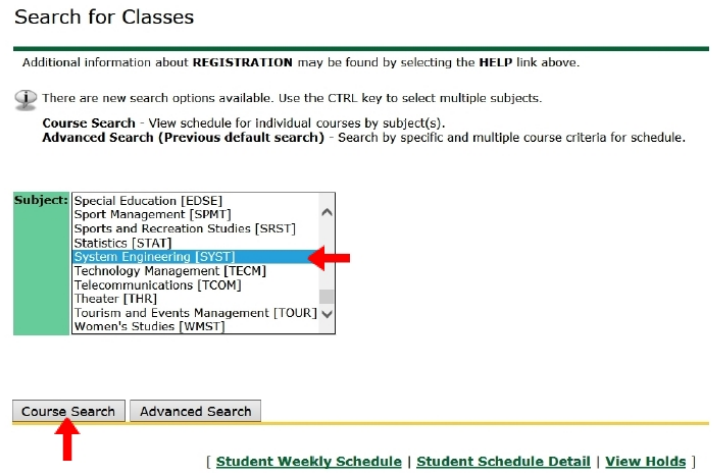
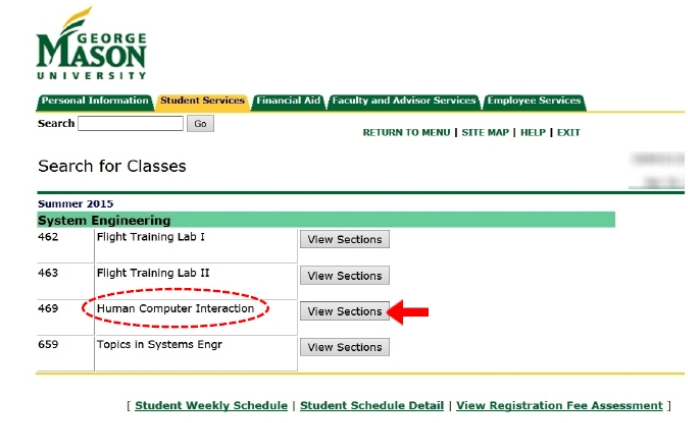
1	On Welcome page, mouse over "**Personal Information"** tab. Select "**View and Update Personal Information**" link from the drop down menu list.	

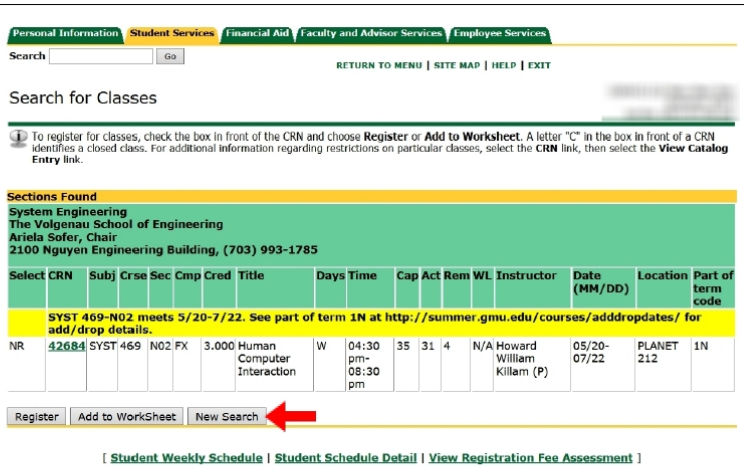
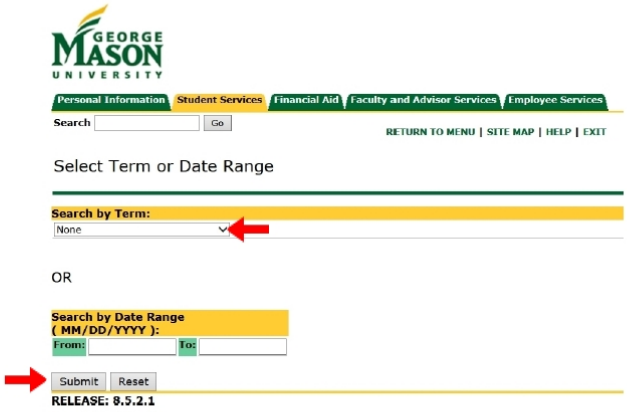
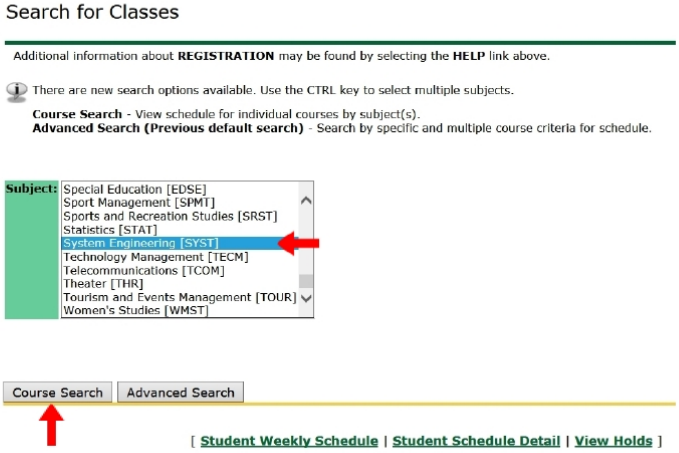
Scenario 5: Search for Classes

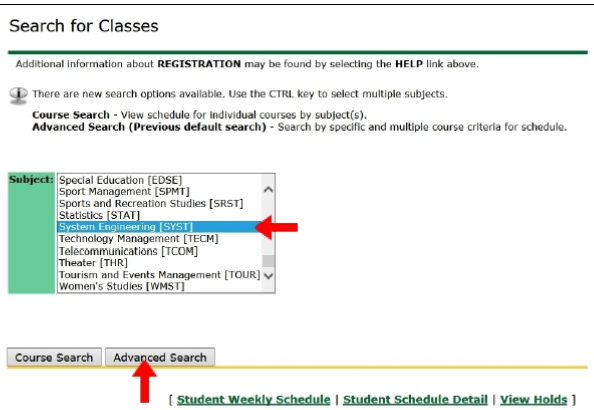
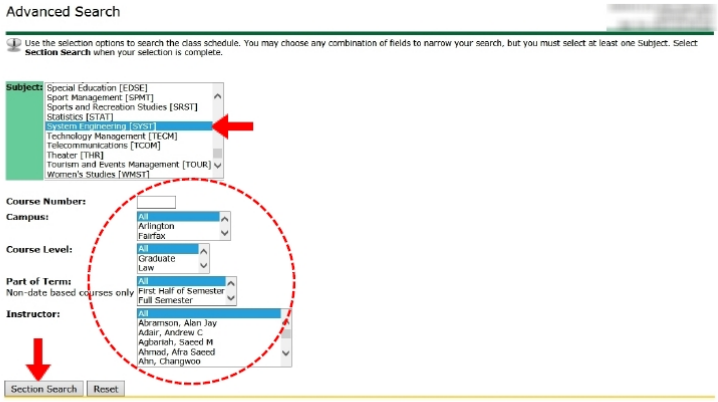
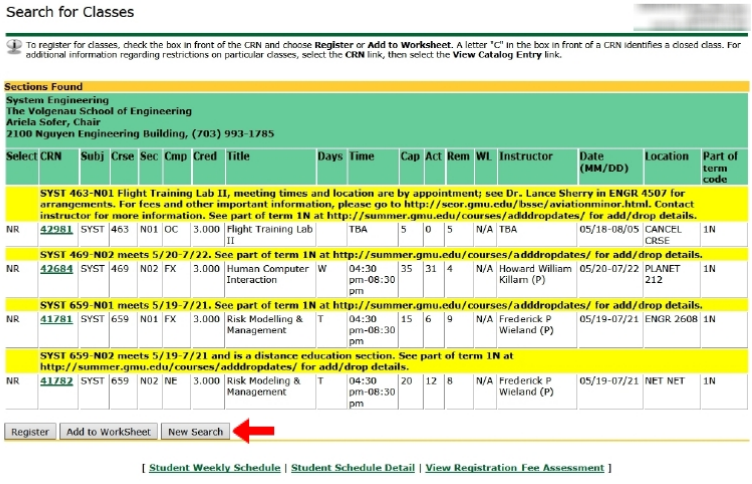
Current PatriotWeb System

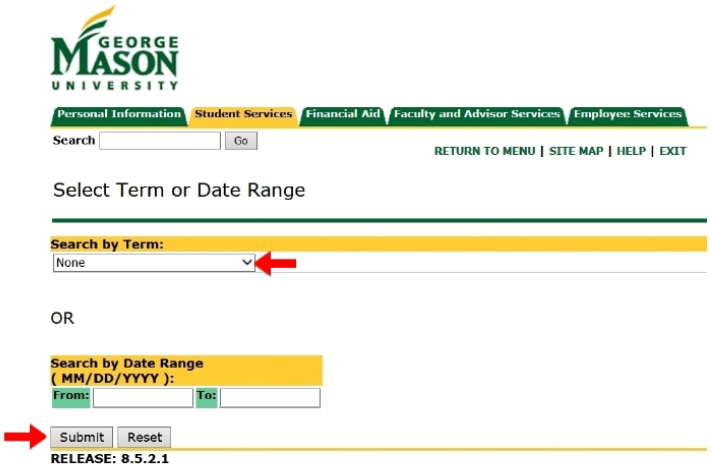
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

1	On Welcome page, click "**Student Services**" tab.	
2	On Student Services page, click "**Registration**" link.	
3	On Registration page, click "**Search for Classes**" link.	

4	On Select Term or Date Range page, select "**Summer 2015**" from the "**Search by Term:**" drop down list. Click "**Submit**" button.	
5	On Search for Classes page, select "**System Engineering [SYST]**" from "**Subject**" list. Click "**Course Search**" button.	
6	Click "**View Sections**" button of "**Human Computer Interaction**" class.	

7	Click "**New Search**" button.	
8	**Repeat from #4:** Repeat the process: On the Select Term or Date Range page, select "**Summer 2015**" from the "**Search by Term:**" drop down list to continue search for classes for the summer 2015 term. Click "**Submit**" button.	
9	**Repeat from #5:** Repeat the process: On Search for Classes page, select "**System Engineering [SYST]**" from "**Subject**" list. Click "**Course Search**" button.	

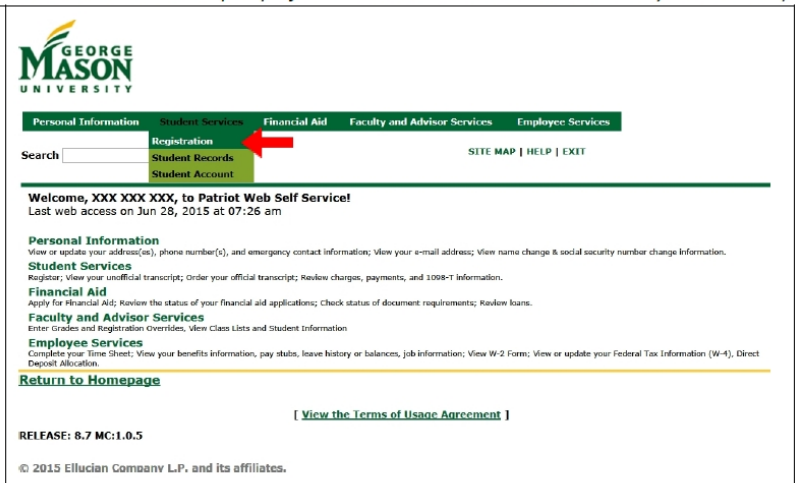
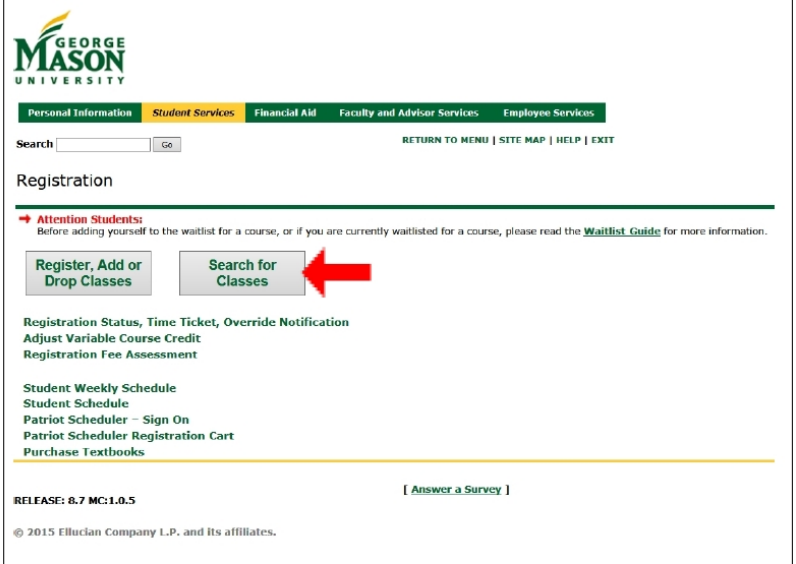
10	**Refer to #5 – *now click the "Advanced Search button"*:** On Search for Classes page, select "**System Engineering [SYST]**" from "**Subject**" list. Click "**Advanced Search**" button.	
11	On Advanced Search page, select "**System Engineering [SYST]**" from "**Subject**" list. Choose any combination of fields to narrow the search. Click "**Section Search**" button.	
11	Click "**New Search**" button.	

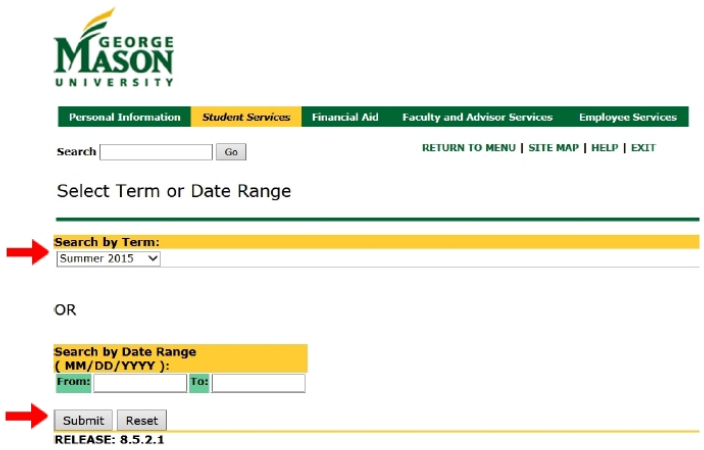
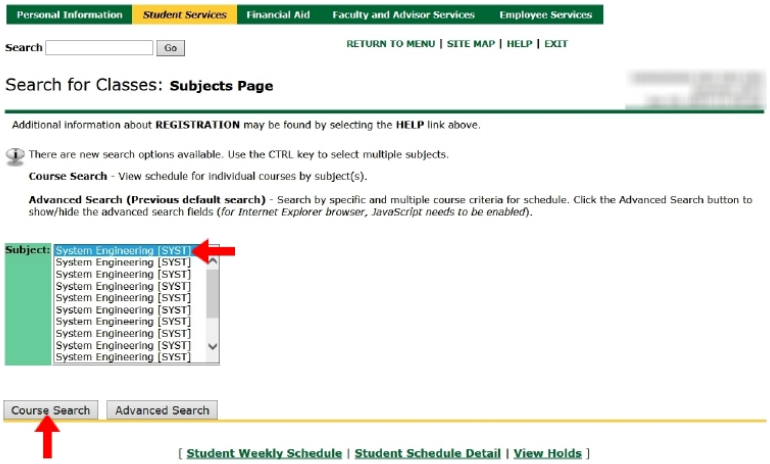
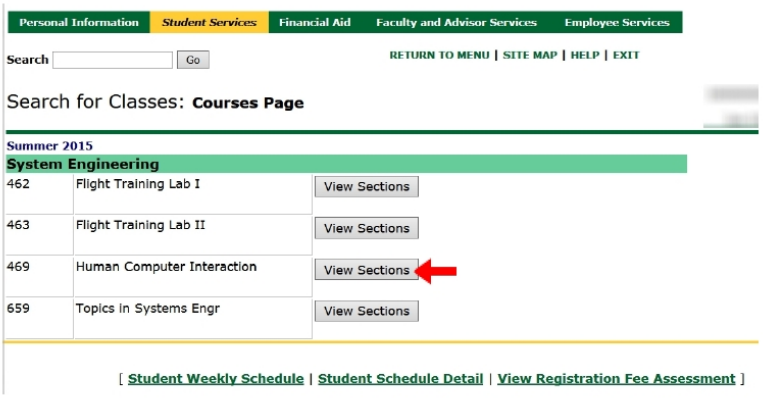
12	**Repeat from #4:** Repeat the process: Select "**Summer 2015**" from the "**Search by Term:**" drop down list to continue search for classes (for other subjects) for the summer 2015 term. *OR*, if you are done with searching for classes for the current term (Summer 2015) and you want to search for classes in other term, then select a new term from the "**Search by Term:**" drop down list to search for classes for other term. Click "**Submit**" button.	

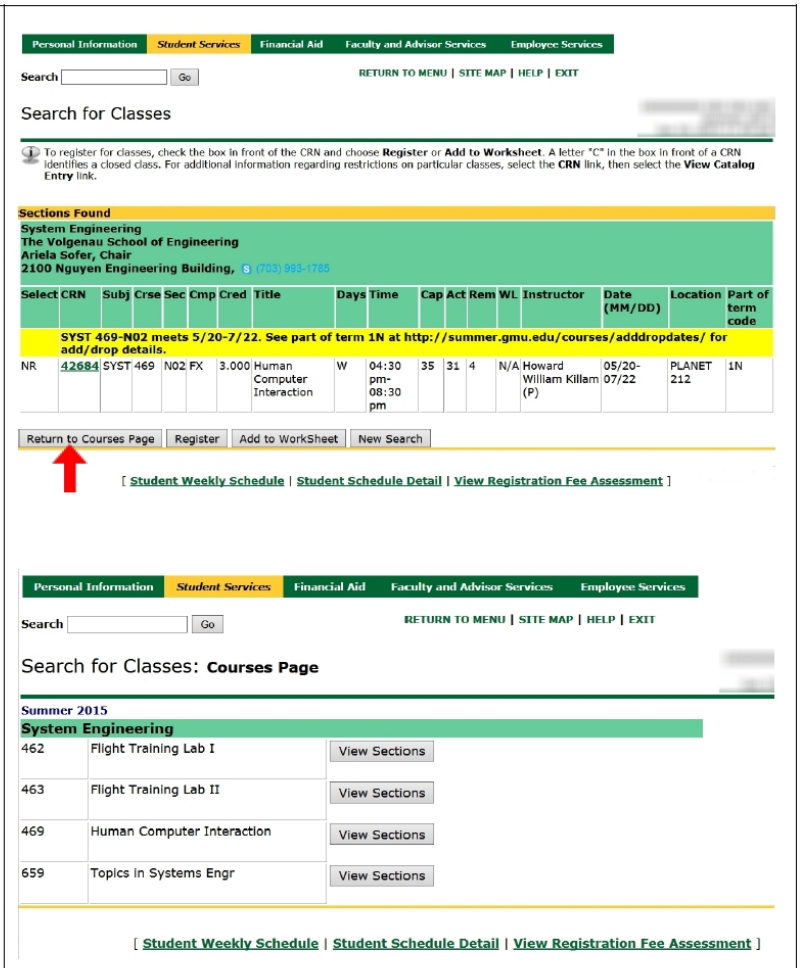
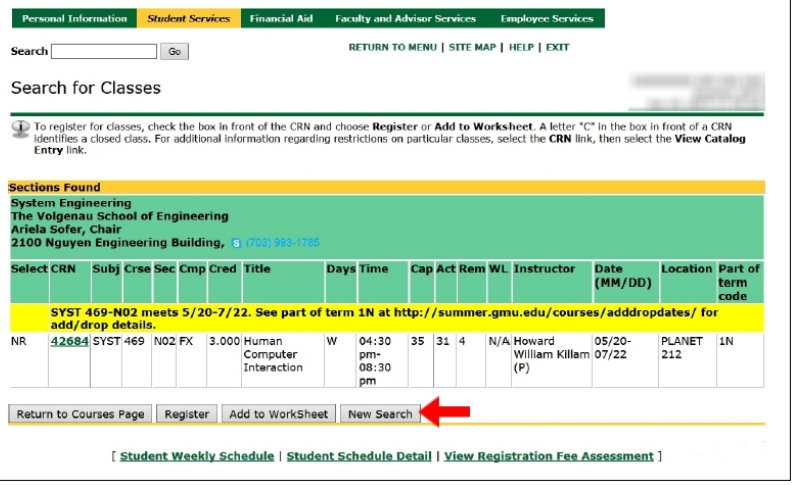
Scenario 5: Search for Classes

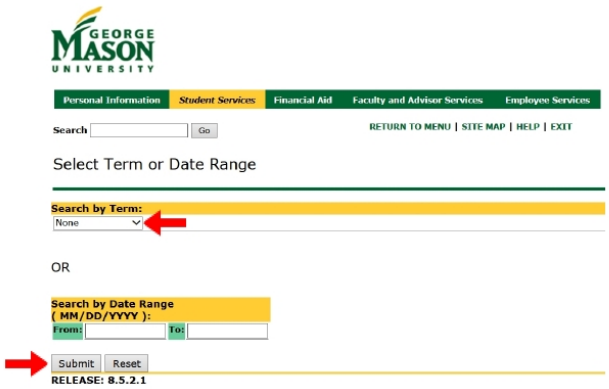
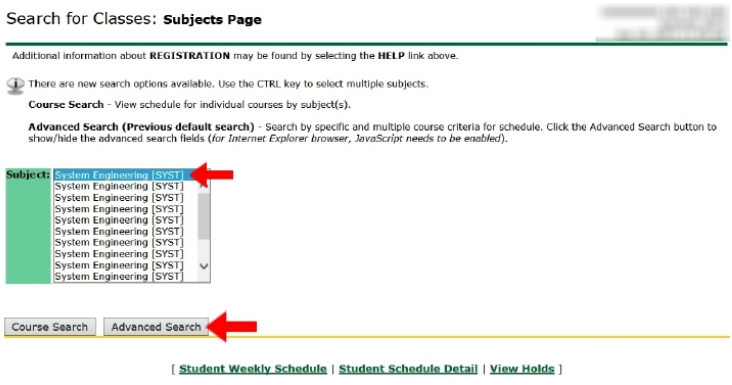
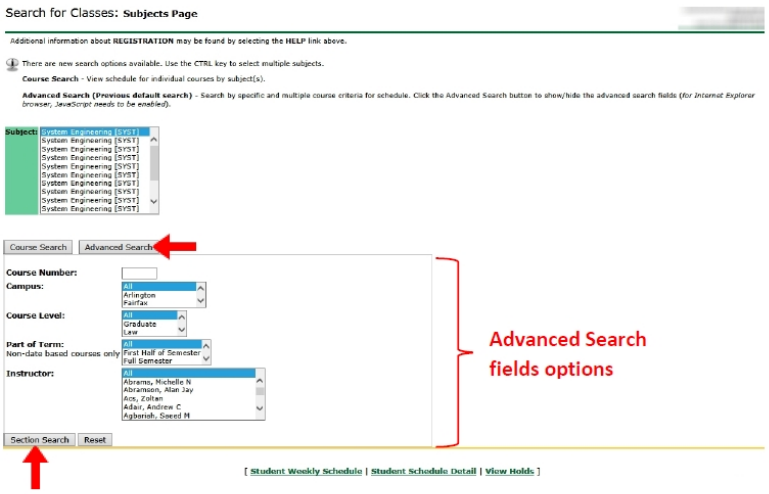
Prototype PatriotWeb System

