An Experience with Three Scenario-Based Methods: Evaluation and Comparison

Dejan Petkovic and Gitesh K. Raikundalia

School of Engineering and Science, Victoria University, PO Box 14428, Melbourne 8001, Australia

Summary
Scenario-based design is a family of methodologies that strives to concretely describe systems during early stages of software process activities. Sound theories behind proposed scenario-based methods suggest a possibility of software process improvement. This paper covers practical application of three selected scenario-based design methods in regards to software system analysis and design. Web browser systems, given that most scenario-based methods are user-centred in nature and web browser is user-centred software, are chosen as a development target. Based on analysis, design and implementation the selected scenario-based methods are evaluated and compared against each other. Consequently, advantages of each scenario-based method are discussed.

Key words:
Scenario-based design, scenario-based modeling, hierarchical, scenario-based engineering process, SEP, scenarios with functions and goals, user-oriented, user-centered

1. Introduction

Unfortunately at present, developed software systems still suffer from many deficiencies. The cause of deficiencies in a final software system in many cases is a lack of adherence to software engineering principles. Much software development failure arises due to poor requirements elicitation and analysis and unsound architecture decisions, leading to poor system design.

Although appropriate software lifecycles, design paradigms and implementation paradigms may be chosen by practitioners, final software product failure does occur. Regrettably a sufficient number of software practitioners disregard prescribed software engineering's process sub-activities, or place minimal importance on these, and rush towards implementation. Implementation results are often materialised with either failure to meet a project deadline, unusable/unstable software product or inadequate software testing leading to costly process of maintenance.

Various software design approaches have been proposed that attempt to either improve design methods or in themselves are ways of producing software. Amongst these approaches, there is a set of techniques collectively termed *scenario-based design*. Scenarios are basically narrative descriptions of sequence of events, culminating in achievement of a specific event. It is expected that narratives, or stories, are understood by every one. Since scenarios hold such basic properties, they allow inclusion of every concerned party into software engineering processes. Inclusion of future users and domain experts provides accurate knowledge acquisition and representation, thereby allowing agreement and consistency of specifications the software system needs to provide. Scenario-based design methods in general cover most important activities leading to implementation: requirements, analysis and design. Furthermore, scenario-based techniques are extended to software architecture, testing and reengineering.

In this paper, the practical application of three selected scenario-based methods to software system development is presented. Through practical application of three scenario-based methods, the aim is to determine does the scenario-based design offer any significant results to software engineering. Selection is based on the criteria that techniques cover at least analysis and design activities leading to implementation. Selected methods are applied to a design of a simple web browser, and each resulting design is implemented. Results of scenario-based methods are evaluated and compared against each other.

2. Scenario-Based Design

It is out of the scope of this paper to cover a topic of scenario-based design in general, as well existing number of proposed methods available in public domain. However, for the interested readers, sound theory can be found in publication by John M. Carroll [1] where benefits of scenario use are discussed.

Further discussion about scenarios and scenario-based design can be found in Carroll's collaboration with Mary Beth Rosson [2]. For readers who might be interested in requirements engineering, work carried out by CREWS (cooperative requirements engineering with scenarios)

Manuscript received January 5, 2009
Manuscript revised January 20, 2009

project [3, 15]. Other source for scenario-based requirements engineering are works by Alistair Sutcliffe and good starting point, besides a number of other proposed methods, would be Sutcliffe's publication on analysis [4]. Efficient use of scenarios and prototypes presented in work by Dzida and Freitag [16] may assists with validation and verification of requirements.

Important aspect of software process is definition of appropriate architecture and Rick Kazman et all [17, 18] have published applicable means of using scenarios to address architecture issues.

Except for above mentioned scenario-based references, there are a substantial numbers of proposed methods that are applicable to requirements, testing and architecture definition. The three selected scenario-based methods we have applied cover analysis and design, and are based on works by Xiaoying Bai et all [6], Erik Mettala et all [8] and Hermann Kaindl [9].

