NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

SCENARIO SELECTION AND STUDENT ASSESSMENT MODULES FOR CYBERCIEGE

by

Tiat Leng, Teo

December 2003

Thesis Advisor:	Cynthia Irvine
Second Reader:	Michael Thompson

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Title (Mix case letters) Scenario Selection and Student Assessment Modules for CyberCIEGE	**5. FUNDING NUMBERS**
6. AUTHOR(S) Tiat Leng, TEO	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

CyberCIEGE aims to provide an Information Assurance (IA) teaching/learning Laboratory in the form of an interactive, entertaining, commercial-grade PC-based computer game. Each game plays as a single scenario that serves to teach a particular IA concept. However, more synergy can be gained if there is higher-order organization of these scenarios, such as by grouping around a set of desired concepts to be taught, or by increasing the complexity of the scenarios built around a common theme. This thesis aims to provide an instructor tool for this purpose.

In addition, by tapping the CyberCIEGE event log files generated at the end of each game, we can reconstruct the game progress to support After Action Reviews (AAR) to assist the instructor and student to analyze game decisions and the student's progress. This provides a constructive follow-up to review and reinforce the concepts being taught.

14. SUBJECT TERMS Information Assurance, Security Education, After Action Review.	**15. NUMBER OF PAGES** 127
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

SCENARIO SELECTION AND STUDENT ASSESSMENT MODULES FOR CYBERCIEGE

Tiat Leng, Teo
Civilian, Singapore
B.Sc, National University of Singapore, 1991
M.Tech(SE), National University of Singapore, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2003

Author:	Tiat Leng, Teo
Approved by:	Cynthia Irvine Thesis Advisor
	Michael Thompson Second Reader
	Peter Denning Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

CyberCIEGE aims to provide an Information Assurance (IA) teaching/learning Laboratory in the form of an interactive, entertaining, commercial-grade PC-based computer game. Each game plays as a single *scenario* that serves to teach a particular IA concept. However, more synergy can be gained if there is higher-order organization of these scenarios, such as by grouping around a set of desired concepts to be taught, or by increasing the complexity of the scenarios built around a common theme. This thesis aims to provide an instructor tool for this purpose.

In addition, by tapping the CyberCIEGE event log files generated at the end of each game, we can reconstruct the game progress to support *After Action Reviews* (AAR) to assist the instructor and student to analyze game decisions and the student's learning progress. This provides a constructive follow-up to review and reinforce the concepts being taught.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I wish to thank Prof Cynthia Irvine and especially Michael Thompson for their inspiring and invaluable guidance during the course of this thesis; the developers from Rivermind who are building CyberCIEGE itself, offering me a first-hand view of a Silicon Valley-based games company in action; and fellow thesis students working on various aspects of the CyberCIEGE project: Ken Johns and Marc Meyers.

I also wish to thank my sponsor, the Defence Science and Technology Agency (DSTA) for the scholarship to participate in this enriching experience at the Naval Postgraduate School in Monterey, and the Ministry of Defence, Singapore, for creating this fascinating joint program between the National University of Singapore and NPS. Being a member of the pioneer batch in this program has been both exciting as well as painful.

And finally, to my wife and two wonderful kids who tolerated my insufferable mania to study at night, and who have thoroughly enjoyed our experiences here in the United States.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

As mentioned by [Salzer 1975], security is often viewed as "a negative kind of requirement". It is not uncommon that when faced with resource and schedule demands, security is often one of the first victims of project management prioritization. The consequences of such decisions are often not seen until the system has gone into operation. The impacts are only felt when security flaws are subsequently exploited or perceived to be exploitable. Traditional information systems have often seen security issues addressed only as an after thought, with dramatically disastrous effects. Attempts at conducting security analysis at that point are especially hard, as the system has become monolithic with hundreds of thousands of lines of codes to be analyzed. Post-implementation defects eventually lead to demands for numerous ad-hoc patches and consequent system down times. On occasions, hastily but poorly tested patches have themselves contributed to introducing further flaws, with patches requiring patches [Livingston 2003], and in some cases exploited by unskilled vandals (e.g. "script-kiddies").

Recognition of the needs for well-designed information security have been recognized as early as the 70s, as exemplified by the [Anderson 1972] study and the [Saltzer 1975] paper. In the military arena, increasing dependence and use of information technology, initially for combat support operations, and in more recent times as integral components of Command, Control and Communication (C3I) and Intelligence, Surveillance and Reconnaissance (ISR) systems have elevated the importance of information security. Military operations that have traditionally been conducted only in the physical space are rapidly moving into the information space as well. To this end, the US Joint doctrine for Information Operations (IO) [Joint 1998] recognizes Information Assurance (IA) as one of the key tenets of an effective IO strategy. As defined in [Joint 1998], IA is "IO that protect and defend information systems by ensuring their availability, integrity, authentication, confidentiality and non-repudiation". It includes provision for restoration of information systems incorporating protection, detection and reaction capabilities (see Figure 1).

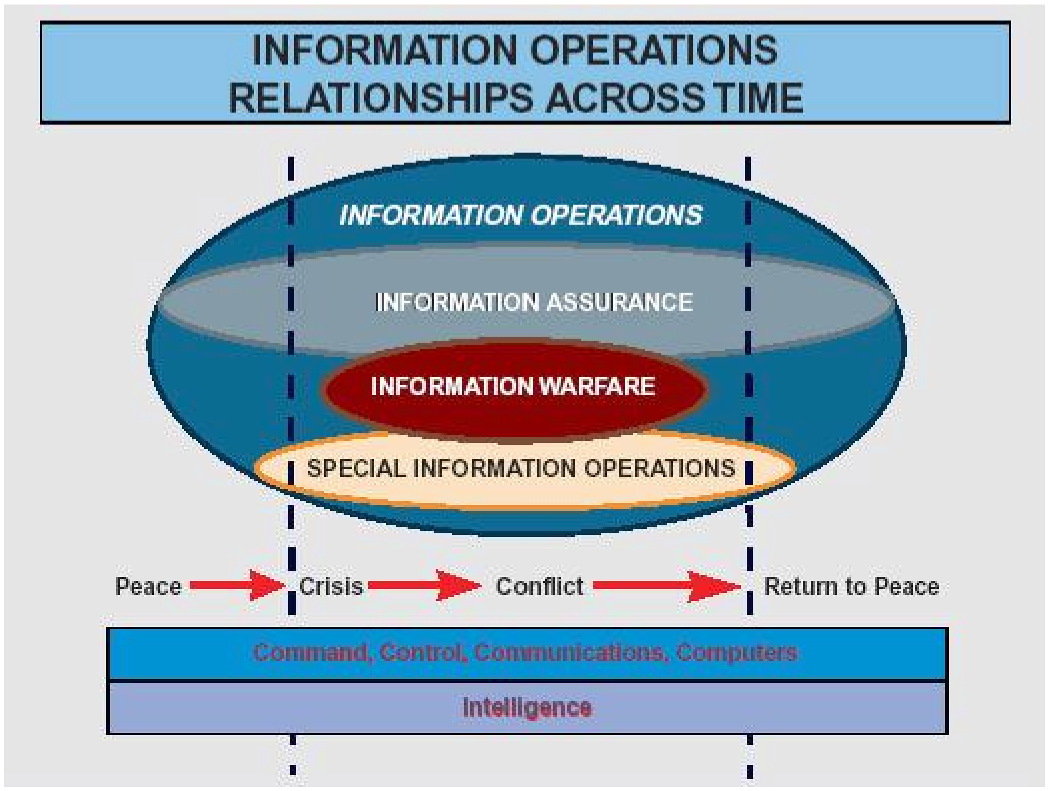

INFORMATION OPERATIONS RELATIONSHIPS ACROSS TIME
INFORMATION OPERATIONS
INFORMATION ASSURANCE
INFORMATION WARFARE
SPECIAL INFORMATION OPERATIONS
Peace
Crisis
Conflict
Return to Peace
Command, Control, Communications, Computers
Intelligence

Figure 1. Information Operations relationships across time. (From [Joint 1998])

As part of an effective implementation plan to further this, it is important to promote awareness of security issues to a wide spectrum of users. Information systems are today reaching a wide spectrum of users. Unfortunately, it is rare for non-security inclined personnel to possess a natural inclination to take an active interest in IA issues. How then can we overcome this gap? IA implementations face challenges in terms of commitment and conformance, this could lead to less effective implementations, resulting in an overall less effective operational IA posture.

An effective means for conducting training and education and to promote awareness in IA is hence crucial. As highlighted by [Tanner 2002], commercial and federal systems are targets of attack by criminals and foreign intelligence agencies. Technical protection measures alone are insufficient to defend against these as human errors and operators errors may leave systems open to other forms of attacks. Consequently, security education, training and awareness constitute an integral part of any effective layered defense strategy [DON 2000, Boyce 2002]. Usually, this involves taking away the warfighter's precious time from his/her operational functions to attend security awareness training. This is both costly and time-consuming. In addition, it is extremely difficult to conduct such training in a manner that would actually captivate the student's attention and interest, and hence improve on the retention of knowledge.

A novel alternative is to package the training in the form of interactive games. To this end, NPS is leading the effort in the development of CyberCIEGE, which could significantly contribute towards this goal. America's Army has paved the way in demonstrating the viability of using commercial-grade games to support a learning-teaching objective [USArmy 2003] as have other similar efforts, demonstrated by the Singapore Armed Forces [DSTA 2003], in utilizing games to supplement conventional training.

CyberCIEGE (see Figure 2), originally conceived as "SimSecurity", aims to provide an IA teaching and learning laboratory in the form of an interactive and entertaining commercial-grade PC-based game. It is conceived as a real-time interactive game with each scenario serving to teach one or more IA concepts. Through playing the game, the student gains insight into IA. The CyberCIEGE game itself is being developed by Rivermind, Inc.

Figure 2. CyberCIEGE.

B. SCOPE

The scope of this thesis is to develop on top of the CyberCIEGE effort, to attempt to answer the following questions:

1. How do we construct and organize scenarios to provide an educational focus on an IA-specific topic?

2. What do we need in order to perform student assessment of the scenarios played?

3. What interfaces must be introduced into CyberCIEGE to achieve this?

To explore to these questions, this thesis involves the analysis, design and implementation of instructor tools while concurrently defining the necessary interfaces. The instructor tools would enable the construction of campaigns by assembling a collection of CyberCIEGE scenarios and provides a means for reviewing student performance on the scenarios played.

In CyberCIEGE, each game plays as an individual scenario that serves to teach a particular IA concept. However, more synergy can be gained if there is a higher-order organization of these scenarios, such as by grouping around a set of desired concepts to be taught, or by increasing the complexity of the scenarios built around a common theme. Such a concept has been demonstrated in numerous strategy and action games such as Warcraft, Starcraft, Warlords, etc. Rather than working with isolated scenarios, the instructor can construct the scenarios to enable the student to progress from one scenario to another in a logical fashion. In this manner, a library of scenarios can be reused and readily assembled to craft different packages for different teaching objectives. We shall refer to these as *campaigns*. Campaigns can then be customized to meet specific training objectives and improve the relevance for the targeted student audience. A tool is hence needed to enable the instructor to construct these campaigns. Such a tool is currently not within the development scope of CyberCIEGE being undertaken by Rivermind.

Similarly, within the game being developed by Rivermind, there is little or no opportunity for the instructor to review the game played by the student unless the instructor is physically present and observes the progress of game play. Such an approach is clearly not preferred and would defeat the objective of a virtual laboratory. On the other hand, if the game play could be instrumented and significant events are logged, the logs can then be used to analyze game decisions and the student's progress. This approach would enable *After Action Reviews* (AAR) to be carried, enabling the instructor to provide the student constructive follow-up reviews and to further reinforce the concepts being taught [DOA 1993, Morrison 1999].

THIS PAGE INTENTIONALLY LEFT BLANK

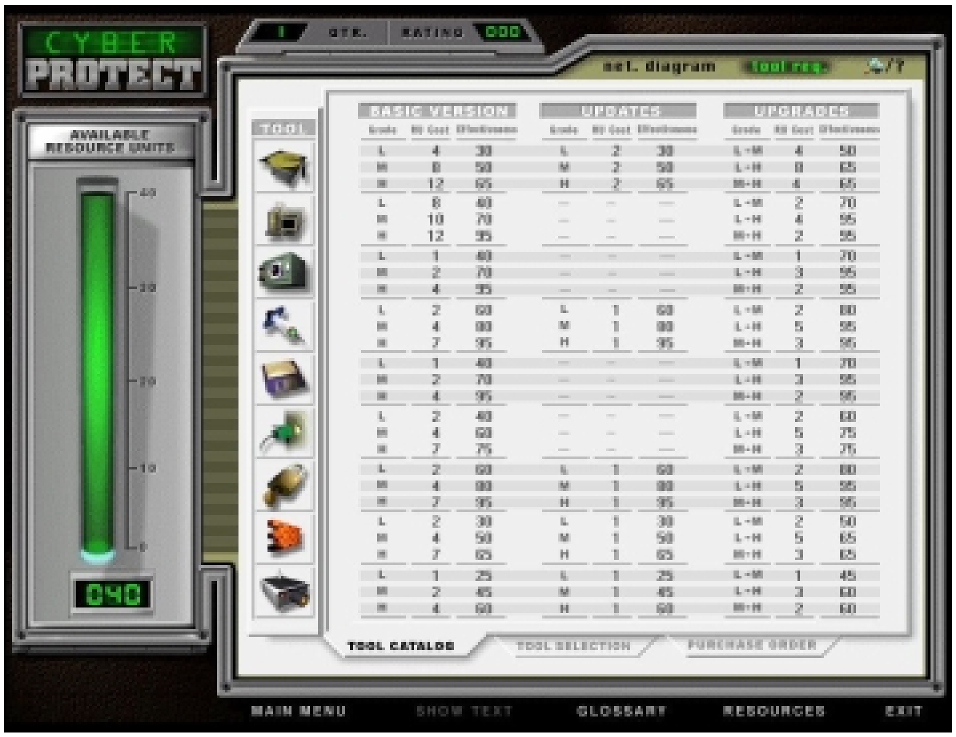
II. REQUIREMENTS

A. EXISTING PRODUCTS

A search for existing games of a similar genre reveals the existence of *CyberProtect* and the *Information Security Wargaming System* (ISWS) described in [Saunders 2003] and *AI Wars: The Awakening* [Nexus 2003].

1. CyberProtect

CyberProtect is a turn-based game sponsored by the Office of the Assistant Secretary of Defense for C3I and the IA Program Management Office of the Defense Information Systems Agency (DISA). The game simulates a fairly simple, small networked system with interconnections to other departmental systems, external sites and the Internet. The game features an introductory tutorial to guide new players through the basic of the game's mechanics and a small reference guide to explain the purpose of each of the countermeasures tools available. CyberProtect is fairly easy to learn, and with these tools, new players can get going very quickly.

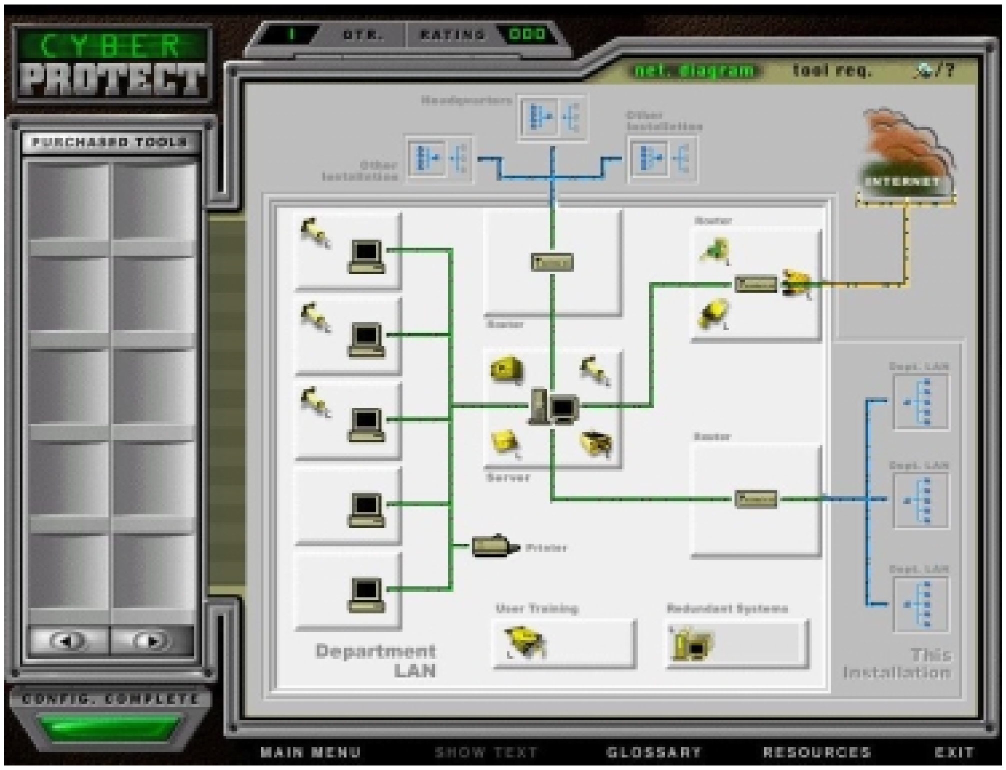


BASIC VERSION			UPDATES			UPGRADES		
Grade	RU Cost	Effectiveness	Grade	RU Cost	Effectiveness	Grade	RU Cost	Effectiveness
L	4	30	L	2	30	L-M	4	50
M	8	50	M	2	50	L-H	8	65
H	12	65	H	2	65	M-H	4	65
L	8	40	—	—	—	L-M	2	70
M	10	70	—	—	—	L-H	4	95
H	12	95	—	—	—	M-H	2	95
L	1	40	—	—	—	L-M	1	70
M	2	70	—	—	—	L-H	3	95
H	4	95	—	—	—	M-H	2	95
L	2	60	L	1	60	L-M	2	80
M	4	80	M	1	80	L-H	5	95
H	7	95	H	1	95	M-H	3	95
L	1	40	—	—	—	L-M	1	70
M	2	70	—	—	—	L-H	3	95
H	4	95	—	—	—	M-H	2	95
L	2	40	—	—	—	L-M	2	60
M	4	60	—	—	—	L-H	5	75
H	7	75	—	—	—	M-H	3	75
L	2	60	L	1	60	L-M	2	80
M	4	80	M	1	80	L-H	5	95
H	7	95	H	1	95	M-H	3	95
L	2	30	L	1	30	L-M	2	50
M	4	50	M	1	50	L-H	5	65
H	7	65	H	1	65	M-H	3	65
L	1	25	L	1	25	L-M	1	45
M	2	45	M	1	45	L-H	3	60
H	4	60	H	1	60	M-H	2	60

* Screenshot from CyberProtect.

Figure 3. CyberProtect - tools acquisition screen.

It is played over four quarters (i.e. turns) and revolves around the acquisition and deployment of a limited set of abstract defensive information assurance measures which are applied to a network of computers at the start of each quarter, as illustrated in Figures 3 and 4. These measures include training, redundant systems, access controls, virus protection, backups, encryption, firewalls and intrusion detection systems, all with varying levels of quality and effectiveness (i.e. low, medium or high), and cost. Given a limited operating budget in the form of Resource Units (RUs), the player never has enough to buy everything desired and careful choices would have to be made. The acquired tools are then deployed on the various elements of the networked system.



* Screenshot from CyberProtect.

Figure 4. CyberProtect - system with countermeasures applied.

The networked system is then subjected to a variety of randomly generated attacks at the end of each quarter. Specifically, these attacks include jamming, virus infections, moles (insider actions), social engineering, packet sniffing, theft, modification, spoofing and disaster effects (e.g. flooding). Each attack is evaluated against the defensive counter-measures in place to determine the extent of success or failure of system penetration. Feedback is provided indicating the type of attack, origin (which external site or insider) and status of the attack (successful or failed). A multimedia narration of the attack is also available. With these clues, the player can then reexamine the existing countermeasures implemented to figure out why the attacks were successful and proceed to acquire further measures or upgrade existing ones to better improve the defensive posture.

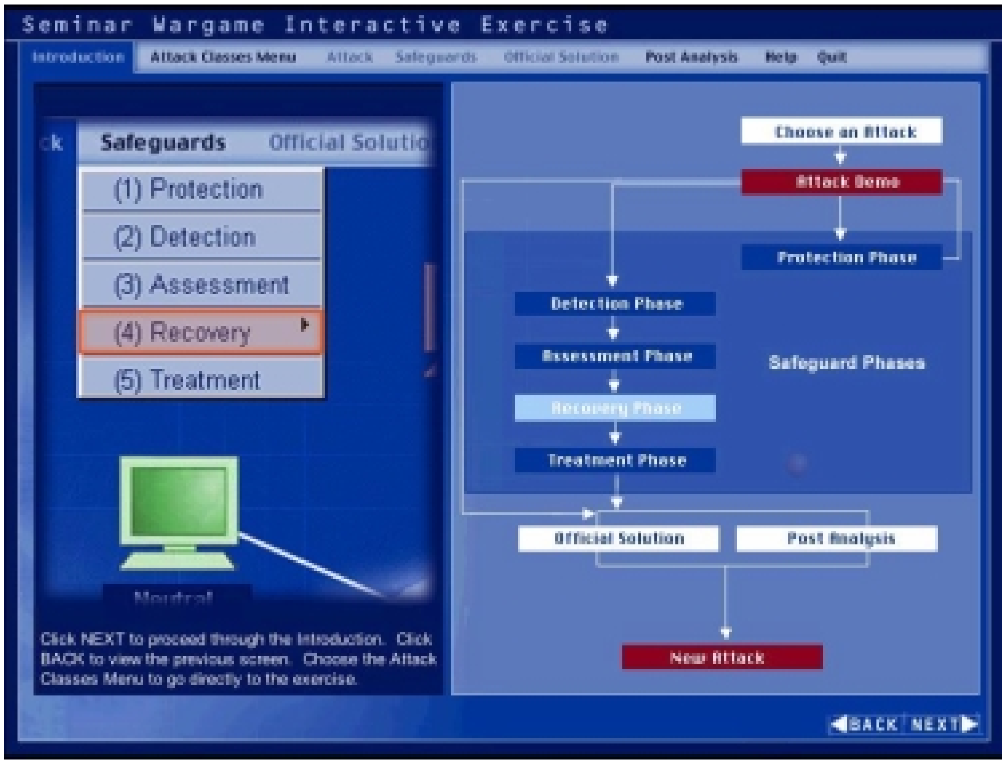
At the end of the quarter, a summary and score of the player's progress is given. Success in the game is defined as achieving a final score of 90 or more, thus providing a quantifiable measure.

2. Information Security Wargaming System (ISWS)

ISWS, developed by Concurrent Technologies Corporation (CTC) for the National Defense University, is a tutorial type simulation that provides in-depth focus on specific attacks, primarily centered on network-based attacks, and the applicable defensive measures. The package has the following teaching objectives:

- Understand network configuration to defend against attacks.
- Identify and recommend countermeasures to network attacks.
- Identify types of security measures to protect and maintain data integrity.
- Make better strategic decisions related to the protection of network attacks.

A short tutorial guides the student through the various phases of implementing the defenses and the tutorial functionalities (see Figure 5).



Click NEXT to proceed through the Introduction. Click BACK to view the previous screen. Choose the Attack Classes Menu to go directly to the exercise.

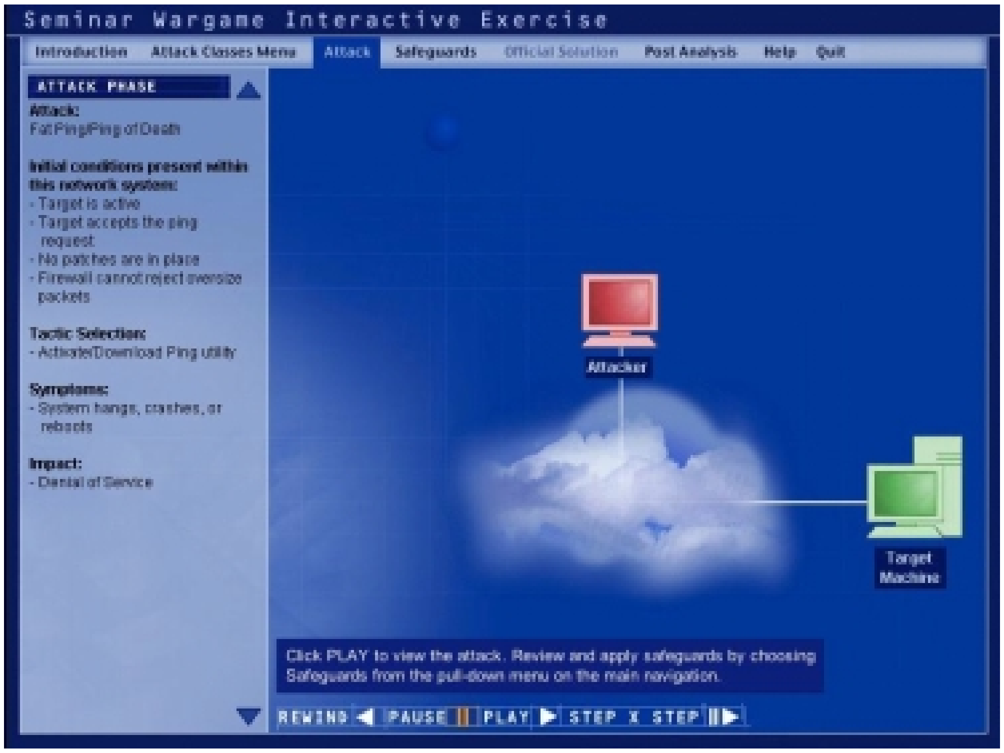
* Screenshot from ISWS.

Figure 5. ISWS - tutorial.

The various forms of attacks are grouped into 10 classes, namely - disruption, modification, destruction, infection, intrusion, theft/fraud, exploitation and observation. From each class of attacks, the student selects a specific attack and ISWS will explain the behavior of a system when subjected to such an attack (Figure 6). The student can then choose from a list of defensive options to be employed. ISWS presents a fairly detailed explanation of the pre-conditions and behavior of a system when undergoing the attack using multimedia presentations.

The student then proceeds to select from a palette of safeguards for each phase against the specific attack. These phases include protection, detection, assessment, recovery and treatment. When applied, the system assesses and provides feedback on the effectiveness of the set of safeguards selected. The student is free to examine various what-if combinations of safeguards or to proceed to the next phase.

Upon completing the phases, the student is presented with the “official” solution.



Attack:
Fat Ping/Ping of Death

Initial conditions present within this network system:
- Target is active
- Target accepts the ping request
- No patches are in place
- Firewall cannot reject oversize packets

Tactic Selection:
- Activate/Download Ping utility

Symptoms:
- System hangs, crashes, or reboots

Impact:
- Denial of Service

Click PLAY to view the attack. Review and apply safeguards by choosing Safeguards from the pull-down menu on the main navigation.

* Screenshot from ISWS.

Figure 6. ISWS - attack scenario.

3. Artificial Intelligence (AI) Wars: The Awakening

AIWars [Nexus 2003] is a futuristic first-person 3D real-time action-strategy game published by Nexus Interactive Studios Inc. It is targeted at the gaming community and is intended purely for its entertainment value only.