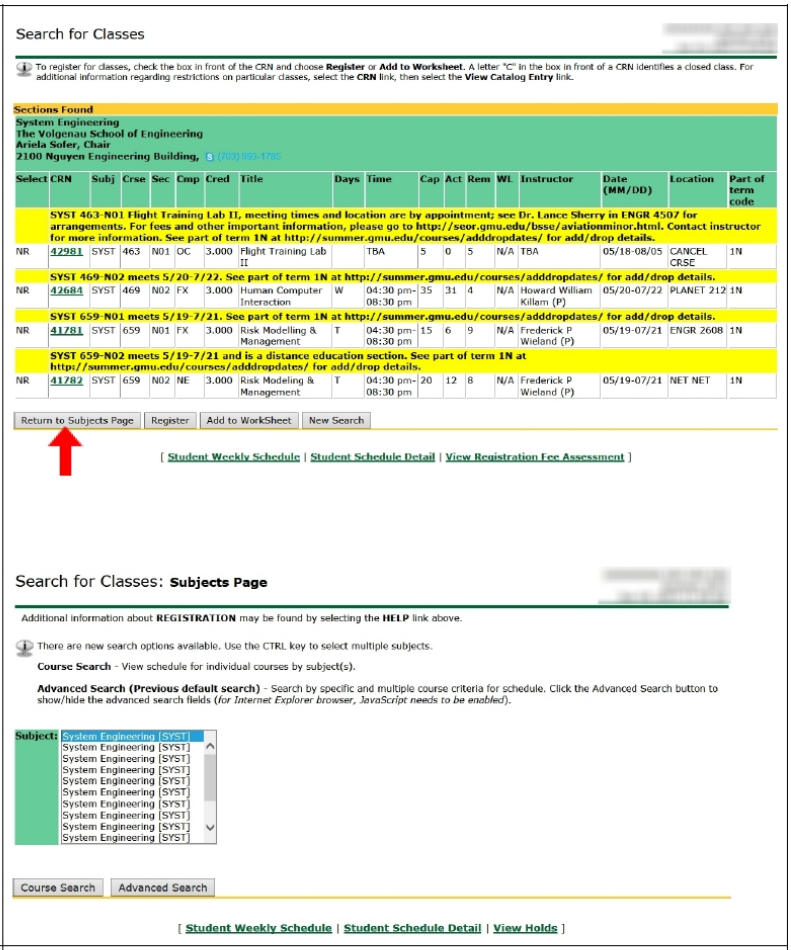
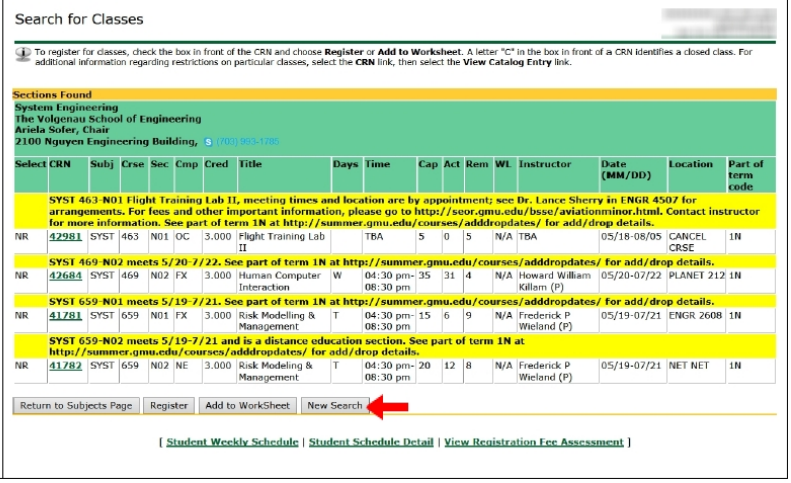
(NOTE: the number listed on first column does not mean the step to perform the task. Additional screenshots may have added.)

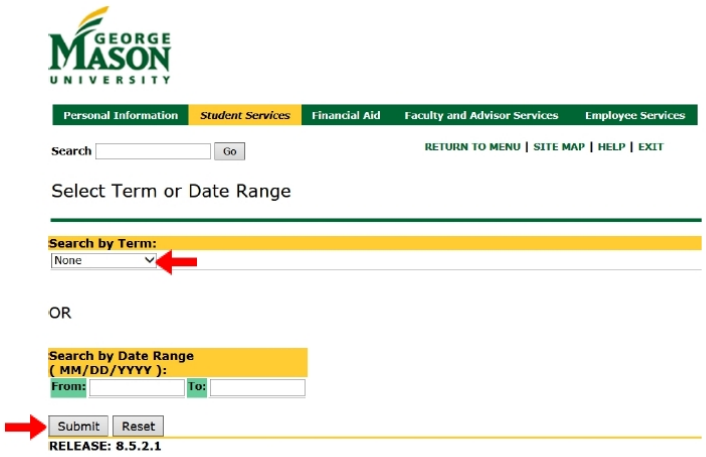
1	On Welcome page, mouse over "**Student Services**" tab. Select "**Registration**" link from the drop down menu list.	
2	On Registration page, click "**Search for Classes**" button.	

3	On Select Term or Date Range page, select "**Summer 2015**" from the "**Search by Term:**" drop down list. Click "**Submit**" button.	
4	On Search for Classes page, select "**System Engineering [SYST]**" from "**Subject**" list (select any one System Engineering [SYST] in the drop down list). Click "**Course Search**" button.	
5	Click "**View Sections**" button of "**Human Computer Interaction**" course.	

6	Click "**Return to Courses Page**" button to select sections of another course from the current selected "System Engineering [SYST]" subject. This will bring you to the previous page, which is the Courses page. On Search for Classes: Courses page, click "**View Sections**" button of another course (of the subject "System Engineering [SYST]").	
7	**Refer to #6 – *now click the "New Search" button*:** Click "**New Search**" button to start a new search on a brand new term and subject.	

8	Select another term from the "**Search by Term:**" drop down list. Click "**Submit**" button.	
10	**Refer to #4 – *now click the "Advanced Search" button*:** On Search for Classes: Subjects page, select "**System Engineering [SYST]**" from "**Subject**" list. Click "**Advanced Search**" button.	
11	A panel slide down below the Advanced Search button reviewing the advanced search fields. The Advanced Search fields are displayed on the same page as the Search for Classes: Subjects page, instead of opening in another page. Choose any combination of fields to narrow the search. Then click "**Section Search**" button.	

12	Click "**Return to Subjects Page**" button to select another subject. This will bring you to the previous page, which is the Subjects page. On Search for Classes: Subjects page, select another subject from the "**Subject**" list. Select either "**Course Search**" button or "**Advanced Search**" button to continue.	
13	**Refer to #12 – *now click the "New Search" button*:** Click "**New Search**" button to start a brand new search on another term and subject.	

14	Select another term from the "**Search by Term:**" drop down list. Click "**Submit**" button.	

APPENDIX H: EVALUATION – PART 1 SURVEY SUMMARIZED DATA

Q1 - Q19: Total Respondents (Final): 68 respondents

Q1 What is your age group?	18-24	49	72%
	25-34	19	28%
Q2 What is your gender?	Male	50	74%
	Female	18	26%
Q3 What is your marital status?	Single	60	88%
	Married	8	12%
Q4 What is your current school status?	Full-Time	62	91%
	Part-Time	6	9%
Q5 What is current GMU degree program?	Computer Sc	30	45%
	Civil Eng	12	18%
	Systems Eng	7	11%
	Bioeng	7	11%
	Applied IT	6	9%
	Math	4	6%
	No Response	2	3%
Q6 What is your education background?	High School Diploma/GED	28	42%
	Associate	38	58%
	No Response	2	3%
Q7 Job Status	Full-Time	0	0%
	Part-Time	34	50%
	Not Employed	34	50%

Q7 Years of Employment	<1 yr		6	9%
	1-3 yrs		10	15%
	3-6 yrs		18	26%
Q8 What device(s) do you currently own?	Desktop Computer		65	96%
	Laptop		68	100%
	Tablet		30	44%
	Mobile/Smart Phone (Internet-capable)		68	100%
Q9 In an average week, how often do you use the following devices?	Desktop Computer	Every Day	43	63%
		4-6 Times A Week	25	37%
	Laptop	Every Day	45	66%
		4-6 Times A Week	23	34%
	Tablet	Every Day	15	22%
		4-6 Times A Week	10	15%
		1-3 Times A Week	5	7%
	Mobile/Smart Phone	Every Day	68	100%
		4-6 Times A Week	0	0%
Q10 How comfortable are you with using the following devices for basic internet browsing/website navigation?	Desktop Computer	Very Comfortable	45	66%
		Comfortable	23	34%
	Laptop	Very Comfortable	50	74%
		Comfortable	18	26%
	Tablet	Very Comfortable	55	81%
		Comfortable	13	19%
	Mobile/Smart Phone	Very Comfortable	58	85%
		Comfortable	10	15%

Q11 How comfortable are you with using the following computing platforms?	UNIX	Comfortable	20	29%	
		Somewhat Comfortable	14	21%	
		Slightly Comfortable	10	15%	
		Not At All Comfortable	24	35%	
	LINUX	Comfortable	15	22%	
		Somewhat Comfortable	4	6%	
		Slightly Comfortable	10	15%	
		Not At All Comfortable	39	57%	
	WINDOWS (8, 7, XP)	Very Comfortable	62	91%	
		Comfortable	6	9%	

Q12 How familiar (in general) are you with the following web literacy items?	Basic Internet Browsing	Very Familiar	68	100%
	Downloading or Uploading Files on the Interent	Very Familiar	64	94%
		Familiar	4	6%
	Basic HTML, CSS	Very Familiar	40	59%
		Familiar	28	41%
	Basic Programming Languages (e.g. Java, Visual Basic, C++, C)	Very Familiar	20	29%
		Familiar	15	22%
		Somewhat Familiar	8	12%
		Slightly Familiar	25	37%

Q13 In a weekly basis, how often (average) do you use the GMU PatriotWeb system?	0-5 Times Per Week	30	44%
	6-10 Times Per Week	30	44%
	>10 Times Per Week	8	12%

Q14 What device do you use most often to access the GMU PatriotWeb system?	Dektop Computer	12	18%
	Laptop	41	60%
	Tablet	10	15%
	Mobile Phone	5	7%

Q15 How do you access GMU PatriotWeb system when you are in a public (outdoor) place where desktop computers are not available?	Laptop	18	26%
	Mobile Phone	40	59%
	Tablet	10	15%
Q16 In what environment do you use the PatriotWeb system most often?	Off-Campus Private Home	28	44%
	Campus Dormitory	20	26%
	Public Places	20	29%
Q17 In what part of the day do you usually access the GMU PatriotWeb system?	Morning	9	13%
	Lunch Hour (12pm-1pm)	11	16%
	Afternoon	12	18%
	Evening	16	24%
	Night	20	29%
Q18 List up to three services you usually use when using the PatriotWeb system?	Search Classes	40	59%
	Registration	25	37%
	Check Schedule	30	44%
	Check Account	22	32%
	Check Finanicla Aid	25	37%
	View Grades/ Transcripts	27	40%

Q19 In your opinon, how important to you are the following factors when using the PatriotWeb system in general?

Ease in performing tasks	Very Important	25	37%
	Important	43	63%
Easy to navigate the website without too much learning	Very Important	8	12%
	Important	40	59%
	Somewhat Important	20	29%
Number of mouse clicks or web pages needed to reach destination page (i.e. speed, task time)	Very Important	20	29%
	Important	43	63%
	Somewhat Important	5	7%
Visibility, consistency, or location of menus, links, or buttons	Very Important	23	34%
	Important	35	51%
	Somewhat Important	10	15%
Menus, links, or buttons are easy to understand	Very Important	20	29%
	Important	40	59%
	Somewhat Important	8	12%
The outcome of clicking the menus, links, or buttons produce the result correctly (i.e. as you would expect)	Very Important	23	34%
	Important	40	59%
	Somewhat Important	5	7%
Attractiveness of the web pages (i.e. color, layout, font style, font size, etc)	Somewhat Important	22	32%
	Low Important	32	47%
	Not At All Important	14	21%
Able to read the texts clearly	Very Important	22	32%
	Important	41	60%
	Somewhat Important	5	7%
Able to remember how the system looks like and what the system does when you visit the system the next time (e.g. a week	Somewhat Important	26	38%
	Low Important	37	54%
	Not At All Important	4	6%

Q21 - Q23 (Section C), Q20 (Section B)

Total Respondents (Final): 68 respondents

Group A (persona applied): 34 respondents

Group B (persona not applied): 34 respondents

			Group A		Group B	
Q21 In comparison to the current PatriotWeb system on the five scenarios, how would you rate the PatriotWeb prototype system?	Item 1 (Efficiency)	Same	4	12%	15	44%
		Better	30	88%	19	56%
	Item 2 (Efficiency)	Same	6	18%	15	44%
		Better	28	82%	19	56%
	Item 3 (Comprehensibility)	Same	4	12%	9	26%
		Better	30	88%	25	74%
	Item 4 (Comprehensibility)	Same	10	29%	16	47%
		Better	24	71%	18	53%
	Item 5 (Comprehensibility)	Same	4	12%	12	35%
		Better	30	88%	22	65%
	Item 6 (Accuracy)	Same	12	35%	16	47%
		Better	22	65%	18	53%
	Item 7 (Learnability)	Same	11	32%	16	47%
		Better	23	68%	18	53%
	Item 8 (Rememberability)	Same	34	100%	34	100%
		Better	0	0%	0	0%
	Item 9 (Reliability)	Same	34	100%	34	100%
		Better	0	0%	0	0%
	Item 10 (Attractiveness)	Same	21	62%	24	71%
		Better	13	38%	10	29%
	Item 11 (Clarity)	Same	25	74%	27	79%
		Better	9	26%	7	21%

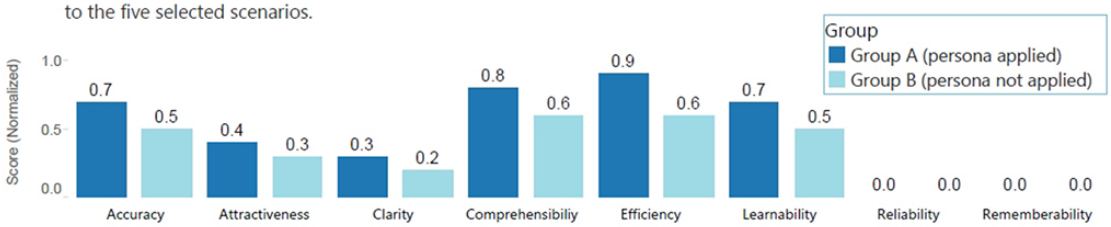
		Group A		Group B	
Q22 Overall how would you rate the PatriotWeb prototype system in comparison to the current PatrioWeb system, based on the five scenarios?	Same	8	24%	14	41%
	Better	26	76%	20	59%

		Group A		Group B	
Q23 What factors are important to you as a user of the system? Please list one to three factors.	Speed, less mouse clicks, reduce unnecessary pages, ease of use [Efficiency]	34	100%	32	94%
	Return page button, no confusing menus, visibility of important links or buttons [Comprehensibility]	26	76%	23	68%
	Accuracy [Accuracy]	20	59%	21	62%

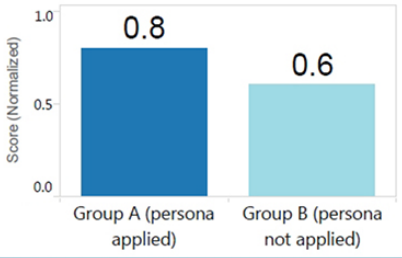
		Group A		Group B	
Q20 (Section B) Do you feel that a mobile version of the GMU PatriotWeb system with streamlined most common or used features is beneficial to you as a user of the system?	Yes	29	85%	24	71%
	No	5	15%	10	29%

APPENDIX I: EVALUATION – PART 1 SURVEY EVALUATION RESULTS

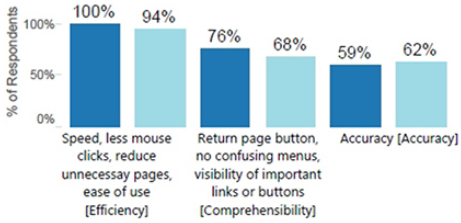
Q21 Comparison of prototype PatriotWeb with current PatriotWeb according to the the eight usability factors with respect to the five selected scenarios.



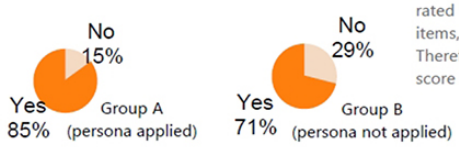
Q22 OVERALL, how would you rate the prototype PatriotWeb in comparison to the current PatriotWeb with respect to the five selected scenarios?



Q23 What factors are important to you as a user of the current PatriotWeb system? Please list one to three factors.



Q20 Do you feel that a mobile version of the GMU PatriotWeb system with streamlined most common or used features is beneficial to you as a user of the system?



menus [*Comprehensibility*]

Computation:

Rating scale (Q21 & Q22):
Worse -2pts, Same (No improvement) 0pt, Better +2pts

Total respondents:
Group A (persona applied) 34
Group B (persona not applied) 34

There are total 11 evaluating usuability items on the survey for Q21. If there are more than one item belongs to a particular usability factor, then average the scores. Score for each usability factor is finally normalized to 1.0

Examples:

a) Attractiveness:
Group A: 13 respondents rated "Better" and 21 respondents rated "Same".
Therefore, (13*2)+(21*0) = 26. Final normalized score = (26/68)*1.0 = 0.4

b) Efficiency:
There are 2 items on Q21 belong to "Efficiency"
Group A: total 58 respondents (for 2 items, 30+28) rated "Better" and total 10 respondents (for 2 items, 4+6) rated "Same".
Therefore, ((58*2)+(10*0))/2 = 58. Final normalized score = (58/68)*1.0 = 0.9

APPENDIX J: EVALUATION – PART 2
PAL CONSTRAINTS, JESS CONSISTENCY AND COMPLETENESS RULES

J.1. PAL Constraints and Jess Rules Natural Language Statements

PAL Constraints: REQUIREMENTS ONTOLOGY	
Constraint No.	**Constraint Name**
1	Every requirement must have a unique id (i.e. no duplicate requirement id).
2	Every requirement category (functional and non-functional) must have a unique id (i.e. no duplicate requirement category id).
3	Every system requirements specification (SRS) must have a unique id (i.e. no duplicate system requirements specification id).
4	All priority values should be between min value 0.0 and max value 1.0.
5	All risk level values should be between min value 0.0 and max value 1.0.
6	Each requirement must have at least one requirement category.
7	Each requirement category must have at least one requirement.
8	Each requirement must be part of at least one system requirements specification (SRS) document.
9	Each system requirements specification (SRS) document must have at least one requirement.
10	A requirement should be refined from no more than one requirement.
11	Each requirement must be a requirement in at least one viewpoint.
12	A requirement category can only be a sub requirement category of at most one requirement category.
13	If two requirement categories have the same category name and if they have sub requirement category, then they must have the same sub requirement category.
14	Every requirement category name must be unique (i.e. no duplicate requirement category name).

PAL Constraints: PERSONA ONTOLOGY	
Constraint No.	**Constraint Name**
15	Every persona must have a unique id (i.e. no duplicate persona id).
16	Every name must have a unique id (i.e. no duplicate name id).
17	Every education must have a unique id (i.e. no duplicate education id).
18	Every occupation must have a unique id (i.e. no duplicate occupation id).
19	Every language proficiency (writing, speaking, and listening) must have a unique id (i.e. no duplicate language proficiency id).
20	Every environment must have a unique id (i.e. no duplicate environment id).
21	Every ability must have a unique id (i.e. no duplicate ability id).
22	Every interest must have a unique id (i.e. no duplicate interest id).
23	Every interest category must have a unique id (i.e. no duplicate interest category id).
24	Every role must have a unique id (i.e. no duplicate role id).
25	Every knowledge (domain, web, and computer) must have a unique id (i.e. no duplicate knowledge id).
26	Every concern must have a unique id (i.e. no duplicate concern id).
27	Every usability preference must have a unique id (i.e. no duplicate usability preference id).
28	Each persona must have at most one name.
29	Each name must have at most one persona.
30	No two personas can have the same name.
31	Each persona must have at most one highest education level.
32	Each persona must personify at least one role.
33	Each persona must engage in at least one environment.
34	Each environment must be engaged by at least one persona.
35	Each role must be personified by at least one persona.
36	Each highest education level must be a highest education level of at least one persona.
37	Each persona must be a persona in at least one viewpoint.
38	Each role must be a role in at least one viewpoint.
39	Each environment must be an environment in at least one viewpoint.
40	Each concern must be addressed by at least one viewpoint.
41	An interest can only have at most one interest category.
42	An interest category can only be a sub interest category of at most

	one interest category.
43	If two interests have the same name, then they must belong to the same interest category.
44	If two interest categories have the same category name and if they have sub interest category, then they must have the same sub interest category.
45	If the ability (physical, cognitive) type is not “Other”, then the ability (physical, cognitive) type other field must be empty.
46	Every physical ability with the same physical type other and ability level must be the physical ability of the same persona.
47	Every cognitive ability with the same cognitive type other and ability level must be the cognitive ability of the same persona.
48	Every ability (physical, cognitive) with the same type other and ability level must be the ability of the same persona.
49	Every speaking language proficiency with the same speaking language name and proficiency level must be the speaking language of the same persona.
50	Every writing language proficiency with the same writing language name and proficiency level must be the writing language of the same persona.
51	Every listening language proficiency with the same listening language name and proficiency level must be the listening language of the same persona.
52	Every computer knowledge with the same computer knowledge name and knowledge level must be the computer knowledge of the same persona.
53	Every domain knowledge with the same domain knowledge name and knowledge level must be the domain knowledge of the same persona.
54	Every web knowledge with the same web knowledge name and knowledge level must be the web knowledge of the same persona.