3. Scenario-Based Methods

Most of the proposed scenario-based methods focus only on specific stages of the development lifecycle, such as requirements engineering, architecture, system testing, etc. Here we will describe the main points of the three selected scenario-based design methods. Selected methods have been chosen on the basis that they cover at least analysis and/or design stages.

Scenario-Based Modelling
Scenario-Based Modelling [6] method is founded on the ideas of hierarchical organisation and functional composition/decomposition of scenarios. The method aims to support modelling of user actions, system functions, system data and system usages/internal processes. Three types of design activities are distinguished in this scenario method *functional view, data view and usage view*.

In first activity, functional view, scenarios are used to model and represent both user actions and system functions. In the functional view activity, scenarios are arranged hierarchically into a scenario tree. Hierarchical organisation creates a scenario model that is to be equivalent to functionalities system provides.

The second activity is creation of a data view model. Data is modelled by identifying and extracting information from hierarchically organised scenarios created in the functional view activity. Identified data is organised hierarchically, once again, and associated with scenarios. The created hierarchical data model represents data processed by the system. Third activity, usage view, aims to model complex system usages and internal processes. This is achieved by identifying number of scenarios; scenarios that, when executed together, result in achievement of specific functionality. The usage view model is created by grouping and sequentially ordering these scenarios using control structures.

Scenarios with Goals and Functions
Idea of creating scenarios in such a way as to hold within information about functions required to execute scenario and information about scenarios purpose (users' goals) can be found in [7, 8]. Scenario is a sequence of actions performed in order to achieve certain objective. Actions contained within scenarios are accomplished by functions of the system or by user interactions with the system. These actions and interactions result in achievement of user goals and validate usefulness of the system. The focus of this method is on determining by what means goals of users are achieved. Scenarios are tools used as the intermediary link between goals of the user and functions of the system.

Scenarios are modelled by combining (linking) them with both user's goals and functions which are required to achieve these goals. Author prescribes simple yet effective design sequences [8] for purposes of knowledge acquisition through scenarios, leading to system design. Combining scenarios, goals and functions, results in a scenario-based model which shows, how functions of the system will serve the goals of the user. Systematic design process based on this scenario model drives the system design. Design process is based on a traversal of set of rules specified by three distinct sequences: by known goals, by known functions or by known scenarios. The resulting design consists of scenarios interlinked with goals and functions.

Scenario-Based Engineering Process (SEP)
Authors [9] define SEP as "user-centred, architecture-based, iterative and prototype-focused" process. Focus of this method is on scenarios and scenarios' task analysis. Task analysis identifies new sets of tasks (smaller scenarios). Task analysis in regards to software engineering is a substantial field of research and good starting points for interested reader are [12, 13].

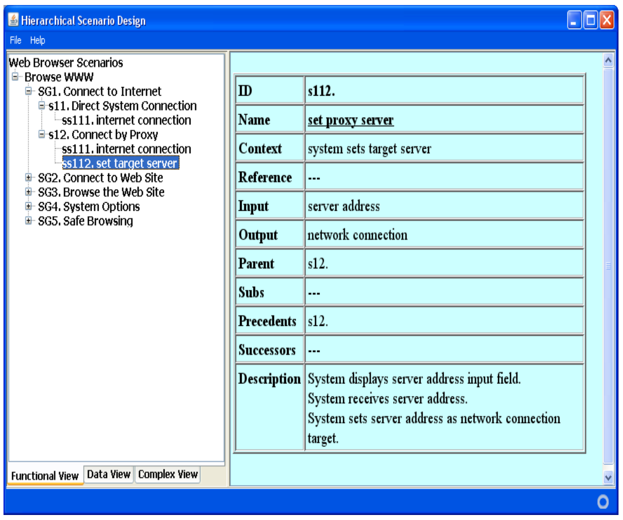
SEP process [9, 10] strives to deliver system analysis/design/development details for architecture-based component reuse. To achieve the reuse, scenarios are employed as a backbone to task analysis, and task analysis as a backbone to Domain-Specific Software Architecture (DSSA) [11] specification. All of created artefacts should

be stored in a database, or as SEP calls it "lessons learned repository", and updated accordingly.
SEP considers that each scenario is a limited domain in itself, hence incremental design and implementation of specific parts of the system should be done. As in incremental processes, early scenarios should be prototyped and directly implemented into the process. Such views promote user involvement in validation and verification of system requirements. This process is supplemented with traditional object-oriented concepts (analysis, design, class diagrams etc), or for that matter any other concept development team is used to work with.