The player has to conduct research in various areas in order to gain more powerful or better software and hardware. In order to understand the cyber-environment, referred to as the Net, the player has to map out the different systems and gather datablocks to sell or to keep. The player possesses various defensive (e.g. Masquerade, Firewall, Antivirus, Spoof, etc) and offensive (Crack, Viral Infections, Infinite Recursive Calculations (IRC), etc) options with which to interact within the Net.

Characters with different predispositions (i.e. usually friendly, sometimes unfriendly) exist in the NET and they can be interacted with. However, if the player goes into a system and starts attacking a Warden character, the player's name will be placed in a hacker log. The next time that the player enters the system, the player would be attacked on sight.

A system is divided into public (e.g. online stores, public datastores, news groups and chat rooms) and private nodes (e.g. system cores, secure datastore and private meeting rooms) with passwords required for gaining entry into a private node. Private nodes are further guarded by a Warden. Without the password, the player can force an entry by launching the Crack software against the Warden. Attempts at such password cracking can trigger an alarm and result in detection. The latter can be mitigated if the player pre-activated the Masquerade software, otherwise the Warden will launch an active alarm and record the player as an intruder. Subsequent character interactions within the system would be hostile and system elements will attempt to eradicate the player. To clear the alarms, the player has to reach the system's Core and interface with it to plant a Backdoor.

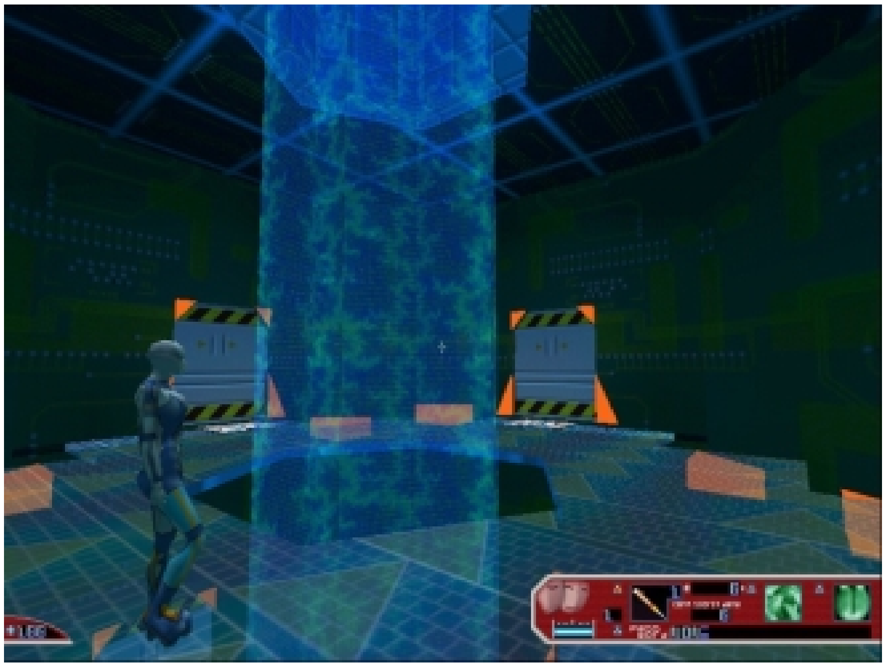


Figure 7. AIWars - Inside a Router. (From [Nexus 2003]).

Success in the game is achieved in any one of three ways: taking control of the net, transferring the player's consciousness and memories into the Net to achieve immortality or by bringing the player's agent to sentience. Taking control of the Net requires that the player place a Backdoor in the Core of each of the key systems. The immortality option requires development of six specific technologies, while the agent sentience option requires a different set of 4 technologies and acquisition of a lot of data for the agent to evolve.

B. BROAD DESIGN OBJECTIVES

[Tanner 2002] hypothesized that simulation-based computer security awareness training can be more focused and less expensive than a lecture- or laboratory-based courses. The design objectives of such a class of tool are defined as follows:

1. Content: Understand the Threat

The first step is to know the enemy who may range from novice hackers and script-kiddies to state-sponsored organized hackers. Novice hackers typically employ readily available tools to exploit known system vulnerabilities. Although pervasive, these are relatively easily mitigated. In contrast, state-sponsored organized hackers are patient, methodical and not limited to well-known vulnerabilities and readily available attack tools. Their activities tend to be covert and not easily detectable.

2. Content: Awareness of Known Weakness and Attack Techniques

The objective for the student is to learn about the weaknesses of a networked computer system and understand how these may be exploited by hackers. The student should be aware of which vulnerabilities can be eliminated and which are unavoidable exposures inherent in the design of the system.

3. Pedagogy: Support Multiple Training Objectives

The purpose of awareness training is to *train* the student. To achieve this goal, the following training objectives may be relevant:

a. Connecting concepts to practice.

b. Repeatability.

c. Progressing from novice through more sophisticated scenarios.

d. Examining “what if’s” by reconfiguring and trying again.

e. Practicing skills in a realistic training environment.

f. Developing problem-solving and decision-making skills.

g. Learning to recognize operational indicators of normal, abnormal and emergency conditions.

4. Pedagogy: Support Multiple Perspectives

A crucial pedagogical need is to enable the student to understand the cause-and-effect relationships, and to examine them from the perspective of the attacker, defender or the forensic analyst.

5. Content: Model the Trainee’s Environment

By matching the simulated environment with the trainee’s operational environment, the training can be more relevant, in-depth, focused and effective.

6. Portable, Self Contained Laboratory

As mentioned before, pulling a warfighter from his/her primary operational duty for training consumes valuable time and usually undesirable. By making the tool portable and self-contained, it can significantly expand the *reach* of the tool by placing it in the hands of the student and enable training at their convenience.

C. COMPARATIVE REVIEW

With the objectives thus defined by [Tanner 2002], we will now re-examine the three games discussed earlier. Each of the three games described are distinctly different but each has some significant attributes that would contribute towards the objective of education, training and awareness in information assurance. We shall highlight their respective strengths and weaknesses here.

CyberProtect is a turn-based game which provides a macro view of resource management and deployment of abstract defensive countermeasures at a system-level. It attempts to teach some broad concepts such as the need for multiple layers of defenses and introduces the various forms of general attacks, including social engineering which is sometimes not apparent. Correctness of concepts is fairly high, although at a fairly abstract level. The abstraction simplifications do help to make the game easy to learn and play. Unfortunately, the canned environment presented (i.e. the fixed network) offers little replay value and the player will soon run out of concepts to explore.

ISWS is a tutorial based multimedia package which takes a detail look at each of the network-based exploitations. It is clearly focused on teaching the student specific forms of attacks in isolation and the ways to mitigate each of them. The what-if’s option in selecting a package of safeguards affords the student some room for exploration. However, it does not provide a temporal element and the effects of organized multi-faceted attacks. The general lack of interactivity between the student and the game would soon turn it into a typical multimedia presentation. Certainly, it offers the least in terms of replay value.

AIWars is a real-time 3D entertainment game. In this respect, its re-playability is probably the highest and is most likely to captivate the student/player. It clearly carries across numerous concepts although the concepts are significantly dramatized for increased playability and hence requires the student to be able to relate the corresponding metaphors with current realities.

	CyberProtect	**ISWS**	**AIWars**
Format	Educational game.	Tutorial.	Entertainment game.
Understanding of the threat	Yes, but it is not clear what the organizational security policy is?	Yes, but addresses each form of attack only in isolation.	Uncertain.
Awareness of known weaknesses and attack techniques	Yes.	Yes.	Yes.
Connecting concepts to practice	Yes, at a system-level.	Yes, for network-specific attacks.	Yes, at a system-level.
Repeatability	Limited replay value.	Low.	High.
Progression from novice attacks to more sophisticated versions	No.	No.	No.
Examine "what if's"	Limited ability.	Yes, only with respect to a single attack type.	Yes.
Realistic training environment	Yes.	Somewhat.	Generally no, as it uses futuristic connotations.
Develop problem-solving and decision-making skills	Yes.	Yes, but constructs are canned.	Yes.

Learn to recognize operational indicators	Limited.	No.	Yes.
Support for multiple perspectives	No. Only defender's view point is supported.	No. Only defender's view point is supported.	Somewhat, as both attacker and defender perspectives are played.
Model the student's environment	No.	No.	No.
Portable, self-contained laboratory	Yes.	Yes.	Yes.

Table 1 Comparison of products reviewed.

CyberProtect provides a summary score at the end of each quarter and at the end of the game, thus providing a quantifiable measure. Unfortunately, the player is not provided with any indication of the intended security policy. For instance, there is no notion of what resources are being protected, and what the threat is. Consequently, the player is left clueless regarding the appropriate strategy to apply. Thus the score does not provide any obvious relationship to the success of the security measures implemented.

From actual games played, the scores do not appear to be useful for comparative purposes. One can apply very few and poor measures and yet achieve a score of 99, while in another game, despite applying similar measures, a player can score as badly as 60. The wide variance stems from the randomness of the attacks. A player who happens to be subjected to an attack for which a defensive measure happens to be already in place will fare very well. In a separate run with the same measures in place, the player may be faced with a flood of attacks for which the same measures are unfortunately ineffective. In this case, the player will fare very poorly. Hence the resultant score does not serve as a useful measure of effectiveness and no real conclusions can actually be drawn.

ISWS on the other hand, only addresses the security objective from the narrow viewpoint of defending against a single threat. Hence, it does not provide a holistic view at an organizational level.

Both CyberProtect and ISWS are fairly restrictive in terms of options and flexibility. As the game scenarios are fairly static, there is little room for exploration and replay value is limited. CyberProtect for instance, has only a single scenario, although randomized attacks generated do create variety. ISWS has no variations at all and plays in a strictly tutorial-like fashion. Both these games do have a very strong flavor of truism to the concepts being taught and are therefore high in educational value.

AIWars' 3D real-time environment is clearly a step above both CyberProtect and ISWS in terms of gaming playability and hence the potential ability to captivate the player. Replay value is high; however, this is achieved at the sacrifice of realism.

D. CYBERCIEGE

CyberCIEGE, as described in [Irvine 2003], simulates a range of scenarios involving networked computer systems with the player assuming the defender role and the computer assuming the attacker role. The player needs to construct computer networks with components such as servers and workstations, and apply appropriate security measures to ensure that the system's security posture is able to meet the organizational goals. The game lies in the tension created by the competing goals of efficient and affordable access to assets and protection of assets from unauthorized disclosures or modification. This is a significant improvement over that of CyberProtect which had only considered the application of protection mechanisms without clearly articulating the organization goals.

Unlike CyberProtect which has a canned scenario, CyberCIEGE has a wider range of options, allowing the player to construct, interconnect and apply protection of the network as well. The player will make decisions that affect the behavior of a set of virtual user characters. Hostile game characters may develop and attack the system, ranging from vandals, disgruntled insiders, incompetent users, to professional attackers. This offers a much richer set of attacks than when compared to CyberProtect.

[Irvine 2003] puts up a case to illustrate the feasibility of having a computer security game that can be both fun and educational. Indeed, CyberCIEGE shares many of the playability attributes of AIWars. Both cast the player into a real-time 3D game world where the player has to interact with a constantly changing environment. In CyberCIEGE, the player starts the game with a finite budget and has to perform resource management to establish an ever-growing enterprise, reaping the benefit of productive users while balancing benefits of protecting their assets against attackers. The resource management aspects are far more dynamic than that in CyberProtect and are probably closer to that of AIWars.

	CyberCIEGE
Format	Educational and entertainment game.
Understanding of the threat	Yes.
Awareness of known weaknesses and attack techniques	Yes.
Connecting concepts to practice	Yes.
Repeatability	Yes.
Progression from novice attacks to more sophisticated versions	No.
Examine "what if's"	Yes.
Realistic training environment	Yes.
Develop problem-solving and decision-making skills	Yes.
Learn to recognize operational indicators	Yes.
Support for multiple perspectives	No. Only defender's view point is supported.
Model the student's environment	Possibly.
Portable, self-contained laboratory	Yes.

Table 2 Comparison with CyberCIEGE.

E. THE GAPS

Based on its current design, CyberCIEGE matches most of the design requirements as proposed by [Tanner 2002]. However, there are certain areas that require further work. Specifically, these will serve as the requirements of the system that we shall develop in this thesis:

1. Campaign Play

CyberCIEGE improves upon the designs of CyberProtect and ISWS by supporting multiple scenarios as opposed to fairly static scenarios. This improvement can be taken a step further by providing a means for a student to progress from novice to sophisticated scenarios as suggested by [Tanner 2002]. This suggests a need for an instructor tool for ordering a sequence of scenarios that progressively introduces new concepts and complexities to the student or to enable the assembly of a set of related scenarios built around a common theme.

In order to support the concept of a *campaign*, there is a need to introduce further semantics to the current structure of the *scenario definition file*. The scenario definition file defines the goals of the organization and the resource options available to the student. This file is fed in as a input to the CyberCIEGE game to initialize the game and to setup the scenario environment that the student will play in. It would be particularly useful to introduce a taxonomy of classifications to each scenario so that instructors can easily assemble campaigns by stringing together relevant scenarios based on a subject focus, rather than having to scan through each scenario one at a time to determine the scope of each scenario. A tool can therefore be developed to support the definition of a taxonomy tree. The Scenario Builder, who is responsible for constructing these scenarios, would then associate the relevant taxonomy tags to each scenario. The latter requirement for taxonomy tagging will be undertaken by a separate thesis effort that involves the development of the Scenario Definition Tool.

As CyberCIEGE currently plays scenarios independently, we have to provide the student with a tool to play through the scenarios of the campaign. This also suggests a need to define some means to interface with the CyberCIEGE game itself.

2. After-Action Reviews

Although not defined by [Tanner 2002], it is clear that a training package could be significantly enhanced if supported by some means to analyze student progress. This brings forth the idea of After-Action Reviews (AARs). As highlighted in [DOA 1993], AARs helps bridge the gap between concept and practice. Problem-solving skills can be improved through AARs. To achieve this, the instructor requires a tool to reconstruct significant elements of the game so that player progress and decisions can be reviewed.

Through a temporal ordering of transpired events, we would wish to be able to answer these questions as suggested by [DOA 1993]:

a. What happened?

b. Why did it happen?

c. How can performance be improved?

The tool can also help to provide useful metrics of the student's progress, such as the amount of time taken, and to present snapshots of the player's game state at significant stages for review.

This scheme is only possible if there is a means to perform real-time interaction with CyberCIEGE to monitor events, or if event logging is performed. To achieve the objective of keeping CyberCIEGE "portable" and widely available, it is clearly not desirable to unnecessarily encumber it with real-time monitoring from an instructor station. Further, it would be undesirable for the instructor to monitor multiple game sessions (i.e. different students) in such a real-time manner. Hence, the latter approach of event logging for post-game analysis is a better fit for the purpose of AAR. In addition, this would further support the use of CyberCIEGE as a distance learning tool. The event log will permit asynchronous monitoring and student assessment. In order to achieve this, an event logging construct has to be built into the CyberCIEGE game itself. This thesis will therefore define the structure and organization of the event log.

3. Support for Multiple Perspectives

CyberCIEGE is currently designed to enable the player to participate only in the defender's role. To enable the player to participate in attacker or forensic roles as suggested by [Tanner 2002] would require changes to CyberCIEGE itself and hence will not be explored within the scope of this thesis. Futher, [Irvine 2003] has also indicated that this is planned for a future iteration of CyberCIEGE's development.

III. ANALYSIS

A. CONCEPT OF OPERATIONS

We shall next describe the broad concept of operations for the tools to be developed (see III.B for descriptions of the respective *actors*):

The Taxonomy Manager will manage a taxonomy of IA terms. When scenarios are created, this taxonomy of terms are used by the Scenario Builder to catalog the scenarios. This will facilitate convenient downstream retrieval when we need to select scenarios for inclusion into a campaign. The taxonomy tagging of scenarios will not be encompassed within the scope of this thesis.

The Instructor can then proceed to create campaigns, where each campaign is made up of a sequence of previously developed scenarios. Once a campaign definition is completed, it can be released and is ready for use by the Student.

The Student will proceed to launch CyberCIEGE to play each of the scenarios. During game play, CyberCIEGE will generate events which are recorded into event logs. The resultant event logs are then used for analysis by the Instructor.

B. ACTORS

An *actor*, as defined in [Leffingwell 1999], is "someone or something, outside the system, that interacts with the system". This includes users who have a role to play and external systems that are interfaced with. Five such actors are defined. Of these, the Taxonomy Manager, Instructor and Student are users who perform a role, while CyberCIEGE and the Scenario Definition Tool are external systems (see Figure 8).

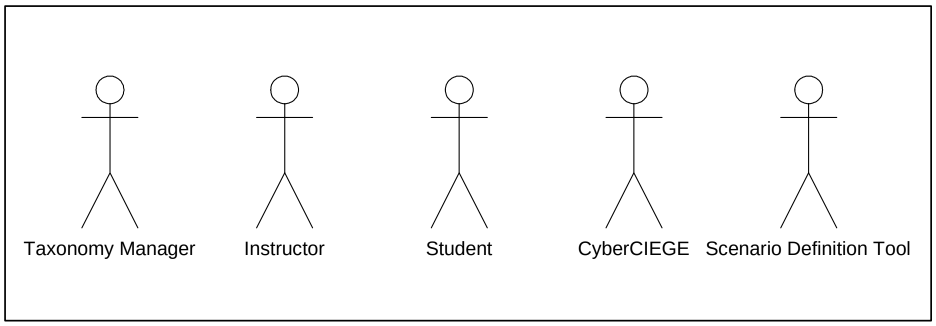


Figure 8. Actor survey.

1. Taxonomy Manager

The Taxonomy Manager is a user responsible for managing the taxonomy library of terms.

2. Instructor

The Instructor is a user responsible for constructing campaigns with a teaching objective in mind and for conducting the AAR with the Student.

3. Student

The Student is responsible for playing through the ordered set of scenarios of a campaign to learn and apply information assurance concepts.

4. CyberCIEGE

CyberCIEGE is the game itself and is treated as an external system.

5. Scenario Definition Tool

The Scenario Definition Tool is an external system used to construct scenario definitions.

C. USE CASES

The Use Case modeling approach of the Unified Process, as described by [Jacobson 1999] and [Leffingwell 1999] is adopted for the development effort.

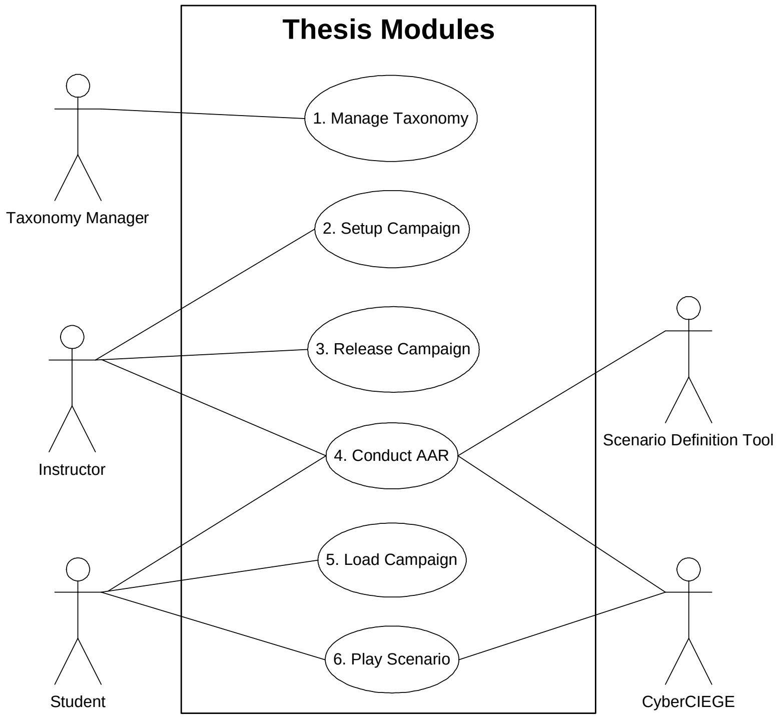


Figure 9. Use case model survey.

The detailed use case specifications are provided in Appendix A. Here we shall briefly summarize and explain the purpose of the use cases defined.

1. [UC.1] Manage Taxonomy

In this use case, The Taxonomy Manager defines and manages the taxonomy of security terms. The terms are created in a hierarchical order and presented as a tree structure.

This is desired in order to establish a common vocabulary for the taxonomy of terms which are used subsequently by people designing scenarios using the Scenario Definition Tool for purposes of cataloging the scenarios, and again in the Campaign Manager to search and retrieve scenarios.

2. [UC.2] Setup Campaign

In this use case, the Instructor either selects an existing campaign or creates a new campaign to work with. The Instructor can then name and describe the objectives of the campaign, and select pre-created scenarios for inclusion in the campaign. A filter can be defined to select specific taxonomy terms, thereby narrowing the set of selectable scenarios to those tagged with the relevant taxonomy terms.

Essentially, this use case has the responsibility for creating and editing the campaign definition.

3. [UC.3] Release Campaign

Once the campaign has been fully defined, this use case supports its release for play.

In this use case, a campaign which is ready for release undergoes an integrity check to finalize the campaign package. This ensures that the referenced Scenario Definition Files are physically present. The campaign is then base-lined and the Campaign Definition File (campaign.xml) and the respective Scenario Definition Files (*.sdf) are copied into a directory defined by the Instructor. The campaign is now ready for the Student to play and is no longer editable.

4. [UC.4] Conduct After-Action Review (AAR)

Once the scenario is started in CyberCIEGE, event logs will begin to be generated. The Instructor can then begin to perform AAR activities. It is not necessary to wait till the scenario has been fully played out to do so.

In this use case, the Instructor selects a campaign to be reviewed by loading the Campaign Definition File. Summary information of all students involved are retrieved from their respective event log files and is displayed. The Instructor is therefore presented with a quick overview of the current status of all his students. The Instructor may find a need to examine further details of a particular Student and can do so by selecting and calling up the event log pertaining to that Student. The Instructor can then analyze the event logs in detail. Where a snapshot of the game "state" has been saved, the Instructor can call it up for analysis and discussion with the Student involved. As the snapshot is implemented as a saved game in CyberCIEGE, we only need to launch CyberCIEGE with the saved game reloaded to restore the game "state".

5. [UC.5] Load Campaign

With the campaign released, each Student can proceed to play the selected scenarios of the campaign.

In this use case, the Student loads the campaign in order to review the objectives of the campaign, and the scenarios to be played.

6. [UC.6] Play Scenario

In this use case, the Student proceeds to play a scenario from the campaign. CyberCIEGE is automatically launched with the scenario loaded at its start state.

The Student will play out the scenario in CyberCIEGE. As the game progresses, CyberCIEGE will automatically log events taking place.

D. INTERFACES WITH CYBERCIEGE (RIVERMIND)

The CyberCIEGE game engine itself is being developed by Rivermind Inc. Interfaces have to be clearly defined to facilitate parallel work. The detailed interface specifications with Rivermind are presented in Appendix C. These include specific requirements for the Scenario Definition File format, the CyberCIEGE command line parameter specification and the Event Log File format.

1. Scenario Definition File Format

When this thesis effort first began, the Scenario Definition File had been partially defined and was still a work-in-progress. It was felt that in order to facilitate the campaign construction; scenarios would need to be given a title. Similarly, in order to have meaningful AAR, the need for game termination conditions were also raised - i.e. to stop the CyberCIEGE game when a win or loss condition is met. Consequently, these were introduced into the Scenario Definition File format.

2. Command-line Parameters

We also need to be able to launch CyberCIEGE under two (subsequently three) situations. Firstly, the Student needs to launch a specific scenario with event logging turned on, in order to play the game. Secondly, the Instructor may also want to do the same in order to examine the start state of a scenario. In this case however, the event logging would not be turned on. A command-line syntax was thus defined to allow these variations:

```
> CyberCIEGE -s <Scenario File> [-i <IDTag> -e <EventLog>]
```

Finally, it was decided that to support the capability to save game states at defined points of a game, this would be implemented by simply having CyberCIEGE to perform a save game function. These defined points would be scripted into the scenario definition as conditions and event triggers. During campaign analysis, we would use the Campaign Analyzer to call up CyberCIEGE with the saved game reloaded. Hence, an additional command-line option was introduced to do this:

```
> CyberCIEGE -l <Saved File>
```

3. Event Logging

Event logging is needed so that we would be able to use the logs to review game progress. The question is, under what circumstance should a log be recorded and how should these be defined? Fixed event loggings would require that all logging points be coded into CyberCIEGE itself. This is clearly inflexible. A scriptable form of logging is preferred so that the Scenario Builder has some latitude in defining under what conditions to trigger a log record, and to specify what should be logged. The syntax and semantics to support this has been included in the Scenario Definition File format.

The Event Log File format was correspondingly defined, and the specification is detailed in Appendix C. The file format is based on XML [W3C 2003] which is an industry-wide open standard. Figure 8 illustrates a sample fragment of the Event Log File:

```
<summaryevent>
        <dtimereal>20030818193015</dtimereal>
        <dtimegame>20030131062359</dtimegame>
        <daily>
                <budget>10000</budget>
        <sales>0</sales>
                <salaries>5000</salaries>
                <hardwareexp>0</hardwareexp>
                <softwareexp>0</softwareexp>
                <misc>0</misc>
                <cash>5000</cash>
        </daily>
</summaryevent>
<controlevent>
        <dtimereal>20030818203015</dtimereal>
        <dtimegame>20030131070100</dtimegame>
        <savetrigger>
                <tagdata>#1</tagdata>
                <filename>1_scenario1_0001.sav</filename>
        </savetrigger>
</controlevent>
<userevent>
        <dtimereal>20030818193015</dtimereal>
        <dtimegame>20030131062359</dtimegame>
        <hire>
                <name>Mr. Gates</name>
                <salary>5000</salary>
        </hire>
</userevent>
<componentevent>
        <dtimereal>20030818193015</dtimereal>
        <dtimegame>20030131062359</dtimegame>
        <buy>
                <catalogname>SuperX 9000 Server</catalogname>
                <componentname>dbserver#1</componentname>
                <cost>3000</cost>
        </buy>
</componentevent>
```

Figure 10. Sample fragment of an Event Log File.

E. INTERFACES WITH SCENARIO EDITOR

The Scenario Editor is being developed by Ken Johns in his thesis. As the Scenario Editor would need to perform the Taxonomy-tagging required by the Campaign Manager module, coordination was required.

1. Taxonomy File Format

The specification of the Taxonomy File format is detailed in Appendix D. The Taxonomy File adopts the XML format to organize the taxonomy terms in a hierarchical order. Shown here is a sample fragment of the Taxonomy File:

```
<simsecuritytaxonomy>
        <tnode>
                <tname>Encryption</tname>
                <tnode>
                        <tname>Public Key Encryption</tname>
                        <tnode>
                                <tname>RSA</tname>
                        </tnode>
                </tnode>
                <tnode>
                        <tname>Symmetric Key Encryption</tname>
                </tnode>
        </tnode>
        <tnode>
                <tname>E-voting</tname>
        </tnode>
</simsecuritytaxonomy>
```

Figure 11. Sample fragment of a Taxonomy File.