PAL Constraints: BEHAVIORAL-GST ONTOLOGY	
Constraint No.	**Constraint Name**
55	Every goal must have a unique id (i.e. no duplicate persona id).
56	Every scenario must have a unique id (i.e. no duplicate scenario id).
57	Every task must have a unique id (i.e. no duplicate task id).
58	Every viewpoint must have a unique id (i.e. no duplicate viewpoint id).
59	Every stakeholder must have a unique id (i.e. no duplicate stakeholder id).
60	Every actor must have a unique id (i.e. no duplicate actor id).
61	Every constraint must have a unique id (i.e. no duplicate constraint id).
62	Every obstacle must have a unique id (i.e. no duplicate obstacle id).
63	Every pre-requisite must have a unique id (i.e. no duplicate pre-requisite id).
64	Every post-requisite must have a unique id (i.e. no duplicate post-requisite id).
65	Each goal must be a goal of at least one viewpoint.
66	Each scenario must be a scenario of at least one viewpoint.
67	Each task must be a task of at least one viewpoint.
68	Each stakeholder must be a stakeholder of at least one viewpoint.
69	Each viewpoint must have at least one requirement.
70	Each viewpoint must have at least one goal.
71	Each viewpoint must have at least one scenario.
72	Each viewpoint must have at least one task.
73	Each scenario must have at least one task.
74	Each task is task of at least one scenario.
75	If a goal has status “withdrawn”, then it should not have any relationships with one or more concepts.
76	No two viewpoints can have the same persona name, environment, and role (i.e. a viewpoint is uniquely identified by persona, environment, and role).

JESS Consistency Rules: REQUIREMENTS ONTOLOGY	
Rule No.	**Rule Name**
1	A functional requirement category cannot also be a non-functional requirement category (no duplicate requirement category).
2	A requirement cannot appear in both functional and non-functional requirement category (no duplicate requirement in functional and non-functional requirement category).
3	If a requirement A is refined into a requirement B, then requirement B must be refined from only requirement A.
4	If a requirement A constrains a requirement B, then requirement B must be constrained by requirement A.
5	If a requirement A requires a requirement B, then requirement B must be required by requirement A.
6	If a requirement A conflicts with a requirement B, then requirement B must also conflict with requirement A.
7	If a requirement A meets a goal A, then goal A must be met by requirement A.
8	If a requirement A is operationalized by a scenario A, then scenario A must operationalize requirement A.
9	If a requirement A is part of a System Requirements Specification (SRS) document A, then SRS document A has requirement A.
10	If a requirement A is a requirement category (functional or non-functional) A, then requirement category (functional or non-functional) A has requirement A.
11	If a requirement category A is a sub-requirement category of a requirement category B, then requirement category B has at least sub-requirement category A.
12	If a requirement A is a requirement in a viewpoint A, then viewpoint A has at least requirement A.

JESS Consistency Rules: PERSONA ONTOLOGY	
Rule No.	**Rule Name**
13	If a persona does not have a name, then add a temporary name PERSONA.
14	If a persona A has a name A, then name A is the name for only persona A.
15	If a persona A has a highest education A, then highest education A is the highest education level for at least persona A.
16	If a persona A has an occupation A, then occupation A is an occupation for at least persona A.
17	If a persona A has a language proficiency A, then language proficiency A is the language proficiency for at least persona A.
18	If a persona A has a concern A, then concern A is a concern of at least persona A.
19	If a persona A engages in an environment A, then environment A is engaged by at least persona A.
20	If a persona A plays (personifies) a role A, then role A is played by (personified by) at least persona A.
21	If a concern A relates to an environment A, then environment A has at least concern A.
22	If a role A participates in an environment A, then environment A is participated by at least role A.
23	If an interest category A is a sub-interest category of interest category B, then interest category B has sub-interest category A.
24	If an interest A is an interest category A, then interest category A has interest A.
25	If a persona A has interest A, then interest A is an interest of at least persona A.
26	A cognitive ability (of type “Other”) cannot also be a physical ability (of type “Other”) (no duplicate cognitive type other).
27	If a persona A has an ability (cognitive or physical) A, then ability (cognitive or physical) A is an ability (cognitive or physical) of at least persona A.
28	If an environment A has an usability preference A, then usability preference A is an usability preference of at least environment A.
29	A domain knowledge cannot also be a web or computer knowledge (no duplicate knowledge).
30	A web knowledge cannot also be a domain or computer knowledge (no duplicate knowledge).
31	A computer knowledge cannot also be a domain or web knowledge (no duplicate knowledge).
32	If a persona A has a knowledge A, then knowledge A is a knowledge of

	at least persona A.
33	If a persona A plays a role A that participates in an environment A, then persona A must also engage in environment A.

JESS Consistency Rules: BEHAVIORAL-GST ONTOLOGY	
Rule No.	**Rule Name**
34	If a goal A is refined into ("AND-refinesInto") a goal B, then goal B must be refined from ("AND-refinesFrom") at least goal A.
35	If a goal A is refined ("OR-refinesInto") into a goal B, then goal B must be refined ("OR-refinesFrom") from at least goal A.
36	If a goal A conflicts with a goal B, then goal B must also conflict with goal A.
37	If a goal A supports a goal B, then goal B must also be supported by goal A.
38	If a goal A has an obstacle A, then obstacle A is an obstacle of goal A.
39	If a goal A has a constraint A, then constraint A is a constraint of goal A.
40	If a goal A has a stakeholder A, then stakeholder A is a stakeholder of goal A.
41	If a goal A is a goal in a viewpoint A, then viewpoint A must have goal A.
42	If a goal A has a goal type A, then goal type A is the goal type of at least goal A.
43	If a goal is a goal in a viewpoint A that includes persona A, then persona A must also have goal A.
44	If a scenario A has an episode A, then episode A is an episode of scenario A.
45	A normal scenario cannot also be an exception scenario (no duplicate scenario).
46	If a scenario A has a task A, then task A is a task of scenario A.
47	If a scenario A has a constraint A, then constraint A is a constraint of scenario A.
48	If a scenario A is a scenario in a viewpoint A, then viewpoint A must have scenario A.
49	If a task A is refined into ("AND-refinesInto") a task B, then task B must be refined from ("AND-refinesFrom") at least task A.
50	If a task A is refined ("OR-refinesInto") into a task B, then task B must be refined ("OR-refinesFrom") from at least task A.
51	If a task A has a tool A, then tool A is a tool of at least task A.
52	If a task A has an action A, then action A is an action of at least task A.
53	If a task A has a constraint A, then constraint A is a constraint of at least task A.
54	If a task A is a task in a viewpoint A, then viewpoint A must have at least task A.
55	If a task A has a pre-requisite A, then pre-requisite A is a pre-requisite of at least task A.

56	If a task A has a post-requisite A, then post-requisite A is a post-requisite of at least task A.
57	If a viewpoint A has a constraint A, then constraint A is a constraint of at least viewpoint A.
58	An instance of Actor sub-class cannot also be an instance of another Actor sub-class (no duplicate actor).
59	If an actor A plays a role A, then role A is played by at least actor A.
60	If a viewpoint A has a persona A, a role A, and an environment A, then viewpoint B cannot also has the same persona A, role A, and environment A (no two viewpoints have the same persona, role, and environment; a view point is uniquely identified by <persona, environment, role>).

JESS Completeness Rules: REQUIREMENTS ONTOLOGY	
Rule No.	**Rule Name**
1	Every requirement must specify attribute id, statement, source, status, priority, validation, and verification.
2	Every requirement must specify whether it is a functional or a non-functional requirement.
3	Every functional and non-functional requirement category must specify requirement category attribute.
4	Every requirement must specify at least one relationship with another requirement, i.e. conflictsWith, constrains, requires, refinesInto.
5	Every requirement must specify at least one "isPartOf" relationship with system requirements specification.
6	If a requirement has status "Implemented", then that requirement must specify at least one meets relationship with a goal.
7	If a requirement has status "Implemented", then that requirement must specify at least one operationalizedBy relationship with a scenario.
8	Every requirement must specify at least one isRequirementsOf relationship with one or more viewpoint.

JESS Completeness Rules: PERSONA ONTOLOGY	
Rule No.	**Rule Name**
9	Every persona must specify attribute id, persona_title, persona_type, age, and gender.
10	Every name must specify attribute id, first_name, middle_name, and last_name.
11	Every education must specify attribute id and highest education level.
12	Every occupation must specify attribute id, position, and occupation group.
13	Every concern must specify attribute id, name, and rating.
14	Every role must specify attribute id, name, and role type.
15	Every environment must specify attribute id, name, location, time of day, and persona attitude.
16	Every usability preference must specify attribute id, usability type, and rating.
17	Every interest must specify attribute id and name.
18	Every interest category must specify attribute id and category.
19	Every ability must specify attribute id, cognitive type or physical type, and ability level. If cognitive type or physical type is “Other”, then it must specify the text field for “Other”.
20	Every language proficiency must specify attribute id, writing language or speaking language or listening language, and proficiency level.
21	Every knowledge must specify attribute id, domain knowledge or web knowledge or computer knowledge, and knowledge level.
22	Every persona must specify at most one hasPersonaName relationship with a name.
23	Every persona must specify at most one hasHighestEducation relationship with an education.
24	Every persona must specify at least one personifies relationship with a role.
25	Every persona must specify at least one engagesIn relationship with an environment.
26	Every interest must specify at most one isInterestPartOf relationship with an interest category.
27	Every persona must specify at least one isPersonaOf relationship with a viewpoint.
28	Every environment must specify at least one isEnvironmentOf relationship with a viewpoint.
29	Every role must specify at least one isRoleOf relationship with a viewpoint.
30	Every concern must specify at least one addressedBy relationship with a viewpoint.

31	Every name must specify at most one isPersonaNameOf relationship with a persona.
32	Every education must specify at most one isHighestEducationOf relationship with a persona.
33	Every role must specify at least one personifiedBy relationship with a persona.
34	Every environment must specify at least one engagedBy relationship with a persona.

JESS Completeness Rules: BEHAVIORAL-GST ONTOLOGY	
Rule No.	**Rule Name**
35	Every goal must specify attribute id, name, priority, and status.
36	Every scenario must specify attribute id, name, status, preconditions, and postconditions.
37	Every tool must specify attribute id and name.
38	Every task must specify attribute id, name, and priority.
39	Every viewpoint must specify attribute id, name, modeling techniques, and models.
40	Every stakeholder must specify attribute id, name, and priority.
41	Every actor (organization, device, system, person) must specify attribute id and name.
42	Every constraint must specify attribute id, name, critical level, and constraint type. If constraint type is “Other”, then it must specify the text field for “Other”.
43	Every obstacle must specify attribute id, name, critical level, and obstacle type. If obstacle type is “Other”, then it must specify the text field for “Other”.
44	Every viewpoint must specify at least one hasGoal relationship with a goal.
45	Every viewpoint must specify at least one hasTask relationship with a task.
46	Every viewpoint must specify at least one hasScenario relationship with a scenario.
47	Every viewpoint must specify at least one hasRequirements relationship with a requirement.
48	Every viewpoint must specify at least one hasStakeholder relationship with a stakeholder.
49	Every viewpoint must specify at most one hasPersona relationship with a persona.
50	Every viewpoint must specify at most one hasEnvironment relationship with a persona.
51	Every viewpoint must specify at most one hasRole relationship with a persona.
52	Every viewpoint must specify at least one addresses relationship with a concern.
53	Every goal must specify at least one relationship with another goal, i.e. conflictsWith, supports, requires, AND-refinesInto, OR-refinesInto.
54	Every goal must specify at least one hasStakeholder relationship with a stakeholder.
55	Every goal must specify at least one isGoalOf relationship with a viewpoint.

56	Every goal must specify at least one hasGoalType relationship with a goal type.
57	Every goal type must specify at least one isGoalTypeOf relationship with a goal.
58	If a goal has status "Met" or “Partially-Met”, then that goal must specify at least one metBy relationship with a requirement.
59	Every scenario must specify at least one isScenarioOf relationship with a viewpoint.
60	If a scenario has status “Active”, then that scenario must specify at least one operationalizes relationship with a requirement.
61	Every scenario must specify at least one hasTask relationship with a task.
62	Every task must specify at least one isTaskOf relationship with a scenario.
63	Every task must specify at least one relationship with another task, i.e. AND-refinesInto, OR-refinesInto.
64	Every task must specify at least one isTaskOf relationship with a viewpoint.
65	Every task must specify at least one hasAction relationship with an action.
66	Every action must specify at least one isActionOf relationship with a task.
67	Every action must specify at least one performedBy relationship with an actor.
68	Every actor must specify at most one represents relationship with a stakeholder.
69	Every actor must specify at least one performes relationship with an action.
70	Every stakeholder must specify at least one representedBy relationship with an actor.
71	Every stakeholder must specify at least one isStakeholderOf relationship with a goal.

J.2. PAL Constraints and Jess Rules

Requirements Ontology - PAL Constraint 1:

Every requirement must have a unique id (i.e. no duplicate requirement id).

```
(defrange ?req1 :FRAME Requirement)
(defrange ?req2 :FRAME Requirement)
(forall ?req1
    (forall ?req2
        (=> (and (own-slot-not-null id ?req1)
                 (own-slot-not-null id ?req2))
            (=> (/= ?req1 ?req2)
            (/= (id ?req1)(id ?req2))))))
```

Requirements Ontology - PAL Constraint 2:

Every requirement category (functional and non-functional) must have a unique id (i.e. no duplicate requirement category id).

```
(defrange ?reqcat1 :FRAME RequirementCategory)
(defrange ?reqcat2 :FRAME RequirementCategory)
(forall ?reqcat1
    (forall ?reqcat2
        (=> (and (own-slot-not-null id ?reqcat1)
                 (own-slot-not-null id ?reqcat2))
            (=> (/= ?reqcat1 ?reqcat2)
            (/= (id ?reqcat1)(id ?reqcat2))))))
```

Requirements Ontology - PAL Constraint 3:

Every system requirements specification (SRS) must have a unique id (i.e. no duplicate system requirements specification id).

```
(defrange ?srs1 :FRAME SystemRequirementsSpecificationSRS)
(defrange ?srs2 :FRAME SystemRequirementsSpecificationSRS)
```

```
(forall ?srs1
    (forall ?srs2
        (=> (and (own-slot-not-null id ?srs1)
                 (own-slot-not-null id ?srs2))
            (=> (/= ?srs1 ?srs2)
            (/= (id ?srs1)(id ?srs2))))))
```

Requirements Ontology - PAL Constraint 4:

All priority values should be between min value 0.0 and max value 1.0.

```
(defrange ?priority :FRAME Requirement)
(forall ?priority (=> (own-slot-not-null priority ?priority)
                  (and (or (> (priority ?priority) 0.0)
                           (= (priority ?priority) 0.0))
                       (or (< (priority ?priority) 1.0)
                           (= (priority ?priority) 1.0)))))
```

Requirements Ontology - PAL Constraint 5:

All risk level values should be between min value 0.0 and max value 1.0.

```
(defrange ?risk :FRAME Requirement)
(forall ?priority (=> (own-slot-not-null risk_level ?risk)
                  (and (or (> (risk_level ?risk) 0.0)
                           (= (isk_level ?risk) 0.0))
                       (or (< (risk_level ?risk) 1.0)
                           (= (risk_level ?risk) 1.0)))))
```

Requirements Ontology - PAL Constraint 6:

Each requirement must have at least one requirement category.

```
(defrange ?req :FRAME Requirement)
(forall ?req (=> (own-slot-not-null id ?req)
                 (own-slot-not-null isReqOf ?req)))
```

Requirements Ontology - PAL Constraint 7:

Each requirement category must have at least one requirement.

```
(defrange ?cat :FRAME RequirementCategory)
(forall ?cat (=> (own-slot-not-null id ?cat)
                 (own-slot-not-null hasReq ?cat)))
```

Requirements Ontology - PAL Constraint 8:

Each requirement must be part of at least one system requirements specification (SRS) document.

```
(defrange ?req :FRAME Requirement)
(forall ?req (=> (own-slot-not-null id ?req)
                 (own-slot-not-null isPartOf ?req)))
```

Requirements Ontology - PAL Constraint 9:

Each system requirements specification (SRS) document must have at least one requirement.

```
(defrange ?req :FRAME SystemRequirementsSpecificationSRS)
(forall ?req (=> (own-slot-not-null id ?req)
                 (own-slot-not-null hasPart ?req)))
```

Requirements Ontology - PAL Constraint 10:

A requirement should be refined from no more than one requirement.

```
(defrange ?req :FRAME Requirement)
(forall ?req (=> (own-slot-not-null id ?req)
            (< (number-of-slot-values refinesFrom ?req) 2)))
```

Requirements Ontology - PAL Constraint 11:

Each requirement must be a requirement in at least one viewpoint.

```
(defrange ?req :FRAME Requirement)
(forall ?req (=> (own-slot-not-null id ?req)
                 (own-slot-not-null isRequirementsOf ?req)))
```

Requirements Ontology - PAL Constraint 12:

A requirement category can only be a sub requirement category of at most one requirement category.

```
(defrange ?reqcat :FRAME RequirementCategory)
(forall ?reqcat (=> (own-slot-not-null id ?reqcat)
            (and (own-slot-not-null isSubReqCategoryOf ?reqcat)
                 (< (number-of-slot-values isSubCategoryOf ?reqcat) 2))))
```

Requirements Ontology - PAL Constraint 13:

If two requirement categories have the same category name and if they have sub requirement category, then they must have the same sub requirement category.

```
(defrange ?reqcat1 :FRAME RequirementCategory)
(defrange ?reqcat2 :FRAME RequirementCategory)
(forall ?reqcat1
    (forall ?reqcat2
        (=> (and (own-slot-not-null isSubReqCategoryOf ?reqcat1)
                 (own-slot-not-null isSubReqCategoryOf ?reqcat2))
        (=> (= (category ?reqcat1)(category ?reqcat2))
            (= (isSubReqCategoryOf ?reqcat1)(isSubReqCategoryOf ?reqcat2))))))
```

Requirements Ontology - PAL Constraint 14:

Every requirement category name must be unique (i.e. no duplicate requirement category name).

```
(defrange ?reqcat1 :FRAME RequirementCategory)
(defrange ?reqcat2 :FRAME RequirementCategory)
(forall ?reqcat1
    (forall ?reqcat2
```

```
(=> (and (own-slot-not-null category ?reqcat1)
         (own-slot-not-null category ?reqcat2))
    (=> (/= (id ?reqcat1)(id ?reqcat2))
    (/= (category ?reqcat1)(category ?reqcat2))))))
```

Persona Ontology - PAL Constraint 15:

Every persona must have a unique id (i.e. no duplicate persona id).

```
(defrange ?pers1 :FRAME Persona)
(defrange ?pers2 :FRAME Persona)
(forall ?pers1
    (forall ?pers2
        (=> (and (own-slot-not-null id ?pers1)
                 (own-slot-not-null id ?pers2))
            (=> (/= ?pers1 ?pers2)
            (/= (id ?pers1)(id ?pers2))))))
```

Persona Ontology - PAL Constraint 16:

Every name must have a unique id (i.e. no duplicate name id).

```
(defrange ?name1 :FRAME Name)
(defrange ?name2 :FRAME Name)
(forall ?name1
    (forall ?name2
        (=> (and (own-slot-not-null id ?name1)
                 (own-slot-not-null id ?name2))
            (=> (/= ?name1 ?name2)
            (/= (id ?name1)(id ?name2))))))
```

Persona Ontology - PAL Constraint 17:

Every education must have a unique id (i.e. no duplicate education id).

```
(defrange ?edu1 :FRAME Education)
(defrange ?edu2 :FRAME Education)
(forall ?edu1
    (forall ?edu2
        (=> (and (own-slot-not-null id ?edu1)
                 (own-slot-not-null id ?edu2))
            (=> (/= ?edu1 ?edu2)
```

```
        (/= (id ?edu1)(id ?edu2))))))
```

Persona Ontology - PAL Constraint 18:

Every occupation must have a unique id (i.e. no duplicate occupation id).

```
(defrange ?occu1 :FRAME Occupation)
(defrange ?occu2 :FRAME Occupation)
(forall ?occu1
    (forall ?occu2
        (=> (and (own-slot-not-null id ?occu1)
                 (own-slot-not-null id ?occu2))
            (=> (/= ?occu1 ?occu2)
            (/= (id ?occu1)(id ?occu2))))))
```

Persona Ontology - PAL Constraint 19:

Every language proficiency (writing, speaking, and listening) must have a unique id (i.e. no duplicate language proficiency id).

```
(defrange ?lp1 :FRAME LanguageProficiency)
(defrange ?lp2 :FRAME LanguageProficiency)
(forall ?lp1
    (forall ?lp2
        (=> (and (own-slot-not-null id ?lp1)
                 (own-slot-not-null id ?lp2))
            (=> (/= ?lp1 ?lp2)
            (/= (id ?lp1)(id ?lp2))))))
```

Persona Ontology - PAL Constraint 20:

Every environment must have a unique id (i.e. no duplicate environment id).

```
(defrange ?env1 :FRAME Environment)
(defrange ?env2 :FRAME Environment)
(forall ?env1
    (forall ?env2
```

```
        (=> (and (own-slot-not-null id ?env1)
                 (own-slot-not-null id ?env2))
            (=> (/= ?env1 ?env2)
            (/= (id ?env1)(id ?env2))))))
```

Persona Ontology - PAL Constraint 21:

Every ability must have a unique id (i.e. no duplicate ability id).

```
(defrange ?ab1 :FRAME Ability)
(defrange ?ab2 :FRAME Ability)
(forall ?ab1
    (forall ?ab2
        (=> (and (own-slot-not-null id ?ab1)
                 (own-slot-not-null id ?ab2))
            (=> (/= ?ab1 ?ab2)
            (/= (id ?ab1)(id ?ab2))))))
```

Persona Ontology - PAL Constraint 22:

Every interest must have a unique id (i.e. no duplicate interest id).

```
(defrange ?int1 :FRAME Interest)
(defrange ?int2 :FRAME Interest)
(forall ?int1
    (forall ?int2
        (=> (and (own-slot-not-null id ?int1)
                 (own-slot-not-null id ?int2))
            (=> (/= ?int1 ?int2)
            (/= (id ?int1)(id ?int2))))))
```