4. Application of Scenario-Based Methods

Scenario-Based Modelling

System modelling using scenarios starts by functional decomposition of system requirements down to *scenarios* (users' point of view of system) and *sub-scenarios* (system's functions). Functionally related scenarios are then grouped into *scenario groups*. These scenarios are then hierarchically organised. Figure 1 shows hierarchical scenario organisation in case of a simple web browser system (connect, display, navigate, favourites, print, save functionalities). Scenario group (SG) "Connect to Internet" is shown and contains two scenarios with its relevant sub-scenarios. When the hierarchical scenario tree is expanded it contains 42 scenarios (excluding exceptional and alternative cases).



```
Web Browser Scenarios
  Browse WWW
    SG1. Connect to Internet
      s11. Direct System Connection
        ss111. internet connection
      s12. Connect by Proxy
        ss111. internet connection
        ss112. set target server
    SG2. Connect to Web Site
    SG3. Browse the Web Site
    SG4. System Options
    SG5. Safe Browsing
```

ID	s112.
Name	set proxy server
Context	system sets target server
Reference	---
Input	server address
Output	network connection
Parent	s12.
Subs	---
Precedents	s12.
Successors	---
Description	System displays server address input field. System receives server address. System sets server address as network connection target.

Figure 1 Hierarchical scenario organisation

Functional view activity, a process based entirely on notions of functional composition/decomposition of scenarios is time consuming and cumbersome. The notion that scenarios are understood by every stakeholder involved in software process is true. However, other existing techniques for analysis and design, such as use cases and interaction diagrams would produce more accurate design, in a shorter period of time. Next two activities of this method, data view and usage view, prescribed by authors, are insufficient in detail on how to perform the steps within. Likewise, applying the data view seemed nothing more then class modelling, which can be achieved more efficiently using a standard object-oriented approach, rather then by constructing scenarios in a scenario model. In regards to the usage view, where a number are scenarios is composed together to model system states and processes – again using known means, like state diagrams of finite state machines, would achieve more efficient results.

Scenarios with Goals and Functions

Traversing prescribed design sequences, in collaboration with the future user, (there are some known user goals, known system functions and known scenarios) produced a set of scenarios. Table 1 shows a structure of a procured scenario during a design process. Except for actions needed to achieve a certain scenario, required system functions are linked to actions within, as well what user goal the scenario achieves.

Table 1 Scenario with Goals and Functions

Scenario: Print	
2. User selects a print type to print the web page. *By-Function*: Select Print Option	1. System displays a web page. *By-Function*: Render Document 3. System checks the print option. *By-Function*: Check Print Option. 4. System processes print request. *By-Function*: Process Print Job. 5. System forwards print job to printing machine. *By-Function*: Initiate Print Job
Goal: Print Document *Goal*: Print Text	

Design sequences have proven to be effective indeed. Starting a design sequence by knowing two basic user goals; traversal of other sequences triggered creation of new scenarios and identification of new system functions.

In case of a simple web browser: user visits websites and user prints documents – six scenarios have been determined and all together 26 system functions have been identified. To further the design specifications the method involves the idea of functional representation to specify required system functions [8, 14].