2. Embedded Taxonomy Tags in the Scenario Definition File

Because the taxonomy tagging is not an integral part of a scenario definition, it was decided that these internal tagging would not be specified as part of the standard Scenario Definition File format. Instead, we take advantage of the embedded comments construct of the scenario definition file to provide the needed extensibility. The symbol "//", just as in C++ and Java, is used in the Scenario Definition File to denote that all character strings following it up to the end-of-line are part of a comment and are ignored during scenario parsing.

Hence, the taxonomy tags construct would appear with the comments prefix as shown in the following example in the Scenario Definition File, bounded by the

"TaxonomyTag:" and ":end" pair. The text in between would correspond to <tnode>s shown in Figure 11.

```
:
// TaxonomyTag: Public Key Encryption :end
// TaxonomyTag: E-voting :end
:
```

Figure 12. Taxonomy tag embedded in a Scenario Definition File.

IV. DESIGN & IMPLEMENTATION

A. DESIGN

In general, the approach taken was to define a Design Use Case for each Analysis Use Case; hence there is a one-to-one mapping from analysis to design.

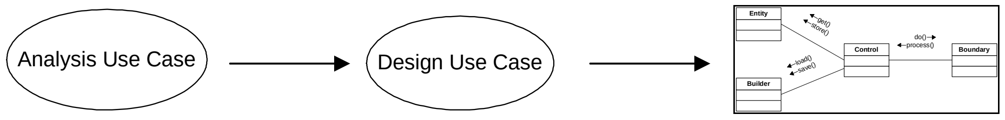


Figure 13. Transitioning from analysis to design.

As described in [Bruegge 2000], an analysis object model consists of entity, boundary and control objects. These stereotypes are defined as such:

- Entity - An entity object represents persistent information tracked by the system.
- Boundary - A boundary object represents the interactions between actors and the system.
- Control - A control object represents the tasks that are performed by the user and supported by the system.

In addition, a Builder object stereotype is introduced. It is an object to interact with a persistent store to perform load and save operations.

This structure for organizing analysis object models was carried over into the design model. Thus, the architectural look-and-feel for each Design Use Case is typically as shown in Figure 14:

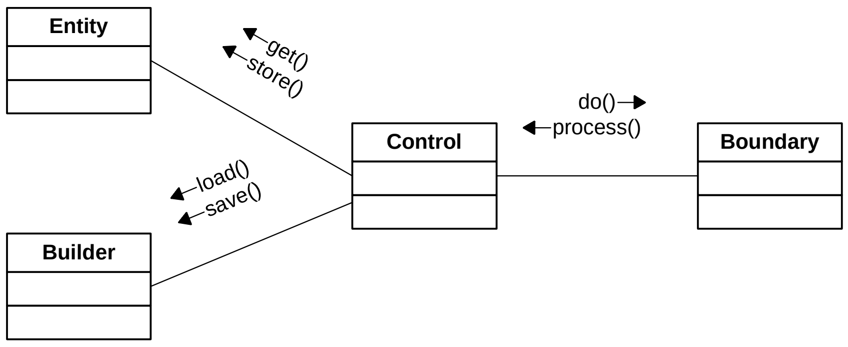


Figure 14. Design model template.

A Control object is defined for each use case. This object handles all the coordination between Graphical User Interface (GUI) objects and entity objects.

GUI objects are created for each user interface that interacts with the actor. Typically, each use case has a main GUI form, supported by some secondary GUIs. GUI objects handle all user interactions, but any processing tasks would be passed onto the Control object to perform.

In sharp contrast, Entity objects are typically passive. These are created for each object whose principle responsibility it is to hold information. In addition, Builder objects were also created to handle the reading and writing of entity objects to the file system as XML files.

The respective Design Use Cases are extensively described in Appendix B, illustrated with Collaboration Diagrams.

B. IMPLEMENTATION

The following table maps the realization of the respective use cases to four software modules developed in this thesis.

Application Modules	**Analysis/Design Use Cases**
TaxonomyManager	[UC.1] Manage Taxonomy
CampaignManager	[UC.2] Setup Campaign, [UC.3] Release Campaign
CampaignPlayer	[UC.5] Load Campaign, [UC.6] Play Campaign
CampaignAnalyzer	[UC.4] Conduct AAR

Table 3 Use case realization.

All the software modules are implemented using the Java 2 SDK v1.4.1 [Sun 2003].

Structurally, the implementation modules are organized into two layers as illustrated in the figure below. The respective application modules (packages) are dependent on the Utilities package. Brief descriptions of the respective code units are provided in Appendix E.

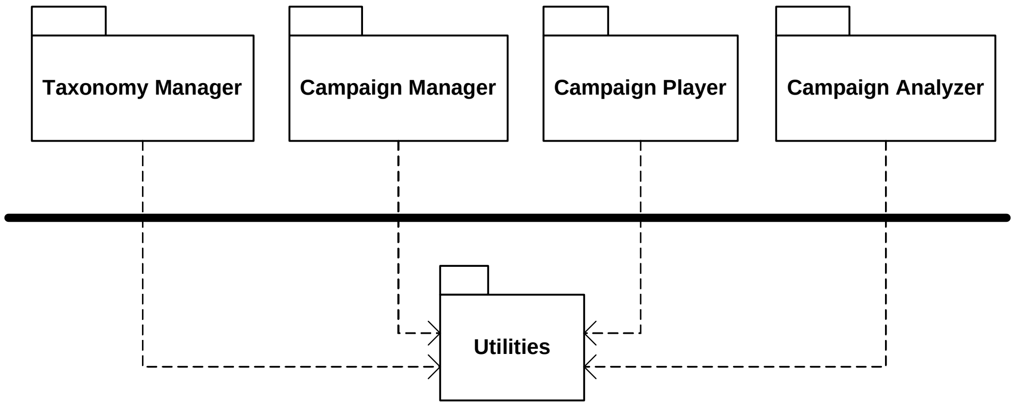


Figure 15. Layered architecture.

The inter-relationship between the modules and the various data files is illustrated in the next block diagram. As shown, the darker rectangle represents the boundary that is within the scope of this thesis.

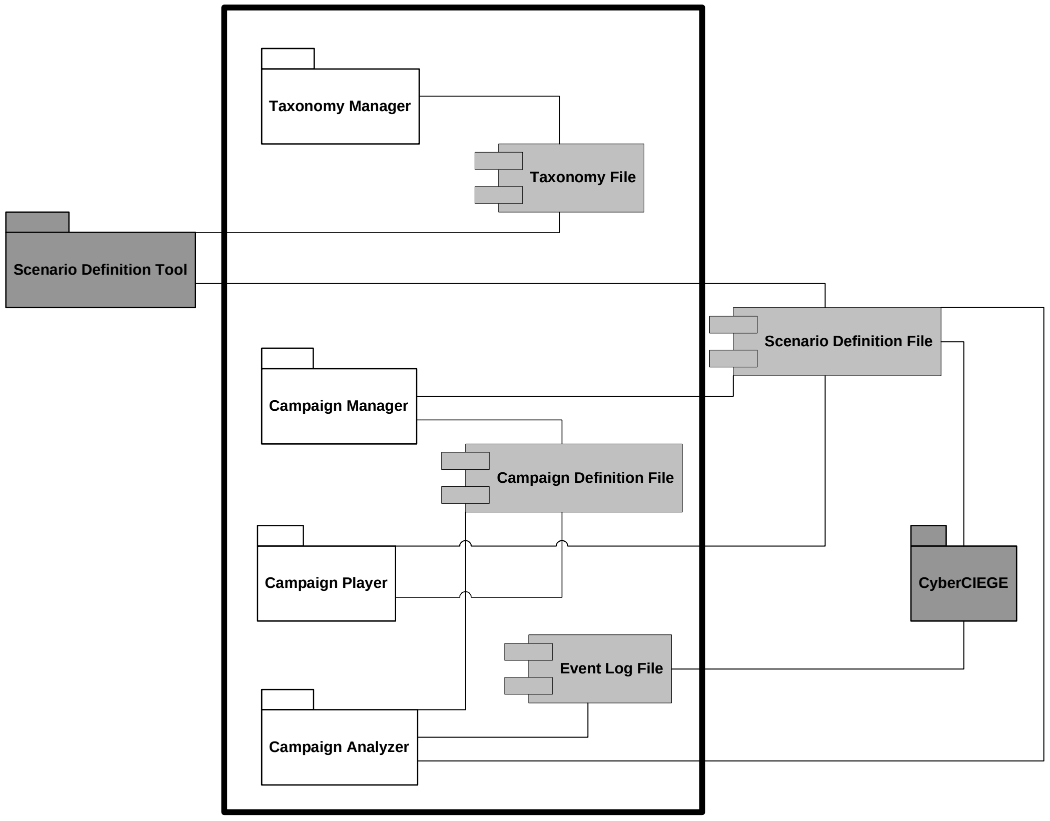


Figure 16. Inter-relationships between the modules and files.

The Taxonomy Manager is responsible for managing the taxonomy terms stored in the Taxonomy File. The later is then used by the Scenario Editor to select Taxonomy terms to be tagged to the Scenario Definition Files that it is creating.

The Campaign Manager processes the respective Scenario Definition Files to obtain a list of scenarios. Specific scenarios are then selected for inclusion into the campaign. Subsequently, the Campaign Manager creates the Campaign Definition File.

The Campaign Player loads the Campaign Definition File and selects a scenario to play. This results in CyberCIEGE being launched using the Scenario Definition File (not shown in figure). Event Log Files are generated by CyberCIEGE.

Finally, the Campaign Analyzer loads a Campaign Definition File and parses the related Scenario Definition Files. Selectively, Event Log Files may be selected to be viewed as well.

C. INTEGRATION

One of the key limitations of this effort is that the CyberCIEGE development is not due for completion till early 2004 while this thesis was to be completed by Dec 2003. In addition, the parallel thesis effort by Ken Johns to develop the Scenario Definition Tool is also not due for completion until Mar 2003.

Due to this mismatch of schedules, stubs were used in lieu of integration testing of the interfaces with CyberCIEGE. Similarly, taxonomy-tagged Scenario Definition Files and the Event Log Files were artificially hand-created to simulate these artifacts.

V. DISCUSSION

A. RESULTS

1. Application Modules

The four application modules have been developed. Shown here are screenshots from the respective modules. This section also serves as a user guide.

a. Taxonomy Manager

In the Taxonomy Manager screen below (Figure 17), we can see the hierarchical structure of the taxonomy terms. The Taxonomy Manager (actor) can add, edit or delete the respective terms. Deleting a parent node will cause all the sub-nodes to be deleted as well.

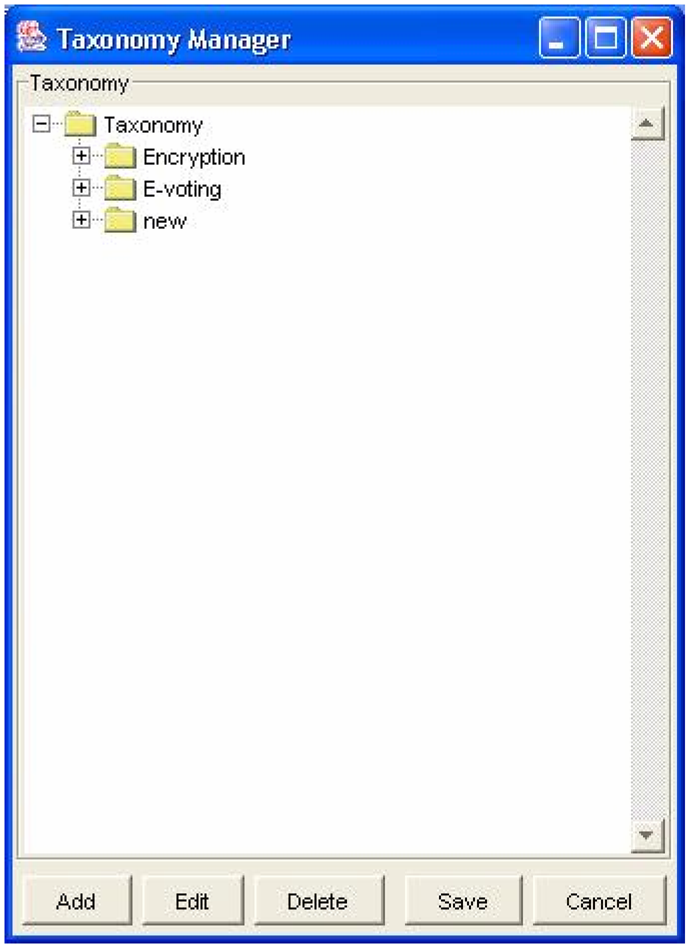
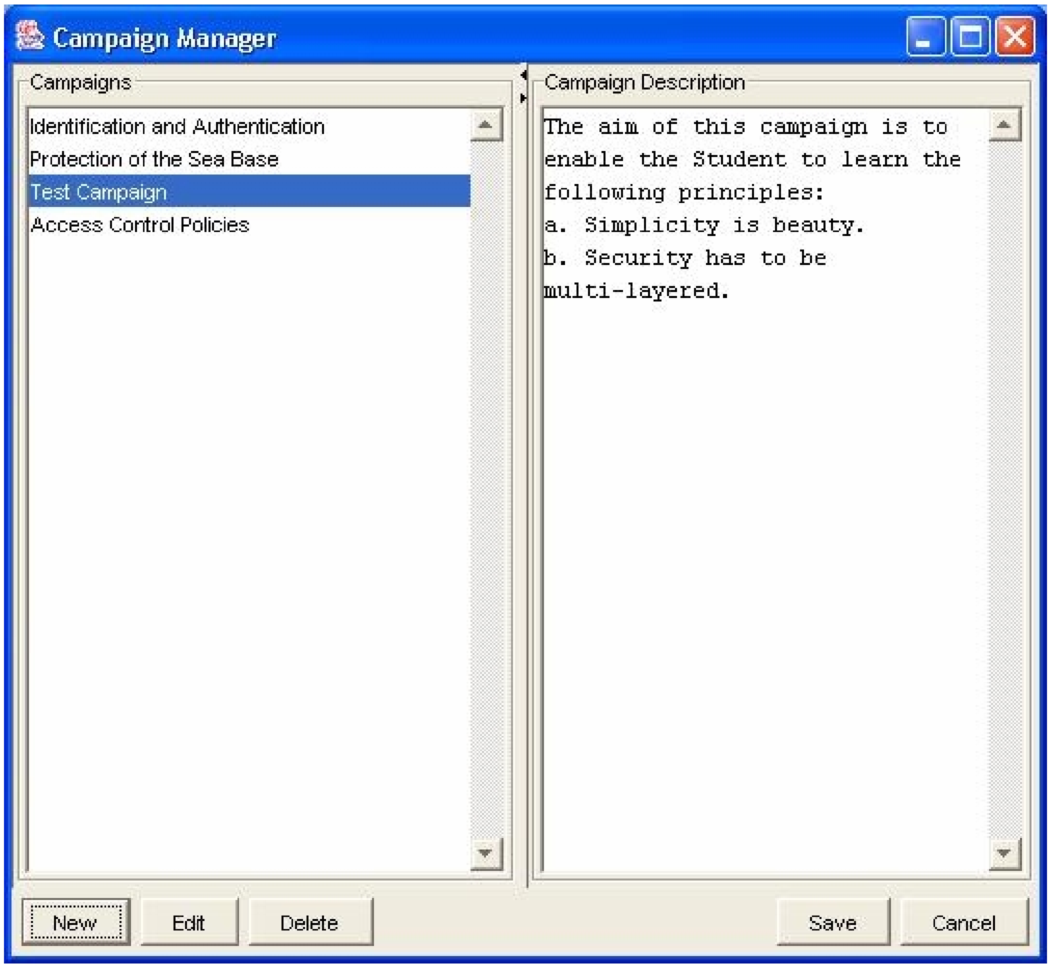


Figure 17. Taxonomy Manager.

Once all the desired changes have been made, the changes can be saved into the persistent store - i.e. as an XML file called “Taxonomy.xml”.

b. Campaign Manager

In the main screen of the Campaign Manager module (Figure 18), the Instructor can review the existing campaigns which have been defined. New campaigns can be created from here, while existing campaigns can be edited or deleted.

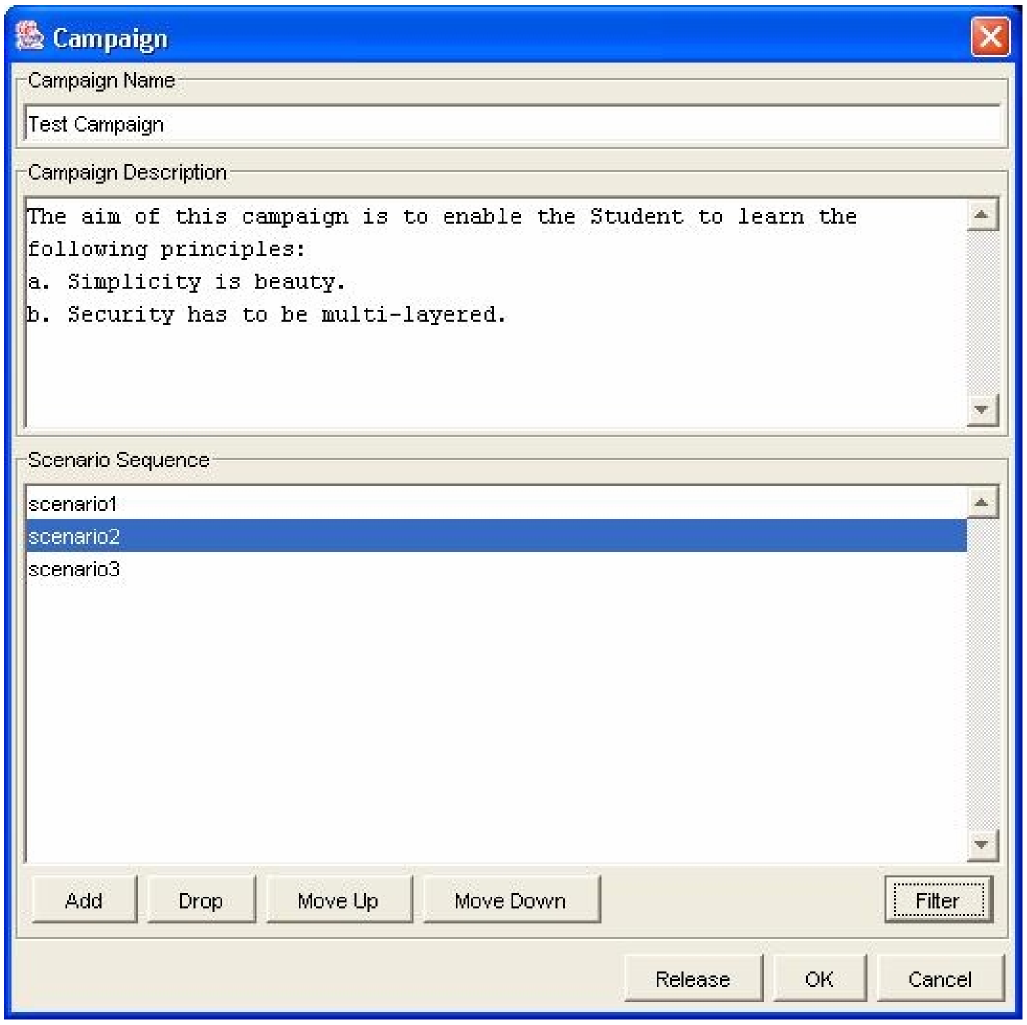


Identification and Authentication
Protection of the Sea Base
Test Campaign
Access Control Policies

```
The aim of this campaign is to
enable the Student to learn the
following principles:
a. Simplicity is beauty.
b. Security has to be
multi-layered.
```

Figure 18. Campaign Manager.

Creating a new campaign or editing an existing one will bring up the Campaign screen as shown in Figure 19.



Test Campaign

The aim of this campaign is to enable the Student to learn the following principles:
a. Simplicity is beauty.
b. Security has to be multi-layered.

scenario1
scenario2
scenario3

Figure 19. Campaign Manager - constructing the campaign.

Here, the Instructor can modify the name and description of the campaign, as well as to select and organize the scenarios. Additional scenarios can be added while existing selections can be dropped. The Instructor can move scenarios up or down, organizing them in the sequence desired.

To narrow the scenarios selectable, the Instructor can call up the filter to define the desired taxonomy. This is especially useful if the campaign is being created around a specific theme. By choosing the relevant taxonomy terms that pertain to this theme, the list of selectable scenarios is reduced to just those scenarios that exhibit one or more of the taxonomy terms selected.

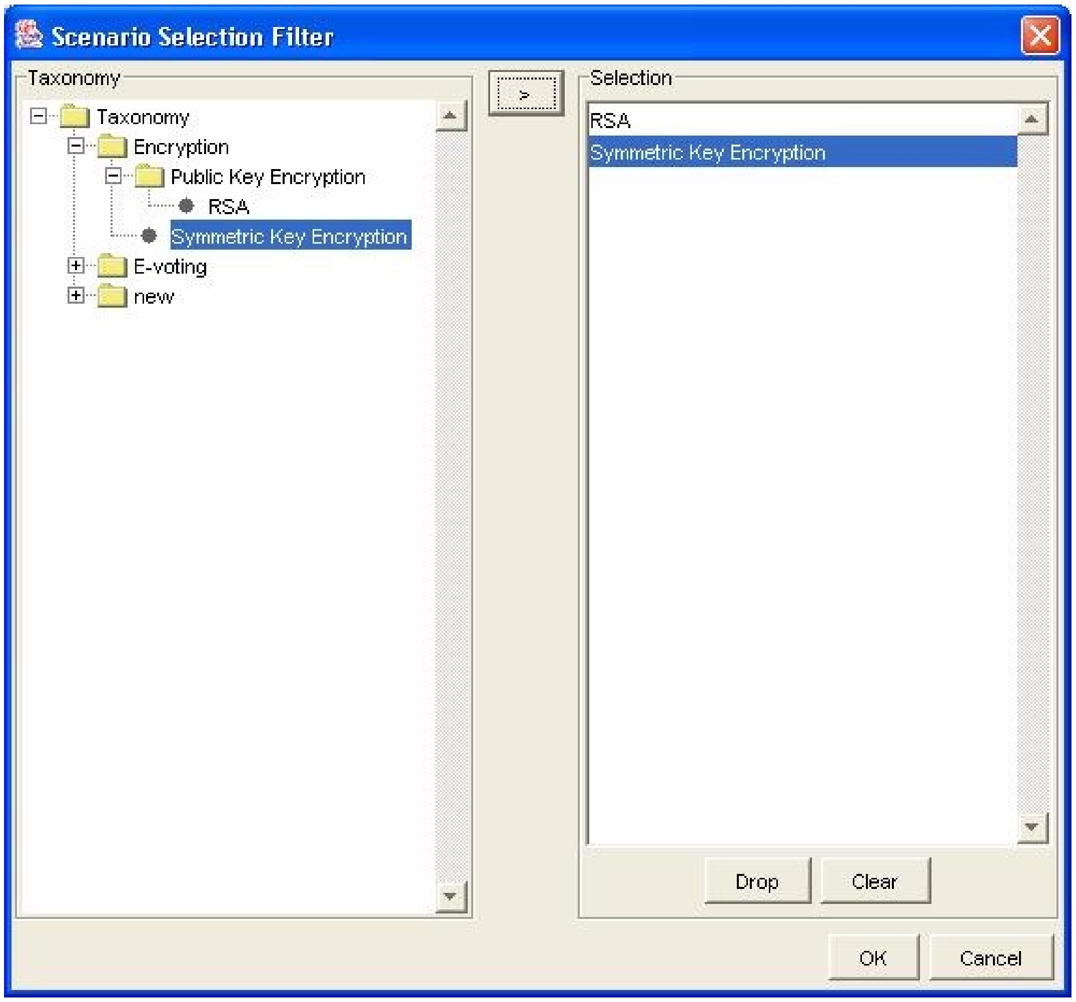


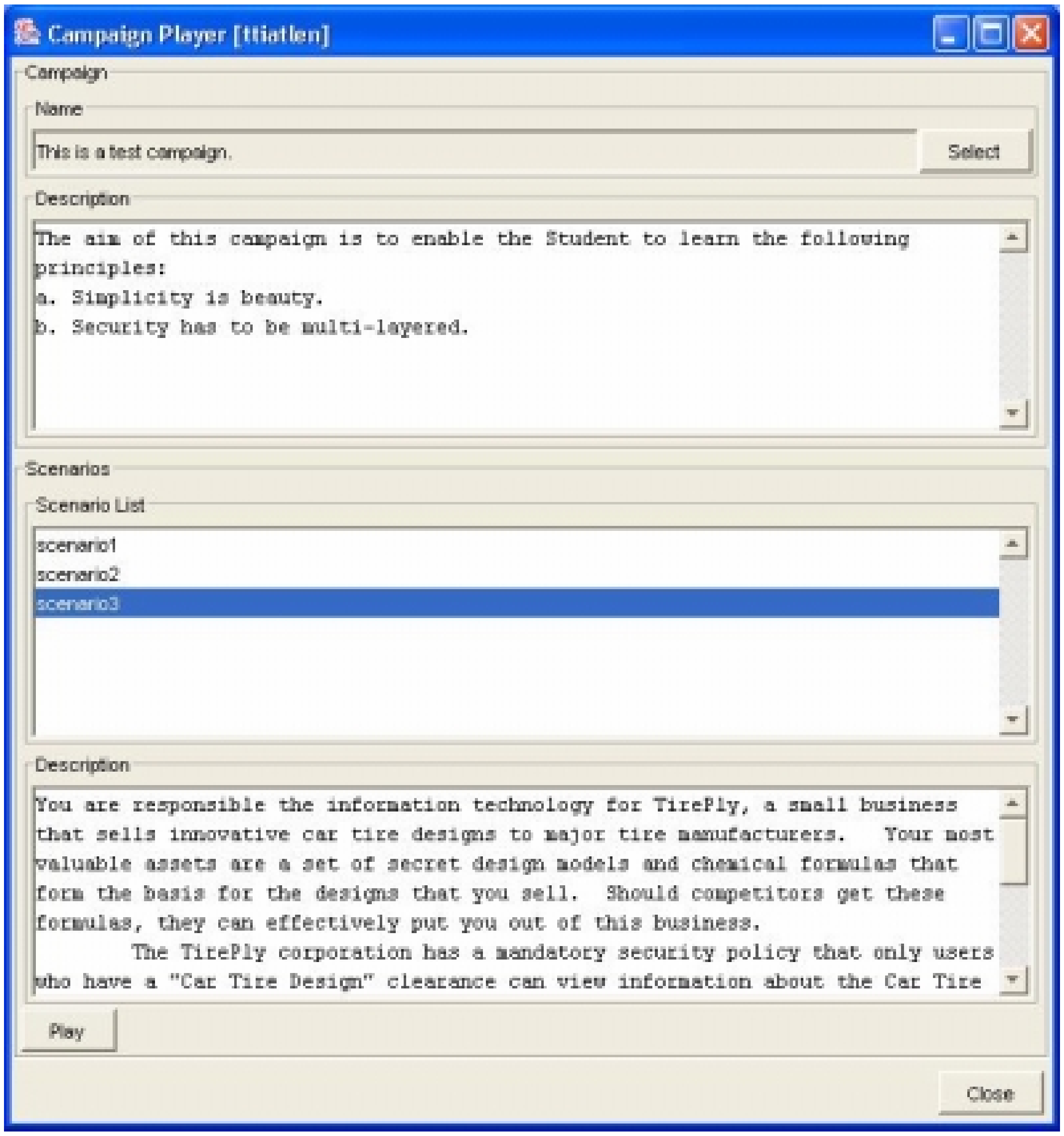
Figure 20. Campaign Manager - taxonomy-based scenario filter.