Persona Ontology - PAL Constraint 23:

Every interest category must have a unique id (i.e. no duplicate interest category id).

```
(defrange ?intcat1 :FRAME InterestCategory)
(defrange ?intcat2 :FRAME InterestCategory)
```

```
(forall ?intcat1
    (forall ?intcat2
        (=> (and (own-slot-not-null id ?intcat1)
                 (own-slot-not-null id ?intcat2))
            (=> (/= ?intcat1 ?intcat2)
            (/= (id ?intcat1)(id ?intcat2))))))
```

Persona Ontology - PAL Constraint 24:

Every role must have a unique id (i.e. no duplicate role id).

```
(defrange ?rol1 :FRAME Role)
(defrange ?rol2 :FRAME Role)
(forall ?rol1
    (forall ?rol2
        (=> (and (own-slot-not-null id ?rol1)
                 (own-slot-not-null id ?rol2))
            (=> (/= ?rol1 ?rol2)
            (/= (id ?rol1)(id ?rol2))))))
```

Persona Ontology - PAL Constraint 25:

Every knowledge (domain, web, and computer) must have a unique id (i.e. no duplicate knowledge id).

```
(defrange ?kno1 :FRAME Knowledge)
(defrange ?kno2 :FRAME Knowledge)
(forall ?kno1
    (forall ?kno2
        (=> (and (own-slot-not-null id ?kno1)
                 (own-slot-not-null id ?kno2))
            (=> (/= ?kno1 ?kno2)
            (/= (id ?kno1)(id ?kno2))))))
```

Persona Ontology - PAL Constraint 26:

Every concern must have a unique id (i.e. no duplicate concern id).

```
(defrange ?con1 :FRAME Concern)
(defrange ?con2 :FRAME Concern)
(forall ?con1
    (forall ?con2
        (=> (and (own-slot-not-null id ?con1)
                 (own-slot-not-null id ?con2))
            (=> (/= ?con1 ?con2)
            (/= (id ?con1)(id ?con2))))))
```

Persona Ontology - PAL Constraint 27:

Every usability preference must have a unique id (i.e. no duplicate usability preference id).

```
(defrange ?up1 :FRAME UsabilityPref)
(defrange ?up2 :FRAME UsabilityPref)
(forall ?up1
    (forall ?up2
        (=> (and (own-slot-not-null id ?up1)
                 (own-slot-not-null id ?up2))
            (=> (/= ?up1 ?up2)
            (/= (id ?up1)(id ?up2))))))
```

Persona Ontology - PAL Constraint 28:

Each persona must have at most one name.

```
(defrange ?persona :FRAME Persona)
(forall ?persona (=> (own-slot-not-null id ?persona)
                     (and (own-slot-not-null hasPersonaName ?persona)
                          (< (number-of-slot-values hasPersonaName ?persona) 2))))
```

Persona Ontology - PAL Constraint 29:

Each name must have at most one persona.

```
(defrange ?name :FRAME Name)
(forall ?name (=> (and (own-slot-not-null first_name ?name)
                       (own-slot-not-null last_name ?name)
                       (own-slot-not-null middle_name ?name))
                  (and (own-slot-not-null isPersonaNameOf ?name)
                       (< (number-of-slot-values isPersonaNameOf ?name) 2))))
```

Persona Ontology - PAL Constraint 30:

No two personas can have the same name.

```
(defrange ?name1 :FRAME Name)
(defrange ?name2 :FRAME Name)
(forall ?name1
    (forall ?name2
        (=> (and (own-slot-not-null isPersonaNameOf ?name1)
                 (own-slot-not-null isPersonaNameOf ?name2))
            (=> (and (= (first_name ?name1)(first_name ?name2))
                     (= (last_name ?name1)(last_name ?name2))
                     (= (middle_name ?name1)(middle_name ?name2)))
                (= (isPersonaNameOf ?name1)(isPersonaNameOf ?name2))))))
```

Persona Ontology - PAL Constraint 31:

Each persona must have at most one highest education level.

```
(defrange ?persona :FRAME Persona)
(forall ?persona (=> (own-slot-not-null id ?persona)
                     (and (own-slot-not-null hasHighestEducation ?persona)
                          (< (number-of-slot-values hasHighestEducation ?persona) 2))))
```

Persona Ontology - PAL Constraint 32:

Each persona must personify at least one role.

```
(defrange ?persona :FRAME Persona)
(forall ?persona (=> (own-slot-not-null id ?persona)
                     (own-slot-not-null personifies ?persona)))
```

Persona Ontology - PAL Constraint 33:

Each persona must engage in at least one environment.

```
(defrange ?persona :FRAME Persona)
(forall ?persona (=> (own-slot-not-null id ?persona)
                     (own-slot-not-null engagesIn ?persona)))
```

Persona Ontology - PAL Constraint 34:

Each environment must be engaged by at least one persona.

```
(defrange ?env :FRAME Environment)
(forall ?env (=> (own-slot-not-null id ?env)
                 (own-slot-not-null engagedBy ?env)))
```

Persona Ontology - PAL Constraint 35:

Each role must be personified by at least one persona.

```
(defrange ?role :FRAME Role)
(forall ?role (=> (own-slot-not-null id ?role)
                  (own-slot-not-null personifiedBy ?role)))
```

Persona Ontology - PAL Constraint 36:

Each highest education level must be a highest education level of at least one persona.

```
(defrange ?edu :FRAME Education)
(forall ?edu (=> (own-slot-not-null id ?edu)
                 (own-slot-not-null isHighestEducationOf ?edu)))
```

Persona Ontology - PAL Constraint 37:

Each persona must be a persona in at least one viewpoint.

```
(defrange ?persona :FRAME Persona)
(forall ?persona (=> (own-slot-not-null id ?persona)
                     (own-slot-not-null isPersonaOf ?persona)))
```

Persona Ontology - PAL Constraint 38:

Each role must be a role in at least one viewpoint.

```
(defrange ?role :FRAME Role)
(forall ?role (=> (own-slot-not-null id ?role)
                  (own-slot-not-null isRoleOf ?role)))
```

Persona Ontology - PAL Constraint 39:

Each environment must be an environment in at least one viewpoint.

```
(defrange ?env :FRAME Environment)
(forall ?env (=> (own-slot-not-null id ?env)
                 (own-slot-not-null isEnvironmentOf ?env)))
```

Persona Ontology - PAL Constraint 40:

Each concern must be addressed by at least one viewpoint.

```
(defrange ?concern :FRAME Concern)
(forall ?concern (=> (own-slot-not-null id ?concern)
                     (own-slot-not-null addressedBy ?concern)))
```

Persona Ontology - PAL Constraint 41:

An interest can only have at most one interest category.

```
(defrange ?int :FRAME Interest)
(forall ?int (=> (own-slot-not-null id ?int)
                 (and (own-slot-not-null isInterestPartOf ?int)
                      (< (number-of-slot-values isInterestPartOf ?int) 2))))
```

Persona Ontology - PAL Constraint 42:

An interest category can only be a sub interest category of at most one interest category.

```
(defrange ?intcat :FRAME InterestCategory)
(forall ?intcat (=> (own-slot-not-null id ?intcat)
                 (and (own-slot-not-null isSubCategoryOf ?intcat)
                      (< (number-of-slot-values isSubCategoryOf ?intcat) 2))))
```

Persona Ontology - PAL Constraint 43:

If two interests have the same name, then they must belong to the same interest category.

```
(defrange ?interest1 :FRAME Interest)
(defrange ?interest2 :FRAME Interest)
(forall ?interest1
    (forall ?interest2
        (=> (and (own-slot-not-null isInterestPartOf ?interest1)
                 (own-slot-not-null isInterestPartOf ?interest2))
            (=> (= (name ?interest1)(name ?interest2))
            (= (isInterestPartOf ?interest1)(isInterestPartOf ?interest2))))))
```

Persona Ontology - PAL Constraint 44:

If two interest categories have the same category name and if they have sub interest category, then they must have the same sub interest category.

```
(defrange ?intcat1 :FRAME InterestCategory)
(defrange ?intcat2 :FRAME InterestCategory)
```

```
(forall ?intcat1
    (forall ?intcat2
        (=> (and (own-slot-not-null isSubCategoryOf ?intcat1)
                 (own-slot-not-null isSubCategoryOf ?intcat2))
            (=> (= (category ?intcat1)(category ?intcat2))
            (= (isSubCategoryOf ?intcat1)(isSubCategoryOf ?intcat2))))))
```

Persona Ontology - PAL Constraint 45:

If the ability (physical, cognitive) type is not "Other", then the ability (physical, cognitive) type other field must be empty.

```
(defrange ?ab :FRAME Ability)
(forall ?ab (=>(and (own-slot-not-null id ?ab)
                    (not (or (physical_type ?ab "Other")
                             (cognitive_type ?ab "Other"))))
           (not (or (own-slot-not-null physical_type_other ?ab)
                    (own-slot-not-null cognitive_type_other ?ab)))))
```

Persona Ontology - PAL Constraint 46:

Every physical ability with the same physical type other and ability level must be the physical ability of the same persona.

```
(defrange ?phy1 :FRAME Physical)
(defrange ?phy2 :FRAME Physical)
(forall ?phy1
    (forall ?phy2
        (=> (and (own-slot-not-null physical_type_other ?phy1)
                 (own-slot-not-null ability_level ?phy1)
                 (own-slot-not-null physical_type_other ?phy2)
                 (own-slot-not-null ability_level ?phy2))
            (=> (and (/= ?phy1 ?phy2)
                     (= (physical_type_other ?phy1) (physical_type_other ?phy2))
                     (= (ability_level ?phy1) (ability_level ?phy2)))
            (=  (isAbilityOf ?phy1) (isAbilityOf ?phy2))))))
```

Persona Ontology - PAL Constraint 47:

Every cognitive ability with the same cognitive type other and ability level must be the cognitive ability of the same persona.

```
(defrange ?cog1 :FRAME Cognitive)
(defrange ?cog2 :FRAME Cognitive)
(forall ?cog1
    (forall ?cog2
        (=> (and (own-slot-not-null cognitive_type_other ?cog1)
                 (own-slot-not-null ability_level ?cog1)
                 (own-slot-not-null cognitive_type_other ?cog2)
                 (own-slot-not-null ability_level ?cog2))
            (=> (and (/= ?cog1 ?cog2)
                     (= (cognitive_type_other ?phy1) (cognitive_type_other ?cog2))
                     (= (ability_level ?cog1) (ability_level ?cog2)))
            (=  (isAbilityOf ?cog1) (isAbilityOf ?cog2))))))
```

Persona Ontology - PAL Constraint 48:

Every ability (physical, cognitive) with the same type other and ability level must be the ability of the same persona.

```
(defrange ?phy :FRAME Physical)
(defrange ?cog :FRAME Cognitive)
(forall ?phy
    (forall ?cog
        (=> (and (own-slot-not-null physical_type_other ?phy)
                 (own-slot-not-null cognitive_type_other ?cog))
            (=> (and (/= ?phy ?cog)
                     (= (physical_type_other ?phy) (cognitive_type_other ?cog))
                     (= (ability_level ?phy) (ability_level ?cog)))
            (=  (isAbilityOf ?phy) (isAbilityOf ?cog))))))
```

Persona Ontology - PAL Constraint 49:

Every speaking language proficiency with the same speaking language name and proficiency level must be the speaking language of the same persona.

```
(defrange ?spk1 :FRAME Speaking)
(defrange ?spk2 :FRAME Speaking)
(forall ?spk1
    (forall ?spk2
        (=> (and (own-slot-not-null speaking_language ?spk1)
                 (own-slot-not-null proficiency_level ?spk1)
                 (own-slot-not-null speaking_language ?spk2)
                 (own-slot-not-null proficiency_level ?spk2))
            (=> (and (/= ?spk1 ?spk2)
                     (= (speaking_language ?spk1) (speaking_language ?spk2))
                     (= (proficiency_level ?spk1) (proficiency_level ?spk2)))
            (=  (isLanguageProficiencyOf ?spk1) (isLanguageProficiencyOf ?spk2))))))
```

Persona Ontology - PAL Constraint 50:

Every writing language proficiency with the same writing language name and proficiency level must be the writing language of the same persona.

```
(defrange ?wr1 :FRAME Writing)
(defrange ?wr2 :FRAME Writing)
(forall ?wr1
    (forall ?wr2
        (=> (and (own-slot-not-null writing_language ?wr1)
                 (own-slot-not-null proficiency_level ?wr1)
                 (own-slot-not-null writing_language ?wr2)
                 (own-slot-not-null proficiency_level ?wr2))
            (=> (and (/= ?wr1 ?wr2)
                     (= (writing_language ?wr1) (writing_language ?wr2))
                     (= (proficiency_level ?wr1) (proficiency_level ?wr2)))
            (=  (isLanguageProficiencyOf ?wr1) (isLanguageProficiencyOf ?wr2))))))
```

Persona Ontology - PAL Constraint 51:

Every listening language proficiency with the same listening language name and proficiency level must be the listening language of the same persona.

```
(defrange ?lis1 :FRAME Listening)
(defrange ?lis2 :FRAME Listening)
(forall ?lis1
    (forall ?lis2
        (=> (and (own-slot-not-null listening_language ?lis1)
                 (own-slot-not-null proficiency_level ?lis1)
                 (own-slot-not-null listening_language ?lis2)
                 (own-slot-not-null proficiency_level ?lis2))
            (=> (and (/= ?lis1 ?lis2)
                     (= (listening_language ?lis1) (listening_language ?lis2))
                     (= (proficiency_level ?lis1) (proficiency_level ?lis2)))
            (=  (isLanguageProficiencyOf ?lis1) (isLanguageProficiencyOf ?lis2))))))
```

Persona Ontology - PAL Constraint 52:

Every computer knowledge with the same computer knowledge name and knowledge level must be the computer knowledge of the same persona.

```
(defrange ?com1 :FRAME Computer)
(defrange ?com2 :FRAME Computer)
(forall ?com1
    (forall ?com2
        (=> (and (own-slot-not-null computer_knowledge ?com1)
                 (own-slot-not-null knowledge_level ?com1)
                 (own-slot-not-null computer_knowledge ?com2)
                 (own-slot-not-null knowledge_level ?com2))
            (=> (and (/= ?com1 ?com2)
                     (= (computer_knowledge ?com1) (computer_knowledge ?com2))
                     (= (knowledge_level ?com1) (knowledge_level ?com2)))
            (=  (isKnowledgeOf ?com1) (isKnowledgeOf ?com2))))))
```

Persona Ontology - PAL Constraint 53:

Every domain knowledge with the same domain knowledge name and knowledge level must be the domain knowledge of the same persona.

```
(defrange ?dom1 :FRAME Domain)
(defrange ?dom2 :FRAME Domain)
(forall ?dom1
    (forall ?dom2
        (=> (and (own-slot-not-null domain_knowledge ?dom1)
                 (own-slot-not-null knowledge_level ?dom1)
                 (own-slot-not-null domain_knowledge ?dom2)
                 (own-slot-not-null knowledge_level ?dom2))
            (=> (and (/= ?dom1 ?dom2)
                     (= (domain_knowledge ?dom1) (domain_knowledge ?dom2))
                     (= (knowledge_level ?dom1) (knowledge_level ?dom2)))
            (=  (isKnowledgeOf ?dom1) (isKnowledgeOf ?dom2))))))
```

<u>Persona Ontology - PAL Constraint 54</u>:

Every web knowledge with the same web knowledge name and knowledge level must be the web knowledge of the same persona.

```
(defrange ?web1 :FRAME Web)
(defrange ?web2 :FRAME Web)
(forall ?web1
    (forall ?web2
        (=> (and (own-slot-not-null web_knowledge ?web1)
                 (own-slot-not-null knowledge_level ?web1)
                 (own-slot-not-null web_knowledge ?web2)
                 (own-slot-not-null knowledge_level ?web2))
            (=> (and (/= ?web1 ?web2)
                     (= (web_knowledge ?web1) (web_knowledge ?web2))
                     (= (knowledge_level ?web1) (knowledge_level ?web2)))
            (=  (isKnowledgeOf ?web1) (isKnowledgeOf ?web2))))))
```

Behavioral-GST Ontology - PAL Constraint 55:

Every goal must have a unique id (i.e. no duplicate persona id).

```
(defrange ?goal1 :FRAME Goal)
(defrange ?goal2 :FRAME Goal)
(forall ?goal1
    (forall ?goal2
        (=> (and (own-slot-not-null id ?goal1)
                 (own-slot-not-null id ?goal2))
            (=> (/= ?goal1 ?goal2)
            (/= (id ?goal1)(id ?goal2))))))
```

Behavioral-GST Ontology - PAL Constraint 56:

Every scenario must have a unique id (i.e. no duplicate scenario id).

```
(defrange ?scen1 :FRAME Scenario)
(defrange ?scen2 :FRAME Scenario)
(forall ?scen1
    (forall ?scen2
        (=> (and (own-slot-not-null id ?scen1)
                 (own-slot-not-null id ?scen2))
            (=> (/= ?scen1 ?scen2)
            (/= (id ?scen1)(id ?scen2))))))
```

Behavioral-GST Ontology - PAL Constraint 57:

Every task must have a unique id (i.e. no duplicate task id).

```
(defrange ?tk1 :FRAME Task)
(defrange ?tk2 :FRAME Task)
(forall ?tk1
    (forall ?tk2
        (=> (and (own-slot-not-null id ?tk1)
                 (own-slot-not-null id ?tk2))
            (=> (/= ?tk1 ?tk2)
```

```
        (/= (id ?tk1)(id ?tk2))))))
```

Behavioral-GST Ontology - PAL Constraint 58:

Every viewpoint must have a unique id (i.e. no duplicate viewpoint id).

```
(defrange ?vp1 :FRAME Viewpoint)
(defrange ?vp2 :FRAME Viewpoint)
(forall ?vp1
    (forall ?vp2
        (=> (and (own-slot-not-null id ?vp1)
                 (own-slot-not-null id ?vp2))
            (=> (/= ?vp1 ?vp2)
            (/= (id ?vp1)(id ?vp2))))))
```

Behavioral-GST Ontology - PAL Constraint 59:

Every stakeholder must have a unique id (i.e. no duplicate stakeholder id).

```
(defrange ?stk1 :FRAME Stakeholder)
(defrange ?stk2 :FRAME Stakeholder)
(forall ?stk1
    (forall ?stk2
        (=> (and (own-slot-not-null id ?stk1)
                 (own-slot-not-null id ?stk2))
            (=> (/= ?stk1 ?stk2)
            (/= (id ?stk1)(id ?stk2))))))
```

Behavioral-GST Ontology - PAL Constraint 60:

Every actor must have a unique id (i.e. no duplicate actor id).

```
(defrange ?act1 :FRAME Actor)
(defrange ?act2 :FRAME Actor)
(forall ?act1
    (forall ?act2
        (=> (and (own-slot-not-null id ?act1)
```

```
                (own-slot-not-null id ?act2))
            (=> (/= ?act1 ?act2)
            (/= (id ?act1)(id ?act2))))))
```

Behavioral-GST Ontology - PAL Constraint 61:

Every constraint must have a unique id (i.e. no duplicate constraint id).

```
(defrange ?con1 :FRAME Constraint)
(defrange ?con2 :FRAME Constraint)
(forall ?con1
    (forall ?con2
        (=> (and (own-slot-not-null id ?con1)
                (own-slot-not-null id ?con2))
            (=> (/= ?con1 ?con2)
            (/= (id ?con1)(id ?con2))))))
```

Behavioral-GST Ontology - PAL Constraint 62:

Every obstacle must have a unique id (i.e. no duplicate obstacle id).

```
(defrange ?ob1 :FRAME Obstacle)
(defrange ?ob2 :FRAME Obstacle)
(forall ?ob1
    (forall ?ob2
        (=> (and (own-slot-not-null id ?ob1)
                (own-slot-not-null id ?ob2))
            (=> (/= ?ob1 ?ob2)
            (/= (id ?ob1)(id ?ob2))))))
```

Behavioral-GST Ontology - PAL Constraint 63:

Every pre-requisite must have a unique id (i.e. no duplicate pre-requisite id).

```
(defrange ?pre1 :FRAME Prerequisite)
(defrange ?pre2 :FRAME Prerequisite)
(forall ?pre1
```

```
    (forall ?pre2
        (=> (and (own-slot-not-null id ?pre1)
                 (own-slot-not-null id ?pre2))
            (=> (/= ?pre1 ?pre2)
            (/= (id ?pre1)(id pre2))))))
```

Behavioral-GST Ontology - PAL Constraint 64:

Every post-requisite must have a unique id (i.e. no duplicate post-requisite id).

```
(defrange ?post1 :FRAME Postrequisite)
(defrange ?post2 :FRAME Postrequisite)
(forall ?post1
    (forall ?post2
        (=> (and (own-slot-not-null id ?post1)
                 (own-slot-not-null id ?post2))
            (=> (/= ?post1 ?post2)
            (/= (id ?post1)(id post2))))))
```

Behavioral-GST Ontology - PAL Constraint 65:

Each goal must be a goal of at least one viewpoint.

```
(defrange ?goal :FRAME Goal)
(forall ?goal (=> (own-slot-not-null id ?goal)
                  (own-slot-not-null isGoalOf ?goal)))
```

Behavioral-GST Ontology - PAL Constraint 66:

Each scenario must be a scenario of at least one viewpoint.

```
(defrange ?scen :FRAME Scenario)
(forall ?scen (=> (own-slot-not-null id ?scen)
                  (own-slot-not-null isScenarioOf ?scen))
```

Behavioral-GST Ontology - PAL Constraint 67:

Each task must be a task of at least one viewpoint.

```
(defrange ?task :FRAME Task)
(forall ?task (=> (own-slot-not-null id ?task)
                  (own-slot-not-null isTaskOf ?task))
```

Behavioral-GST Ontology - PAL Constraint 68:

Each stakeholder must be a stakeholder of at least one viewpoint.

```
(defrange ?stake :FRAME Stakeholder)
(forall ?stake (=> (own-slot-not-null id ?stake)
                  (own-slot-not-null isStakeholderOf ?stake))
```

Behavioral-GST Ontology - PAL Constraint 69:

Each viewpoint must have at least one requirement.

```
(defrange ?vp :FRAME Viewpoint)
(forall ?vp (=> (own-slot-not-null id ?vp)
                  (own-slot-not-null hasRequirements ?vp))
```

Behavioral-GST Ontology - PAL Constraint 70:

Each viewpoint must have at least one goal.

```
(defrange ?vp :FRAME Viewpoint)
(forall ?vp (=> (own-slot-not-null id ?vp)
                  (own-slot-not-null hasGoal ?vp))
```

Behavioral-GST Ontology - PAL Constraint 71:

Each viewpoint must have at least one scenario.

```
(defrange ?vp :FRAME Viewpoint)
(forall ?vp (=> (own-slot-not-null id ?vp)
                  (own-slot-not-null hasScenario ?vp))
```

Behavioral-GST Ontology - PAL Constraint 72:

Each viewpoint must have at least one task.

```
(defrange ?vp :FRAME Viewpoint)
(forall ?vp (=> (own-slot-not-null id ?vp)
                (own-slot-not-null hasTask ?vp))
```

Behavioral-GST Ontology - PAL Constraint 73:

Each scenario must have at least one task.

```
(defrange ?scen :FRAME Scenario)
(forall ?scen (=> (own-slot-not-null id ?scen)
                (own-slot-not-null hasTask ?scen))
```

Behavioral-GST Ontology - PAL Constraint 74:

Each task is task of at least one scenario.

```
(defrange ?task :FRAME Task)
(forall ?task (=> (own-slot-not-null id ?task)
                (own-slot-not-null isTaskOf ?task))
```

Behavioral-GST Ontology - PAL Constraint 75:

If a goal has status "withdrawn", then it should not have any relationships with one or more concepts.

```
(defrange ?goal :FRAME Goal)
(forall ?goal (=>(and (own-slot-not-null id ?goal)
                      (status ?goal "Withdrawn"))
                 (and (not (own-slot-not-null hasConstraint ?goal))
                      (not (own-slot-not-null isGoalOf ?goal))
                      (not (own-slot-not-null hasGoalType ?goal))
                      (not (own-slot-not-null hasStakeholder ?goal))
                      (not (own-slot-not-null supports ?goal))
                      (not (own-slot-not-null supportedBy ?goal))
                      (not (own-slot-not-null hasObstacle ?goal))
                      (not (own-slot-not-null conflictsWith ?goal))
                      (not (own-slot-not-null AND-refinesFrom ?goal))
                      (not (own-slot-not-null OR-refinesInto ?goal))
                      (not (own-slot-not-null OR-refinesFrom ?goal)))))
```

Behavioral-GST Ontology - PAL Constraint 76:

No two viewpoints can have the same persona name, environment, and role (i.e. a viewpoint is uniquely identified by persona, environment, and role).

```
(defrange ?vpname1 :FRAME Viewpoint)
(defrange ?vpname2 :FRAME Viewpoint)
(forall ?vpname1
    (forall ?vpname2
        (=> (and (own-slot-not-null hasPersona ?vpname1)
                 (own-slot-not-null hasEnvironment ?vpname1)
                 (own-slot-not-null hasRole ?vpname1)
                 (own-slot-not-null hasPersona ?vpname2)
                 (own-slot-not-null hasEnvironment ?vpname2)
                 (own-slot-not-null hasRole ?vpname2))
            (=> (/= (id ?vpname1)(id ?vpname2))
                (or (/= (hasPersona ?vpname1)(hasPersona ?vpname2))
                    (/= (hasEnvironment ?vpname1)(hasEnvironment ?vpname2))
                    (/= (hasRole ?vpname1)(hasRole ?vpname2)))))))
```

Requirements Ontology – Jess Consistency Rule 1:

A functional requirement category cannot also be a non-functional requirement category (no duplicate requirement category).

```
(mapclass Functional)
(mapclass NonFunctional)
(defrule no_identical_requirement_category
  (object (is-a Functional) (category ?c) (id ?i))
  (object (is-a NonFunctional) (category ?c) (id ~?i))
  =>
(printout t "The category " ?c " appears in both Functional and Non-Functional requirement category and should be resolved." crlf))
```

Requirements Ontology – Jess Consistency Rule 2:

A requirement cannot appear in both functional and non-functional requirement category (no duplicate requirement in functional and non-functional requirement category).

```
(mapclass Functional)
(mapclass NonFunctional)
(defrule no_identical_requirement_in_both_requirement_category
  (object (is-a Functional) (OBJECT ?obj1) (category ?c) (hasReq $? ?r1 $?))
  (object (is-a NonFunctional)(OBJECT ?obj2) (category ~?c) (hasReq $? ?r2 $? &:(eq (slot-get ?r2 id) (slot-get ?r1 id))))
  =>
  (printout t "Requirement (" (slot-get ?r1 id) ") appears in both Functional (" (slot-get ?obj1 category) ") and Non-Functional (" (slot-get ?obj2 category) ") requirement category and should be resolved." crlf))
```

Requirements Ontology – Jess Consistency Rule 3:

If a requirement A is refined into a requirement B, then requirement B must be refined from only requirement A.

```
(mapclass Requirement)
(defrule refinesInto_refinesFrom
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (refinesInto $? ?r1 $?))
  (object (is-a Requirement) (OBJECT ?r1) (refinesFrom ?r2&:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Requirement " ?i " and refinement relationship (" (slot-get ?r1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 4:

If a requirement A constrains a requirement B, then requirement B must be constrained by requirement A.

```
(mapclass Requirement)
(defrule constrains_constrainedBy
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (constrains $? ?c1 $?))
  (object (is-a Requirement) (OBJECT ?c1) (constrainedBy $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Requirement " ?i " and constrains relationship (" (slot-get ?c1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 5:

If a requirement A requires a requirement B, then requirement B must be required by requirement A.

```
(mapclass Requirement)
(defrule requires_requiredBy
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (requires $? ?r1 $?))
  (object (is-a Requirement) (OBJECT ?r1) (requiredBy $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t “Requirement " ?i " and requires relationship (" (slot-get ?r1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 6:

If a requirement A conflicts with a requirement B, then requirement B must also conflict with requirement A.

```
(mapclass Requirement)
(defrule conflictsWith
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (conflictsWith $? ?c1 $?))
  (object (is-a Requirement) (OBJECT ?c1) (conflictsWith $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Requirement " ?i " and conflictsWith relationship (" (slot-get ?c1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 7:

If a requirement A meets a goal A, then goal A must be met by requirement A.

```
(mapclass Requirement)
(mapclass Goal)
(defrule meets_metBy
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (meets $? ?m1 $?))
  (object (is-a Goal) (OBJECT ?m1) (metBy ?m2&:(neq (slot-get ?m2 id) ?i)))
  =>
(printout t "Requirement " ?i " and meets relationship (" (slot-get ?m1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 8:

If a requirement A is operationalized by a scenario A, then scenario A must operationalize requirement A.

```
(mapclass Requirement)
(mapclass NormalScenario)
(mapclass ExceptionScenario)
(defrule operationalizedBy_operationalizes
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (operationalizedBy $? ?op1 $?))
  (object (is-a NormalScenario | ExceptionScenario) (OBJECT ?op1) (operationalizes ?op2&:(neq (slot-get ?op2 id) ?i)))
  =>
(printout t " Requirement " ?i " and operationalization relationship (" (slot-get ?op1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 9:

If a requirement A is part of a System Requirements Specification (SRS) document A, then SRS document A has requirement A.

```
(mapclass Requirement)
(mapclass SystemRequirementsSpecificationSRS)
(defrule isPartOf_hasPart
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (isPartOf $? ?p1 $?))
  (object (is-a SystemRequirementsSpecificationSRS) (OBJECT ?p1) (hasPart $? ?p2 $? &:(neq (slot-get ?p2 id) ?i)))
  =>
(printout t "Requirement " ?i " and isPartOf relationship (" (slot-get ?p1 id) ") are inconsistent." crlf))
```

<u>Requirements Ontology – Jess Consistency Rule 10</u>:

If a requirement A is a requirement category (functional or non-functional) A, then requirement category (functional or non-functional) A has requirement A.

```
(mapclass Requirement)
(mapclass Functional)
(mapclass NonFunctional)
(defrule isRequirementCategoryOf_hasRequirementCategory
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (isReqOf $? ?r1 $?))
  (object (is-a Functional | NonFunctional) (OBJECT ?r1) (hasReq $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Requirement " ?i " and isReqOf relationship (" (slot-get ?r1 id) ") are inconsistent." crlf))
```

<u>Requirements Ontology – Jess Consistency Rule 11</u>:

If a requirement category A is a sub-requirement category of a requirement category B, then requirement category B has at least sub-requirement category A.

```
(mapclass Functional)
(mapclass NonFunctional)
(defrule isSubRequirementCategoryOf_hasSubRequirementCategory
  (object (is-a Functional | NonFunctional) (OBJECT ?obj) (id ?i) (hasSubReqCategory $? ?r1 $?))
  (object (is-a Functional | NonFunctional) (OBJECT ?r1) (isSubReqCategoryOf ?r2&:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Requirement " ?i " and sub-requirement category relationship (" (slot-get ?r1 id) ") are inconsistent." crlf))
```

Requirements Ontology – Jess Consistency Rule 12:

If a requirement A is a requirement in a viewpoint A, then viewpoint A has at least requirement A.

```
(mapclass Requirement)
(mapclass Viewpoint)
(defrule isRequirementsOfViewpoint_viewpointHasRequirements
  (object (is-a Requirement) (OBJECT ?obj) (id ?i) (isRequirementsOf $? ?r1 $?))
  (object (is-a Viewpoint) (OBJECT ?r1) (hasRequirements $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Requirement " ?i " and isRequirementsOf relationship (" (slot-get ?r1 id) ") are inconsistent." crlf))
```

Persona Ontology - Jess Consistency Rule 13:

If a persona does not have a name, then add a temporary name PERSONA.

```
(mapclass Persona)
(mapclass Name)
(defrule set_temp_name_if_no_name
(object (is-a Persona) (hasPersonaName nil))
=>
(make-instance PERSONA of Name (first_name "PERSONA") (last_name "?")))
```

Persona Ontology - Jess Consistency Rule 14:

If a persona A has a name A, then name A is the name for only persona A.

```
(mapclass Persona)
(mapclass Name)
(defrule hasPersonaName_isPersonaNameOf
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasPersonaName ?n1))
  (object (is-a Name) (OBJECT ?n1) (isPersonaNameOf ?n2&:(eq (slot-get ?n2 id) ?i)))
  =>
(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and persona name (\"" (slot-get ?n1 first_name) " " (slot-get ?n1 middle_name) " " (slot-get ?n1 last_name) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 15:

If a persona A has a highest education A, then highest education A is the highest education level for at least persona A.

```
(mapclass Persona)
(mapclass Education)
(defrule hasHighestEducation_isHighestEducationOf
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasHighestEducation ?e1))
  (object (is-a Education) (OBJECT ?e1) (isHighestEducationOf $? ?e2 $? &:(eq (slot-get ?e2 id) ?i)))
  =>
(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasHighestOccupation relationship (\"" (slot-get ?e1 id) " - " (slot-get ?e1 highest_education_level) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 16:

If a persona A has an occupation A, then occupation A is an occupation for at least persona A.

```
(mapclass Persona)

(mapclass Occupation)

(defrule hasOccupation_isOccupationOf

  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasOccupation ?o1))

  (object (is-a Occupation) (OBJECT ?o1) (isOccupationOf $? ?o2 $? &:(eq (slot-get ?o2 id) ?i)))

  =>

(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasOccupation relationship (\"" (slot-get ?o1 id) " - " (slot-get ?o1 position) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 17:

If a persona A has a language proficiency A, then language proficiency A is the language proficiency for at least persona A.

```
(mapclass Persona)

(mapclass Writing)

(mapclass Speaking)

(mapclass Listening)

(defrule hasLanguageProficiency_isLanguageProficiencyOf

  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasLanguageProficiency $? ?lp1 $?))

  (object (is-a Writing | Speaking | Listening) (is-a-name ?type1) (OBJECT ?lp1) (isLanguageProficiencyOf $? ?lp2 $? &:(eq (slot-get ?lp2 id) ?i)))

  =>

(if (= ?type1 Writing)
then (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasLanguageProficiency relationship (\"" (slot-get ?lp1 id) " - " (slot-get ?lp1 writing_language) "\") are consistent." crlf)

else (if (= ?type1 Speaking)
then (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasLanguageProficiency relationship (\"" (slot-get ?lp1 id) " - " (slot-get ?lp1 speaking_language) "\") are consistent." crlf)

else (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasLanguageProficiency relationship (\"" (slot-get ?lp1 id) " - " (slot-get ?lp1 listening_language) "\") are consistent." crlf))))
```

Persona Ontology - Jess Consistency Rule 18:

If a persona A has a concern A, then concern A is a concern of at least persona A.

```
(mapclass Persona)
(mapclass Concern)
(defrule hasCocnern_isConcernOf
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasConcern $? ?c1 $?))
  (object (is-a Concern) (OBJECT ?c1) (isConcernOf $? ?c2 $? &:(eq (slot-get ?c2 id) ?i)))
  =>
(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasConcern relationship (\"" (slot-get ?c1 id) " - " (slot-get ?c1 name) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 19:

If a persona A engages in an environment A, then environment A is engaged by at least persona A.

```
(mapclass Persona)
(mapclass Environment)
(defrule engagesIn_engagedBy
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (engagesIn $? ?e1 $?))
  (object (is-a Environment) (OBJECT ?e1) (engagedBy $? ?e2 $? &:(eq (slot-get ?e2 id) ?i)))
  =>
(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and engagesIn relationship (\"" (slot-get ?e1 id) " - " (slot-get ?e1 name) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 20:

If a persona A plays (personifies) a role A, then role A is played by (personified by) at least persona A.

```
(mapclass Persona)
(mapclass Role)
(defrule personifies_personifiedBy
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (personifies $? ?p1 $?))
  (object (is-a Role) (OBJECT ?p1) (personifiedBy $? ?p2 $? &:(eq (slot-get ?p2 id) ?i)))
  =>
(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and personaifies relationship (\"" (slot-get ?p1 id) " - " (slot-get ?p1 name) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 21:

If a concern A relates to an environment A, then environment A has at least concern A.

```
(mapclass Role)
(mapclass Environment)
(defrule participatesIn_participatedBy
  (object (is-a Role) (OBJECT ?obj) (id ?i) (participatesIn $? ?p1 $?))
  (object (is-a Environment) (OBJECT ?p1) (participatedBy $? ?p2 $? &:(eq (slot-get ?p2 id) ?i)))
  =>
(printout t "Role (\"" ?i " - " (slot-get ?obj name) "\") and participatesIn relationship (\"" (slot-get ?p1 id) " - " (slot-get ?p1 name) "\") are consistent." crlf))
```

Persona Ontology - Jess Consistency Rule 22:

If a role A participates in an environment A, then environment A is participated by at least role A.

```
(mapclass Concern)
(mapclass Environment)
(defrule relatesTo_presents
  (object (is-a Concern) (OBJECT ?obj) (id ?i) (relatesTo $? ?r1 $?))
  (object (is-a Environment) (OBJECT ?r1) (presents $? ?r2 $? &:(eq (slot-get ?r2 id) ?i)))
  =>
(printout t "Concern (\"" ?i " - " (slot-get ?obj name) "\") and relatesTo relationship (\"" (slot-get ?r1 id) " - " (slot-get ?r1 name) "\") are consistent." crlf))
```

Persona Ontology – Jess Consistency Rule 23:

If an interest category A is a sub-interest category of interest category B, then interest category B has sub-interest category A.

```
(mapclass InterestCategory)
(defrule isSubCategoryOf_hasSubCategory
  (object (is-a InterestCategory) (OBJECT ?obj) (id ?i) (hasSubCategory $? ?c1 $?))
  (object (is-a InterestCategory) (OBJECT ?c1) (isSubCategoryOf ?c2&:(eq (slot-get ?c2 id) ?i)))
  =>
(printout t "Interest category (\"" ?i " - " (slot-get ?obj category) "\") and hasSubCategory relationship (\""
(slot-get ?c1 id) " - " (slot-get ?c1 category) "\") are consistent." crlf))
```

Persona Ontology – Jess Consistency Rule 24:

If an interest A is an interest category A, then interest category A has interest A.

```
(mapclass Interest)
(mapclass InterestCategory)
(defrule isInterestPartOf_hasInterestPart
  (object (is-a Interest) (OBJECT ?obj) (id ?i) (isInterestPartOf ?ip1))
  (object (is-a InterestCategory) (OBJECT ?ip1) (hasInterestPart $? ?ip2 $? &:(eq (slot-get ?ip2 id) ?i)))
  =>
(printout t "Interest (\"" ?i " - " (slot-get ?obj name) "\") and isInterestPartOf relationship (\"" (slot-get ?ip1
id) " - " (slot-get ?ip1 category) "\") are consistent." crlf))
```

Persona Ontology – Jess Consistency Rule 25:

If a persona A has interest A, then interest A is an interest of at least persona A.

```
(mapclass Persona)
(mapclass Interest)
(defrule hasInterest_isInterestOf
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasInterest $? ?in1 $?))
  (object (is-a Interest) (OBJECT ?in1) (isInterestOf $? ?in2 $? &:(eq (slot-get ?in2 id) ?i)))
  =>
(printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasInterest relationship (\"" (slot-get
?in1 id) " - " (slot-get ?in1 name) "\") are consistent." crlf))
```

Persona Ontology – Jess Consistency Rule 26:

A cognitive ability (of type "Other") cannot also be a physical ability (of type "Other")

(no duplicate cognitive type other).

```
(mapclass Cognitive)
(mapclass Physical)
(defrule no_identical_ability_other
  (object (is-a Cognitive) (is-a-name ?type1) (OBJECT ?obj) (cognitive_type_other ?t&:(neq ?t nil)))
  (object (is-a Physical) (is-a-name ?type2) (OBJECT ~?obj) (physical_type_other ?t&:(neq ?t nil)))
  =>
(printout t "Ability of type other \"" ?t "\" appears in both \"" ?type1 "\" and \"" ?type2 "\" and should be resolved." crlf))
```

Persona Ontology – Jess Consistency Rule 27:

If a persona A has an ability (cognitive or physical) A, then ability (cognitive or physical) A is an ability (cognitive or physical) of at least persona A.

```
(mapclass Persona)
(mapclass Cognitive)
(mapclass Physical)
(defrule hasAbility_isAbilityOf
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasAbility $? ?a1 $?))
  (object (is-a Cognitive | Physical) (is-a-name ?type1) (OBJECT ?a1) (isAbilityOf $? ?a2 $? &:(eq (slot-get ?a2 id) ?i)))
  =>
(if (= ?type1 Cognitive)
then (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasAbility relationship (\"" (slot-get ?a1 id) " - " (slot-get ?a1 cognitive_type) "\") are consistent." crlf)
else (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasAbility relationship (\"" (slot-get ?a1 id) " - " (slot-get ?a1 physical_type) "\") are consistent." crlf)))
```

Persona Ontology – Jess Consistency Rule 28:

If an environment A has an usability preference A, then usability preference A is an usability preference of at least environment A.

```
(mapclass Environment)
(mapclass UsabilityPref)
(defrule hasUsabilityPref_isUsabilityPrefOf
  (object (is-a Environment) (OBJECT ?obj) (id ?i) (hasUsabilityPref $? ?u1 $?))
  (object (is-a UsabilityPref) (OBJECT ?u1) (isUsabilityPrefOf $? ?u2 $? &:(eq (slot-get ?u2 id) ?i)))
  =>
(printout t "Environment (\"" ?i " - " (slot-get ?obj name) "\") and hasUsabilityPref relationship (\"" (slot-get ?u1 id) " - " (slot-get ?u1 usability_type) "\") are consistent." crlf))
```

Persona Ontology – Jess Consistency Rule 29:

A domain knowledge cannot also be a web or computer knowledge (no duplicate knowledge).

```
(mapclass Domain)
(mapclass Web)
(mapclass Computer)
(defrule no_identical_knowledge_category
  (object (is-a Domain) (is-a-name ?type) (OBJECT ?obj) (domain_knowledge ?k))
  (or (object (is-a Web) (is-a-name ~?type) (OBJECT ~?obj) (web_knowledge ?k))
  (object (is-a Computer) (is-a-name ~?type) (OBJECT ~?obj) (computer_knowledge ?k)))
  =>
(printout t "Duplicate knowledge \"" ?type " - " ?k "\" appears in more than one knowledge category and should be resolved." crlf))
```

Persona Ontology – Jess Consistency Rule 30:

A web knowledge cannot also be a domain or computer knowledge (no duplicate knowledge).

```
(mapclass Domain)
(mapclass Web)
(mapclass Computer)
(defrule no_identical_knowledge_category
  (object (is-a Web) (is-a-name ?type) (OBJECT ?obj) (web_knowledge ?k))
  (or (object (is-a Domain) (is-a-name ~?type) (OBJECT ~?obj) (domain_knowledge ?k))
  (object (is-a Computer) (is-a-name ~?type) (OBJECT ~?obj) (computer_knowledge ?k)))
  =>
(printout t "Duplicate knowledge \"" ?type " - " ?k "\" appears in more than one knowledge category and should be resolved." crlf))
```

Persona Ontology – Jess Consistency Rule 31:

A computer knowledge cannot also be a domain or web knowledge (no duplicate knowledge).

```
(mapclass Domain)
(mapclass Web)
(mapclass Computer)
(defrule no_identical_knowledge_category
  (object (is-a Computer) (is-a-name ?type) (OBJECT ?obj) (computer_knowledge ?k))
  (or (object (is-a Domain) (is-a-name ~?type) (OBJECT ~?obj) (domain_knowledge ?k))
  (object (is-a Web) (is-a-name ~?type) (OBJECT ~?obj) (web_knowledge ?k)))
  =>
(printout t "Duplicate knowledge \"" ?type " - " ?k "\" appears in more than one knowledge category and should be resolved." crlf))
```

Persona Ontology – Jess Consistency Rule 32:

If a persona A has a knowledge A, then knowledge A is a knowledge of at least persona A.

```
(mapclass Persona)

(mapclass Domain)

(mapclass Web)

(mapclass Computer)

(defrule hasKnowledge_isKnowledgeOf

  (object (is-a Persona) (OBJECT ?obj) (id ?i) (hasKnowledge $? ?k1 $?))

  (object (is-a Domain | Web | Computer) (is-a-name ?type1) (OBJECT ?k1) (isKnowledgeOf $? ?k2 $?
&:(eq (slot-get ?k2 id) ?i)))

  =>

(if (= ?type1 Domain)
then (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasKnowledge relationship (\""
(slot-get ?k1 id) " - " (slot-get ?k1 domain_knowledge) "\") are consistent." crlf)

else (if (= ?type1 Web)
then (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasKnowledge relationship (\""
(slot-get ?k1 id) " - " (slot-get ?k1 web_knowledge) "\") are consistent." crlf)

else (printout t "Persona (\"" ?i " - " (slot-get ?obj persona_title) "\") and hasKnowledge relationship (\""
(slot-get ?k1 id) " - " (slot-get ?k1 computer_knowledge) "\") are consistent." crlf))))
```

Persona Ontology - Jess Consistency Rule 33:

If a persona A plays a role A that participates in an environment A, then persona A must also engage in environment A.

```
(mapclass Persona)

(mapclass Role)

(mapclass Environment)

(defrule persona_role_environment

  (object (is-a Persona) (OBJECT ?obj) (id ?i) (personifies $? ?r1 $?) (engagesIn $? ?e1 $?))

  (object (is-a Role) (OBJECT ?r1) (participates $? ?p1 $?))

  (object (is-a Environment) (OBJECT ?p1) (engagedBy $? ?e2 $? &:(neq (slot-get ?e2 id) ?i) &:(neq (slot-
get ?e1 id) (slot-get ?p1 id))))

  =>

(printout t "Persona " ?i " (\"" (slot-get ?obj persona_title) "\"), role (\""(slot-get ?r1 name) "\"), and
environment (\""(slot-get ?e1 name) "\") are inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 34:

If a goal A is refined into ("AND-refinesInto") a goal B, then goal B must be refined

from ("AND-refinesFrom") at least goal A.