It could be said that procured scenarios depicted no more then sets of functional requirements. However, efficient use of scenarios in this approach can be thought of as a valuable knowledge acquisition tool. In that way, a tool that does offer good insight into the system needed to be built. By using rules specified by design sequence of known goals, known functions and known scenarios — not only scenarios of use were elicited showing user/system interaction. Observing the scenarios clarified sequence of actions required to satisfy user's goal. Every action within a scenario had to have a function linked to it, a function that needs to be developed and integrated in the system. Scenarios are structured in such a way that on the left side user's actions/functions are specified and on the right side are system's responsibilities. Having a distinct view of user's actions gave a clear view of what is required from the user, and therefore the possibility to consider user interface design decisions.

Scenario-Based Engineering Process (SEP)
Applying SEP was a well defined, systematic design practice – main activities within SEP were domain analysis, DSSA [11] specification and application design. Domain analysis started by identifying, in collaboration with the user, a set of applications (required functionalities) system needed to support. The next step was creation of possible scenarios for each identified application, followed by the process of task analysis of each identified scenario. Scenarios and task analysis produced 53 well specified artefacts that were a basis for architecture definition and system application design.

Task analysis is the backbone of SEP and Table 2 shows one of the results of applying task analysis to Browse Websites application scenarios. In Table 2 the Search Query scenario has undergone process of the task analysis. Each identified sub-task has been refined further using alike task analysis structure. Final result of task analysis was a task model that provided information about required functionalities and what is needed to accomplish these. Scenarios combined with application of task analysis produced a wealth of information that were relied upon and employed in next stages of design activities. Inclusion of a formal design technique, in this case object-oriented concepts, assisted with specifics on how system should be structured.

Table 2 Task Analysis of Search Query Scenario

Tasks Analysis of Scenario 7	Process Search Query
Objective	search term forward by the system to the internet search engine
Task Responsibility	user and system
Precondition	keyboard interface, mouse interface, internetwork connection
Input	characters,numerals, combination of both, mouse click, keyboard event
Output	web document displaying results of search query
Task Description	User enters characters or combination of numerals and characters into the system. User forwards entry to the system by a mouse click or keyboard "enter" key event. System detects user action – either mouse click or keyboard event and validates user's entry. System validates if the entry is a search term or a domain name. System connects to internet search engine and forwards the search term, including application (agent) identification. Internet search engine returns search query results.
Sub-tasks	1. Detect user actions 2. Validate user entry. 3. Provide application identification and forward search term

5. Implementation and Results of Scenario-Based Methods Designs

In general, scenario-based design methods in theory seem highly applicable. However, they do suffer from lack of diagrammatic representations of the system to be built. Specifying and explaining behaviour of machines in natural human language does not produce sufficient information. Lack of diagrams is a major drawback to system design and implementation. However implementation differences between design results of each selected three methods are substantial.

Scenario-Based Modelling

From our experience Scenario-Based Modelling is no more then a requirements generator. The output of a design process, a number of functionally decomposed and grouped scenarios that are hierarchically organised provide developers with just that. A number of scenarios holding descriptions of what system has to provide. Essentially, the developer is left with a set of requirements. Implementing requirements adds a time consuming burden to software process, since design is non-existent. Trial-and-error development is something to be avoided at all costs. This scenario-based design method is good for determining what functionalities are required for the system and its use should stop there. Therefore, from our experience it can hardly be used to model the system and produce sufficient design that leads to implementation.

Scenarios with Goals and Functions

On the other hand, implementation of design by Combining Scenario with Goals and Functions method is far more effective. Implementation is somewhat easier due to the clear structure of scenarios. Having a number of scenarios that clearly state the user's goal, and clearly state how that goal is to be accomplished through execution of actions is valuable. Each action in a scenario has a function linked to it, a function that needs to be coded. Identified functions are specified using functional representation. Implementing such information at least gives developers clearer and well defined set of operations coupled with purpose. Figure 2 shows implementation of scenario Print from previous Table 1. Each scenario was implemented as a module since it was self-sufficient in its own right.