In the filter screen shown in Figure 20 for instance, if the taxonomy term "Public Key Encryption" is selected, then any scenario which has been tagged with either "Public Key Encryption" or "RSA" would be selectable.

Clearing the filter implies that no taxonomy filtering is to be applied. Therefore, all scenarios would be selectable.

c. Campaign Player

The Student will be primarily using this module to launch scenarios to play. Selecting a campaign will bring up the campaign details as defined by the Instructor. Scrolling through each of the scenarios, the Student can review the description (i.e. briefing notes) of the selected scenario.



This is a test campaign.

The aim of this campaign is to enable the Student to learn the following principles:
a. Simplicity is beauty.
b. Security has to be multi-layered.

scenario1
scenario2
scenario3

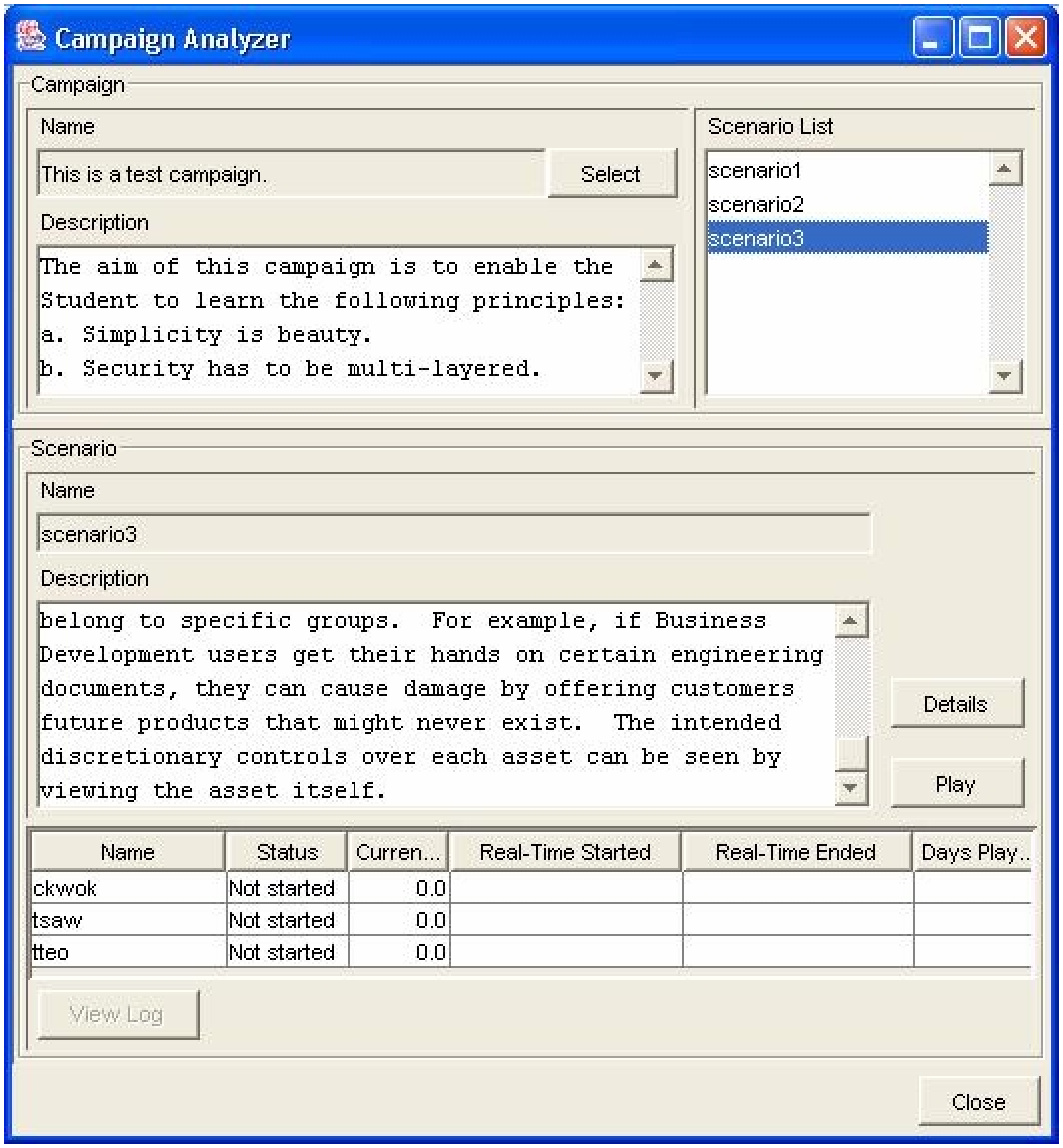
You are responsible the information technology for TirePly, a small business that sells innovative car tire designs to major tire manufacturers. Your most valuable assets are a set of secret design models and chemical formulas that form the basis for the designs that you sell. Should competitors get these formulas, they can effectively put you out of this business.
The TirePly corporation has a mandatory security policy that only users who have a "Car Tire Design" clearance can view information about the Car Tire

Figure 21. Campaign Player.

Once the Student is ready, he/she can play that scenario. This will cause the CyberCIEGE game to be launched with that scenario.

d. Campaign Analyzer

The Instructor uses the Campaign Analyzer to view the progress of the Students. By selecting a campaign, the Campaign Analyzer will call up all the details of the campaign. The module also automatically checks for the status of each Student and summarizes their current status on the display as shown at the bottom of the next screen.



This is a test campaign.

```
The aim of this campaign is to enable the
Student to learn the following principles:
a. Simplicity is beauty.
b. Security has to be multi-layered.
```

scenario1
scenario2
scenario3

scenario3

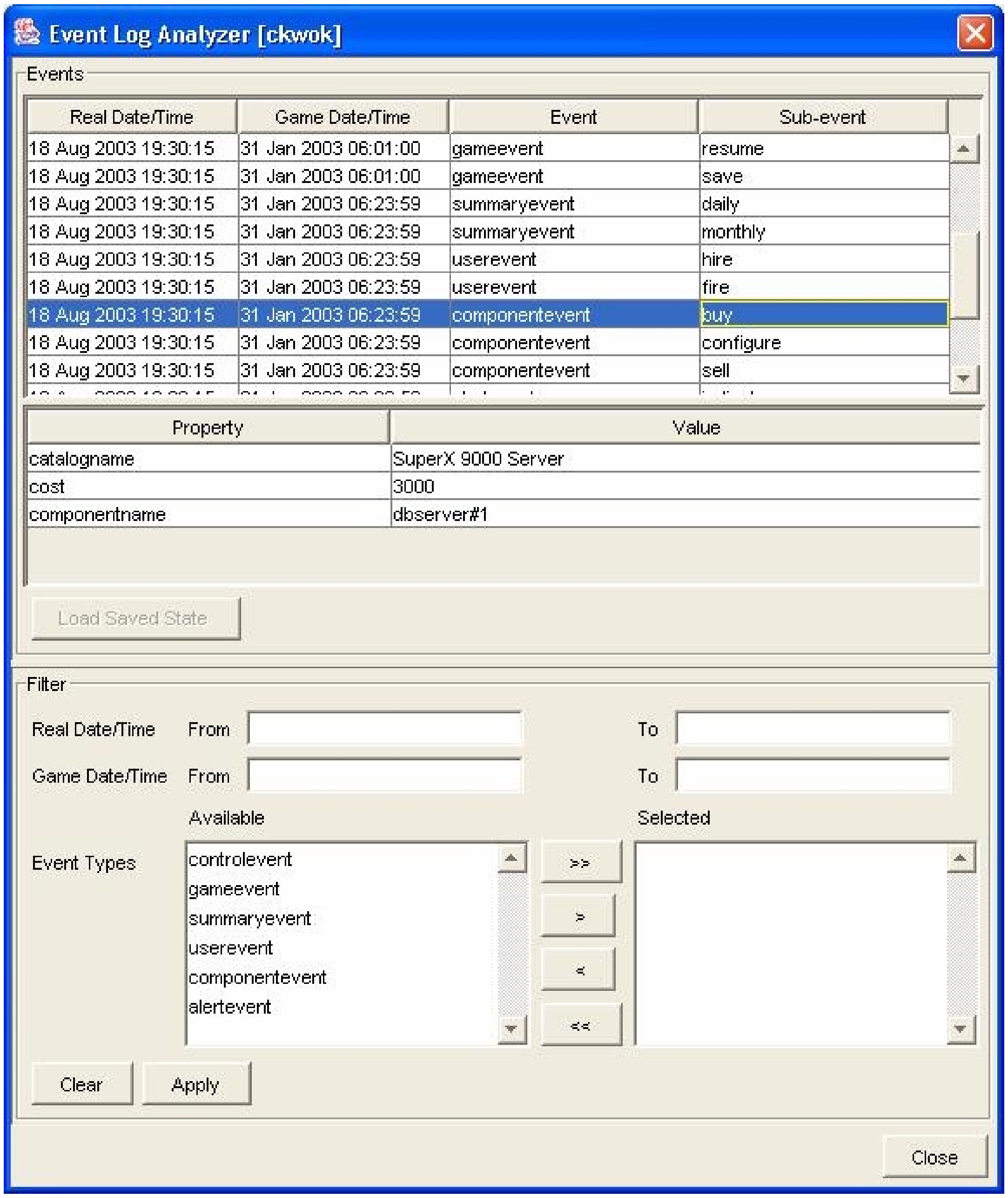
```
belong to specific groups.  For example, if Business
Development users get their hands on certain engineering
documents, they can cause damage by offering customers
future products that might never exist.  The intended
discretionary controls over each asset can be seen by
viewing the asset itself.
```

Name	Status	Curren...	Real-Time Started	Real-Time Ended	Days Play...
ckwok	Not started	0.0			
tsaw	Not started	0.0			
tteo	Not started	0.0			

Figure 22. Campaign Analyzer.

The status summary can provide an indication of the Student's status to the Instructor. The status column indicates whether the Student has started ("Started", "Not started") or completed ("Won", "Lost") the scenario. Other information include the Student's current budget status, the time that the scenario was started and ended, and the total number of days elapsed in terms of real gaming time. At a glance, the Instructor will be able to identify Students who may be way behind schedule on that scenario - e.g. not started yet, or started but has yet to complete. Or the Instructor may notice that a particular Student is taking a lot more time to play than the rest. This may warrant further investigation of that particular Student's progress. To accomplish this, the Instructor can choose to view that Student's logs to find out the cause.

Choosing "View Log" will bring up the Event Log Analyzer (Figure 23). Here, the Instructor can systematically browse through each of the events logged for that Student's game.



Real Date/Time	Game Date/Time	Event	Sub-event
18 Aug 2003 19:30:15	31 Jan 2003 06:01:00	gameevent	resume
18 Aug 2003 19:30:15	31 Jan 2003 06:01:00	gameevent	save
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	summaryevent	daily
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	summaryevent	monthly
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	userevent	hire
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	userevent	fire
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	componentevent	buy
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	componentevent	configure
18 Aug 2003 19:30:15	31 Jan 2003 06:23:59	componentevent	sell

Property	Value
catalogname	SuperX 9000 Server
cost	3000
componentname	dbserver#1

controlevent
gameevent
summaryevent
userevent
componentevent
alertevent

Figure 23. Event Log Analyzer.

For the scope of this thesis, only a partial set of events are processed and presented. These are shown in Table 4:

EventType	Event Sub-Type	Properties
controlevent	debuglog	tagdata, message
	logtrigger	tagdata, message
	popuptrigger	tagdata, message
	tickertrigger	tagdata, message
	savetrigger	tagdata, filename
gameevent	start	
	end	"win" or "lose"
	pause	
	resume	
	save	
	exit	
	quit	
summaryevent	daily	budget, status, salaries, hardwareexp, softwareexp, misc, cash
	monthly	budget, status, salaries, hardwareexp, softwareexp, misc, cash
userevent	hire	name, salary
	fire	name, salary
componentevent	buy	catalogname, componentname, cost
	sell	catalogname, componentname, cost
	configure	
alertevent	indicator	securitytarget, targetname, message
	attack	securitytarget, targetname, attacktype, result

Table 4 Implemented set of events.

Where a game saved state has been recorded, the Instructor can review that saved state. Effectively, this will bring up the CyberCIEGE game, loading the saved game file. The Instructor can then examine the state of the Student's game within CyberCIEGE itself to gain insights.

2. Interface Specifications

Interface specifications are essential to ensure that different developers are able to work in parallel loosely-coupled, whilst developing modules that can be tightly-integrated.

Various interface specifications were defined in the course of this thesis research. These are described in more details in Appendices B and C. This has enabled the respective parties (i.e. Rivermind and Ken Johns) to proceed with their development efforts in parallel. Their respective efforts are not due for completion until much later downstream.

3. Influences on the Design of CyberCIEGE

During the early and mid stages of this thesis research, NPS and Rivermind were engaged in active discussions defining the Scenario Definition Format. In these early stages, the focus of the discussions was generally centered on what the Scenario Builder needed in order to present CyberCIEGE with the desired scenario. This involved numerous meeting iterations defining the various elements of the game; such as the components, and the security and configuration attributes.

As this thesis effort was focused on developing tool support for the Instructor, it helped to bring a new perspective to these discussions. In particular, we had to examine how an Instructor interacts with CyberCIEGE in various ways. For instance, to effect some form of "control" of the CyberCIEGE game itself, command-line parameter constructs were defined to launch the CyberCIEGE game to do different things. Conditions and triggers had to be defined within the Scenario Definition File itself to enable the Scenario Builder to influence the game as it progresses. And finally, to extract meaningful data from a game in progress which can be used to analyze the Student's progress. These helped to ensure the *completeness* of the Scenario Definition Format specification effort and serves as a form of *validation* check of CyberCIEGE's design.

In particular, we note that in addition to presenting the scenario to the game engine, the scenario definition also has to provide scripted control of the game so that an

Instructor is able to conduct post-game analysis (or after-action reviews).

In the same light, there are also various thesis research initiatives being undertaken, involving the development of educational scenarios. To facilitate their research efforts, it would be necessary for the CyberCIEGE game to be able to produce results which can be analyzed. The event logging mechanism would be a useful means to support this.

B. THE QUESTIONS ANSWERED

This thesis was initiated to examine three key questions. We shall now discuss these.

The first question was, "How do we construct and organize scenarios to provide an educational focus on an IA-specific topic?" What we have done has been to demonstrate the viability of layering the notion of a *campaign* on top of the *scenario* to achieve this. By organizing a set of scenarios into a campaign and having the student play through the scenarios, we enable the student to step through progressive scenarios, learning concepts one step at a time.

In order to facilitate campaign construction, we have further identified that it is beneficial to support a notion of *taxonomy-tagging* of scenarios. As this function is optimally best performed by the Scenario Builder, this has also resulted in additional requirements for the Scenario Editor.

The second question was, "What do we need in order to perform student assessment of the scenarios played?" As has been extensively discussed, this involved the introduction of conditions and triggers into the scenario design, and to have the CyberCIEGE game to perform event logging. Parsing the event logs, we can then present the necessary information to the Instructor to perform analysis.

Finally, the last question was, "What interfaces must be introduced into CyberCIEGE to achieve this?" This is answered through the interface specifications defined in Appendix C.

VI. CONCLUSION

A. CONCLUSION

With the concept and the tools developed from this thesis research, it is now possible to make the CyberCIEGE game into a more complete instructional package. By layering the idea of a campaign on top of the scenarios, we have provided a way for organizing progressive and/or focused training packages. The tools provides for the construction and support for measurable assessment of student performance of the given scenarios to assess learning progress.

It has also served to enable the Scenario Definition File format to be more comprehensively defined, thereby enabling the necessary control and interaction with the CyberCIEGE game to support educational objectives.

As a more complete package, the CyberCIEGE effort is a step closer towards serving as an educational tool to promote greater cyber-defense awareness and understanding.

B. FURTHER WORK

As an initial effort in this area, there is certainly a lot more room for improvement. The following are some suggested areas for further research.

1. Improved Details & Usability

The graphical interfaces of the tools could be further improved to provide more information and improved usability. Presently, to demonstrate its viability, only critical elements are presented in the user interfaces. There may be more information that the Instructor would find useful when constructing the campaigns. For example, it would be useful during campaign creation if the Instructor were able to call up the Scenario Definition Tool to examine the details of a specific scenario, to obtain a better understanding of the scenario and to make selection decisions. In the scenario selection screen (Figure 19), it may be better to display the filter together with the list of scenarios.

2. Taxonomy of Terms

It may be worth examining how best to define a taxonomy of terms so that its usefulness to the Instructor is maximized to support scenario selection.

Presently, the structure of the taxonomy of terms allows the Taxonomy Manager to define the same term under more than one sub-tree. When the Instructor is constructing a campaign, the filter will enable selection of scenarios which have that taxonomy term defined, regardless of the sub-tree from which that term was actually selected. An improvement to this would be to fully qualify the taxonomy term. For example, “Cryptography” defined as “Confidentiality:Cryptography” would be differentiated from “Authentication:Cryptography”.

3. Event Log Analysis

The present tool for event log analysis only displays the logged events in a tabular format for the Instructor to review. More improvements could be made in this area to provide a graphically-based time-line display of the events and other analysis tools.

The current implementation has only incorporated a subset of the events being logged. The event log format is also likely to be expanded in the future, incorporating more event types. Consequently, the implementation would also need to be enhanced to provide commensurate support.

APPENDIX A. USE CASE SPECIFICATIONS

A. PURPOSE

This appendix documents the Analysis Use Case Specification, describing each Use Case in detail.

B. USE CASES

1. [UC.1] Manage Taxonomy

Brief Description:

The Taxonomy Manager defines and manages a taxonomy of security terms.

Primary Actor(s):

Taxonomy Manager.

Secondary Actor(s):

Nil.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with the Taxonomy Manager loading the taxonomy of security terms. The taxonomy is expressed as a hierarchy of security terms (Figure 24). The Taxonomy Manager can then proceed to update the hierarchy by performing any of steps 2 to 4 iteratively.

2. The Taxonomy Manager can add a new term to the hierarchy.

3. The Taxonomy Manager can select an existing term and delete it. This will also delete any terms which are sub-nodes of the hierarchy.

4. The Taxonomy Manager can edit an existing term by renaming it.

5. At any point after completing any of steps 2 to 4, the Taxonomy Manager can save the updated hierarchy.

6. Alternatively, the Taxonomy Manager could discard all the changes made and revert to the last saved version.

7. When the Taxonomy Manager is done, he can close the taxonomy editor and end the taxonomy editing. However, if there has been a change since the last save, the Taxonomy Manager is prompted to save or discard the changes.

8. The use case then ends.

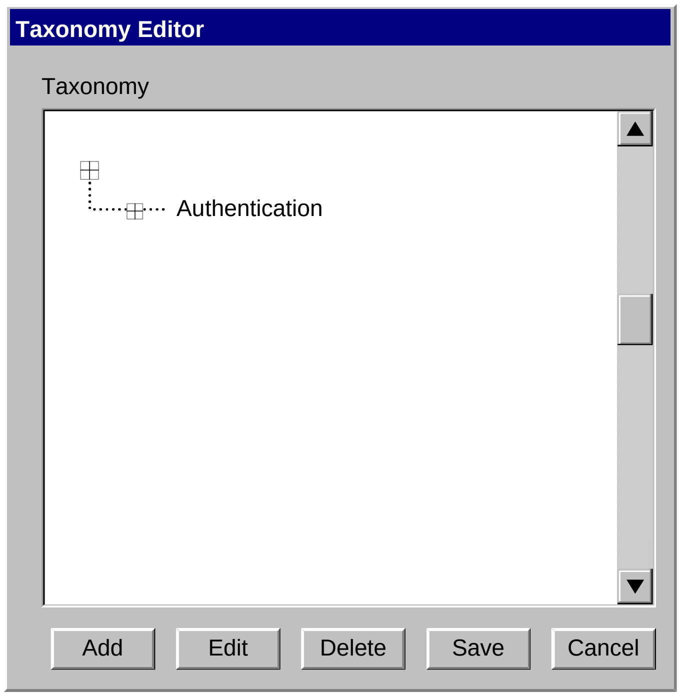


Figure 24. Taxonomy Editor.

<u>Alternate Flows</u>

Nil.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

2. [UC.2] Setup Campaign

Brief Description:

The Instructor sets up a campaign by describing the objectives of the campaign and by composing a collection of related scenarios.

Primary Actor(s):

Instructor.

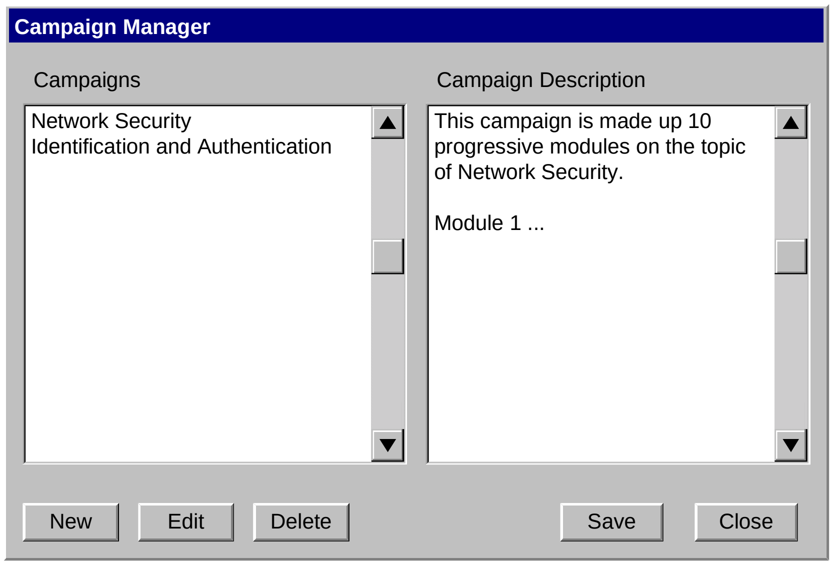
Secondary Actor(s):

Nil.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with the Instructor calling up the Campaign Manager and selecting a campaign to work on or create a new one if desired.



Network Security
Identification and Authentication

This campaign is made up 10 progressive modules on the topic of Network Security.

Module 1 ...

Figure 25. Campaign Manager.

2. When creating a new campaign, the Instructor will assign a name and a description to the campaign. The Instructor can also rename or modify the description of an existing campaign.

3. With the campaign defined, the Instructor can proceed to select scenarios to be added into it. He can do this by selecting a scenario from the available list of scenarios.

4. Scenario selection can be accelerated by filtering the scenarios based on taxonomy tags.

5. Once a scenario has been added to the campaign, the scenario sequence can be rearranged.

6. The Instructor can then save or discard the campaign.

7. The Instructor can also delete a campaign.

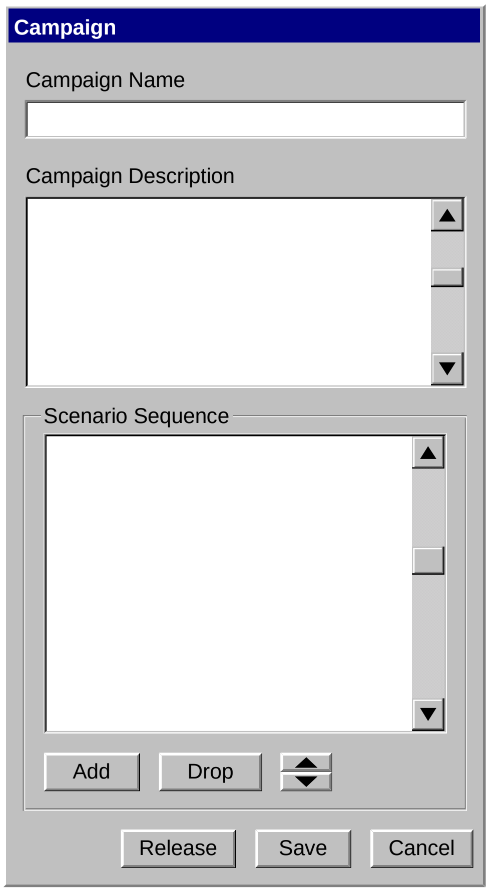


Figure 26. Campaign editor.

<u>Alternate Flows</u>

Nil.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

3. [UC.3] Release Campaign

Brief Description:

A campaign which is ready for release undergoes an integrity check to finalize the campaign package. The campaign is then base-lined and ready for the

Student to play.

Primary Actor(s):

Instructor.

Secondary Actor(s):

Nil.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with the Instructor selecting a campaign.

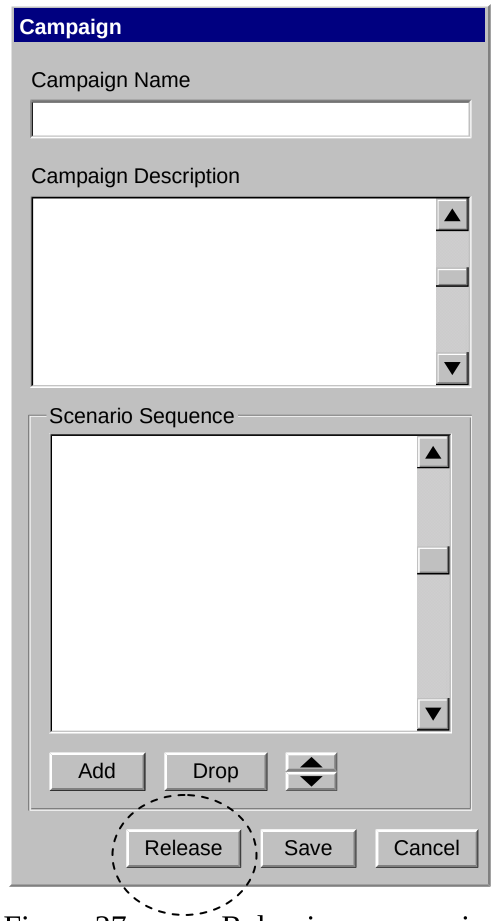


Figure 27. Releasing a campaign.

2. The Instructor can then "release" it for Students to play it. This will result in an integrity check to ensure that all the scenarios of the campaign are consistent - i.e. the corresponding scenario definition files are physically present.

3. Once the checks are successful, the campaign can be accessed by the Student for playing.

4. If there are any errors, the Instructor is informed of the sources of errors. The Instructor updates the campaign definition as necessary to fix the problem. For instance, the scenario file itself may be missing and hence need to be removed.

5. The use case ends when the Instructor completes the release of the campaign, or abandons the attempt to release the campaign.

Alternate Flows

Nil.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Campaign is now released for Student to play.