```
(mapclass Goal)
(defrule AND-refinesInto_AND-refinesFrom
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (AND-refinesInto $? ?r1 $?))
  (object (is-a Goal) (OBJECT ?r1) (AND-refinesFrom $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Goal " ?i " AND-refinement relationship (" (slot-get ?r1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 35:

If a goal A is refined ("OR-refinesInto") into a goal B, then goal B must be refined ("OR-refinesFrom") from at least goal A.

```
(mapclass Goal)
(defrule OR-refinesInto_OR-refinesFrom
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (OR-refinesInto $? ?r1 $?))
  (object (is-a Goal) (OBJECT ?r1) (OR-refinesFrom $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Goal " ?i " OR-refinement relationship (" (slot-get ?r1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 36:

If a goal A conflicts with a goal B, then goal B must also conflict with goal A.

```
(mapclass Goal)
(defrule conflictsWith
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (conflictsWith $? ?c1 $?))
  (object (is-a Goal) (OBJECT ?c1) (conflictsWith $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Goal " ?i " conflictsWith relationship (" (slot-get ?c1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 37:

If a goal A supports a goal B, then goal B must also be supported by goal A.

```
(mapclass Goal)
(defrule supports_supportedBy
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (supports $? ?s1 $?))
  (object (is-a Goal) (OBJECT ?s1) (supportedBy $? ?s2 $? &:(neq (slot-get ?s2 id) ?i)))
  =>
(printout t "Goal " ?i " supports relationship (" (slot-get ?s1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 38:

If a goal A has an obstacle A, then obstacle A is an obstacle of goal A.

```
(mapclass Goal)
(defrule hasObstacle_isObstacleOf
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (hasObstacle $? ?o1 $?))
  (object (is-a Goal) (OBJECT ?o1) (isObstacleOf $? ?o2 $? &:(neq (slot-get ?o2 id) ?i)))
  =>
(printout t "Goal " ?i " hasObstacle relationship (" (slot-get ?o1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 39:

If a goal A has a constraint A, then constraint A is a constraint of goal A.

```
(mapclass Goal)
(mapclass Constraint)
(defrule hasConstraint_isConstraintOf
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (hasConstraint $? ?c1 $?))
  (object (is-a Constraint) (OBJECT ?c1) (isConstraintOf $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Goal " ?i " hasConstraint relationship (" (slot-get ?c1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 40:

If a goal A has a stakeholder A, then stakeholder A is a stakeholder of goal A.

```
(mapclass Goal)
(mapclass Stakeholder)
(defrule hasStakeholder_isStakeholderOf
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (hasStakeholder $? ?s1 $?))
  (object (is-a Stakeholder) (OBJECT ?s1) (isStakeholderOf $? ?s2 $? &:(neq (slot-get ?s2 id) ?i)))
  =>
(printout t "Goal " ?i " hasStakeholder relationship (" (slot-get ?s1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 41:

If a goal A is a goal in a viewpoint A, then viewpoint A must have goal A.

```
(mapclass Goal)
(mapclass Viewpoint)
(defrule isGoalOf_hasGoal
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (isGoalOf $? ?g1 $?))
  (object (is-a Viewpoint) (OBJECT ?g1) (hasGoal $? ?g2 $? &:(neq (slot-get ?g2 id) ?i)))
  =>
(printout t "Goal " ?i " isGoalOf relationship (" (slot-get ?g1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 42:

If a goal A has a goal type A, then goal type A is the goal type of at least goal A.

```
(mapclass Goal)
(mapclass BusinessGoal)
(mapclass SystemGoal)
(mapclass PersonGoal)
(defrule hasGoalType_isGoalTypeOf
  (object (is-a Goal) (OBJECT ?obj) (id ?i) (hasGoalType $? ?gt1 $?))
  (object (is-a BusinessGoal | SystemGoal | PersonGoal) (is-a-name ?type1) (OBJECT ?gt1) (isGoalTypeOf $? ?gt2 $? &:(neq (slot-get ?gt2 id) ?i)))
  =>
(printout t "Goal " ?i " (\"" (slot-get ?obj name) "\") and hasGoalType relationship (\"" ?type1 " - " (slot-get ?gt1 id) "\") are inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 43:

If a goal is a goal in a viewpoint A that includes persona A, then persona A must also have goal A.

```
(mapclass Goal)
(mapclass Viewpoint)
(mapclass Persona)
(defrule goal_viewpoint_persona
  (object (is-a Persona) (OBJECT ?obj) (id ?i) (personifies $? ?r1 $?) (engagesIn $? ?e1 $?))
  (object (is-a Role) (OBJECT ?r1) (participates $? ?p1 $?))
  (object (is-a Environment) (OBJECT ?p1) (engagedBy $? ?e2 $? &:(eq (slot-get ?e2 id) ?i) &:(eq (slot-get ?e1 id) (slot-get ?p1 id))))
  =>
(printout t "Persona " ?i " (\"" (slot-get ?obj persona_title) "\") plays a role (\""(slot-get ?r1 name) "\") in an environment (\""(slot-get ?e1 name) "\") is consistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 44:

If a scenario A has an episode A, then episode A is an episode of scenario A.

```
(mapclass Scenario)
(defrule hasEpisode_isEpisodeOf
  (object (is-a Scenario) (OBJECT ?obj) (id ?i) (hasEpisode $? ?e1 $?))
  (object (is-a Scenario) (OBJECT ?s1) (isEpisodeOf $? ?e2 $? &:(neq (slot-get ?e2 id) ?i)))
  =>
(printout t "Scenario " ?i " hasEpisode relationship (" (slot-get ?e1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology - Jess Consistency Rule 45:

A normal scenario cannot also be an exception scenario (no duplicate scenario).

```
(mapclass NormalScenario)
(mapclass ExceptionScenario)
(defrule no_identical_scenario
  (object (is-a NormalScenario) (is-a-name ?type1) (OBJECT ?obj) (name ?n))
  (object (is-a ExceptionScenario) (is-a-name ?type2) (OBJECT ~?obj) (name ?n))
  =>
(printout t "Scenario \"" ?n "\" appears in both \"" ?type1 "\" and \"" ?type2 "\" and should be resolved." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 46:

If a scenario A has a task A, then task A is a task of scenario A.

```
(mapclass Scenario)
(mapclass Task)
(defrule hasTask_isTaskOf
  (object (is-a Scenario) (OBJECT ?obj) (id ?i) (hasTask $? ?t1 $?))
  (object (is-a Task) (OBJECT ?t1) (isTaskOf $? ?t2 $? &:(neq (slot-get ?t2 id) ?i)))
  =>
(printout t "Scenario " ?i " hasTask relationship (" (slot-get ?t1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 47:

If a scenario A has a constraint A, then constraint A is a constraint of scenario A.

```
(mapclass Scenario)
(mapclass Constraint)
(defrule hasConstraint_isConstraintOf
  (object (is-a Scenario) (OBJECT ?obj) (id ?i) (hasConstraint $? ?c1 $?))
  (object (is-a Constraint) (OBJECT ?c1) (isConstraintOf $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Scenario " ?i " hasConstraint relationship (" (slot-get ?c1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 48:

If a scenario A is a scenario in a viewpoint A, then viewpoint A must have scenario A.

```
(mapclass Scenario)
(mapclass Viewpoint)
(defrule isScenarioOf_hasScenario
  (object (is-a Scenario) (OBJECT ?obj) (id ?i) (isScenarioOf $? ?s1 $?))
  (object (is-a Viewpoint) (OBJECT ?s1) (hasScenario $? ?s2 $? &:(neq (slot-get ?s2 id) ?i)))
  =>
(printout t "Scenario " ?i " isScenarioOf relationship (" (slot-get ?s1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 49:

If a task A is refined into ("AND-refinesInto") a task B, then task B must be refined from ("AND-refinesFrom") at least task A.

```
(mapclass Task)
(defrule AND-refinesInto_AND-refinesFrom
  (object (is-a Task) (OBJECT ?obj) (id ?i) (AND-refinesInto $? ?r1 $?))
  (object (is-a Task) (OBJECT ?r1) (AND-refinesFrom $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Task " ?i " AND-refinement relationship (" (slot-get ?r1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 50:

If a task A is refined ("OR-refinesInto") into a task B, then task B must be refined ("OR-refinesFrom") from at least task A.

```
(mapclass Task)
(defrule OR-refinesInto_OR-refinesFrom
  (object (is-a Task) (OBJECT ?obj) (id ?i) (OR-refinesInto $? ?r1 $?))
  (object (is-a Task) (OBJECT ?r1) (OR-refinesFrom $? ?r2 $? &:(neq (slot-get ?r2 id) ?i)))
  =>
(printout t "Task " ?i " OR-refinement relationship (" (slot-get ?r1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 51:

If a task A has a tool A, then tool A is a tool of at least task A.

```
(mapclass Task)
(mapclass Tool)
(defrule hasTool_isToolOf
  (object (is-a Task) (OBJECT ?obj) (id ?i) (hasTool $? ?t1 $?))
  (object (is-a Tool) (OBJECT ?t1) (isToolOf $? ?t2 $? &:(neq (slot-get ?t2 id) ?i)))
  =>
(printout t "Task " ?i " hasTool relationship (" (slot-get ?t1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 52:

If a task A has an action A, then action A is an action of at least task A.

```
(mapclass Task)
(mapclass Action)
(defrule hasAction_isActionOf
  (object (is-a Task) (OBJECT ?obj) (id ?i) (hasAction $? ?a1 $?))
  (object (is-a Action) (OBJECT ?a1) (isActionOf $? ?a2 $? &:(neq (slot-get ?a2 id) ?i)))
  =>
(printout t "Task " ?i " hasAction relationship (" (slot-get ?a1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 53:

If a task A has a constraint A, then constraint A is a constraint of at least task A.

```
(mapclass Task)
(mapclass Constraint)
(defrule hasConstraint_isConstraintOf
  (object (is-a Task) (OBJECT ?obj) (id ?i) (hasConstraint $? ?c1 $?))
  (object (is-a Constraint) (OBJECT ?c1) (isConstraintOf $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Scenario " ?i " hasConstraint relationship (" (slot-get ?c1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 54:

If a task A is a task in a viewpoint A, then viewpoint A must have at least task A.

```
(mapclass Task)
(mapclass Viewpoint)
(defrule isTaskOf_hasTask
  (object (is-a Task) (OBJECT ?obj) (id ?i) (isTaskOf $? ?t1 $?))
  (object (is-a Viewpoint) (OBJECT ?t1) (hasTask $? ?t2 $? &:(neq (slot-get ?t2 id) ?i)))
  =>
(printout t "Task " ?i " isTaskOf relationship (" (slot-get ?t1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 55:

If a task A has a pre-requisite A, then pe-requisite A is a pre-requisite of at least task A.

```
(mapclass Task)
(mapclass Prerequisite)
(defrule hasPrerequisite_isPrerequisiteOf
  (object (is-a Task) (OBJECT ?obj) (id ?i) (hasPrerequisite $? ?p1 $?))
  (object (is-a Prerequisite) (OBJECT ?p1) (isPrerequisiteOf $? ?p2 $? &:(neq (slot-get ?p2 id) ?i)))
  =>
(printout t "Task " ?i " hasPrerequisite relationship (" (slot-get ?p1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 56:

If a task A has a post-requisite A, then post-requisite A is a post-requisite of at least task A.

```
(mapclass Task)
(mapclass Postrequisite)
(defrule hasPostrequisite_isPostrequisiteOf
  (object (is-a Task) (OBJECT ?obj) (id ?i) (hasPostrequisite $? ?p1 $?))
  (object (is-a Postrequisite) (OBJECT ?p1) (isPostrequisiteOf $? ?p2 $? &:(neq (slot-get ?p2 id) ?i)))
  =>
(printout t "Task " ?i " hasPostrequisite relationship (" (slot-get ?p1 id) ") is inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 57:

If a viewpoint A has a constraint A, then constraint A is a constraint of at least viewpoint A.

```
(mapclass Viewpoint)
(mapclass Constraint)
(defrule hasConstraint_isConstraintOf
  (object (is-a Viewpoint) (OBJECT ?obj) (id ?i) (hasConstraint $? ?c1 $?))
  (object (is-a Constraint) (OBJECT ?c1) (isConstraintOf $? ?c2 $? &:(neq (slot-get ?c2 id) ?i)))
  =>
(printout t "Viewpoint " ?i " and hasConstraint relationship (" (slot-get ?c1 id) ") are inconsistent." crlf))
```

Behavioral-GST Ontology - Jess Consistency Rule 58:

An instance of Actor sub-class cannot also be an instance of another Actor sub-class (no duplicate actor).

```
(mapclass Organization)
(mapclass Device)
(mapclass System)
(mapclass Person)
(defrule no_identical_actor
  (object (is-a Organization | Device | System | Person) (is-a-name ?type) (OBJECT ?obj) (name ?n))
  (object (is-a Organization | Device | System | Person) (is-a-name ~?type)(OBJECT ~?obj) (name ?n))
  =>
(printout t "Actor \"" ?n "\" appears in \"" ?type "\" and should be resolved." crlf))
```

Behavioral-GST Ontology - Jess Consistency Rule 59:

If an actor A plays a role A, then role A is played by at least actor A.

```
(mapclass Role)
(mapclass Organization)
(mapclass Device)
(mapclass Person)
(mapclass System)
(defrule plays_playedBy
  (object (is-a Role) (OBJECT ?obj) (id ?i) (playedBy $? ?p1 $?))
  (object (is-a Organization | Device | Person | System) (is-a-name ?type1) (OBJECT ?p1) (plays $? ?p2 $? &:(neq (slot-get ?p2 id) ?i)))
  =>
(printout t "Actor (\"" ?type1 " - " (slot-get ?p1 id) "\") and plays relationship (" ?i ") are inconsistent." crlf))
```

Behavioral-GST Ontology – Jess Consistency Rule 60:

If a viewpoint A has a persona A, a role A, and an environment A, then viewpoint B cannot also has the same persona A, role A, and environment A (no two viewpoints have the same persona, role, and environment; a view point is uniquely identified by <persona, environment, role>).

```
(mapclass Viewpoint)
(defrule no_identical_viewpoint
  (object (is-a Viewpoint) (OBJECT ?obj1) (id ?i) (hasPersona ?p1) (hasEnvironment ?e1) (hasRole ?r1))
  (object (is-a Viewpoint) (OBJECT ?obj2) (id ~?i)
  (hasPersona ?p2 &:(eq (slot-get ?p2 id) (slot-get ?p1 id)))
  (hasEnvironment ?e2 &:(eq (slot-get ?e2 id) (slot-get ?e1 id)))
  (hasRole ?r2 &:(eq (slot-get ?r2 id) (slot-get ?r1 id))))
  =>
(printout t "Duplicate viewpoint (\"" (slot-get ?obj1 id) "\" " (slot-get ?obj2 id) "\") detected and should be resolved. A viewpoint is uniquely identified by <persona, environment, role>." crlf))
```

Requirements Ontology – Jess Completeness Rule 1:

Every requirement must specify attribute id, statement, source, status, priority, validation, and verification.

```
(mapclass Requirement)
(defrule specify_required_requirement_attribute
(or (object (is-a Requirement) (OBJECT ?obj) (id nil))
(object (is-a Requirement) (OBJECT ?obj) (statement nil))
(object (is-a Requirement) (OBJECT ?obj) (source $?s&:(= (length$ ?s) 0)))
(object (is-a Requirement) (OBJECT ?obj) (status nil))
(object (is-a Requirement) (OBJECT ?obj) (priority nil))
(object (is-a Requirement) (OBJECT ?obj) (validation nil))
(object (is-a Requirement) (OBJECT ?obj) (verification nil)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, statement, source, status, priority, validation, and verification." crlf))
```

Requirements Ontology – Jess Completeness Rule 2:

Every requirement must specify whether it is a functional or a non-functional requirement.

```
(mapclass Requirement)
(defrule specify_functional_nonfunctional
(object (is-a Requirement) (OBJECT ?obj) (isReqOf $?req&:(= (length$ ?req) 0)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. You did not specify whether it is a functional or a non-functional requirement." crlf))
```

Requirements Ontology – Jess Completeness Rule 3:

Every functional and non-functional requirement category must specify requirement category attribute.

```
(mapclass Functional)
(mapclass NonFunctional)
(defrule specify_category
(object (is-a Functional | NonFunctional) (OBJECT ?obj) (category nil))
=>
(printout t "Requirement (" (slot-get ?obj id) ") is incomplete. You did not specify the category of the requirement." crlf))
```

Requirements Ontology – Jess Completeness Rule 4:

Every requirement must specify at least one relationship with another requirement, i.e. conflictsWith, constrains, requires, refinesInto.

```
(mapclass Requirement)
(defrule specify_at_least_one_relationship_with_another_requiremnt
(object (is-a Requirement) (OBJECT ?obj) (refinesInto $?ri&:(= (length$ ?ri) 0)))
(object (is-a Requirement) (OBJECT ?obj) (requires $?rq&:(= (length$ ?rq) 0)))
(object (is-a Requirement) (OBJECT ?obj) (constrains $?ct&:(= (length$ ?ct) 0)))
(object (is-a Requirement) (OBJECT ?obj) (conflictsWith $?cf&:(= (length$ ?cf) 0)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one relationship (refinesInto, requires, constrains, conflicts) with another requirement." crlf))
```

Requirements Ontology – Jess Completeness Rule 5:

Every requirement must specify at least one "isPartOf" relationship with system requirements specification.

```
(mapclass Requirement)
(defrule specify_at_least_one_isPartOf_relationship_with_SRS
(object (is-a Requirement) (OBJECT ?obj) (isPartOf $?ipo&:(= (length$ ?ipo) 0)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"isPartOf\" relationship with System Requirements Specification (SRS)." crlf))
```

Requirements Ontology – Jess Completeness Rule 6:

If a requirement has status "Implemented", then that requirement must specify at least

one meets relationship with a goal.

```
(mapclass Requirement)
(defrule specify_at_least_one_meets_relationship_with_goal
(object (is-a Requirement) (OBJECT ?obj) (status Implemented) (meets $?m&:(= (length$ ?op) 0)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"meets\" relationship with a goal." crlf))
```

Requirements Ontology – Jess Completeness Rule 7:

If a requirement has status "Implemented", then that requirement must specify at least one operationalizedBy relationship with a scenario.

```
(mapclass Requirement)
(defrule specify_at_least_one_meets_relationship_with_scenario
(object (is-a Requirement) (OBJECT ?obj) (status Implemented) (operationalizedBy $?op&:(= (length$ ?op) 0)))
=>
(printout t "Requirement (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"operationalizedBy\" relationship with a scenario." crlf))
```

Requirements Ontology – Jess Completeness Rule 8:

Every requirement must specify at least one isRequirementsOf relationship with one or more viewpoint.

```
(mapclass Requirement)
(defrule specify_at_least_one_isRequirementsOf_relationship_with_Viewpoint
(object (is-a Requirement) (OBJECT ?obj) (isRequirementsOf $?iro&:(= (length$ ?iro) 0)))
=>
(printout t "Requirement (" (slot-get ?obj id) ") is incomplete. You did not specify at least one \"isRequirementsOf\" relationship with Viewpoint." crlf))
```

Persona Ontology - Jess Completeness Rule 9:

Every persona must specify attribute id, persona_title, persona_type, age, and gender.

```
(mapclass Persona)
(defrule specify_required_persona_attribute
(or (object (is-a Persona) (OBJECT ?obj) (id nil))
(object (is-a Persona) (OBJECT ?obj) (persona_title nil))
(object (is-a Persona) (OBJECT ?obj) (persona_type nil))
(object (is-a Persona) (OBJECT ?obj) (age nil))
(object (is-a Persona) (OBJECT ?obj) (gender nil)))
=>
(printout t "Persona (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, persona-title, persona_type, age, and gender." crlf))
```

Persona Ontology - Jess Completeness Rule 10:

Every name must specify attribute id, first_name, middle_name, and last_name.

```
(mapclass Persona)
(defrule specify_required_name_attribute
(or (object (is-a Name) (OBJECT ?obj) (id nil))
(object (is-a Name) (OBJECT ?obj) (first_name nil))
(object (is-a Name) (OBJECT ?obj) (middle_name nil))
(object (is-a Name) (OBJECT ?obj) (last_name nil)))
=>
(printout t "Name (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, first_name, middle_name, and last_name." crlf))
```

Persona Ontology - Jess Completeness Rule 11:

Every education must specify attribute id and highest education level.

```
(mapclass Education)
(defrule specify_required_education_attribute
(or (object (is-a Education) (OBJECT ?obj) (id nil))
(object (is-a Education) (OBJECT ?obj) (highest_education_level nil)))
=>
(printout t "Education (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id and highest_education_level." crlf))
```

Persona Ontology - Jess Completeness Rule 12:

Every occupation must specify attribute id, position, and occupation group.

```
(mapclass Occupation)
(defrule specify_required_occupation_attribute
(or (object (is-a Occupation) (OBJECT ?obj) (id nil))
(object (is-a Occupation) (OBJECT ?obj) (position nil))
(object (is-a Occupation) (OBJECT ?obj) (occupation_group nil)))
=>
(printout t "Occupation (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, position, and occupation_group." crlf))
```