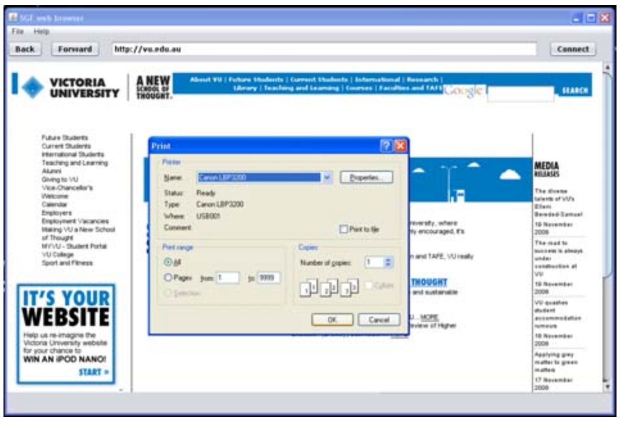


Figure 2 Scenarios Goals and Functions result

Scenario-Based Engineering Process (SEP)

Scenario-Based Engineering Process is a well structured requirements elicitation, task identification and software system design method. From a development point of view, scenarios give information about overall functionality that developers can refer to. Task analysis gives information that can be useful for specification of objects and methods. However, the final product of SEP design that developer refers to is a full class diagrams with well defined attributes and operators. These diagrams themselves are an abundance of information that developer needs. Figure 3 shows implementation result of storing a web address into a web browser.

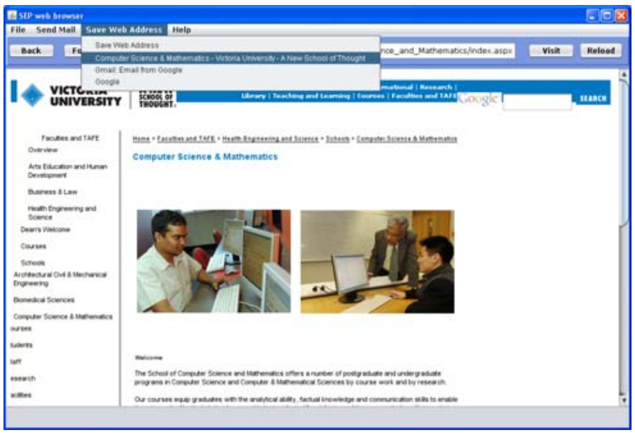

SEP web browser
File Send Mail Save Web Address Help
Back Visit Reload
Save Web Address
Computer Science & Mathematics - Victoria University - A New School of Thought
Gmail: Email from Google
Google
VICTORIA UNIVERSITY
Computer Science & Mathematics
SEARCH

Figure 3 SEP implementation result

From implementation point of view, SEP design produced well rounded information for developers. Of course, combination with object-oriented concept assisted a great deal. However, any other paradigm that designers and developers are comfortable with can be used, since SEP is flexible by nature in such sense. Consequently, it can be determined that the main strength of SEP design is in application of knowledge acquisition, using scenarios and task analysis.

6. Scenario-Based Methods' Design Process

Combining Scenarios with Goals and Functions is a user-centred goal-oriented technique that ensured elicitation of user goals and identification of required system functions. The user was constantly involved in scenario creation. Each scenario contained significant information that was applicable to implement the design. Structure of scenarios and information contained within clearly indicated user and system actions. Each scenario held association with user's goal. From implementation point of view, scenarios could be treated as modules that fulfil goals and actions within provided required functions for a specific module. On the other hand, interpretation of scenarios could be done in object-oriented terms (e.g. applying sequence diagrams, class modelling, etc) and implementation can be done on same terms.

Scenario-Based Engineering Process was a well defined set of knowledge acquisition, architecture definition and design activities. During the domain analysis activity, knowledge acquisition was supported by scenarios, task analysis and object-oriented analysis. The user defined scenarios in own terms and was involved throughout the domain analysis. Prototypes were used to convey design ideas to the user and incorporated into the final system. Furthermore design was supported by inclusion of reusable concept of DSSA and strengths of object-oriented design concepts.