4. [UC.4] Conduct After-Action Review

Brief Description:

The Instructor conducts an after-action review of the Students' performance by analyzing the event logs.

Primary Actor(s):

Instructor.

Secondary Actor(s):

Student, CyberCIEGE, Scenario Definition Tool.

Flow of Events

Basic Flow

1. The use case begins when the Instructor selects a campaign to be analyzed. This may be conducted together with the Student.

2. The campaign details are loaded, including the scenarios defined for the campaign. The first scenario is selected by default.

3. When a scenario is selected, the display will show each Student's status.

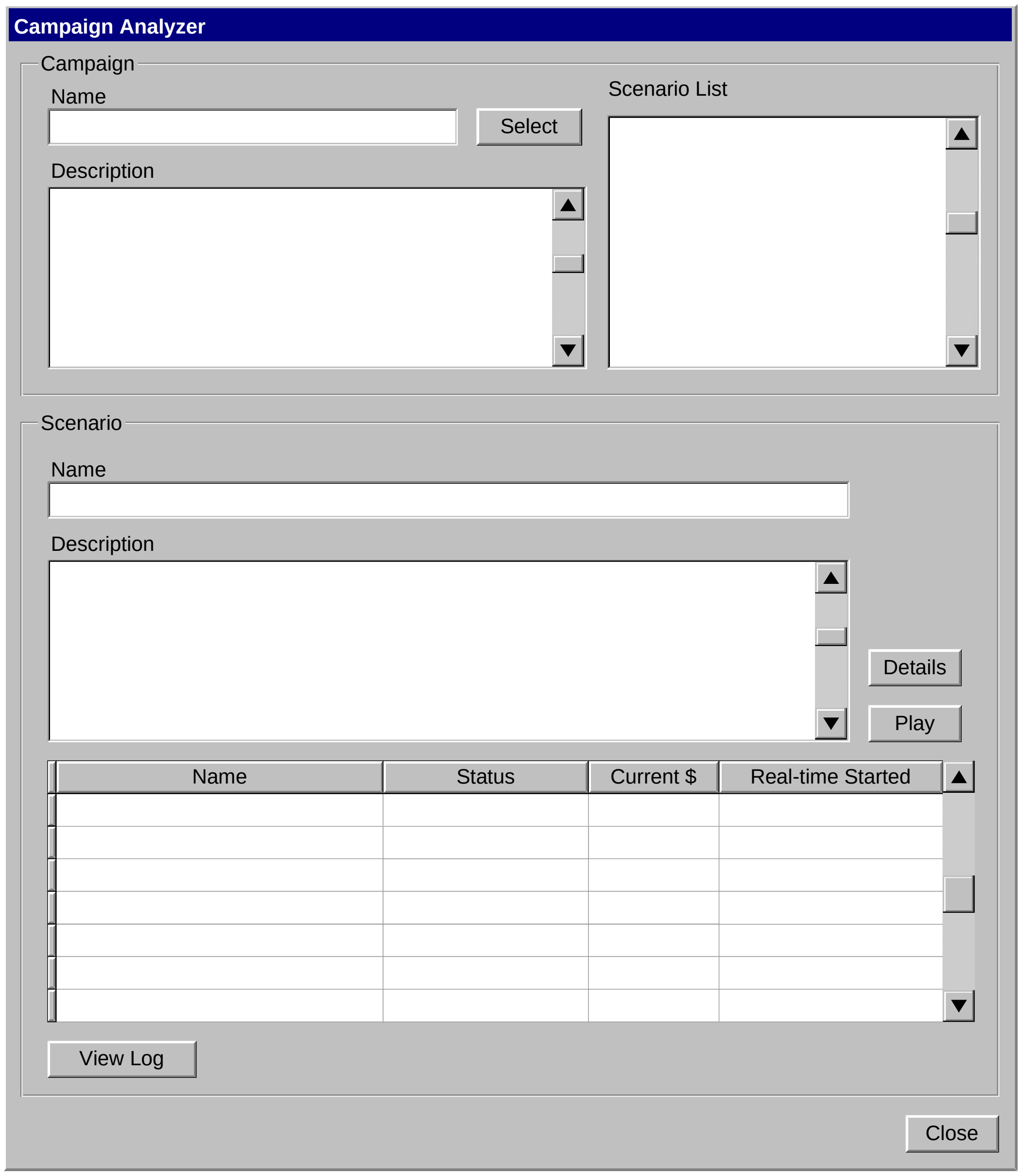


Figure 28. Campaign Analyzer.

4. If the Instructor so chooses, he/she can select to view a Student's event logs to analyze further.

5. The event log is then loaded, and its details displayed (Figure 29). This displays the events sorted by default in real date/time order.

6. As an event is selected, the corresponding details (i.e. event property and value) for that event are displayed.

7. If the event is a saved game state, the Instructor can load the saved state for further analysis. This will actually load CyberCIEGE with the associated saved game file.

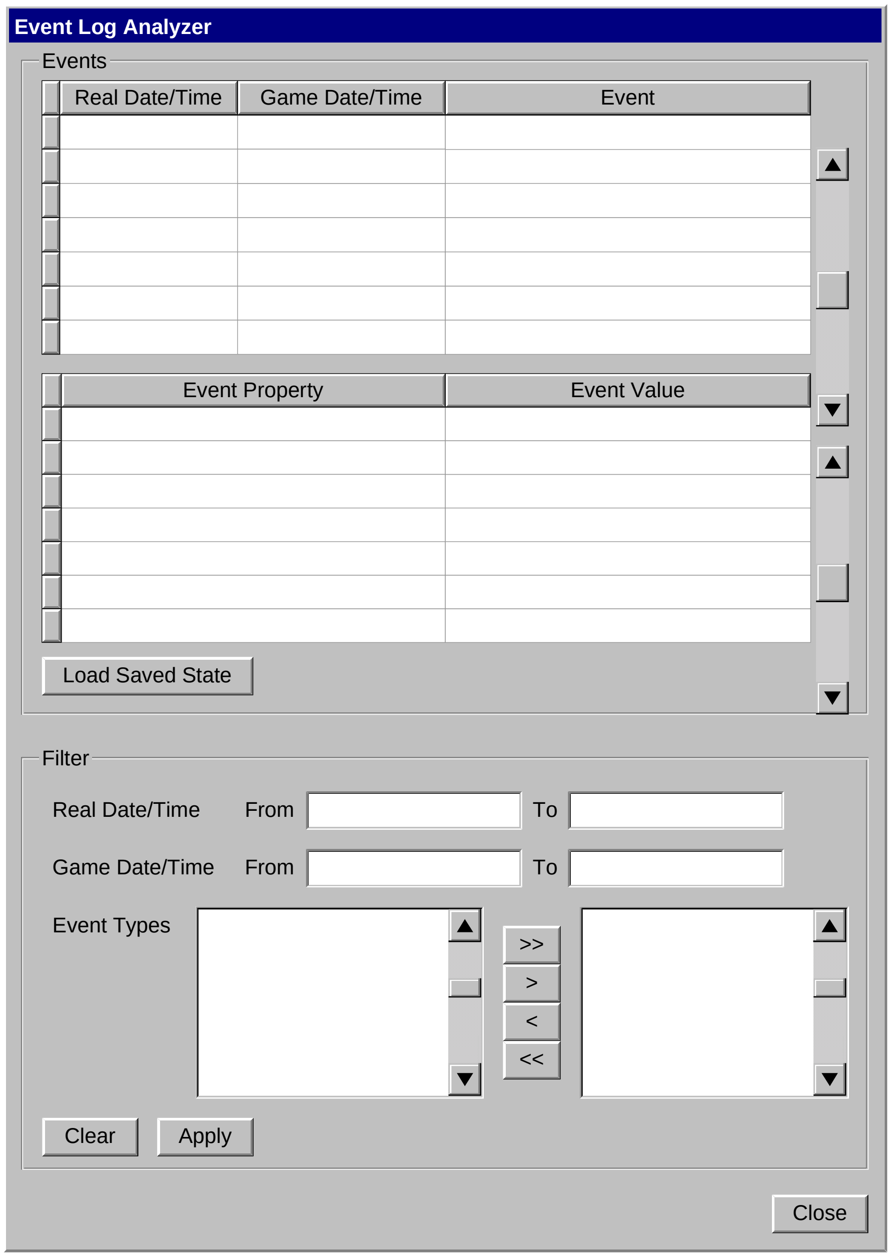


Figure 29. Event Log Analyzer.

8. The Instructor can apply filter conditions to filter the events being displayed. Filter options include selection of a date/time window in terms of the real-time and/or game-time, and by event types.

9. The Instructor can clear any existing filter options and redefine a new one.

10. Once all the filter options have been set, the Instructor can apply the filter. This will filter those events that meet the filter options to be displayed on the events table.

11. The use case ends when the Instructor is done analyzing the campaign and related scenarios.

Alternate Flows

Nil.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

5. [UC.5] Load Campaign

Brief Description:

The Student loads the campaign to review the objectives of the campaign, and the scenarios to be played.

Primary Actor(s):

Student.

Secondary Actor(s):

Nil.

Flow of Events

Basic Flow

1. The use case begins with the Student calling up the Campaign Player module (Figure 30). The Student's identity is automatically determined from the operating system.

2. The Student then selects a campaign definition file.

3. This loads the campaign and its details such as the campaign name and campaign description. The sequence of scenario names involved is also displayed.

4. The Student can repeatedly review each scenario by selecting the scenario.

5. As a scenario is selected, the scenario name and its description are displayed.

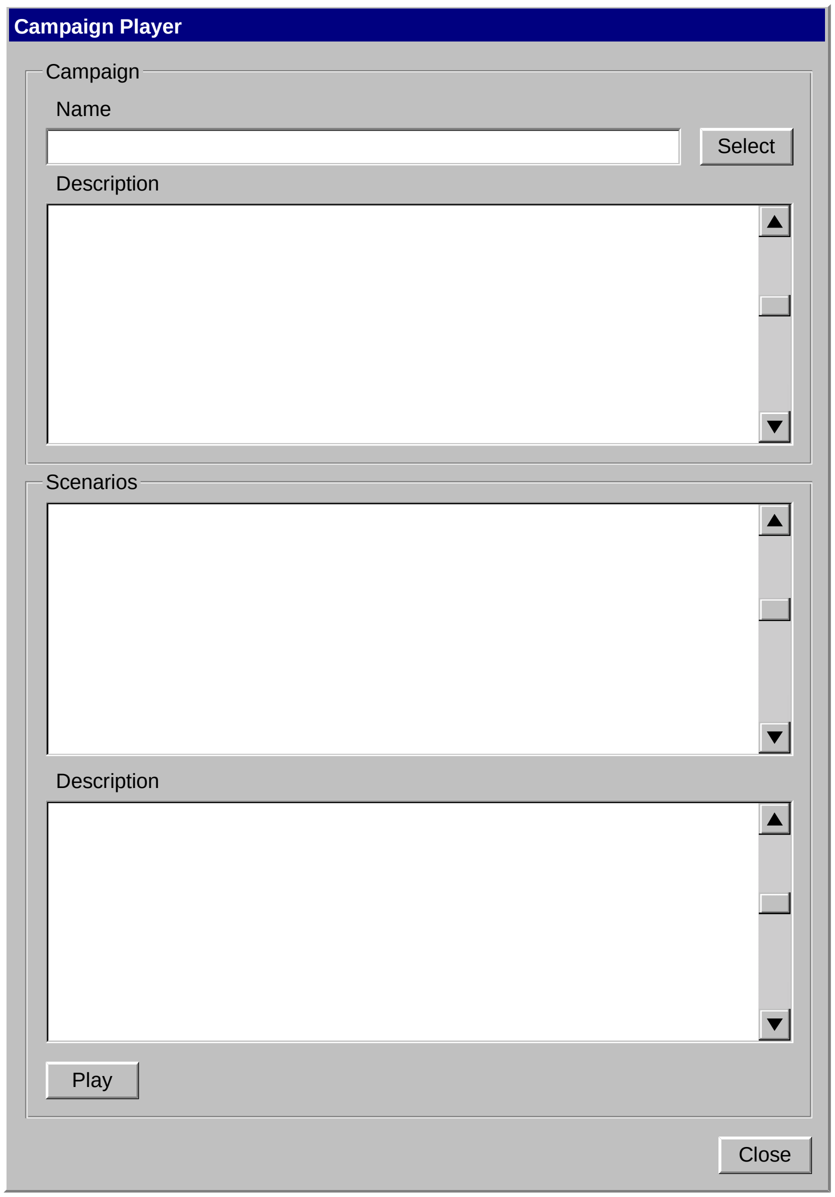


Figure 30. Campaign Player.

6. The use case ends when the Student has finished reviewing the campaign and the scenario descriptions.

Alternate Flows

Nil.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

6. [UC.6] Play Scenario

Brief Description:

The Student proceeds to play a scenario of the campaign.

Primary Actor(s):

Student.

Secondary Actor(s):

CyberCIEGE.

Flow of Events

Basic Flow

1. The use case begins with the Student having selected a scenario of the campaign.

2. The Student can decide to play the currently selected scenario.

3 This will result in CyberCIEGE being launched to play that scenario.

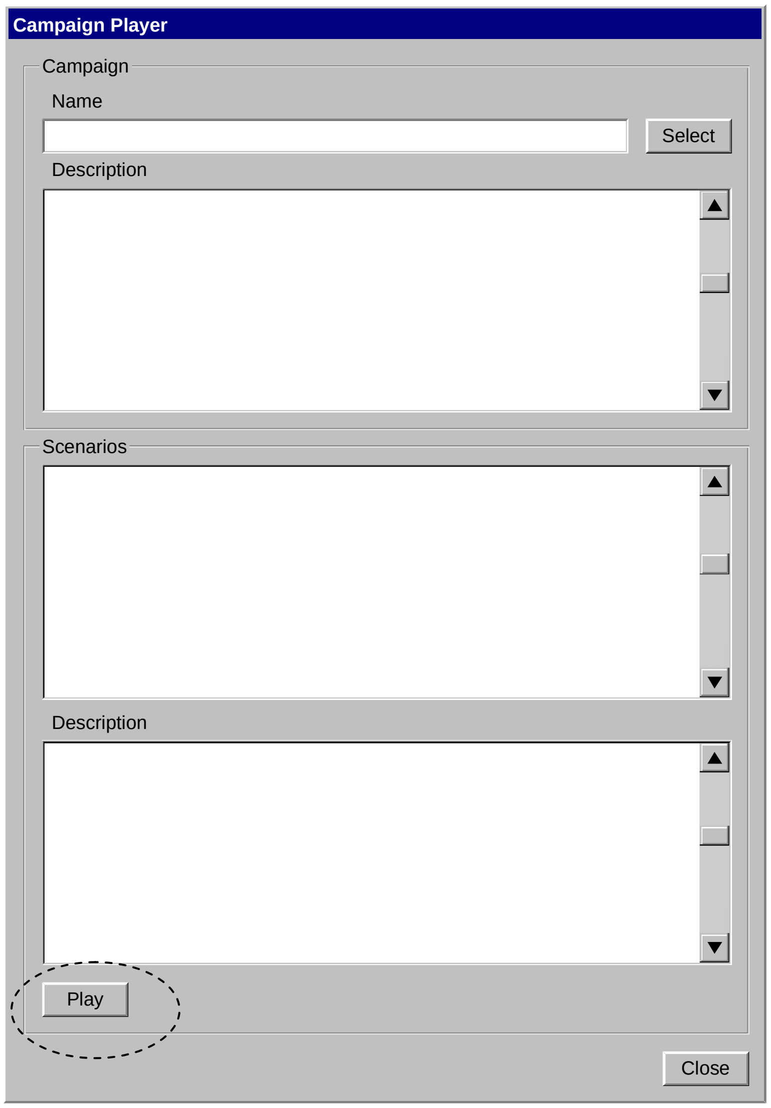


Figure 31. Playing a scenario.

Alternate Flows

Nil.

Special Requirements

Nil.

Pre-conditions

UC.5 must have been performed prior, to load a campaign to work on.

Post-conditions

Nil.

APPENDIX B. DETAILED DESIGN

A. PURPOSE

The purpose of this appendix is to document the detailed design of the application modules. These are described using Design Use Cases, supported by collaboration diagrams.

B. DYNAMIC VIEW - USE CASE DESIGN

For the description of the design use cases, the convention adopted is to use bold font for classes/objects and the method calls. Methods are suffixed with parenthesis but parameters are not presented. GUI interactions are typically not described in details, except for key interactions that significantly affect the use case.

1. [UC.1] Manage Taxonomy

Brief Description:

The Taxonomy Manager (actor) defines and manages the taxonomy of security terms.

Primary Actor(s):

Taxonomy Manager.

Secondary Actor(s):

Nil.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with **TaxonomyManager** creating a **new**() instance of **TaxonomyCtl**.

2. **TaxonomyCtl** in turn creates a **new**() instance of TaxonomyBuilder. TaxonomyBuilder is responsible for loading the Taxonomy hierarchy.

3. **TaxonomyCtl** then retrieves the root Taxonomy object by doing **getTaxonomyRoot**() from **TaxonomyBuilder**.

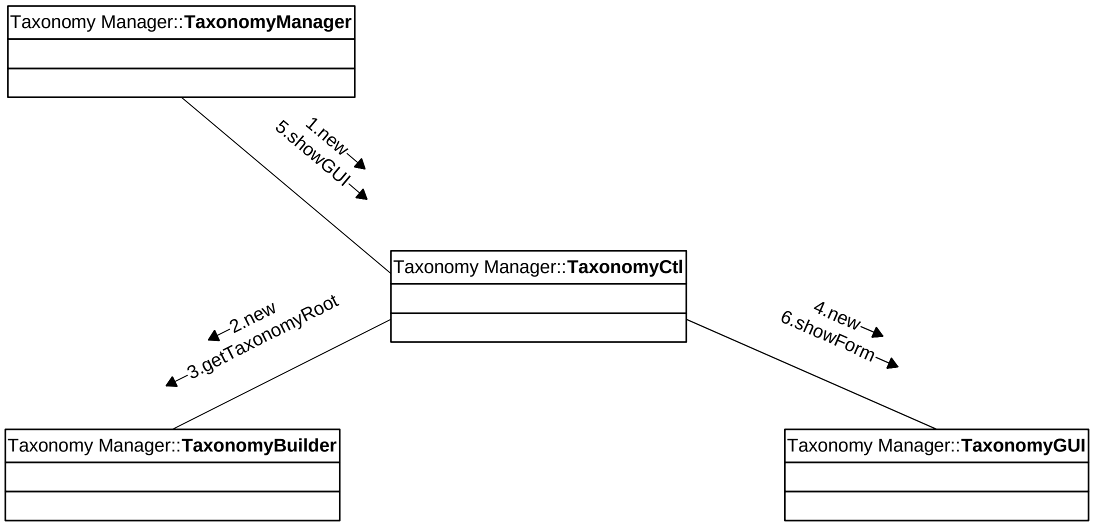

Taxonomy Manager::TaxonomyManager
1.new
5.showGUI
Taxonomy Manager::TaxonomyCtl
2.new
3.getTaxonomyRoot
4.new
6.showForm
Taxonomy Manager::TaxonomyBuilder
Taxonomy Manager::TaxonomyGUI

Figure 32. [UC1.0] Basic Flow: Manage Taxonomy.

4. **TaxonomyCtl** then creates a **new**() instance of the **TaxonomyGUI**.

5. **TaxonomyManager** then calls **TaxonomyCtl** to **showGUI**().

6. And **TaxonomyCtl** in turn calls **TaxonomyGUI** to **showForm**().

7. From here, the Instructor can interact with the TaxonomyGUI to [UC1.1] Add Taxonomy, [UC1.2] Edit Taxonomy, [UC1.3] Delete Taxonomy or [UC1.4] Save Taxonomy. The use case ends when the Taxonomy Manager (actor) performs [UC1.5] Cancel/Close Taxonomy Session.

<u>[UC1.1] Scenario: Add Taxonomy</u>

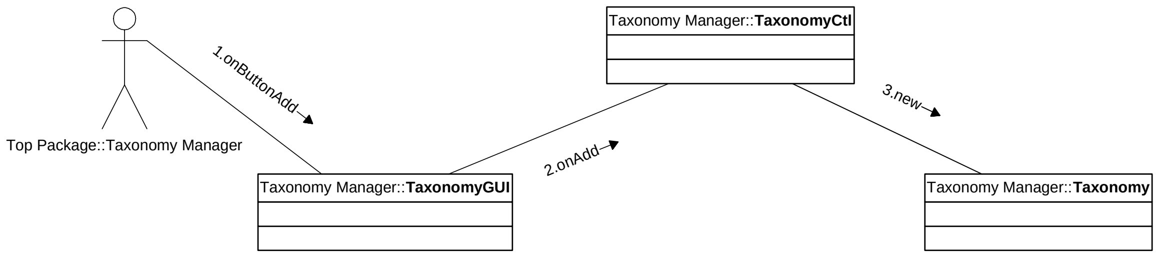

Top Package::Taxonomy Manager
1.onButtonAdd
Taxonomy Manager::TaxonomyCtl
2.onAdd
3.new
Taxonomy Manager::TaxonomyGUI
Taxonomy Manager::Taxonomy

Figure 33. [UC1.1] Scenario: Add Taxonomy.

1. In this scenario, the Taxonomy Manager selects the Add button which triggers **onButtonAdd**() of **TaxonomyGUI**. **TaxonomyGUI** will ask the Instructor to key in the taxonomy term.

2. Once the Taxonomy Manager is done with the entry, **TaxonomyGUI** then calls **onAdd**() of **TaxonomyCtl**. If the Taxonomy Manager cancels the addition request, the creation is abandoned and the scenario ends.

3. **TaxonomyCtl** will then create a **new**() **Taxonomy** object, and the scenario ends.

[UC1.2] Scenario: Edit Taxonomy

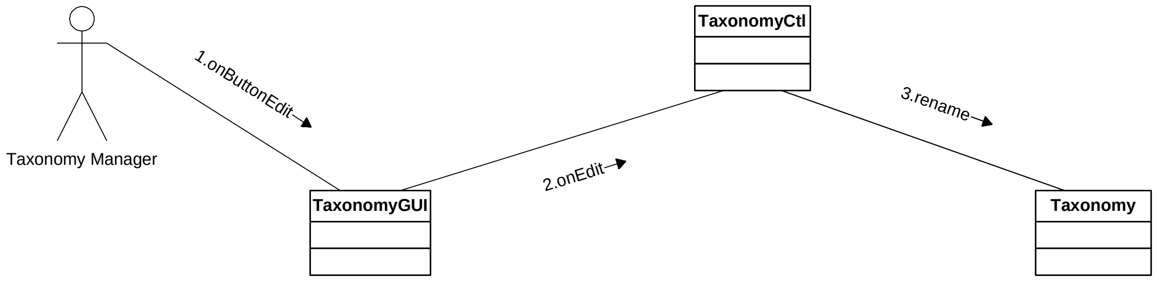


Figure 34. [UC1.2] Scenario: Edit Taxonomy.

1. In this scenario, the Taxonomy Manager selects the Edit button which triggers **onButtonEdit**() of **TaxonomyGUI**. **TaxonomyGUI** will display the current taxonomy value and ask the Taxonomy Manager to edit it.

2. Once the Taxonomy Manager is done, **TaxonomyGUI** then calls **onEdit**() of **TaxonomyCtl**. If the Taxonomy Manager cancelled the edit, the editing is abandoned and the scenario ends.

3. **TaxonomyCtl** will then **rename**() the **Taxonomy** object, and the scenario ends.

[UC1.3] Scenario: Delete Taxonomy

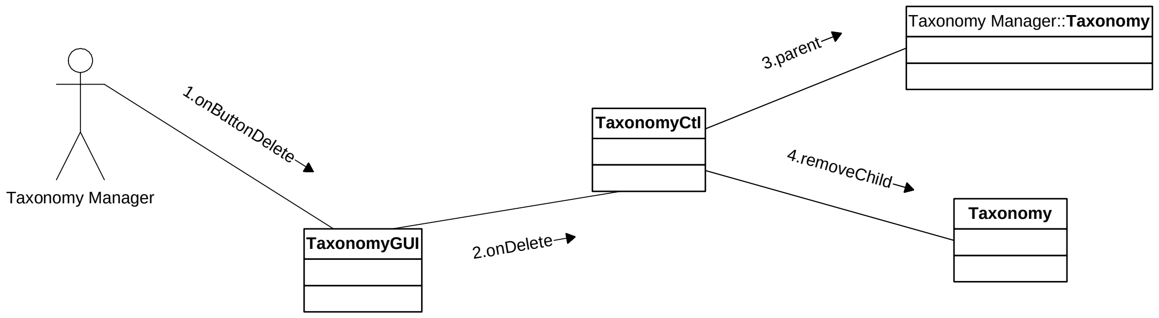


Figure 35. [UC1.3] Scenario: Delete Taxonomy.

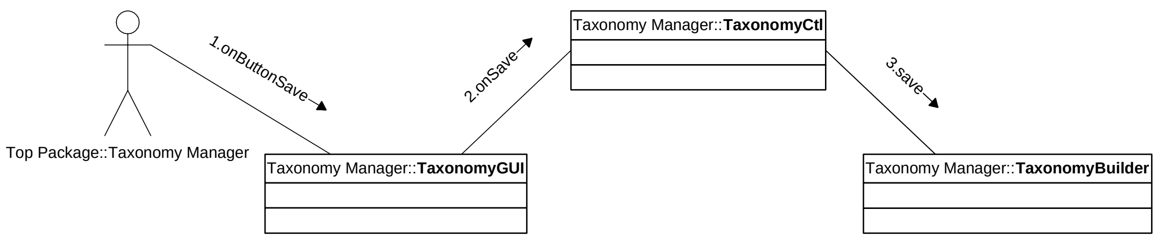
1. In this scenario, the Taxonomy Manager selects the Delete button which triggers **onButtonDelete**() of **TaxonomyGUI**. **TaxonomyGUI** will ask the Instructor to confirm the deletion request.

2. If the Taxonomy Manager confirms the deletion request, **TaxonomyGUI** then calls **onDelete**() of **TaxonomyCtl**, else the deletion request is abandoned and the scenario ends.

3. **TaxonomyCtl** will then call **parent**() of the **Taxonomy** object to retrieve the parent **Taxonomy** object.

4. Using the parent **Taxonomy** object, **TaxonomyCtl** then calls it to **removeChild**(), and the scenario ends.

[UC1.4] Scenario: Save Taxonomy



Top Package::Taxonomy Manager
1.onButtonSave
2.onSave
Taxonomy Manager::TaxonomyCtl
3.save
Taxonomy Manager::TaxonomyGUI
Taxonomy Manager::TaxonomyBuilder

Figure 36. [UC1.4] Scenario: Save Taxonomy.

1. In this scenario, the Taxonomy Manager selects the Save button which triggers **onButtonSave**() of **TaxonomyGUI**.

2. **TaxonomyGUI** then calls **onSave**() of **TaxonomyCtl**.

3. **TaxonomyCtl** will then call the **TaxonomyBuilder** to **save**(), and the scenario ends.