Persona Ontology - Jess Completeness Rule 13:

Every concern must specify attribute id, name, and rating.

```
(mapclass Concern)
(defrule specify_required_concern_attribute
(or (object (is-a Concern) (OBJECT ?obj) (id nil))
(object (is-a Concern) (OBJECT ?obj) (name nil))
(object (is-a Concern) (OBJECT ?obj) (rating nil)))
=>
(printout t "Concern (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, and rating. crlf))
```

Persona Ontology - Jess Completeness Rule 14:

Every role must specify attribute id, name, and role type.

```
(mapclass Role)
(defrule specify_required_role_attribute
(or (object (is-a Role) (OBJECT ?obj) (id nil))
(object (is-a Role) (OBJECT ?obj) (name nil))
(object (is-a Role) (OBJECT ?obj) (role_type nil)))
=>
(printout t "Role (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, and role_type. crlf))
```

Persona Ontology - Jess Completeness Rule 15:

Every environment must specify attribute id, name, location, time of day, and persona attitude.

```
(mapclass Environment)
(defrule specify_required_environment_attribute
(or (object (is-a Environment) (OBJECT ?obj) (id nil))
(object (is-a Environment) (OBJECT ?obj) (name nil))
(object (is-a Environment) (OBJECT ?obj) (location nil))
(object (is-a Environment) (OBJECT ?obj) (time_of_day nil))
(object (is-a Environment) (OBJECT ?obj) (persona_attitude $?pa&:(=(length$ ?pa) 0))))
=>
(printout t "Environment (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, location, time_of_day, and persona_attitude." crlf))
```

Persona Ontology - Jess Completeness Rule 16:

Every usability preference must specify attribute id, usability type, and rating.

```
(mapclass UsabilityPref)
(defrule specify_required_usability_preference_attribute
(or (object (is-a UsabilityPref) (OBJECT ?obj) (id nil))
(object (is-a UsabilityPref) (OBJECT ?obj) (usability_type nil))
(object (is-a UsabilityPref) (OBJECT ?obj) (rating nil)))
=>
(printout t "Usability preference (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, usability_type, and rating" crlf))
```

Persona Ontology - Jess Completeness Rule 17:

Every interest must specify attribute id and name.

```
(mapclass Interest)
(defrule specify_required_interest_attribute
(or (object (is-a Interest) (OBJECT ?obj) (id nil))
(object (is-a Interest) (OBJECT ?obj) (name nil)))
=>
(printout t "Interest (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id and name." crlf))
```

Persona Ontology - Jess Completeness Rule 18:

Every interest category must specify attribute id and category.

```
(mapclass InterestCategory)
(defrule specify_required_interest_category_attribute
(or (object (is-a InterestCategory) (OBJECT ?obj) (id nil))
(object (is-a InterestCategory) (OBJECT ?obj) (category nil)))
=>
(printout t "Interest category (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id and category." crlf))
```

Persona Ontology - Jess Completeness Rule 19:

Every ability must specify attribute id, cognitive type or physical type, and ability level. If cognitive type or physical type is "Other", then it must specify the text field for "Other".

```
(mapclass Cognitive)

(mapclass Physical)

(defrule specify_required_ability_attribute

(or (object (is-a Cognitive | Physical) (is-a-name ?type1) (OBJECT ?obj) (id nil))

(object (is-a Cognitive | Physical) (is-a-name ?type1) (OBJECT ?obj) (ability_level nil))

(object (is-a Cognitive | Physical) (is-a-name ?type1) (OBJECT ?obj) (cognitive_type nil | physical_type nil))

(object (is-a Cognitive | Physical) (is-a-name ?type1) (OBJECT ?obj) (cognitive_type Other) (cognitive_type_other nil))

(object (is-a Cognitive | Physical) (is-a-name ?type1) (OBJECT ?obj) (physical_type Other) (physical_type_other nil)))

=>

(printout t  ?type1 " ability (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, cognitive_type or physical_type, and ability_level." crlf))
```

Persona Ontology - Jess Completeness Rule 20:

Every language proficiency must specify attribute id, writing language or speaking language or listening language, and proficiency level.

```
(mapclass Writing)

(mapclass Speaking)

(mapclass Listening)

(defrule specify_required_language_proficiency_attribute

(or (object (is-a Writing | Speaking | Listening) (is-a-name ?type1) (OBJECT ?obj) (id nil))

(object (is-a Writing | Speaking | Listening) (is-a-name ?type1) (OBJECT ?obj) (proficiency_level nil))

(object (is-a Writing | Speaking | Listening) (is-a-name ?type1) (OBJECT ?obj) (writing_language nil | speaking_language nil | listening_language nil)))

=>

(printout t  ?type1 " proficiency (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, writing_language or speaking_language or listening_language, and proficiency_level." crlf))
```

Persona Ontology - Jess Completeness Rule 21:

Every knowledge must specify attribute id, domain knowledge or web knowledge or computer knowledge, and knowledge level.

```
(mapclass Domain)
(mapclass Web)
(mapclass Computer)
(defrule specify_required_knowledge_attribute
(or (object (is-a Domain | Web | Computer) (is-a-name ?type1) (OBJECT ?obj) (id nil))
(object (is-a Domain | Web | Computer) (is-a-name ?type1) (OBJECT ?obj) (knowledge_level nil))
(object (is-a Domain | Web | Computer) (is-a-name ?type1) (OBJECT ?obj) (domain_knowledge nil | web_knowledge nil | computer_knowledge nil)))
=>
(printout t  ?type1 " knowledge (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, domain_knowledge or web_knowledge or computer_knowledge, and knowledge_level." crlf))
```

Persona Ontology - Jess Completeness Rule 22:

Every persona must specify at most one hasPersonaName relationship with a name.

```
(mapclass Persona)
(defrule specify_hasPersonaName_relationship_with_name
(object (is-a Persona) (OBJECT ?obj) (hasPersonaName nil))
=>
(printout t "Persona ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at most one \"hasPersonaName\" relationship with a name." crlf))
```

Persona Ontology - Jess Completeness Rule 23:

Every persona must specify at most one hasHighestEducation relationship with an education.

```
(mapclass Persona)
(defrule specify_hasHighestEducation_relationship_with_education
(object (is-a Persona) (OBJECT ?obj) (hasHighestEducation nil))
=>
(printout t "Persona ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at most one \"hasHighestEducation\" relationship with an education." crlf))
```

Persona Ontology - Jess Completeness Rule 24:

Every persona must specify at least one personifies relationship with a role.

```
(mapclass Persona)
(defrule specify_personifies_relationship_with_role
(object (is-a Persona) (OBJECT ?obj) (personifies $?p&:(= (length$ ?p) 0)))
=>
(printout t "Persona ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"personifies\" relationship with a role." crlf))
```

Persona Ontology - Jess Completeness Rule 25:

Every persona must specify at least one engagesIn relationship with an environment.

```
(mapclass Persona)
(defrule specify_engagesIn_relationship_with_environment
(object (is-a Persona) (OBJECT ?obj) (engagesIn $?ei&:(= (length$ ?ei) 0)))
=>
(printout t "Persona ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"engagesIn\" relationship with an environment." crlf))
```

Persona Ontology - Jess Completeness Rule 26:

Every interest must specify at most one isInterestPartOf relationship with an interest category.

```
(mapclass Interest)
(defrule specify_isInterestPartOf_relationship_with_interest_category
(object (is-a Interest) (OBJECT ?obj) (isInterestPartOf nil))
=>
(printout t "Interest ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at most one \"isInterestPartOf\" relationship with an interest category." crlf))
```

Persona Ontology - Jess Completeness Rule 27:

Every persona must specify at least one isPersonaOf relationship with a viewpoint.

```
(mapclass Persona)
(defrule specify_at_least_one_isPersonaOf_relationship_with_viewpoint
(object (is-a Persona) (OBJECT ?obj) (isPersonaOf $?ipot&:(= (length$ ?ipo) 0)))
=>
(printout t "Persona ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"isPersonaOf\" relationship with a viewpoint." crlf))
```

Persona Ontology - Jess Completeness Rule 28:

Every environment must specify at least one isEnvironmentOf relationship with a viewpoint.

```
(mapclass Environment)
(defrule specify_at_least_one_isEnvironmentOf_relationship_with_viewpoint
(object (is-a Environment) (OBJECT ?obj) (isEnvironmentOf $?ieot&:(= (length$ ?ieo) 0)))
=>
(printout t "Environment ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"isEnvironmentOf\" relationship with a viewpoint." crlf))
```

Persona Ontology - Jess Completeness Rule 29:

Every role must specify at least one isRoleOf relationship with a viewpoint.

```
(mapclass Role)
(defrule specify_at_least_one_isRoleOf_relationship_with_viewpoint
(object (is-a Role) (OBJECT ?obj) (isRoleOf $?irot&:(= (length$ ?iro) 0)))
=>
(printout t "Role ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"isRoleOf\" relationship with a viewpoint." crlf))
```

Persona Ontology - Jess Completeness Rule 30:

Every concern must specify at least one addressedBy relationship with a viewpoint.

```
(mapclass Concern)
(defrule specify_at_least_one_addressedBy_relationship_with_viewpoint
(object (is-a Concern) (OBJECT ?obj) (addressedBy $?ab&:(= (length$ ?ab) 0)))
=>
(printout t "Concern ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"addressedBy\" relationship with a viewpoint." crlf))
```

Persona Ontology - Jess Completeness Rule 31:

Every name must specify at most one isPersonaNameOf relationship with a persona.

```
(mapclass Name)
(defrule specify_isPersonaNameOf_relationship_with_persona
(object (is-a Name) (OBJECT ?obj) (isPersonaNameOf nil))
=>
(printout t "Name ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at most one \"isPersonaNameOf\" relationship with a persona." crlf))
```

Persona Ontology - Jess Completeness Rule 32:

Every education must specify at most one isHighestEducationOf relationship with a persona.

```
(mapclass Education)
(defrule specify_isHighestEducationOf_relationship_with_persona
(object (is-a Education) (OBJECT ?obj) (isHighestEducationOf $?iheo&:(= (length$ ?iheo) 0)))
=>
(printout t "Name ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at most one \"isPersonaNameOf\" relationship with a persona." crlf))
```

Persona Ontology - Jess Completeness Rule 33:

Every role must specify at least one personifiedBy relationship with a persona.

```
(mapclass Role)
(defrule specify_personifiedBy_relationship_with_persona
(object (is-a Role) (OBJECT ?obj) (personifiedBy $?pb&:(= (length$ ?pb) 0)))
=>
(printout t "Role ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"personifiedBy\" relationship with a persona." crlf))
```

Persona Ontology - Jess Completeness Rule 34:

Every environment must specify at least one engagedBy relationship with a persona.

```
(mapclass Environment)
(defrule specify_engagedBy_relationship_with_persona
(object (is-a Environment) (OBJECT ?obj) (engagedBy $?eb&:(= (length$ ?eb) 0)))
=>
(printout t "Environment ("(slot-get ?obj :NAME) " - " (slot-get ?obj id) ") is incomplete. You did not specify at least one \"engagedBy\" relationship with a persona." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 35:

Every goal must specify attribute id, name, priority, and status.

```
(mapclass Goal)
(defrule specify_required_goal_attribute
(or (object (is-a Goal) (OBJECT ?obj) (id nil))
(object (is-a Goal) (OBJECT ?obj) (name nil))
(object (is-a Goal) (OBJECT ?obj) (priority nil))
(object (is-a Goal) (OBJECT ?obj) (status nil)))
=>
(printout t "Goal (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, priority, and status." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 36:

Every scenario must specify attribute id, name, status, preconditions, and postconditions.

```
(mapclass Scenario)
(defrule specify_required_scenario_attribute
(or (object (is-a Scenario) (OBJECT ?obj) (id nil))
(object (is-a Scenario) (OBJECT ?obj) (name nil))
(object (is-a Scenario) (OBJECT ?obj) (status nil))
(object (is-a Scenario) (OBJECT ?obj) (preconditions $?prc&:(=(length$ ?prc) 0)))
(object (is-a Scenario) (OBJECT ?obj) (postconditions $?poc&:(=(length$ ?poc) 0))))
=>
(printout t "Scenario (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, status, preconditions, and postconditions." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 37:

Every tool must specify attribute id and name.

```
(mapclass Tool)
(defrule specify_required_tool_attribute
(or (object (is-a Tool) (OBJECT ?obj) (id nil))
(object (is-a Tool) (OBJECT ?obj) (name nil)))
=>
(printout t "Tool (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id and name." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 38:

Every task must specify attribute id, name, and priority.

```
(mapclass Task)
(defrule specify_required_task_attribute
(or (object (is-a Task) (OBJECT ?obj) (id nil))
(object (is-a Task) (OBJECT ?obj) (name nil))
(object (is-a Task) (OBJECT ?obj) (priority nil)))
=>
(printout t "Task (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, and priority." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 39:

Every viewpoint must specify attribute id, name, modeling techniques, and models.

```
(mapclass Viewpoint)
(defrule specify_required_viewpoint_attribute
(or (object (is-a Viewpoint) (OBJECT ?obj) (id nil))
(object (is-a Viewpoint) (OBJECT ?obj) (name nil))
(object (is-a Viewpoint) (OBJECT ?obj) (modeling_techniques $?mt&:(=(length$ ?mt) 0)))
(object (is-a Viewpoint) (OBJECT ?obj) (models $?m&:(=(length$ ?m) 0))))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, modeling techniques, and models." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 40:

Every stakeholder must specify attribute id, name, and priority.

```
(mapclass Stakeholder)
(defrule specify_required_stakeholder_attribute
(or (object (is-a Stakeholder) (OBJECT ?obj) (id nil))
(object (is-a Stakeholder) (OBJECT ?obj) (name nil))
(object (is-a Stakeholder) (OBJECT ?obj) (priority nil)))
=>
(printout t "Stakeholder (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, and priority." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 41:

Every actor (organization, device, system, person) must specify attribute id and name.

```
(mapclass Organization)
(mapclass Device)
(mapclass System)
(mapclass Person)
(defrule specify_required_actor_attribute
(or (object (is-a Organization | Device | System | Person) (is-a-name ?type1) (OBJECT ?obj) (id nil))
(object (is-a Organization | Device | System | Person) (is-a-name ?type1) (OBJECT ?obj) (name nil)))
=>
(printout t  ?type1 " actor (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id and name." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 42:

Every constraint must specify attribute id, name, critical level, and constraint type. If constraint type is "Other", then it must specify the text field for "Other".

```
(mapclass Constraint)
(defrule specify_required_constraint_attribute
(or (object (is-a Constraint) (OBJECT ?obj) (id nil))
(object (is-a Constraint) (OBJECT ?obj) (name nil))
(object (is-a Constraint) (OBJECT ?obj) (critical_level nil))
(object (is-a Constraint) (OBJECT ?obj) (constraint_type $?ct&:(=(length$ ?ct) 0)))
(object (is-a Constraint) (OBJECT ?obj) (constraint_type Other) (constraint_type_other nil)))
=>
(printout t "Constraint (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, critical_level, constraint_type, and constraint_type_other (if constraint_type is \"Other\")." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 43:

Every obstacle must specify attribute id, name, critical level, and obstacle type. If obstacle type is "Other", then it must specify the text field for "Other".

```
(mapclass Obstacle)
(defrule specify_required_obstacle_attribute
(or (object (is-a Obstacle) (OBJECT ?obj) (id nil))
(object (is-a Obstacle) (OBJECT ?obj) (name nil))
(object (is-a Obstacle) (OBJECT ?obj) (critical_level nil))
(object (is-a Obstacle) (OBJECT ?obj) (obstacle_type $?ot&:(=(length$ ?ct) 0)))
(object (is-a Obstacle) (OBJECT ?obj) (obstacle_type Other) (obstacle_type_other nil)))
=>
(printout t "Obstacle (" (slot-get ?obj :NAME) ") is incomplete. One or more required attributes are not specified. Required attributes are id, name, critical_level, obstacle_type, and obstacle_type_other (if obstacle_type is \"Other\")." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 44:

Every viewpoint must specify at least one hasGoal relationship with a goal.

```
(mapclass Viewpoint)
(defrule specify_at_least_one_hasGoal_relationship_with_goal
(object (is-a Viewpoint) (OBJECT ?obj) (hasGoal $?hg&:(= (length$ ?hg) 0)))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasGoal\" relationship with a goal." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 45:

Every viewpoint must specify at least one hasTask relationship with a task.

```
(mapclass Viewpoint)
(defrule specify_at_least_one_hasTask_relationship_with_task
(object (is-a Viewpoint) (OBJECT ?obj) (hasTask $?ht&:(= (length$ ?ht) 0)))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasTask\" relationship with a task." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 46:

Every viewpoint must specify at least one hasScenario relationship with a scenario.

```
(mapclass Viewpoint)
(defrule specify_at_least_one_hasScenario_relationship_with_scenario
(object (is-a Viewpoint) (OBJECT ?obj) (hasScenario $?hs&:(= (length$ ?hs) 0)))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasScenario\" relationship with a scenario." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 47:

Every viewpoint must specify at least one hasRequirements relationship with a requirement.

```
(mapclass Viewpoint)
(defrule specify_at_least_one_hasRequirements_relationship_with_requirement
(object (is-a Viewpoint) (OBJECT ?obj) (hasRequirements $?hr&:(= (length$ ?hr) 0)))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasRequirements\" relationship with a requirement." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 48:

Every viewpoint must specify at least one hasStakeholder relationship with a stakeholder.

```
(mapclass Viewpoint)
(defrule specify_at_least_one_hasStakholder_relationship_with_stakeholder
(object (is-a Viewpoint) (OBJECT ?obj) (hasStakeholder $?hst&:(= (length$ ?hst) 0)))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasStakeholder\" relationship with a stakeholder." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 49:

Every viewpoint must specify at most one hasPersona relationship with a persona.

```
(mapclass Viewpoint)
(defrule specify_at_most_one_hasPersona_relationship_with_persona
(object (is-a Viewpoint) (OBJECT ?obj) (hasPersona nil))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at most one \"hasPersona\" relationship with a persona." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 50:

Every viewpoint must specify at most one hasEnvironment relationship with a persona.

```
(mapclass Viewpoint)
(defrule specify_at_most_one_hasEnvironment_relationship_with_persona
(object (is-a Viewpoint) (OBJECT ?obj) (hasEnvironment nil))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at most one \"hasEnvironment\" relationship with a persona." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 51:

Every viewpoint must specify at most one hasRole relationship with a persona.

```
(mapclass Viewpoint)
(defrule specify_at_most_one_hasRole_relationship_with_persona
(object (is-a Viewpoint) (OBJECT ?obj) (hasRole nil))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at most one \"hasRole\" relationship with a persona." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 52:

Every viewpoint must specify at least one addresses relationship with a concern.

```
(mapclass Viewpoint)
(defrule specify_at_least_one_addresses_relationship_with_requirement
(object (is-a Viewpoint) (OBJECT ?obj) (addresses $?a&:(= (length$ ?a) 0)))
=>
(printout t "Viewpoint (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"addresses\" relationship with a concern." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 53:

Every goal must specify at least one relationship with another goal, i.e. conflictsWith, supports, requires, AND-refinesInto, OR-refinesInto.

```
(mapclass Goal)
(defrule specify_at_least_one_relationship_with_another_goal
(object (is-a Goal) (OBJECT ?obj) (AND-refinesInto $?ari&:(= (length$ ?ari) 0)))
(object (is-a Goal) (OBJECT ?obj) (OR-refinesInto $?ori&:(= (length$ ?ori) 0)))
(object (is-a Goal) (OBJECT ?obj) (supports $?s&:(= (length$ ?s) 0)))
 (object (is-a Goal) (OBJECT ?obj) (conflictsWith $?cf&:(= (length$ ?cf) 0)))
=>
(printout t "Goal (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one relationship (AND-refinesInto, OR-refinesInto, conflictsWith, supports) with another goal." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 54:

Every goal must specify at least one hasStakeholder relationship with a stakeholder.

```
(mapclass Goal)
(defrule specify_at_least_one_hasStakeholder_relationship_with_stakeholder
(object (is-a Goal) (OBJECT ?obj) (hasStakeholder $?hs&:(= (length$ ?hs) 0)))
=>
(printout t "Goal (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasStakeholder\" relationship with a stakeholder." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 55:

Every goal must specify at least one isGoalOf relationship with a viewpoint.

```
(mapclass Goal)
(defrule specify_at_least_one_isGoalOf_relationship_with_viewpoint
(object (is-a Goal) (OBJECT ?obj) (isGoalOf $?igo&:(= (length$ ?igo) 0)))
=>
(printout t "Goal (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one  \"isGoalOf\" relationship with a viewpoint." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 56:

Every goal must specify at least one hasGoalType relationship with a goal type.

```
(mapclass Goal)
(defrule specify_at_least_one_hasGoalType_relationship_with_goal_type
(object (is-a Goal) (OBJECT ?obj) (hasGoalType $?hgt&:(= (length$ ?hgt) 0)))
=>
(printout t "Goal (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasGoalType\" relationship with a goal type." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 57:

Every goal type must specify at least one isGoalTypeOf relationship with a goal.

```
(mapclass BusinessGoal)
(mapclass SystemGoal)
(mapclass PersonGoal)
(defrule specify_at_least_one_isGoalTypeOf_relationship_with_goal
(object (is-a BusinessGoal | SystemGoal | PersonGoal) (OBJECT ?obj) (isGoalTypeOf $?igto&:(= (length$ ?igto) 0)))
=>
(printout t "Goal type (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at most one \"isGoalTypeOf\" relationship with a goal." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 58:

If a goal has status "Met" or "Partially-Met", then that goal must specify at least one metBy relationship with a requirement.

```
(mapclass Goal)
(defrule specify_at_least_one_metBy_relationship_with_goal
(object (is-a Goal) (OBJECT ?obj) (status Met | Partially-Met) (metBy $?mb&:(= (length$ ?mb) 0)))
=>
(printout t "Goal (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"metBy\" relationship with a requirement." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 59:

Every scenario must specify at least one isScenarioOf relationship with a viewpoint.

```
(mapclass NormalScenario)

(mapclass ExceptionScenario)

(defrule specify_at_least_one_isScenarioOf_relationship_with_viewpoint

(object (is-a NormalScenario | ExceptionScenario) (OBJECT ?obj) (isScenarioOf $?iso&:(= (length$ ?iso) 0)))

=>

(printout t "Scenario (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"isScenarioOf\" relationship with a viewpoint." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 60:

If a scenario has status "Active", then that scenario must specify at least one operationalizes relationship with a requirement.

```
(mapclass NormalScenario)

(mapclass ExceptionScenario)

(defrule specify_at_least_one_operationalizes_relationship_with_scenario

(object (is-a NormalScenario | ExceptionScenario) (OBJECT ?obj) (status Active) (operationalizes $?o&:(= (length$ ?o) 0)))

=>

(printout t "Scenario (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"operationalizes\" relationship with a requirement." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 61:

Every scenario must specify at least one hasTask relationship with a task.