Scenario-Based Modelling was a lengthy repetitive process of obtaining scenarios by applying functional decomposition to system requirements, then subsequently scenarios themselves to identify basic system functions. The user was not involved in scenario creation. The method aims to create a system model, and instead a set of system requirements was produced. Implementation based on requirements turned into a build-and-fix process.

7. Conclusion

Instead of traditional design methods, scenario-based design methodology was selected for system design as a proof of concept and inquiry into alternatives to system design. Work presented in this paper applied three scenario-based methods to development of a simple web browser system.

In regards to application of selected scenario-based design methods, positive experience acquired is that methods do concretely describe system to be built during early stages of software process. Another fact is that user collaboration is good, user felt positive about the outcome of software project. Collaboration and scenario creation constantly raised questions on design issues and promoted work focus. Lack of diagrammatic representation is major downfall. It is not feasible, using natural language, to convey design ideas of complex subject such is a software system. Therefore, scenario-based design ideas should be used either as analysis method or as a supplement to proven analysis/design techniques.

References

[1] Carroll J.M. (2000), Five reasons for scenario-based design, Interacting with Computers, Volume 13, Number 1, pp. 43-60

[2] Rosson M.B. & Carroll J.M. (2002), Scenario-Based Design, Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications. Lawrence Erlbaum Associates, 2002, pp. 1032-1050.

[3] http://crinfo.univ-paris1.fr/CREWS/Corps.htm

[4] Sutcliffe A. Scenario-Based Requirements Analysis, 1998

[5] Kazman, R.; Abowd, G..; Bass, L.; Clements, P. (1996), Scenario-based analysis of software architecture. IEEE Software pp47-55

[6] Bai, X. Tsai, W.T. Paul, R. Feng, K. Yu, L. (2002), Scenario-based modeling and its applications., Proceedings of the Seventh International Workshop on Object-Oriented Real-Time Dependable Systems, pp253-260

[7] Kaindl, H. (1995), An integration of scenarios with their purposes in task modeling. Proceedings of the 1st conference on Designing interactive systems: processes, practices, methods, & techniques, pp227 - 235

[8] Kaindl, H. (2000), A design process based on a model combining scenarios with goals and functions. IEEE Transactions on Systems, Man and Cybernetics, Part A,Volume 30, Issue 5, pp537 – 551

[9] Mettala, E., Cook, D.J., Harbison, K. (1996), Application of the Scenario-Based Engineering Process to the Unmanned Ground Vehicle Project, ARPA96 (627-642)

[10] Priest, J.W. Burnell, L. Haddock, G. Silva, J. (1998), Scenario-based systems design for quality engineering. IEEE International Conference on Systems, Man, and Cybernetics, pp4866-4871

[11] E. Mettala & M. Graham (1992), The Domain-Specific Software Architecture Program., Technical Report CMU/SEI-92-SR-009

[12] Johnson, P. Johnson, H. Waddington, R. Shouls, A. -Task-related knowledge structures: analysis, modelling and application. Proceedings of the Fourth Conference of the British Computer Society on People and computers IV, pp35 - 62

[13] Cuno Duursma, Olle Olsson, Ulf Sundin - Task Model definition and Task Analysis process (1994) , research project partially funded by the ESPRIT Programme

[14] Chandrasekaran, B. Goel, A.K. Iwasaki, Yumi (1993), Functional Representation as Design Rationale, Computer archive Volume 26, Issue 1, pp48-56

[15] http://sunsite.informatik.rwth-aachen.de/CREWS/crews-sum.htm

[16] Dzida, W. & Freitag, R. (1998), Making use of scenarios for validating analysis and design, IEEE Transactions on Software Engineering Volume 24, Issue 12, pp1182 – 1196

[17] Kazman, R.; Abowd, G..; Bass, L.; Clements, P. (1996), Scenario-based analysis of software architecture. IEEE Software pp47-55

[18] Kazman, R.; Klein, M.; Clements, P. (2000), ATAM: Method for architecture evaluation, CMU SEI Technical Note CMU/SEI-2000-TR-004, ADA382629, Software Engineering Institute, Pittsburgh, PA