[UC1.5] Scenario: Cancel/Close Taxonomy Session

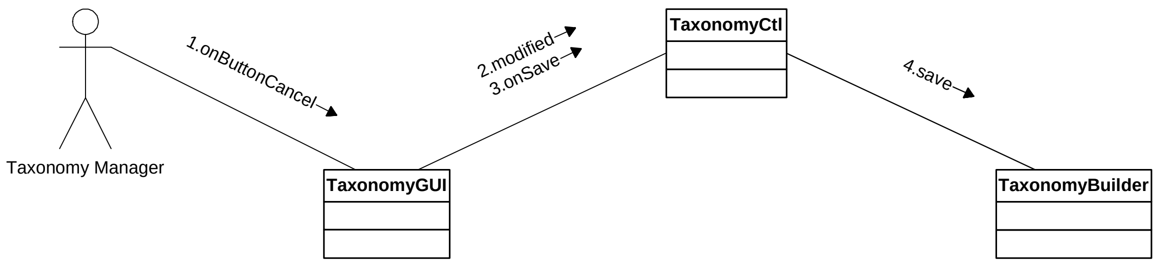


Figure 37. [UC1.5] Scenario: Cancel Taxonomy Session.

1. In this scenario, the Taxonomy Manager selects the Cancel button which triggers **onButtonCancel**() of **TaxonomyGUI**.

2. **TaxonomyGUI** will check with **TaxonomyCtl** if it has been **modified**().

3. If it has been modified, the Taxonomy Manager will be asked if the changes should be saved, ignored or to cancel this Cancel selection. If it is cancelled, then the scenario ends. If it is to be saved, then **TaxonomyCtl** is called to do **onSave**().

4. In which case, **TaxonomyCtl** will in turn call **TaxonomyBuilder** to **save**().

5. The application then shuts down, and the scenario ends.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

2. [UC.2] Setup Campaign

Brief Description:

The Instructor sets up a campaign by describing the objectives of the campaign and by composing a collection of related scenarios.

Primary Actor(s):

Instructor.

Secondary Actor(s):

Nil.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with the **CampaignManager** creating a **new**() instance of **CampaignManagerCtl**.

2. **CampaignManagerCtl** in turn creates a **new**() instance of the **ScenarioCatalogBuilder**.

3. And calls **ScenarioCatalogBuilder** to **load**() the catalog of scenarios.

4. Finally, it calls **getScenarios**() from **ScenarioCatalogBuilder**.

5. **CampaignManagerCtl** next creates a **new**() instance of **CampaignCatalogBuilder**.

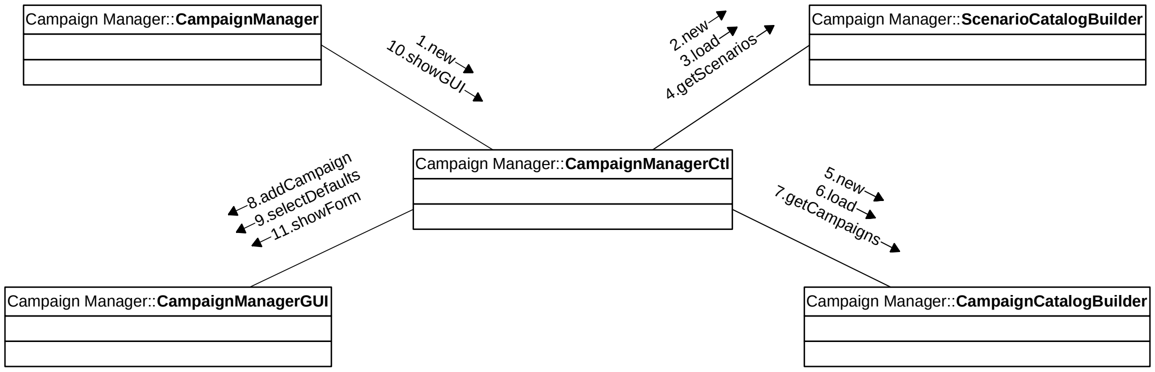

Campaign Manager::**CampaignManager**
1.new→
10.showGUI→
2.new→
3.load→
4.getScenarios→
Campaign Manager::**ScenarioCatalogBuilder**
Campaign Manager::**CampaignManagerCtl**
←8.addCampaign
←9.selectDefaults
←11.showForm
5.new→
6.load→
7.getCampaigns→
Campaign Manager::**CampaignManagerGUI**
Campaign Manager::**CampaignCatalogBuilder**

Figure 38. [UC2.0] Basic Flow: Setup Campaign.

6. And calls **CampaignCatalogBuilder** to **load**() the catalog of campaigns.

7. Finally, it calls **getCampaigns**() from **CampaignCatalogBuilder**.

8. Next, for each **Campaign** retrieved, **CampaignManagerCtl** informs the **CampaignManagerGUI** to **addCampaign**() into the GUI.

9. Finally, it informs **CampaignManagerGUI** to **selectDefaults**().

10. **CampaignManager** then calls **CampaignManagerCtl** to **showGUI**().

11. Which, in turn, performs a similar function with **CampaignManagerGUI** by calling **showForm**().

12. The Instructor can then proceed to interact with the CampaignManagerGUI using [UC2.1] New Campaign, [UC2.2] Edit Campaign, [UC2.3] Delete Campaign or [UC2.4] Save Campaigns. The use case ends when the Instructor performs [UC2.5] Cancel/Close Campaign Management Session.

[UC 2.1] Scenario: New Campaign

1. In this scenario, the Instructor selects the New button causing **onButtonNew**() to be called on **CampaignManagerGUI**.

2. It, in turn, calls **CampaignManagerCtl** to **showAddCampaign**().

3. **CampaignManagerCtl** checks if the **CampaignGUI** is already created and **isVisible**(). It is, then there is already an instance of it active and the scenario ends.

4. If it is not, then it creates a **new**() instance of **CampaignGUI**.

5. It then updates the **CampaignGUI** with the list of scenarios by invoking **setScenarios**().

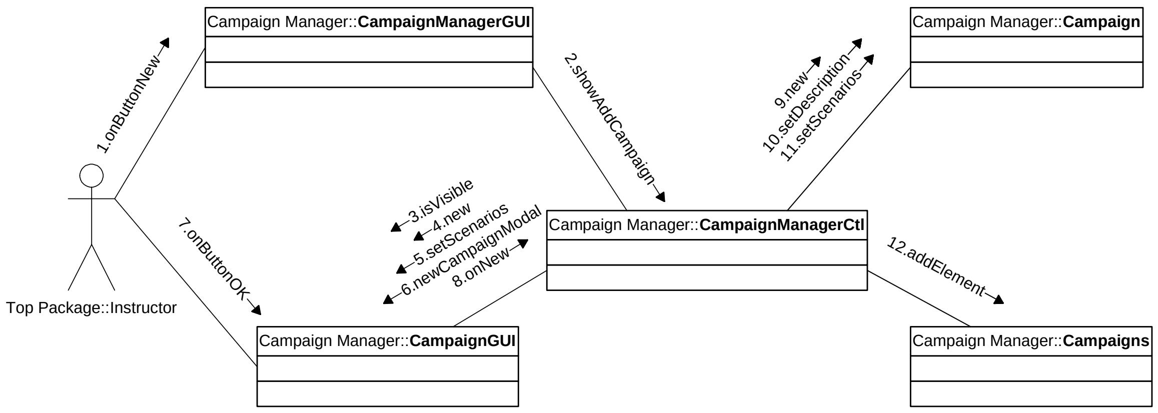


Figure 39. [UC2.1] Scenario: Add Campaign.

6. Finally, it calls **newCampaignModal**() of **CampaignGUI** to setup the GUI as a modal dialog.

7. The Instructor interacts with the **CampaignGUI** to edit accordingly. Once done, the Instructor selects the OK button, calling **onButtonOK**() on **CampaignGUI**. If Cancel is selected, then the creation attempt is cancelled and this scenario ends.

8. **CampaignGUI** next notifies **CampaignManagerCtl** via **onNew**().

9. Consequently, **CampaignManagerCtl** then creates a **new**() instance of a **Campaign**.

10. And updates it by doing **setDescription**().

11. And **setScenarios**().

12. This **Campaign** is then added into the list of campaigns by invoking **addElement**().

13. Finally, **CampaignManagerCtl** updates the **CampaignManagerGUI** by providing the **newCampaign**(). The scenario then ends.

<u>[UC 2.2] Scenario: Edit Campaign</u>

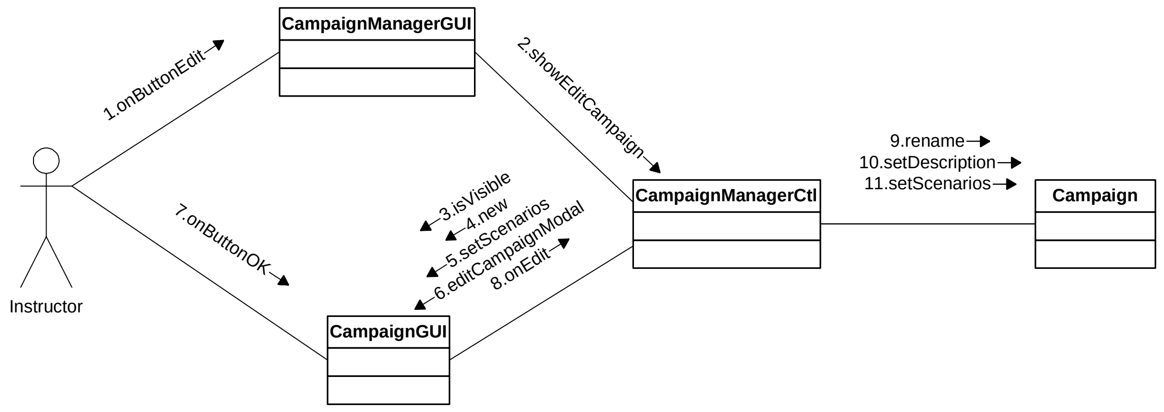


Figure 40. [UC2.2] Scenario: Edit Campaign.

1. In this scenario, the Instructor selects the Edit button causing **onButtonEdit**() of the **CampaignManagerGUI** to be called.

2. It then calls **CampaignManagerGUI** to **showEditCampaign**().

3. **CampaignManagerGUI,** in turn, checks if **CampaignGUI isVisiable**().

4. If it is not, then a **new**() instance of **CampaignGUI** is created.

5. And it updates **CampaignGUI** by invoking **setScenarios**(), supplying the list of all possible scenarios.

6. Finally, it calls **editCampaignModal**() to display the **Campaign** as a modal dialog.

7. The Instructor amends the campaign definition accordingly. Once completed, the Instruct selects OK, causing **onButtonOK**() to be called on **CampaignGUI**. If Cancel is selected, the editing is abandoned and the scenario ends.

8. Else, **Campaign** will call **CampaignManagerCtl** to process the editing done by calling **onEdit**().

9. **CampaignManagerCtl** performs the update by starting with a **rename**() on the **Campaign** object to update its name.

10. Then it calls **setDescription**() to update the description.

11. And lastly, **setScenarios**() to update the list of scenarios for the campaign.

12. With that done, it proceeds to notify the **CampaignManagerGUI** with **editCampaign**() to update the campaigns listed, and the scenario ends.

[UC 2.3] Scenario: Delete Campaign

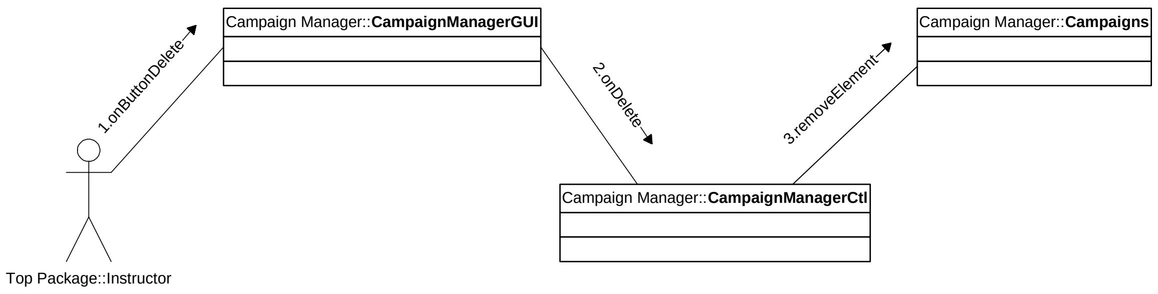

Campaign Manager::CampaignManagerGUI
Campaign Manager::Campaigns
1.onButtonDelete
2.onDelete
3.removeElement
Campaign Manager::CampaignManagerCtl
Top Package::Instructor

Figure 41. [UC2.3] Scenario: Delete Campaign.

1. In this scenario, the Instructor selects the Delete button, causing **onButtonDelete**() to be called on **CampaignManagerGUI**.

2. It then passes this on to **CampaignManagerCtl** to handle **onDelete**().

3. **CampaignManagerCtl** will **removeElement**() from the list of campaigns. The scenario then ends.

[UC 2.4] Scenario: Save Campaigns

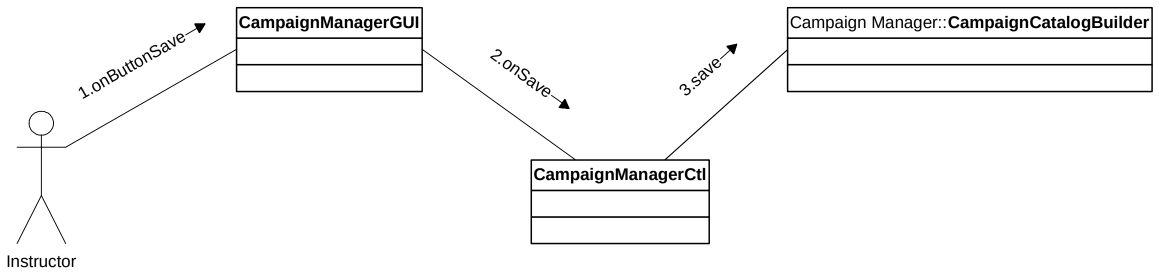


Figure 42. [UC2.4] Scenario: Save Campaigns.

1. In this scenario, the Instructor selects the Save button to save all the campaign changes, causing **onButtonSave**() to be called on **CampaignManagerGUI**.

2. It, in turn, calls **onSave**() of **CampaignManagerCtl** to process it.

3. **CampaignManagerCtl** finally calls **CampaignCatalogBuilder** to do the **save**() itself, and the scenario ends.

<u>[UC 2.5] Scenario: Cancel/Close Campaign Management Session</u>

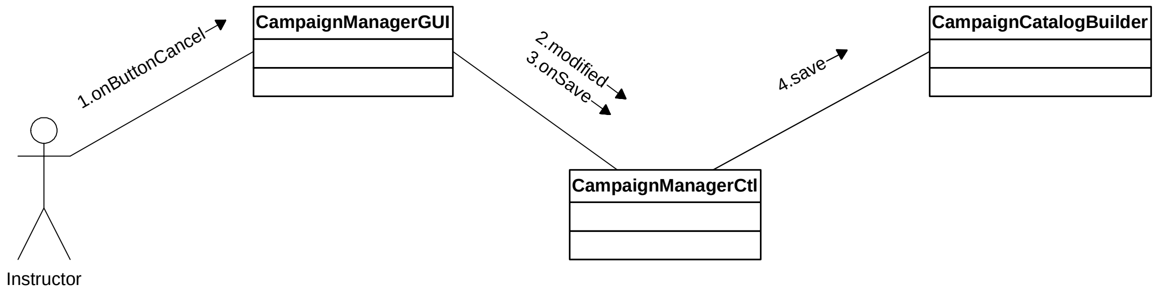


Figure 43. [UC2.5] Scenario: Cancel/Close Campaign Management Session.

1. In this scenario, the Instructor selects the Cancel button to close/cancel the session. This causes **onButtonCancel**() to be called on the **CampaignManagerGUI**.

2. **CampaignManagerGUI** checks with **CampaignManagerCtl** to see if it has been **modified**().

3. If there are changes made which have not been saved, the Instructor is asked to verify if the changes should be saved first, ignored, or to cancel the cancel/close request altogether. If ignored, then the changes are simply abandoned, and we proceed to step 4. If it is cancelled, the scenario ends and the session remain intact. Else, **CampaignManagerCtl** is called to do **onSave**().

4. It then passes the saving to the **CampaignCatalogBuilder** to perform the **save**().

5. The application is then shutdown, ending the scenario.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

3. [UC.3] Release Campaign

Brief Description:

A campaign which is ready for release undergoes an integrity check to finalize the campaign package. The campaign is then base-lined and is ready for Students to play it.

Primary Actor(s):

Instructor.

Secondary Actor(s):

Nil.

Flow of Events

<u>[UC3.1] Scenario: Release on New Campaign</u>

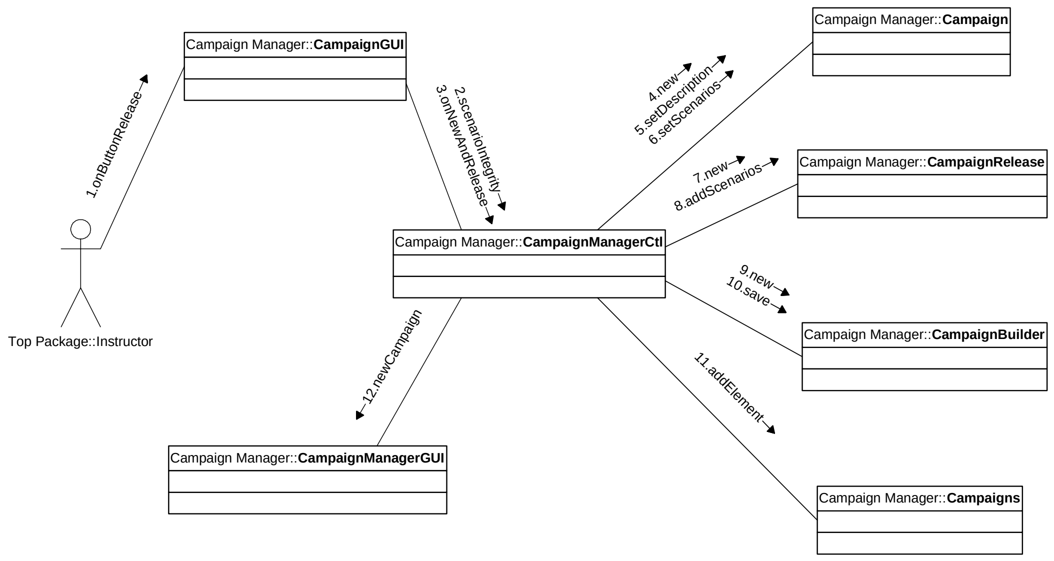

Campaign Manager::Campaign
Campaign Manager::CampaignGUI
1.onButtonRelease
2.scenarioIntegrity
3.onNewAndRelease
4.new
5.setDescription
6.setScenarios
7.new
8.addScenarios
Campaign Manager::CampaignRelease
Campaign Manager::CampaignManagerCtl
9.new
10.save
Campaign Manager::CampaignBuilder
Top Package::Instructor
12.newCampaign
11.addElement
Campaign Manager::CampaignManagerGUI
Campaign Manager::Campaigns

Figure 44. [UC3.1] Scenario: Release on New Campaign.

1. The use case begins with the **CampaignGUI** in the “new campaign” mode [UC2.1] with the Instructor selecting the Release button to release the campaign. This causes **onButtonRelease**() to be called against **CampaignGUI**.

2. **CampaignGUI** then performs an integrity check to ensure that all the scenarios of the campaign are consistent by calling **CampaignManagerCtl** to check **scenarioIntegrity**() iteratively for each scenario defined in the campaign.

3. If all scenario files are intact, it then calls **CampaignManagerCtl** to perform **onNewAndRelease**().

4. **CampaignManagerCtl** then creates a **new**() instance of **Campaign**.

5. And updates **Campaign** by invoking **setDescription**().

6. As well as **setScenarios**().

7. It then creates a copy for release by creating a **new**() instance of **CampaignRelease**.

8. And **addScenarios**() to it.

9. It then creates a **new**() instance of **CampaignBuilder**.

10. And asks **CampaignBuilder** to **save**() the releasable campaign copy. This will include export of all the required files, including the campaign definition files and the (copies of) scenario files.

11. The campaign that was created is then added to **Campaigns** by doing an **addElement**().

12. Finally, it updates **CampaignManagerGUI** with **newCampaign**(). The scenario then ends.

<u>[UC3.2] Scenario: Release on Edit Campaign</u>

1. The use case begins with the CampaignGUI in the “edit campaign” mode [UC2.1] with the Instructor selecting the Release button to release the campaign. This causes **onButtonRelease**() to be called against **CampaignGUI**.

2. **CampaignGUI** then performs an integrity check to ensure that all the scenarios of the campaign are consistent by calling **CampaignManagerCtl** to check **scenarioIntegrity**() iteratively for each scenario defined in the campaign.

3. If all scenario files are intact, it then calls **CampaignManagerCtl** to do **onEditAndRelease**().

4. **CampaignManagerCtl** then **renames**() the **Campaign**.

5. And updates **Campaign** by invoking **setDescription**().

6. Followed by **setScenarios**().

7. It then creates a copy for release by creating a **new**() instance of **CampaignRelease**.

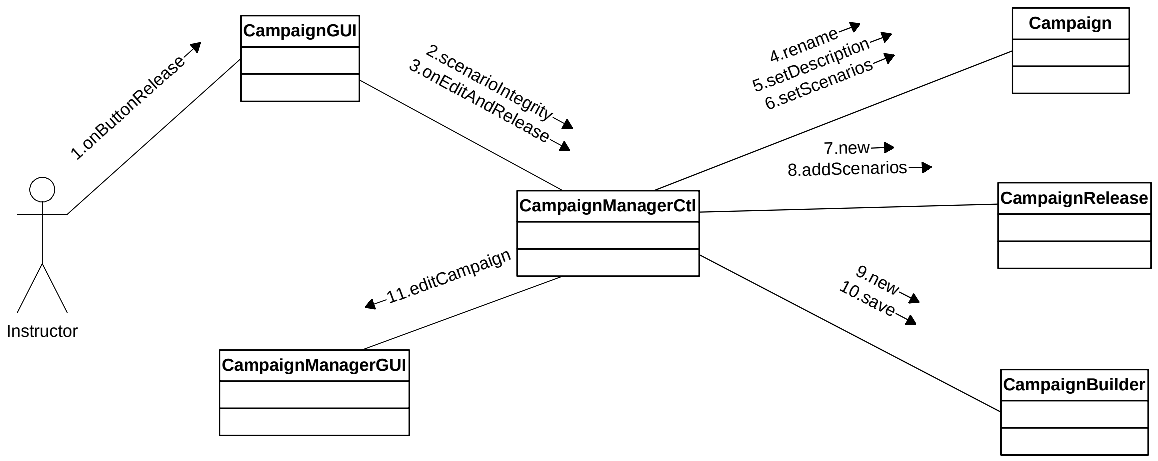


Figure 45. [UC3.2] Scenario: Release on Edit Campaign.

8. And **addScenarios**() to it.

9. It then creates a **new**() instance of **CampaignBuilder**.

10. And asks **CampaignBuilder** to **save**() the releasable campaign copy. It will include exporting all the required files, including the campaign definition files and the (copies of) scenario files.

11. Finally, it updates **CampaignManagerGUI** with **editCampaign**(). The scenario then ends.

Special Requirements

Nil.

Pre-conditions

[UC.2] Setup Campaign has to be performed before this, and more specifically, either [UC2.1] or [UC2.2] is in progress.

Post-conditions

Campaign is now released for Student to play.

4. [UC.4] Conduct After-Action Review

Brief Description:

The Instructor conducts an after-action review with the Student by analyzing the event logs.

Primary Actor(s):

Instructor.

Secondary Actor(s):

CyberCIEGE, Scenario Definition Tool.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with **CampaignAnalyzer** creating a **new**() instance of **CampaignAnalyzerCtl**.

2. **CampaignAnalyzerCtl** in turn creates a **new**() instance of **CampaignAnalyzerGUI**.

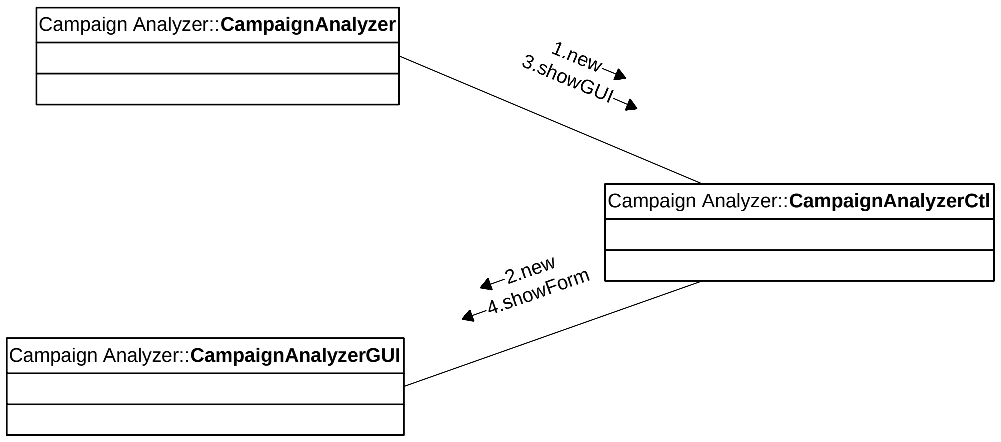

Campaign Analyzer::CampaignAnalyzer
1.new
3.showGUI
Campaign Analyzer::CampaignAnalyzerCtl
2.new
4.showForm
Campaign Analyzer::CampaignAnalyzerGUI

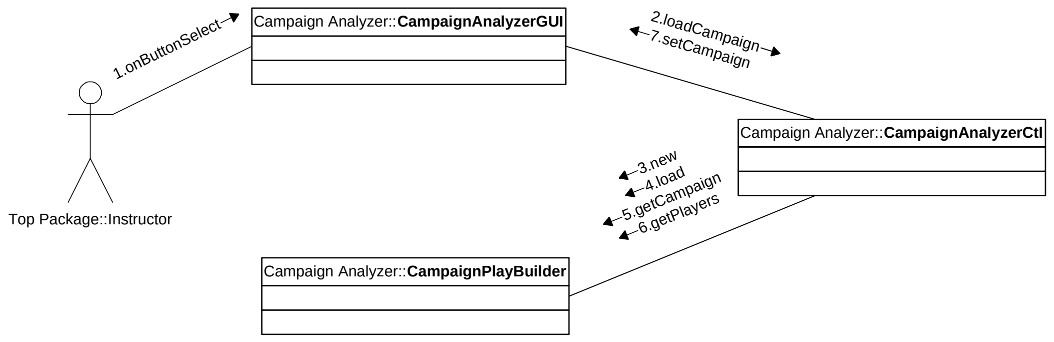
Figure 46. [UC4.0] Basic Flow: Conduct After-Action Review.

3. **CampaignAnalyzer** will then call **CampaignAnalyzerCtl** to showGUI().