```
(mapclass NormalScenario)

(mapclass ExceptionScenario)

(defrule specify_at_least_one_hasTask_relationship_with_viewpoint

(object (is-a NormalScenario | ExceptionScenario) (OBJECT ?obj) (hasTask $?ht&:(= (length$ ?ht) 0)))

=>

(printout t "Scenario (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"hasTask\" relationship with a task." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 62:

Every task must specify at least one isTaskOf relationship with a scenario.

```
(mapclass Task)
(defrule specify_at_least_one_isTaskOf_relationship_with_scenario
(object (is-a Task) (OBJECT ?obj) (isTaskOf $?ito&:(= (length$ ?ito) 0)))
=>
(printout t "Task (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one  \"isTaskOf\" relationship with a scenario." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 63:

Every task must specify at least one relationship with another task, i.e. AND-refinesInto, OR-refinesInto.

```
(mapclass Task)
(defrule specify_at_least_one_relationship_with_another_task
(object (is-a Task) (OBJECT ?obj) (AND-refinesInto $?ari&:(= (length$ ?ari) 0)))
(object (is-a Task) (OBJECT ?obj) (OR-refinesInto $?ori&:(= (length$ ?ori) 0)))
=>
(printout t "Task (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one relationship (AND-refinesInto, OR-refinesInto) with another task." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 64:

Every task must specify at least one isTaskOf relationship with a viewpoint.

```
(mapclass Task)
(defrule specify_at_least_one_isTaskOf_relationship_with_viewpoint
(object (is-a Task) (OBJECT ?obj) (isTaskOf $?ito&:(= (length$ ?ito) 0)))
=>
(printout t "Task (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one  \"isTaskOf\" relationship with a viewpoint." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 65:

Every task must specify at least one hasAction relationship with an action.

```
(mapclass Task)
(defrule specify_at_least_one_hasAction_relationship_with_action
(object (is-a Task) (OBJECT ?obj) (hasAction $?ha&:(= (length$ ?ha) 0)))
=>
(printout t "Task (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one  \"hasAction\" relationship with an action." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 66:

Every action must specify at least one isActionOf relationship with a task.

```
(mapclass Action)
(defrule specify_at_least_one_isActionOf_relationship_with_task
(object (is-a Action) (OBJECT ?obj) (isActionOf $?iao&:(= (length$ ?iao) 0)))
=>
(printout t "Action (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"isActionOf\" relationship with an action." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 67:

Every action must specify at least one performedBy relationship with an actor.

```
(mapclass Action)
(defrule specify_at_least_one_performedBy_relationship_with_actor
(object (is-a Action) (OBJECT ?obj) (performedBy $?pb&:(= (length$ ?pb) 0)))
=>
(printout t "Action (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"performedBy\" relationship with an actor." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 68:

Every actor must specify at most one represents relationship with a stakeholder.

```
(mapclass Organization)
(mapclass Device)
(mapclass System)
(mapclass Person)
(defrule specify_at_most_one_represents_relationship_with_stakeholder
(object (is-a Organization | Device | System | Person) (OBJECT ?obj) (represents nil))
=>
(printout t "Actor (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at most one  \"represents\" relationship with a stakeholder." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 69:

Every actor must specify at least one performes relationship with an action.

```
(mapclass Organization)
(mapclass Device)
(mapclass System)
(mapclass Person)
(defrule specify_at_least_one_performes_relationship_with_action
(object (is-a Organization | Device | System | Person) (OBJECT ?obj) (performes $?p&:(= (length$ ?p) 0)))
=>
(printout t "Actor (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at most one  \"performes\" relationship with an action." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 70:

Every stakeholder must specify at least one representedBy relationship with an actor.

```
(mapclass Stakeholder)
(defrule specify_at_least_one_representedBy_relationship_with_actor
(object (is-a Stakeholder) (OBJECT ?obj) (representedBy $?rb&:(= (length$ ?rb) 0)))
=>
(printout t "Action (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"isActionOf\" relationship with an action." crlf))
```

Behavioral-GST Ontology - Jess Completeness Rule 71:

Every stakeholder must specify at least one isStakeholderOf relationship with a goal.

```
(mapclass Stakeholder)
(defrule specify_at_least_one_isStakeholderOf_relationship_with_goal
(object (is-a Stakeholder) (OBJECT ?obj) (isStakeholderOf $?iso&:(= (length$ ?iso) 0)))
=>
(printout t "Stakeholder (" (slot-get ?obj :NAME) ") is incomplete. You did not specify at least one \"isStakeholderOf\" relationship with a goal." crlf))
```

APPENDIX K: EVALUATION – PART 2 EVALUATION TESTS RESULTS

Table 6. PART 2 Evaluation Tests Results (First Round)

Criteria	No. of Rules	No. of Success	% of Success
PAL Constraints Check	76	56	**74%**
Jess Consistency Check	60	45	**75%**
Jess Completeness Check	71	54	**76%**
TOTAL	207	155	**75%**

Table 7. PART 2 Evaluation Tests Results (Second Round)

Criteria	No. of Rules	No. of Success	% of Success
PAL Constraints Check	76	66	**87%**
Jess Consistency Check	60	51	**85%**
Jess Completeness Check	71	58	**82%**
TOTAL	207	175	**85%**

REFERENCES

[ALEXANDER 04] Alexander, I. F., Maiden, N., Wiley, J. P. (2004). *Scenarios, Stories, Use Cases Through the System Development Life-Cycle*, John Wiley & Sons, New York, 2004.

[ALSPAUGH 08] Alspaugh, T., Antón, A. (2008). Scenario Support for Effective Requirements. *Information and Software Technology*, *50*(3), 2008, pp. 198-220, 2008.

[ANNETT 67] Annett, J., Duncan, K. (1967). Task Analysis and Training in Design. *Occupational Psychology*, *41*, pp. 211–221.

[ANTON 96] Antón, A. (1996). Goal-Based Requirements Analysis. *International Conference on Requirements Engineering (ICRE'96)*, Colorado Springs, Colorado, USA, April 1996, pp. 136-144.

[AOYAMA 05] Aoyama, M. (2005). Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products. *Proc. RE'05*, Aug.-Sept. 2005, Paris, pp. 85-94.

[AOYAMA 07] Aoyama, M. (2007). Persona-Scenario-Goal Methodology for User-Centered Requirements Engineering. *Proc. IEEE RE'07*, pp. 185-194.

[BOOCH 05] Booch, G., Rumbaugh, J., Jacobson, I. (2005). *The Unified Modeling Language User Guide* (2nd Edition). Addison-Wesley, 2005.

[BORST 97] Borst, P. (1997). *Construction of Engineering Ontologies for Knowledge Sharing and Reuse* (PhD thesis). Universiteit Twente.

[BUHR 98] Buhr, R.J.A. (1998). Use Case Maps as Architectural Entities for Complex Systems. *In Transactions on Software Engineering, IEEE*, *24*(12), pp. 1131-1155.

[BREITMAN 03] Breitman, K. K., Leite, J. C. S. P. (2003). Ontology as a Requirement Engineering Product. In *Proceedings of the 11th IEEE International Requirements Engineering Conference*, Monterey Bay, California, USA, pp. 309-319.

[CHIMAERA 00] Chimaera Ontology Environment. http://www.ksl.stanford.edu/software/chimaera.

[CLELAND-H 05] Cleland-Huang, J., Settimi, R., Benkhadra, O. (2005). Goal-Centric Traceability for Managing Non-Functional Requirements. In *Proceedings on the 27th International Conference on Software Engineering (ICSE'05)*, St. Louis, MO, USA, May 15-21, 2005, pp. 362-371.

[CASTRO 08] Castro, J., Acua, S. T., Juristo, N. (2008). Integrating the Personas Techniques into the Requirements Analysis Activity. *Computer Science, Mexican International Conference on*, pp, 104-112.

[CHARREL 03] Charrel, P-J. (2003). Viewpoints for Knowledge Management in System Design, *Dynamics and Change in Organizations*, pp. 263-277.

[COCKBURN 97a] Cockburn, A. (1997). Goals and Use Cases. *Journal of Object-Oriented Programming*, *10*, 35-40.

[COCKBURN 97b] Cockburn, A. (1997). Using Goal-Based Use Cases. *Journal of Object-Oriented Programming*, *10*, 56-62.

[CONSTANTINE 99] Constantine, L., Lockwood, L. (1999). *Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design*. Addison-Wesley Professional, 1999.

[COOPER 99] Cooper, A. (1999). *The Inmates Are Running The Asylum*. New York Macmillan, 1999.

[COOPER 07] Cooper, A., Reimann, R., Cronin, D. (2007). *About Face 3.0: The Essentials of Interaction Design*. Indianapolis, Wiley, 2007.

[DAML] DAML Ontology Library. http://www.daml.org/ontologies/.

[DANTIN 05] Dantin, U. (2005). Application of Personas in User Interface Design for Educational Software. *Proceedings of the 7th*

Australasian Conference on Computing education, pp.239-247, January 01, 2005, Newcastle, New South Wales, Australia.

[DARDENNE 93] Dardenne, A., Lamsweerde, A. van, Fickas, S. (1993). Goal-Directed Requirements Acquisition. *Science of Computer Programming*, *20*, 3-50.

[ERIKSSON 03] Eriksson, H. (2003). Using JessTab to Integrate Protégé and Jess, *IEEE Intelligent Systems Journal*, *18*, 43-50.

[FINKELSETIN 92] Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke., M. (1992). Viewpoints: A Framework for Integrating Multiple perspectives in System Development, *International Journal of Software Engineering and Knowledge Engineering*, 2(*1*), 31 -57.

[GAO 08] General Accounting Office. (2008). *Defense Acquisitions: Significant Challenges Ahead in Developing and Demonstrating Future Combat System's Network and Software*. GAO-08-409, Washington, D.C., March 2008. http://www.gao.gov/new.items/d08409.pdf

[GASEVIC 06] Gasevic, D., Djuric, D., Devedzic, V. (2006). *Model Driven Architecture and Ontology Development.* Springer, 2006.

[GOLEMATI 07] Golemati, M., Katifori, A., Vassilakis, C., Lepouras, G., Halatsis, C. (2007). Creating an Ontology for the User Profile: Method and Applications. In *Proceedings of the First International Conference on Research Challenges in Information Science (RCIS)*, April 23-26, 2007, Ouarzazate, Morocco.

[GOMEZ-PEREZ 94] Gomez-Perez, A. (1994). Some ideas and examples to evaluate ontologies. Knowledge Systems Laboratory, Stanford University.

[GRUBER 93a] Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, *5*(2), 199-2020.

[GRUBER 93b] Gruber, T. R. (1993). Towards principles for the design of ontologies used for knowledge sharing. In Guarino, N. and Poli, R., editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands. Kluwer Academic Publishers.

[GRUDIN 02] Grudin, J., Pruitt, J. (2002). Personas, Participatory Design, and Product Development: An Infrastructure for Engagement. *The Participatory Design Conference*, Malmo, Sweden, 2002, 144-161.

[GUARINO 02] Guarino, N., Welty, C. (2002). Evaluating ontological decisions with OntoClean. *Communications of the ACM, 45*(2), 61-65.

[HORRIDGE 04] Horridge, M., Knublauch, H., Rector, A., Stevens, R., Wroe, C. (2004). *A Practical Guide to Building OWL Ontologies using the Protege-OWL Plugin and CO-ODE Tools, Edition 1.0*, University of Manchester, CO-ODE project, 2004.

[IEEE 10] Institute of Electrical and Electronics Engineers. (2010). IEEE Systems and Software Engineering Vocabulary. *IEEE Standard 24765-2010 (superseded IEEE Std. 610.12-1990)*.Institute of Electrical and Electronics Engineers, New York, 2010.

[IEEE 11 a] Institute of Electrical and Electronics Engineers. (2011). IEEE Systems and software engineering – Life cycle processes – Requirements engineering. *IEEE Standard 29148-2011.* Institute of Electrical and Electronics Engineers, New York, 2011.

[IEEE 11 b] Institute of Electrical and Electronics Engineers. (2011). ISO/IEC/IEEE Systems and software engineering - Architecture *Standard 42010-2011.* Institute of Electrical and Electronics Engineers, New York, 2011.

[IEEE 00] Institute of Electrical and Electronics Engineers. (2000). IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. *IEEE Standard 1471-2000,* 1-23, 2000.

[ISO 06] Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs). *ISO*, *ISO9241-11*, 2006.

[JESS] Java Expert System Shell. http://www.jessrules.com/jess/

[KAIYA 02] Kaiya, H., Horai, H., Saeki, M. (2002) AGORA: Attributed Goal-Oriented Requirements Elicitation Method. In *International Conference on Requirements Engineering*, Essen, Germany, 2002.

[KAIYA 05] Kaiya, H., Saeki, M. (2005). Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach. *Proc. QSIC'05*, Sept. 2005, 223-230.

[KAIYA 06] Kaiya, H., Saeki, M. (2006). Using Domain Ontology as Domain Knowledge for Requirements Elicitation. *Proc. RE'06*, Sept. 2006, 189-198.

[KATIFORI 08] Katifori, A., Vassilakis, C., Daradimos, I., Lepouras, G., Ioannidis, Y., Dix, A., Poggi, P., Catarci, T. (2008). Personal Ontology Creation and Visualization for a Personal Interaction Management System. *Workshop on The Disappearing Desktop: Personal Information Management 2008. CHI2008*, Florence, 5th & 6th April 2008.

[KOAY 09] Koay, N., Kataria, P., Juric, R., Oberndorf, P., Terstyanszky, G. (2009). Ontolgical Support for Managing Non-Functional Requirements in Pervasive Healthcare. *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences (HICSS 42)*, Hawaii, Big Island, USA, Jan. 5-8, 2009, pp. 1-10.

[KOTONYA 92] Kotonya, G., Sommerville, I. (1992). Viewpoints for Requirements Definition, *Software Engineering Journal*, 7(*6*), 375-387.

[LEE 05] Lee, S. W., Gandhi, R. (2005). Ontology-Based Active Requirements Engineering Framework. In *Proceedings of the 12th Asia-Pacific Software Engineering Conference (APSEC'05)*, Taipei, Taiwan, Dec. 15-17, 2005, pp. 481-490.

[LIU 04] Liu, L., Yu, E. (2004). Designing Information Systems in Social Context: A Goal and Scenario Modeling Approach. *Information Systems*, *29*(2), 187-203.

[LIU 08] Liu, W., Jin, F., Zhang, X. (2008). Ontology-Based User Modeling for E-Commerce System. Proc. *ICPCA'08*, *1*, 260-263.

[MARKOPOULOS 00] Markopoulos, P. (2000). Modeling user tasks with the Unified Modeling Language. In *Proc. of Workshop TUPIS2000 in <<UML2000>> International Conference*, York, UK, 2000.

[MIKKELSON 00] Mikkelson, N., Lee, W. O. (2000). Incorporating User Archetypes into Scenario-Based Design. *9th Annual Usability Professionals' Association (UPA) Conference*, Asheville, North Carolina.

[MILLER 08] Miller, G., Williams, L. *Personas: Moving Beyond Role-Based Requirements Engineering*. http://agile.csc.ncsu.edu/SEMaterials/Personas.pdf.

[MULDER 07] Mulder, S., Yaar, Z. (2007). *The User Is Always Right.* New Riders, 2007.

[NIELSEN 93] Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann, 1993.

[NOY 00a] Noy, F., Fergerson, W., and Musen, A. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *Dieng, R. and Corby, O., editors, Knowledge Acquisition, Modeling and Management, 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, 2000, Proceedings, volume 1937 of Lecture Notes in Computer Science*, pp. 17–32. Springer-Verlag.

[NOY 00b] Noy, F., Musen, A. (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI/IAAI*, pp. 450-455.

[NOY 01] Noy, F., Mcguinness, L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[PAL] Protégé Axiom Language. http://protege.stanford.edu/plugins/paltabs/pal-quickguide/

[PATERNO 00] Paternó, F. (2000). Model-Based Design of Interactive Applications. *ACM Intelligence*, *11*(4), 26-38.

[PATRIOTWEB a] George Mason University's PatriotWeb system. https://patriotweb.gmu.edu/

[PATRIOTWEB b] Prototype of the George Mason University's PatriotWeb system. http://mason.gmu.edu/~wsim/Main_Menu.htm

[PROTÉGÉ 00] Protégé, *The Protégé Project.* http://protege.stanford.edu.

[PROTÉGÉ a] Protégé Ontology Library. http://protegewiki.stanford.edu/wiki/Protege_Ontology_Library.

[PROTÉGÉ b] Protégé-Frames User's Guide. http://protegewiki.stanford.edu/wiki/PrF_UG.

[PRUITT 03] Pruitt, J., Grudin, J. (2003). Personas: Practice and Theory. *Conference on Designing for User Experiences*, San Francisco, CA, 2003, pp. 1-15.

[PRUITT 06] Pruitt, J., Adlin, T. (2006). *The Persona lifecycle: Keeping People in Mind Throughout Product Design*. Morgan Kaufman, 2006.

[PU 09] Pu, Y-N., Liu, Q. (2009). A Viewpoint-Oriented Requirements Elicitation Integrated with Aspects. *CSIE*, *7*, 706-711.

[QUESENBERY 06] Qiesenbery, W. (2006). Storytelling and Narrative. In Pruitt, J. & Adlin T. *Persona Lifecycle: Keeping People in Mind Throughout Product Design*. Morgan Kaufman Publishers, San Francisco, pp. 520-554.

[RANDOLPH 04] Randolph, G. (2004). Use-Cases and Personas: A Case Study in Light-Weight User Interaction Design for Small Development Projects. *Information Science Journal*, *7*, 105-116.

[ROLLAND 98] Rolland, C., Souveyet, C., Achour, B. (1998). Guiding Goal Modeling Using Scenarios. *IEEE Transactions on Software Engineering, Special Issue on Scenario Management*, *24*(12), 1055-1071.

[RUSSELL 03] Russell, S., Norvig, P. (2003). *Artificial Intelligence: A Modern Approach* (2nd Edition). Prentice Hall, 2003, Chapter 10 Knowledge Representation, pp. 320-374.

[SHIBAOKA 07] Shibaoka, M., Kaiya, H., Saeki, M. (2007). GOORE: Goal-Oriented and Driven Requirements Elicitation Method. *ER Workshops 2007*, pp. 225-237.

[SIM 14a] Sim, W., Brouse, P. (2014). Empowering Requirements Engineering Activities with Personas. *In: 12th Annual Conference on Systems Engineering Research (CSER 2014), Procedia Computer Science, 28*, 237–246. http://dx.doi.org/10.1016/j.procs.2014.03.030

[SIM 14b] Sim, W., Brouse, P. (2014). Towards an Ontology-Based Persona-Driven Requirements and Knowledge Engineering. *In: Conference on Complex Adaptive Systems (2014), Procedia Computer Science, 36*, 314–321. http://dx.doi.org/10.1016/j.procs.2014.03.030

[SOMMERVILLE 97] Sommerville, I., Sawyer, P. (1997). Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering, *Annals of Software Engineering*, *3*(1), 101-130.

[SOMMERVILLE 10] Sommerville, I. (2010). *Software Engineering* (9th ed.). Addison-Wesley, 2010.

[STUDER 98] Studer, R., Benjamins, V. R., Fensel, D. (1998). Knowledge Engineering: Principles and Methods. *Data and Knowledge Engineering*, *25*(1-2), 161-197.

[SUTCLIFFE 98] Sutcliffe, A. G., Maiden, N, A. M., Minocha, S., Manuel, D. (1998). Supporting Scenario-Based Requirements Engineering. *Software Engineering, IEEE Transactions on*, *24*(12), 1072-1088.

[SUTCLIFFE 02] Sutcliffe, A. (2002). *User-Centered Requirements Engineering: Theory and Practice*. Springer, 2002.

[SWRL] Semantic Web Rule Language. http://www.w3.org/Submission/SWRL/.

[SQWRL] Semantic Query-Enhanced Web Rule Language. http://protege.cim3.net/cgi-bin/wiki.pl?SQWRL.

[TECUCI 98] Tecuci, G. (1998). *Building Intelligent Agents.* Academic Press, 1998, pp. 33-77.

[TECUCI 08] Tecuci, G., Boicu, M. (2008). *A Guide for Ontology Development with Disciple.* Research Report 3, Learning Agents Center, George Mason University, 2008.

[VAN LAMSWEERDE 01] Van Lamsweerde, A. (2001). Goal-Oriented Requirements Engineering: A Guided Tour. *In Proceedings of the 5th IEEE International Symposium on Requirements Engineering RE'01*, pp. 249–263. Los Alamitos, CA:

[VAN LAMSWEERDE 09] Van Lamsweerde, A. (2009). *Requirements Engineering: From System Goals to UML Models to UML Models to Software Specifications*. Wiley, 2009.

[VERNER 05] Verner, J., Cox, K., Bleistein, S., Carpa, N. (2005). Requirements Engineering And Software Project Success: An Industrial Survey in Australia And The U.S. *Australasian Journal of Information Systems*, *13*(1), 225-238.

[W3C 04] W3C. (2004). *W3C recommendation*, *OWL Web Ontology Language Overview, February 2004.* http://www.w3.org/TR/owl-features/.

[YU 93] Yu, E. (1993). Modeling Organizations for Information Systems Requirements Engineering. *Proc. RE'93 – 1st Intl Symp. on Requirements Engineering, IEEE*, 1993, pp. 33-41.

[YU 11a] Yu, E., Giorgini, P., Maiden, N., Mylopoulos, J. (2011) *Social Modeling for Requirements Engineering*. MIT Press, 2011.

[YU 11b] Yu, E. (2011). i*: an agent- and goal-oriented modelling framework. http://www.cs.toronto.edu/km/istar/#roadmap

[ZAVE 97] Zave, P. (1997). Classification of Research Efforts in Requirements Engineering. *ACM Computing Surveys*, *29*(4), 315-321.

BIOGRAPHY

Wee Wee Sim is a Ph.D. candidate in Information Technology in The Volgenau School of Engineering at George Mason University. His dissertation research focuses on persona driven and ontology based user requirements modeling, Wee Wee Sim earned a Master's degree in Computer Science and a bachelor's degree in Mathematics, in 2004 and 1995 respectively.