4. Finally, **CampaignAnalyzerCtl** will call **CampaignAnalyzerGUI** to showForm().

5. From here, the Instructor can interact with the **CampaignAnalyzerGUI** to [UC4.1] Select Campaign, [UC4.2] Examine Details of a Scenario, [UC4.3] Play Scenario or [UC4.7] Close the Campaign Analyzer.

[UC4.1] Scenario: Select Campaign



1.onButtonSelect
Campaign Analyzer::CampaignAnalyzerGUI
2.loadCampaign
7.setCampaign
Campaign Analyzer::CampaignAnalyzerCtl
3.new
4.load
5.getCampaign
6.getPlayers
Top Package::Instructor
Campaign Analyzer::CampaignPlayBuilder

Figure 47. [UC4.1] Scenario: Select Campaign.

1. In this scenario, the Instructor selects the “Select” button to pick a campaign. This causes **onButtonSelect**() to be called on **CampaignAnalyzerGUI**.

2. **CampaignAnalyzerGUI** will then call **CampaignAnalyzerCtl** to **loadCampaign**().

3. **CampaignAnalyzerCtl** will **create**() a new instance of **CampaignPlayBuilder** to collect information about the campaign selected.

4. It does this by asking **CampaignPlayBuilder** to **load**().

5. And proceeds to invoke **getCampaign**() to obtain the campaign details.

6. And also **getPlayers**() to obtain all the players involved.

7. It then updates the **CampaignAnalyzerGUI** by calling **asetCampaign**(). The scenario then ends.

[UC4.2] Scenario: Examine Details of a Scenario

1. The scenario begins with the Instructor selecting the “Details” button, causing **onButtonDetails**() to be called on **CampaignAnalyzerGUI**.

2. **CampaignAnalyzerGUI** then calls **CampaignAnalyzerCtl** to **loadScenario**().

3. **CampaignAnalyzerCtl** in turn calls **CampaignPlay** to **getScenarioEditorCommand**().

4. Finally, it will **execute**() the command to launch the Scenario Editor Tool with the scenario definition file loaded. The Instructor then proceeds to interact with the Scenario Editor Tool, ending this scenario.

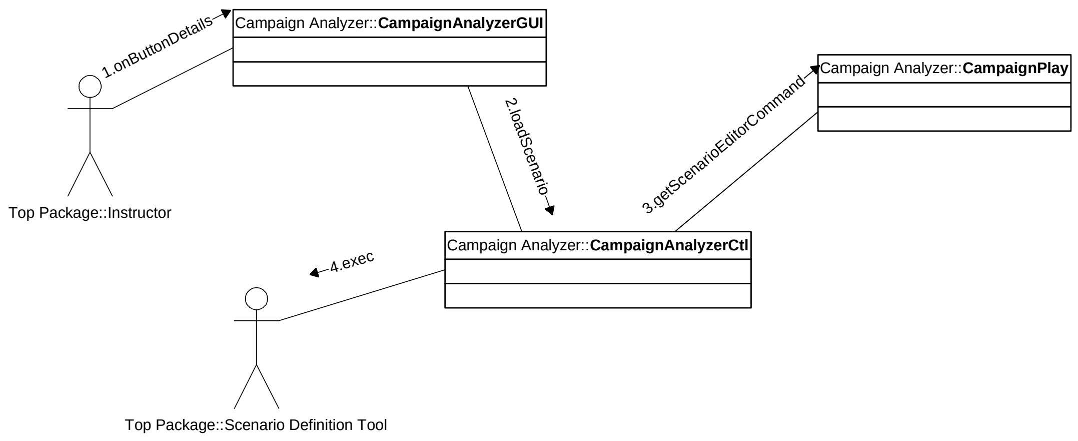

Campaign Analyzer::CampaignAnalyzerGUI
1.onButtonDetails
Top Package::Instructor
2.loadScenario
Campaign Analyzer::CampaignPlay
3.getScenarioEditorCommand
Campaign Analyzer::CampaignAnalyzerCtl
4.exec
Top Package::Scenario Definition Tool

Figure 48. [UC4.2] Scenario: Examine Details of a Scenario.

[UC4.3] Scenario: Play Scenario

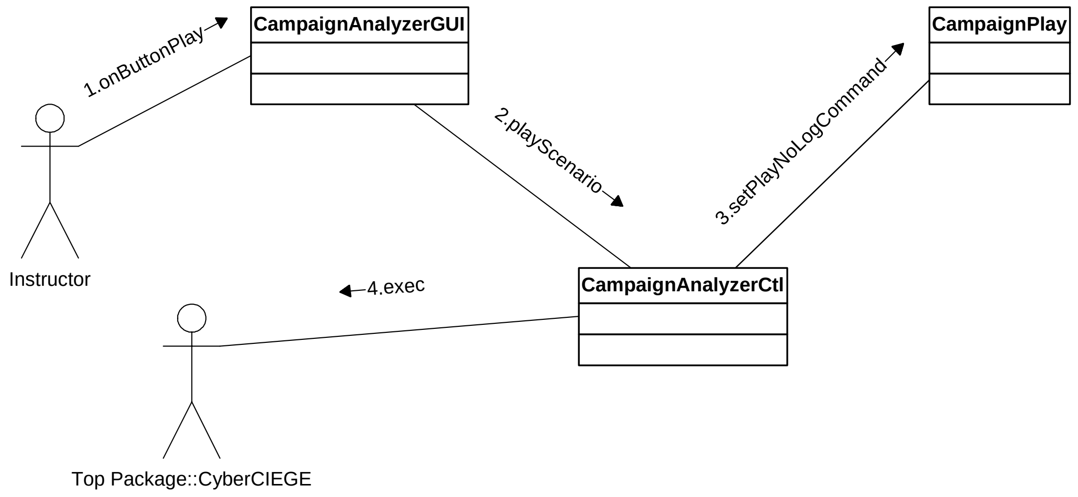


Figure 49. [UC4.3] Scenario: Play Scenario.

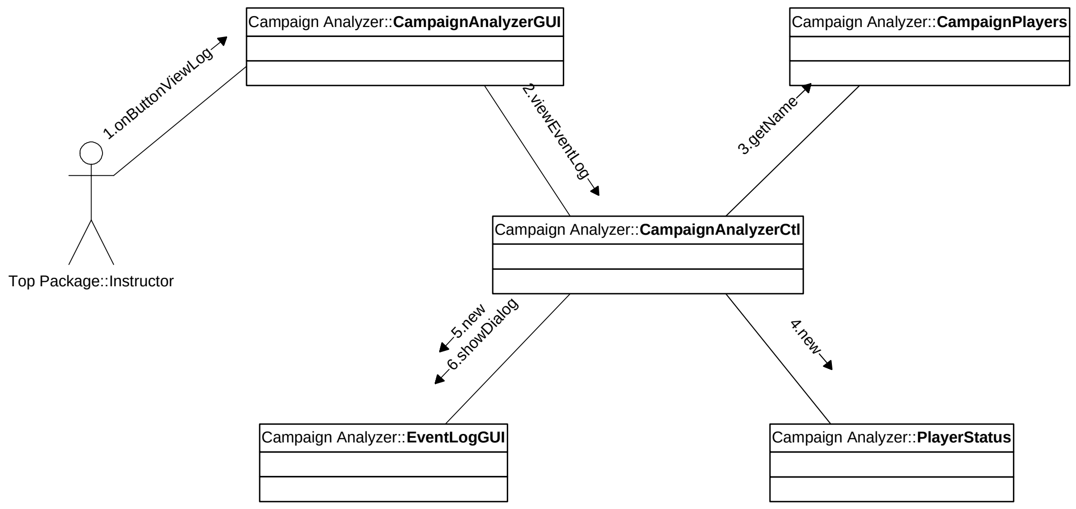
1. In this scenario, the Instructor selects the "Play" button, causing **onButtonPlay**() to be called on the **CampaignAnalyzerGUI**.

2. **CampaignAnalyzerGUI** then calls **CampaignAnalyzerCtl** to playScenario().

3. **CampaignAnalyzerCtl** in turn calls **CampaignPlay** to **getPlayNoLogCommand**().

4. Finally, it will **execute**() the command to launch CyberCIEGE with the given scenario loaded. No event logging is specified in this case. The Instructor then interacts with CyberCIEGE to review the scenario selected, ending this scenario.

[UC4.4] Scenario: View Log



Campaign Analyzer::**CampaignAnalyzerGUI**
Campaign Analyzer::**CampaignPlayers**
1.onButtonViewLog
2.viewEventLog
3.getName
Top Package::Instructor
Campaign Analyzer::**CampaignAnalyzerCtl**
5.new
6.showDialog
4.new
Campaign Analyzer::**EventLogGUI**
Campaign Analyzer::**PlayerStatus**

Figure 50. [UC4.4] Scenario: View Log.

1. The scenario begins with the Instructor selecting the "View" button causing **onButtonViewLog**() to be called on **CampaignAnalyzerGUI**.

2. **CampaignAnalyzerGUI** in turn calls **CampaignAnalyzerCtl** to **viewEventLog**().

3. For each **CampaignPlayer**, the **CampaignAnalyzerCtl** will call **getName**() to compare against the player selected.

4. Once a match is found, it will create a **new**() instance of **PlayerStatus** to obtain all the summary status values of that player.

5. It then creates a **new**() instance of **EventLogGUI**.

6. And calls **EventLogGUI** to **showDialog**(), displaying the event log of that player.

7. The Instructor can continue to interact with the **EventLogGUI** to [UC4.5] Load Saved State and finally to using [UC4.6] Close the Event Log Analyzer to end the session. The scenario then ends.

[UC4.5] Scenario: Load Saved State

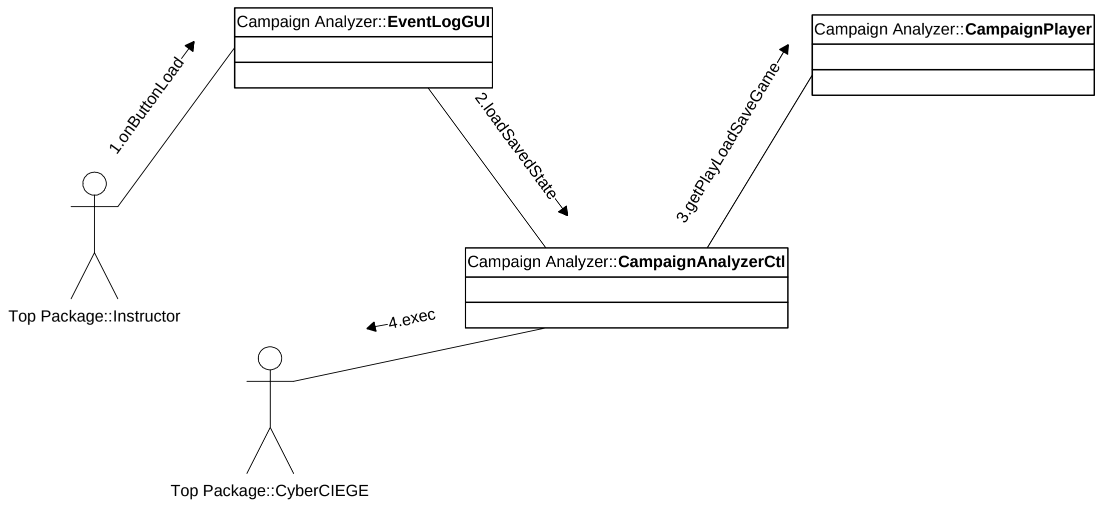


Figure 51. [UC4.5] Scenario: Load Saved State.

1. In this scenario, the Instructor selects the “Load” button causing **onButtonLoad**() to be called on the **EventLogGUI**.

2. **EventLogGUI** in turn calls **CampaignAnalyzerCtl** to **loadSavedState**().

3. **CampaignAnalyzerCtl** then calls **CampaignPlayer** to **getPlayLoadSaveGame**(), obtaining the command to be executed to load the saved state.

4. Finally, it will **execute**() the command, causing CyberCIEGE to be launched with the saved game file loaded. The Instructor continues to interact with CyberCIEGE to review the state of the player’s game. This scenario then ends.

[UC4.6] Scenario: Close the Event Log Analyzer

1. In this scenario, the Instructor selects the “Close” button causing **onButtonClose**() to be called on the **EventLogGUI**.

2. **EventLogGUI** will then **close**() and **dispose**() itself, ending this scenario.

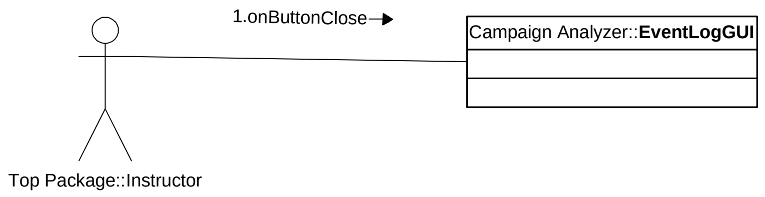


Figure 52. [UC4.6] Scenario: Close the Event Log Analyzer.

[UC4.7] Scenario: Close the Campaign Analyzer

1. In this scenario, the Instructor selects the “Close” button causing **onButtonClose**() to be called on the **CampaignAnalyzerGUI**.

2. **CampaignAnalyzerGUI** will then shutdown the **Campaign Analyzer**.

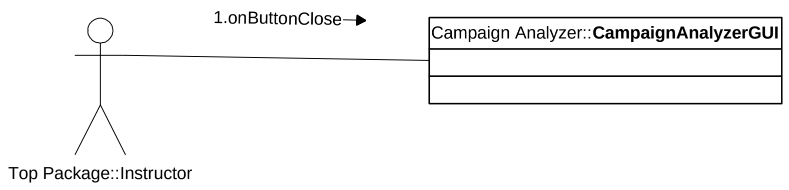


Figure 53. [UC4.7] Scenario: Close the Campaign Analyzer.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

5. [UC.5] Load Campaign

Brief Description:

The Student loads the campaign to review the objectives of the campaign, and the scenarios to be played.

Primary Actor(s):

Student.

Secondary Actor(s):

Nil.

Flow of Events

Basic Flow

1. The use case begins with **CampaignPlayerTool** creating a **new**() instance of **CampaignPlayerTool**.

2. **CampaignPlayerCtl** is then responsible for creating a **new**() instance of **CampaignPlayerGUI**.

3. **CampaignPlayerTool** will then ask **CampaignPlayerCtl** to **showGUI**().

4. **CampaignPlayerCtl** in turn calls **CampaignPlayerGUI** to **showForm**().

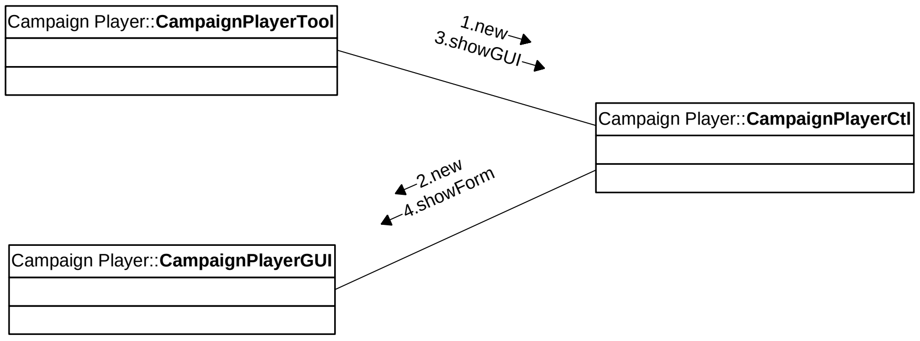

Campaign Player::CampaignPlayerTool
1.new
3.showGUI
Campaign Player::CampaignPlayerCtl
2.new
4.showForm
Campaign Player::CampaignPlayerGUI

Figure 54. [UC5.0] Basic Flow: Load Campaign.

5. The Student can then proceed to perform [UC5.1] Select Campaign.

6. The use case ends when the Student performs [UC5.2] Close Campaign Tool.

[UC 5.1] Scenario: Select Campaign

1. The scenario begins when the Student clicks the Select button, causing **onButtonSelect**() to be called on **CampaignPlayerGUI**. This enables the Student to select a “campaign.xml” file, which is a campaign definition file, to load.

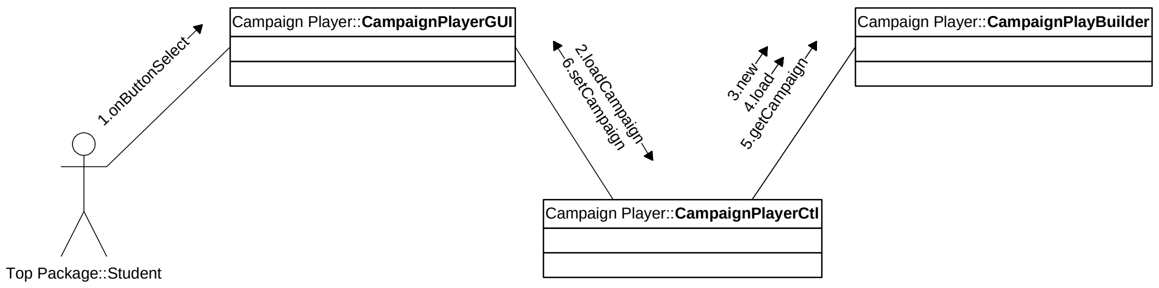
2. Once selected, it in turns calls **CampaignPlayerCtl** to **loadCampaign**().

3. **CampaignPlayerCtl** then creates a **new**() instance of **CampaignPlayerBuilder**.

4. And uses it to **load**() the campaign itself.

5. **CampaignPlayerCtl** then retrieves the resulting **Campaign** object by doing a **getCampaign**() from **CampaignPlayerBuilder**.

6. Finally, **CampaignPlayerCtl** will **setCampaign**() to the **CampaignPlayerGUI**, updating the display with the campaign details, thus ending the scenario.



Campaign Player::CampaignPlayerGUI
Campaign Player::CampaignPlayBuilder
Campaign Player::CampaignPlayerCtl
Top Package::Student
1.onButtonSelect
2.loadCampaign
6.setCampaign
3.new
4.load
5.getCampaign

Figure 55. [UC5.1] Scenario: Select Campaign.

[UC 5.2] Scenario: Close Campaign Tool

1. The scenario begins with the Student selecting the Close button. This causes **onButtonClose**() to be called on **CampaignPlayerGUI**.

2. **CampaignPlayerGUI** will then shutdown the tool, thus ending the scenario.

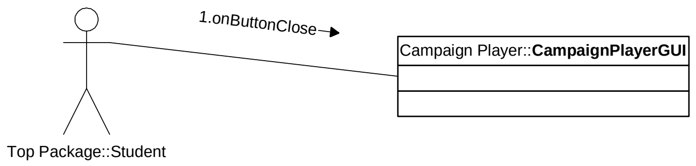


Figure 56. [UC5.2] Scenario: Close Campaign Tool.

Special Requirements

Nil.

Pre-conditions

Nil.

Post-conditions

Nil.

6. [UC.6] Play Scenario

Brief Description:

The Student proceeds to play a scenario of the campaign.

Primary Actor(s):

Student.

Secondary Actor(s):

CyberCIEGE.

Flow of Events

<u>Basic Flow</u>

1. The use case begins with the Student having selected a scenario of the campaign in [UC5.1]. The Student then selects the Play button, causing **onButtonPlay**() to be called on **CampaignPlayerGUI**.

2. It in turn calls the **CampaignPlayerCtl** to **playScenario**().

3. And **CampaignPlayerCtl** then calls **Campaign** to **getPlayCommand**(), obtaining an executable shell command to launch CyberCIEGE. Finally it executes the command and the Student can proceed to play in CyberCIEGE, thus ending the use case.

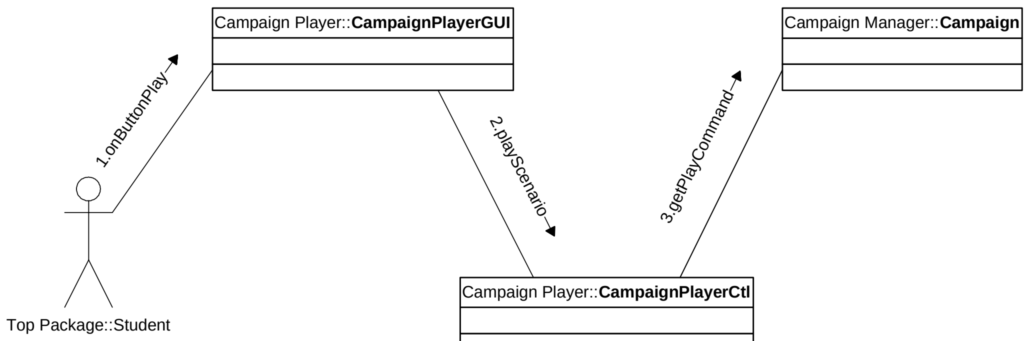

Campaign Player::CampaignPlayerGUI
Campaign Manager::Campaign
1.onButtonPlay
2.playScenario
3.getPlayCommand
Top Package::Student
Campaign Player::CampaignPlayerCtl

Figure 57. [UC6.0] Basic Flow: Play Campaign.

Special Requirements

Nil.

Pre-conditions

[UC5.1] must be performed prior to this use case so that the campaign to be played is already loaded.

Post-conditions

Nil.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. INTERFACE SPECIFICATIONS WITH RIVERMIND

A. PURPOSE

The purpose of this appendix is to define the interface specifications with Rivermind.

B. SCENARIO FILE FORMAT

The following changes/additions proposed for the Scenario File were agreed upon:

1. Title/name (descriptive but short) to the scenario.

2. Game termination condition settings. Each of these could be optionally specified. Only when specified will they affect game termination. For instance, if none are specified, the game plays indefinitely.

   a. Upper/lower-bound thresholds for various game play attributes that will cause the game to end when the threshold is exceeded.

   b. Game turn limit - game ends when the time unit of game play exceeds the limit.

C. CYBERCIEGE PROGRAM PARAMETERS

To enable the Campaign Player module to launch CyberCIEGE with the desired scenario and generate an EventLog, CyberCIEGE shall support the following program parameters:

CyberCIEGE -s <Scenario File> [-i <IDTag>-e <EventLog>]

To enable a saved game file to be reloaded, the following is required:

CyberCIEGE -l <Saved Game>

Program Parameter	Description	Format
-s <Scenario File>	Filename of the scenario to be played.	Standard filename format.
-i <IDTag>	Identifies the campaign-scenario-player for this game. The tag <IDTag> (in the form <campaign name>/<scenario name>/<userID>) is intended to be written	String of at most 100 characters.

	out to the event log file as a header information.	
-e <EventLog>	Name of the event log. CyberCIEGE shall append “-999.log” to this name to fully qualify the filename. “999” shall be a running number starting from “001”, increasing by 1 till a maximum of “999”, when the log is split into multiple log files. Rather than having a single huge event log file, it is thus possible to have a number of smaller event log files instead.	String of at most 100 characters. e.g. “example” becomes “example-001.log”, “example-002.log”, …
-l <Saved Game>	Load the CyberCIEGE saved game file supplied.	Standard filename format.

Table 5 CyberCIEGE program parameters.

D. EVENT LOG

It should be noted that the format is case-sensitive.

The DTD is defined as follows:

```
<!--
   Name:     EventLog.dtd
   Version:  1.0
-->

<!ELEMENT simsecurityeventlog (version, header,
   (controlevent | gameevent | summaryevent | userevent |
   componentevent | zoneevent | alertevent)*)>
<!ELEMENT version              (#PCDATA)>
<!ELEMENT header               (idtag?, scenario)>
<!ELEMENT idtag                (#PCDATA)>
<!ELEMENT scenario             (#PCDATA)>

<!ELEMENT controlevent         (dtimereal, dtimegame,
   (debuglog | logtrigger | popuptrigger | tickertrigger | savetrigger))>
<!ELEMENT debuglog             (tagdata?, message)>
<!ELEMENT logtrigger           (tagdata?, message)>
<!ELEMENT popuptrigger         (tagdata?, message)>
<!ELEMENT tickertrigger        (tagdata?, message)>
<!ELEMENT savetrigger          (tagdata?, filename)>
<!ELEMENT dtimereal            (#PCDATA)>
<!ELEMENT dtimegame            (#PCDATA)>
<!ELEMENT tagdata              (#PCDATA)>
<!ELEMENT message              (#PCDATA)>
<!ELEMENT filename             (#PCDATA)>

<!ELEMENT gameevent            (dtimereal, dtimegame,
   (start | end | pause | resume | save | exit | quit))>
```

```
<!ELEMENT start              EMPTY>
<!ELEMENT end                (#PCDATA)>
<!ELEMENT pause              EMPTY>
<!ELEMENT resume             EMPTY>
<!ELEMENT save               (filename)>
<!ELEMENT exit               EMPTY>
<!ELEMENT quit               EMPTY>

<!ELEMENT summaryevent       (dtimereal, dtimegame, (daily | monthly))>
<!ELEMENT daily              (budget, sales, salaries, hardwareexp, softwareexp, misc,
cash)>
<!ELEMENT monthly            (budget, sales, salaries, hardwareexp, softwareexp, misc,
cash)>
<!ELEMENT budget             (#PCDATA)>
<!ELEMENT sales              (#PCDATA)>
<!ELEMENT salaries           (#PCDATA)>
<!ELEMENT hardwareexp        (#PCDATA)>
<!ELEMENT softwareexp        (#PCDATA)>
<!ELEMENT misc               (#PCDATA)>
<!ELEMENT cash               (#PCDATA)>

<!ELEMENT userevent          (dtimereal, dtimegame, (hire | fire))>
<!ELEMENT hire               (name, salary)>
<!ELEMENT fire               (name, salary)>
<!ELEMENT name               (#PCDATA)>
<!ELEMENT salary             (#PCDATA)>

<!ELEMENT componentevent     (dtimereal, dtimegame, (buy | sell | configure))>
<!ELEMENT buy                (catalogname, componentname, cost)>
<!ELEMENT sell               (catalogname, componentname, cost)>
<!ELEMENT configure          (componentname, config?, procsec?)>
<!ELEMENT config             ((software | network | configbool | configval)*)>
<!ELEMENT software           (softwarename, boolean)>
<!ELEMENT network            (networkname, boolean)>
<!ELEMENT configbool         (field, boolean)>
<!ELEMENT configval          (field, value)>
<!ELEMENT procsec            (
   (procsecbool | procsecval | secrecyrange | integrityrange | access)*)>
<!ELEMENT procsecbool        (field, boolean)>
<!ELEMENT procsecval         (field, value)>
<!ELEMENT secrecyrange       (min?, max?)>
<!ELEMENT integrityrange     (min?, max?)>
<!ELEMENT access             (user, accessmode)>
<!ELEMENT catalogname        (#PCDATA)>
<!ELEMENT componentname      (#PCDATA)>
<!ELEMENT cost               (#PCDATA)>
<!ELEMENT softwarename       (#PCDATA)>
<!ELEMENT networkname        (#PCDATA)>
<!ELEMENT field              (#PCDATA)>
<!ELEMENT boolean            (#PCDATA)>
<!ELEMENT value              (#PCDATA)>
```

```
            <!ELEMENT min                   (#PCDATA)>
            <!ELEMENT max                   (#PCDATA)>
            <!ELEMENT accesslist            (#PCDATA)>
            <!ELEMENT accessmode            (#PCDATA)>

            <!ELEMENT zoneevent             (dtimereal, dtimegame,
               zonename, secrecy?, integrity?, physicalsecurity?)>
            <!ELEMENT physicalsecurity      ((physecbool | permitteduser)*)>
            <!ELEMENT physecbool            (field, boolean)>
            <!ELEMENT permitteduser         (user, boolean)>
            <!ELEMENT zonename              (#PCDATA)>
            <!ELEMENT secrecy               (#PCDATA)>
            <!ELEMENT integrity             (#PCDATA)>
            <!ELEMENT user                  (#PCDATA)>

            <!ELEMENT alertevent            (dtimereal, dtimegame, (indicator | attack))>
            <!ELEMENT indicator             (securitytarget, targetname, message)>
            <!ELEMENT attack                (securitytarget, targetname, attacktype, result)>
            <!ELEMENT securitytarget        (#PCDATA)>
            <!ELEMENT targetname            (#PCDATA)>
            <!ELEMENT attacktype            (#PCDATA)>
            <!ELEMENT result                (#PCDATA)>
```
Figure 58. Event Log Document Type Definition.

The following table details the elements and attributes defined in the XML/DTD file.

Element/ Attribute	Description	Reqd *	Format
version	Version of the event log format.	R	9[99].9[99] e.g. “1.0”
header	**Header block.**	**R**	
idtag	The idtag associates the log file with the campaign-scenario-player involved. If the idtag is absent, it implies that the scenario was played independently.	O	String of at most 128 characters: <campaign name>/<scenario name>/<userID
scenario	Name of the scenario	R	String of at most 128 characters
control event	**Control event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)

			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

debuglog	[controlevent : debuglog] For general-purpose debug logging, typically generated by CyberCIEGE and is of no interest for Campaign Analysis purposes.	C	
tagdata	Can be optionally defined by the scenario designer to provide data to associate this particular log.	O	String of at most 256 characters.
message	The debug message itself.	R	String of at most 1024 characters.
logtrigger	[controlevent : logtrigger] logtrigger is a control event for scenario-defined logging. It is generated when the trigger condition is met.	C	
tagdata	Can be optionally defined by the scenario designer to provide data to associate this particular log.	O	String of at most 256 characters.
message	The log message itself.	R	String of at most 1024 characters.
popup trigger	[controlevent : popuptrigger] popuptrigger is a control event for scenario-defined popups. It is generated when the trigger condition is met. In CyberCIEGE, this corresponds to a pop-up dialog appearing on the screen.	C	
tagdata	Can be optionally defined by the scenario designer to provide data to associate this particular log.	O	String of at most 256 characters.
message	The popup message.	R	String of at most 1024 characters.

tickertrigger	[controlevent : tickertrigger] tickertrigger is a control event for scenario-defined ticker messages. It is generated when the trigger condition is met. In CyberCIEGE, this corresponds to a ticker message scrolling across the screen.	C	
tagdata	Can be optionally defined by the scenario designer to provide data to associate this particular log.	O	String of at most 256 characters.
message	The ticker message.	R	String of at most 1024 characters.
savetrigger	[controlevent : savetrigger] savetrigger is a control event for scenario-defined snapshots. It is generated when the trigger condition is met. In CyberCIEGE, this corresponds to a game being saved without student involvement. Effectively, we get a snapshot of the state of the game. The initial version of CyberCIEGE will not support this feature. Instead, CyberCIEGE will cue the student to do a manual save instead.	C	
tagdata	Can be optionally defined by the scenario designer to provide data to associate this particular log.	O	String of at most 256 characters.
filename	The filename of the saved game. The filename provided must be relative to the event log directory only. Hence, the directory path up to the event log directory shall not be included.	R	Filename.
gameevent	**Game event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss

			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

start	Indicates the start of the game - i.e. when the student is able to start performing actions. This occurs after the scenario has been loaded.	C	Empty - i.e. contains no value.
end	Indicates the result at the end of the game, where a game termination condition is reached. This is applicable only when the scenario has a termination condition.	C	“win” | “lose”
pause	Game was paused.	C	Empty - i.e. contains no value.
resume	Game resumed following a pause, upon completion of saving or upon completion of a reload of a previously saved game.	C	Empty - i.e. contains no value.
exit	Game was exited. May be “resume”d subsequently by reloading.	C	Empty - i.e. contains no value.
quit	Game was terminated before reaching termination condition.	C	Empty - i.e. contains no value.
save	[gameevent : save] Game was saved	O	
filename	Filename of the saved game. This shall be a fully-qualified filename.	C	Filename.
summary event	**Summary event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss

			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

daily	[summaryevent : daily] Daily summary. Generated at the end of each game day (at 2359H).	C	
budget	Daily funds budgeted in $.	R	Dollar amount.
sales	Daily sales in $.	R	Dollar amount.
salaries	Daily salaries paid in $.	R	Dollar amount.
hardware exp	Hardware bought today in $.	R	Dollar amount.
software exp	Software bought today in $.	R	Dollar amount.
misc	Misc daily fixed costs in $.	R	Dollar amount.
cash	Current cash balance in $. Cash balance should be = (budget + sales) - (salaries + hardwareexp + softwareexp + misc)	R	Dollar amount.
monthly	[summaryevent : monthly] Monthly summary. Generated at the end of each game month (at 2359H of the last day of the month). An initial monthly summary is to be generated immediately *before* the “game”-“start” event is logged. This is to indicate the start state.	C	
budget	Monthly budget in $.	R	Dollar amount.
sales	Monthly sales in $.	R	Dollar amount.
salaries	Monthly salaries paid in $.	R	Dollar amount.
hardware exp	Hardware bought this month in $.	R	Dollar amount.
software exp	Software bought this month in $.	R	Dollar amount.
misc	Misc fixed costs for this month $.	R	Dollar amount.

cash	Current cash balance in $. Cash balance should be = (budget + sales) - (salaries + hardwareexp + softwareexp + misc)	R	Dollar amount.
userevent	**User event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

hire	[userevent : hire] User hired.	C	
name	Name of user.	R	String of at most 64 characters.
salary	Salary of user in $.	R	Dollar amount.
fire	[userevent : fire] User fired.	C	
name	Name of user.	R	String of at most 64 characters.
salary	Salary of user in $.	R	Dollar amount.
component event	**Component event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

buy	[componentevent : buy] Component bought.	C	
catalog name	Catalog name that this component is an instance of.	R	String of at most 64 characters
component name	Name of component bought.	R	String of at most 64 characters.
cost	Cost of the component in $.	R	Dollar amount.
sell	[componentevent: sell] Component sold.	C	
catalog name	Catalog name that this component is an instance of.	R	String of at most 64 characters
component name	Name of component sold.	R	String of at most 64 characters.
cost	Cost of the component in $.	R	Dollar amount.
configure	[componentevent : configure] Component being configured.	C	
component name	Name of the component being configured.	R	String of at most 64 characters.
config	[componentevent : configure : config] Configuration setup group.	O	
software	[componentevent : configure : config : software] Software configuration.	C	
software name	Name of the software.	R	String of at most 64 characters.
boolean	Indicates whether the software is being	R	“true” | “false”

	installed (“true”) or uninstalled (“false”).		
network	[componentevent : configure : config : network]	C	
network name	Name of the network.		String of at most 64 characters.
boolean	Indicates whether the component is being attached (“true”) or detached (“false”) from the network.		“true” | “false”
configbool	[componentevent : configure : config : configbool] Configuration setting which is a boolean type.	C	
field	Name of the configuration setting. e.g.: “RemoteAuthentication”| “AcceptPKICerts”| “UseOneTimePasswordToken”| “UseBiometrics”| “UseTokenPKICerts”| “UseClientPKICerts”| “VPNClient”| “ScanEmailAttachments”| “StripEmailAttachments”| “AutomaticLockLogout”| “SelfAdminister”| “SelfAdministerMAC”| “AdministerSoftwareControl”| “BlockRemovableMedia”| “BlockLocalStorage”| “BrowserSettingLoose”| “BrowserSettingNormal”| “BrowserSettingStrict”| “EmailSettingsLoose”| “EmailSettingsNormal”| “EmailSettingsStrict”| “UpdatePatchesAsReleased”| “UpdatePatchesRoutinely”| “UpdatePatchesAutomatically”| “UpdateAntivirusRegular”| “UpdateAntivirusAutomatic”| “UninterruptiblePower”| “AdminBackup”| “OffsiteBackup”	R	String of at most 64 characters.

boolean	Indicates whether the configuration setting is being applied (“true”) or removed (“false”).	R	“true” | “false”
configval	[componentevent : configure : config : configval] Configuration setting which is a value-based type.	C	
field	Name of the configuration setting. e.g. “PasswordLength” | “ChangeFrequency” | “PasswordComplexity” | …	R	String of at most 64 characters.
value	Value of the configuration setting. Note that there are valid values associated with each specific field.	R	String of at most 64 characters.
procsec	[componentevent : configure : procsec] Procedural security.	O	
procsecbool	[componentevent : configure : procsec : procsecbool] Procedural security setting which is a boolean type.	C	
field	Name of the procedural security. e.g.: “HoldsUserAsset” | “ProtectWithACL” | “WriteDownPasswords” | “LockerLogoff” | “NoEmailAttachmentExecute” | “NoExternalSoftware” | “NoUseOfModems” | “NoWebMail” | “NoMediaLeaveZone” | “UpdateAntiVirus” | “ApplyPatches” | “LeaveMachinesOn” | “NoPhysicalModifications” | “UserBackup”	R	String of at most 64 characters.
boolean	Indicates whether the procedural security is being applied (“true”) or removed (“false”).	R	“true” | “false”

procsecval	[componentevent : configure : procsec : procsecval] Procedural security setting which is a value-based type.	C	
field	Name of the procedural security. e.g. “PasswordLength” | “PasswordCharacterSet” | “PasswordChangeFrequency” | …	R	String of at most 64 characters.
value	Value of the procedural security. Note that there are valid values associated with each specific field.	R	String of at most 64 characters.
secrecy range	[componentevent : configure : procsec : secrecyrange] Secrecy range.	C	
min	Minimum secrecy level.	R	String of at most 64 characters.
max	Maximum secrecy level.	R	String of at most 64 characters.
integrity range	[componentevent : configure : procsec : integrityrange] Integrity range.	C	
min	Minimum integrity level.	R	String of at most 64 characters.
max	Maximum integrity level.	R	String of at most 64 characters.
access	[componentevent : configure : procsec : access] Access rights to the component (accesslist).	C	
user	Name of user or group.		String of at most 64 characters.
accessmode	Access mode is specified by 4 attributes of read, write, control and execute.		AAAA Each A can be “Y”es, “N”o or “X” for don’t care. e.g. “YYXX” has read and write,

			but control and execute are don't care.
zoneevent	**Zone event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month ("01" to "12"
			DD = Day ("01" to "31")
			hh = Hour ("00 to 23")
			mm = Minutes ("00" to "59")
			ss = Seconds ("00" to "59")

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month ("01" to "12"
			DD = Day ("01" to "31")
			hh = Hour ("00 to 23")
			mm = Minutes ("00" to "59")
			ss = Seconds ("00" to "59")

zonename	Name of the zone.	R	String of at most 64 characters.
secrecy	Intended secrecy level for people entering the zone.	O	String of at most 64 characters.
integrity	Intended integrity level for people entering the zone.	O	String of at most 64 characters.
physical security	[zoneevent : physicalsecurity] Physical security measures applied to the zone.	O	
physecbool	[zoneevent : physicalsecurity : physecbool] Physical security measure setting of a boolean type.		
field	Name of the physical security measure. e.g. "Receptionist"| "GuardAtDoor"| "PatrollingGuard"| "ProhibitMedia"| "ProhibitPhoneDevices"| "ExpensivePerimeterAlarms"| "ModeratePerimeterAlarms"| "ReinforcedWalls"|	R	String of at most 64 characters.

	“SurveillanceCameras”| “PermitEscortedVisitors”| “VisualPeopleInspection”| “XrayPackages”| “KeyLockOnDoor”| “CipherLockOnDoor”| “ExpensiveIrisScanner”| “ModerateIrisScanner”| “Badges”		
boolean	Indicates whether the physical security measure is being applied (“true”) or not (“false”).	R	“true” | “false”
permitted user	[zoneevent : physicalsecurity : permitteduser] Permitted users to a zone.	O	
user	Name of user or group.	R	String of at most 64 characters.
boolean	Indicates whether the user is being added (“true”) or removed (“false”) from the list of permitted personnels for entering the zone.	R	“true” | “false”
alertevent	**Alert event block.**	**O**	

dtimereal	Date/time of real world.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

dtimegame	Date/time of game simulation.	R	YYYYMMDDhhmmss
			
			YYYY = Year
			MM = Month (“01” to “12”
			DD = Day (“01” to “31”)
			hh = Hour (“00 to 23”)
			mm = Minutes (“00” to “59”)
			ss = Seconds (“00” to “59”)

indicator	[alertevent : indicator] Indicator of a possible ongoing attack (including false positives).	C	
security	Security target that the indicator pertains to.	R	“zone” | “component”

target	e.g. “zone”, “component”, …		
targetname	Name of the target.	R	String of at most 64 characters.
message	The indicator message (which is also displayed to the student in CyberCIEGE).	R	String of at most 1024 characters.
attack	[alertevent: attack] Actual attack that was generated by CyberCIEGE. This represents the actual occurrence of the attack that is not revealed to the student in CyberCIEGE, in contrast to the indicator.	C	
security target	Security target that the indicator pertains to. e.g. “zone”, “component”, …	R	“zone” | “component”
targetname	Name of the target.	R	String of at most 64 characters.
attacktype	Type of attack (refer to “Legal Attacker Moves” document).	R	String of at most 1024 characters.
result	Result of the attack - whether was it successful (“true”) or if the defensive measures were successful in stopping it (“false”).	R	“true” | “false”

- Reqd - indicates whether the element is (R)equired or (O)ptional within the parent block. (C)hoice implies that at least one of the elements must appear within the block.
- Dollar amount is specified in the form 9[9…]. e.g. $25 is “25”, $1,200 is “1200”.

Table 6 Element/attribute description.

APPENDIX D. INTERFACE SPECIFICATIONS WITH KEN JOHNS (SCENARIO DEFINITION TOOL)

A. PURPOSE

The purpose of this appendix is to define the interface specifications with Ken John’s thesis effort which aims to develop the Scenario Definition Tool.

B. TAXONOMY FILE FORMAT

The Taxonomy terms are defined as hierarchical relationships. The XML format is a convenient syntax for representing such a structure. Defined here is the DTD for this purpose:

```
<!--
   Name:     Taxonomy.dtd
   Version:  1.0
-->

<!ELEMENT simsecuritytaxonomy  (tnode)*>
<!ELEMENT tnode                (tname, (tnode)*)>
<!ELEMENT tname                (#PCDATA)>
```

Figure 59. Taxonomy Document Type Definition.

The XML data file holding the Taxonomy terms are stored in the corresponding Taxonomy.xml file. An example is shown as follows:

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE simsecuritytaxonomy SYSTEM “Taxonomy.dtd”>

<simsecuritytaxonomy>
        <tnode>
                <tname>Encryption</tname>
                <tnode>
                        <tname>Public Key Encryption</tname>
                        <tnode>
                                <tname>RSA</tname>
                        </tnode>
                </tnode>
                <tnode>
                        <tname>Symmetric Key Encryption</tname>
                </tnode>
        </tnode>
        <tnode>
                <tname>E-voting</tname>
        </tnode>
```

```
</simsecuritytaxonomy>
```

Figure 60. Taxonomy.xml sample.

C. EMBEDDED TAXONOMY TAGS

When Scenario Definition Files are created, scenario tagging can be optionally performed. This is done by selecting Taxonomy terms from the Taxonomy.xml file. The Taxonomy terms can be embedded anywhere within the Scenario Definition File, prefixed by comments (i.e. "//"). Each Taxonomy term is bounded by a "TaxonomyTag:" prefix and ":end" suffix pair as shown below:

```
:
// TaxonomyTag: Taxonomy Term #1 :end
// TaxonomyTag: Taxonomy Term #2 :end
:
```

Figure 61. Embedded Taxonomy tags.

D. SCENARIO EDITOR PROGRAM PARAMETERS

To enable the Campaign Analyzer to review the scenario definition, the Scenario Definition Tool will need to support the following program parameters:

ScenarioDefinitionTool -s <Scenario File>

This loads the given <Scenario File> into the the Scenario Definition Tool to enable the Instructor to review the details.

Program Parameter	Description	Format
-s <Scenario File>	Filename of the scenario to be loaded.	Standard filename format.

APPENDIX E. SOURCE CODES

A. PURPOSE

The purpose of this appendix is to list the source code and configuration filenames for the application modules developed. The sections are organized according to the directory structure of the Java source codes and configuration files.

Each application module is placed in a separate directory. In addition, base classes and utility classes are stored in the "Utility" directory. Configuration files are stored in a separate "bin" directory. A brief description of each file is provided.

B. TAXONOMY MANAGER

Filename	Description
TaxonomyManager.java	The main program class for the Taxonomy Manager module.
TaxonomyCtl.java	Controller (control object) for the Taxonomy Manager.
TaxonomyGUI.java	Main GUI (boundary object) for the Taxonomy Manager.
TaxonomyBuilder.java	Builder class for reading/writing to the Taxonomy.xml.
Taxonomy.java	Taxonomy (entity object).

Table 7 Taxonomy Manager source codes.

C. CAMPAIGN MANAGER

Filename	Description
CampaignManager.java	The main program class for the Campaign Manager module.
CampaignManagerCtl.java	Controller (control object) for the Campaign Manager.

CampaignManagerGUI.java	Main GUI (boundary object) for the Campaign Manager.
CampaignGUI.java	GUI (boundary object) for editing a campaign.
ScenarioFilterGUI.java	GUI (boundary object) for defining the filter.
CampaignCatalogBuilder.java	Builder class for reading/writing to the CampaignCatalog.xml.
CampaignBuilder.java	Builder class for exporting a campaign to a folder.
CampaignRelease.java	Stores the campaign data for a releasable campaign (entity object).
Campaign.java	Stores the campaign data (entity object) for a campaign being edited.
Scenario.java	Stores the scenario data.
ScenarioCatalogBuilder.java	Builder class to recursively parse and assemble a list of available scenarios.

Table 8 Campaign Manager source codes.

D. CAMPAIGN PLAYER

Filename	Description
CampaignPlayerTool.java	The main program class for the Campaign Player module.
CampaignPlayerCtl.java	Controller (control object) for the Campaign Player.
CampaignPlayerGUI.java	Main GUI (boundary object) for the Campaign Player.
XMLFilter.java	XML filter.
CampaignPlayBuilder.java	Builder class for reading a campaign.xml file.
CampaignPlayer.java	Defines a student (entity object).
CampaignPlay.java	Defines a campaign being played (entity object).

ScenarioPlay.java	Defines a scenario being played (entity object).
ScenarioDefinitionFile.java	Defines a scenario definition (entity object).

Table 9 Campaign Player source codes.

E. CAMPAIGN ANALYZER

Filename	Description
CampaignAnalyzer.java	The main program class for the Campaign Analyzer module.
CampaignAnalyzerCtl.java	Controller (control object) for the Campaign Analyzer.
CampaignAnalyzerGUI.java	Main GUI (boundary object) for the Campaign Analyzer which provides a campaign-level summary view of student event logs.
EventLogGUI.java	GUI for the Event Log Analyzer which presents the detailed event log of a single student.
EventTableModel.java	A model for holding a table structure of logged events.
PropertyTableModel.java	A model for holding a table structure of the sub-events.
StudentTableModel.java	A model for holding a table structure of student data.
ScenarioEventLog.java	Defines a Scenario Event Log File (entity object).
LogEvent.java	Defines a single log event (entity object).
PlayerStatus.java	Maintains the summary status values of a student for a given scenario.

Table 10 Campaign Analyzer source codes.

F. UTILITY

Filename	Description
CampaignResource.java	Utility class to handle resource property definitions
CustomDateTime.java	A customized date/time class to handle the date/time formats used.
Helper.java	A singleton class providing miscellaneous useful functions.
SDFilenameFilter.java	FilenameFilter class for Scenario Definition Files.
StdTableModel.java	Base class for the TableModel used in JTable.
XMLBuilder.java	Base class for XML-based builder classes.
XMLFilter.java	Implements a FileFilter for XML files (i.e. *.xml).
XMLHelper.java	Defines various XML-related constants.
StringVector.java	Implements a Vector class of String objects.

Table 11 Utility and base classes.

G. BIN

Filename	Description
Campaign.properties	The resource property definitions
Taxonomy.dtd	DTD for the Taxonomy data.
Taxonomy.xml	XML file to store the Taxonomy hierarchy.
CampaignCatalog.dtd	DTD for the catalog of campaigns.
CampaignCatalog.xml	XML file to store the catalog of campaigns.
CampaignRelease.dtd	DTD for released campaigns.
EventLog.dtd	DTD for the event log file.

DTD: Document Type Definition; XML: Extensible Mark-up Language.

Table 12 Resource files.

LIST OF REFERENCES

[Anderson 1972] James Anderson. "Computer Security Technology Planning Study". Technical report ESD-TR-73-5, vol II, USAF Electronics Systems Division, pp. 1 to 5. Oct 1972.

[Boyce 2002] Joseph Boyce and Dan Jennings. "Information Assurance: Managing Organization IT Security Risks". Butterworth-Heinemann, Woburn, MA, pp. 32 to 34. 2002.

[Bruegge 2000] Bernd Bruegge and Allen Dutoit. "Object-Oriented Software Engineering: Conquering Complex and Changing Systems". Prentice Hall, Upper Saddle River, NJ, pp 134 to 135. 2000.

[DOA 1993] Department of the Army. "A Leader's Guide to After-Action Review". http://call.army.mil/products/spc_prod/tc25-20/table.htm, Training Circular 25-20, US Army Combined Arms Center. Last accessed: Sep 1993.

[DON 2000] Department of the Navy. "Introduction to Information Assurance Publication". IA Pub 5239-01, Department of the Navy, pp. 2 to 3, 15 to 17. May 2000.

[DSTA 2003] Singapore Defence Science and Technology Agency. "PC War Games to Enhance Soldiers' Training". DSTA Vista newsletter, Vol 21, pp. 1, 10. May 2003.

[Irvine 2003] Cynthia Irvine and Michael Thompson. "Teaching Objectives of a Simulation Game for Computer Security". Proceedings of Informing Science and Information Technology Joint Conference, Pori, Finland. Jun 2003.

[Jacobson 1999] Ivar Jacobson, Grady Booch and James Rumbaugh. "Unified Software Development Process", Addison-Wesley. Feb 1999.

[Joint 1998] Joint Chiefs of Staff. "Joint Doctrine for Information Operations". Joint Pub 3-13, published under the direction of the Chairman of the US Joint Chiefs of Staff, pp. I-1 to I-6, III-1 to III-4. Oct 1998.

[Leffingwell 1999] Dean Leffingwell and Don Widrig. "Managing Software Requirements: A Unified Approach". Addison-Wesley, Indianapolis, IN, pp. 42, 252 to 270. 1999.

[Livingston 2003] Brian Livingston. "Patches that Patch". eWeek magazine, 17 Nov 2003 issue, pp. 54. Nov 2003.

[Morrison 1999] John Morrison and Larry Meliza. "Foundations of the After Action Review Process". http://call.army.mil/products/spc_prod/aar/aar.htm, Special Report 42 by the US Army Research Institute for the Bahavioral and Social Sciences. Jul 1999.

[Nexus 2003] Nexus Interactive. "AI Wars: The Awakening". http://www.aiwars.com. Last accessed: Nov 2003.

[Rivermind 2003] Rivermind, Inc. "CyberCIEGE: Scenario Definition File Format". File format for specifying the scenario definition co-developed by NPS and Rivermind. 2003.

[Salzer 1975] Jerome Saltzer and Michael Schroeder. "The Protection of Information in Computer System". Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308. Sep 1975.

[Seo 2002] James Jung-Hoon Seo. "Reading the Look and Feel: Interface Design and Critical Theories". http://acg.media.mit.edu/people/jseo/courses/cms800/final-paper.html, pp. 2 to 5. Jan 2002.

[Saunders 2003] John Saunders. "The Case for Modeling and Simulation of Information Security". http://www.johnsaunders.com/papers/securitysimulation.htm, National Defense University. Last accessed: Dec 2003.

[Sun 2003] Sun Microsystems. "Java 2 Platform, Standard Edition (J2SE)". Download site for the Java 2 API. http://java.sun.com/j2se/. Last accessed: Aug 2003.

[Tanner 2002] Michael Tanner, Christopher Elsasser and Gregory Whittaker. "Security Awareness Training Simulation". http://www.mitre.org/work/tech_papers/tech_papers_01/tanner_security/tanner_se curity.pdf. Cognitive Science and Artificial Intelligence Center, The MITRE Corporation, pp. 1 to 3. 14 Jan 2002.

[USArmy 2003] US Army. "America's Army". US Army website for America's Army. http://www.americasarmy.com/. Last accessed: Aug 2003.

[W3C 2003] World Wide Web Consortium (W3C). "Extensible Mark-up Language (XML)". W3C website for XML resources. http://www.w3.org/XML/. Last accessed: Oct 2003.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Dr. Ernest McDuffie
   National Science Foundation
   Arlington, VA

4. Dr. Carl Landwehr
   National Science Foundation
   Arlington, VA

5. RADM Zelebor
   N6/Deputy DON CIO
   Arlington, VA

6. Russell Jones
   N641
   Arlington, VA

7. David Wirth
   N641
   Arlington, VA

8. David Wennergren
   Headquarters U.S. Navy
   Arlington, VA

9. CAPT Sheila McCoy
   Headquarters U.S. Navy
   Arlington, VA

10. CAPT Robert Zellmann
    CNO Staff N614
    Arlington, VA

11. Dr. Ralph Wachter
    ONR
    Arlington, VA

12. Steve LaFountain
NSA
Fort Meade, MD

13. Dr. Vic Maconachy
NSA
Fort Meade, MD

14. Richard Hale
DISA
Falls Church, VA

15. George Bieber
OSD
Washington, DC

16. Deborah Cooper
DC Associates, LLC
Roslyn, VA

17. David Ladd
Microsoft Corporation
Redmond, WA

18. Marshall Potter
Federal Aviation Administration
Washington, DC

19. Ernest Lucier
Federal Aviation Administration
Washington, DC

20. RADM Joseph Burns
Fort George Meade, MD

21. Dr. Greg Larson
IDA
Alexandria, VA

22. Daniel Wolf
NSA
Fort Meade, MD

23. Penny Lehtola
NSA
Fort Meade, Maryland

24. Ray A. Letteer
Head, Information Assurance, HQMC C4 Directorate
Washington, DC

25. Cynthia Irvine
Naval Postgraduate School
Monterey, CA

26. Michael Thompson
Naval Postgraduate School
Monterey, CA

27. Yeo Tat Soon
Temasek Defence Systems Institute
National University of Singapore
Singapore

28. Seah Siew Hwee
Defence Science & Technology Agency
Singapore

29. Tang Bee Theng
Defence Science & Technology Agency
Singapore

30. Teo Tiat Leng
Defence Science & Technology Agency
Singapore