SCENARIO-BASED SPECIFICATION AND EVALUATION OF ARCHITECTURES FOR HEALTH MONITORING OF AEROSPACE STRUCTURES[1]

Ravi Mukkamala[2], M.Britton and P. Sundaram

Department of Computer Science,

Old Dominion University, Norfolk, VA.

1. Introduction

HUMS systems have been an area of increased research in the recent times due to two main reasons: (a) increase in the occurrences of accidents in the aerospace, and (b) stricter FAA regulations on aircrafts maintenance [2]. There are several problems associated with the maintenance of aircrafts that the HUMS systems can solve through the use of several monitoring technologies. Currently, a variety of maintenance programs are institutionalized by the aircraft carriers that mostly involve visual inspections and hence are error-prone [3]. Automatic, continuous health monitoring systems could simplify the maintenance tasks as well as improve the efficiency of the operation, thereby enhancing the safety of air travel and also lowering the total lifecycle costs of aircrafts [1].

In addition to the common, well-known objectives, HUMS systems for aircrafts have expanded their array of activities in track with the developments in other HUMS domains [See 3-10]. The general objectives of HUMS include the following:

- To perform structural health monitoring,
- To perform operational health monitoring,
- To monitor usage of aircraft components,
- To perform prognostic and diagnostic analyses

All the objectives require a multi-disciplinary approach for successful implementation. For example, collective advancements in mechanical engineering, electronics and instrumentation, computer science etc., are essential for accomplishing the objectives.

Besides, HUMS systems must prove to be cost-effective as well as efficient in their functioning for the acceptance of the aircraft carriers in their maintenance programs [2]. Several problems continue to affect the credibility of HUMS systems like (a) high incidence of false alarm rates, (b) abundance and ambiguity of data present, (c) high cost of implementation, etc. [4]. So, researchers are focused on the development of tools and techniques that promote successful implementation of HUMS systems.

Scenarios are tools that are increasingly finding applications in several stages of system development lifecycle. They have been demonstrated for their effectiveness in requirements analysis, specification, design, documentation, user testing, and evaluation. We have employed scenarios in the development of HUMS in three main areas. They are: (a) to improve reusability by using scenarios as a library indexing tool and as a domain analysis tool, (b) to improve maintainability by recording design rationales from two perspectives – problem domain and solution domain, and (c) to evaluate the software architecture. It is our belief that scenarios can be highly valuable for developing systems that involve

[1] This work is supported in part by a research grant from NASA Langley Research Center, Hampton, Virginia.

[2] Contact author: Email: mukka@cs.odu.edu

designers from multiple disciplines and that exhibit real-time constraints.

This paper documents our methodology of employing scenarios in the specification and evaluation of architecture for HUMS. Section 2 investigates related works that use scenarios in software development. Section 3 describes how we use scenarios in our work, which is followed by a demonstration of our methods in the development of HUMS in section 4. Conclusion summarizes the results.

2. Related Work

Applications of scenarios could be classified into two categories depending on the target usage: (a) Human-computer interaction, where scenarios take an active role in improving usability, understanding user needs, enabling team building, etc., [15, 16, 18, 19] and (b) Software engineering, where scenarios are used to generate designs, to evaluate designs, to specify requirements, etc. [21-23].

Kyng explains how scenarios could be employed to effect cooperative design or user-centered design [17]. Three different types of scenarios – *use scenarios* for system envisionment, *explanation scenarios* for technical feasibility verification, *exploration scenarios* for new possibilities investigation – are explained. Our work differs from Kyng's in that we use scenarios especially to capture the effect of different constraints on the desired quality attributes of the system.

Scenarios could also be used as a framework of evaluation as in enabling designers to reuse design insights in a retrospective manner [20]. Our work too focuses on using scenarios for improving reusability in two ways: (a) as an indexing tool for the reuse library, (b) as a domain analysis tool.

SAAM is an evaluation method that uses scenarios in analyzing architecture for different quality attributes [23]. The architecture quality is analyzed by measuring the extent of code modifications required to implement a scenario. SAAM is well suited to be used during implementation stage, while ours is appropriate to be used during design stage to uncover the problems with the architecture and to improvise it before implementation.

ATAM[SM] is an architecture analysis method that enables the developers to understand the tradeoffs involved in the different design decisions and hence to evolve a better design [29]. Our use of scenarios in evaluation is based on ATAM[SM] but we improve on it by taking a microscopic view of effect of different quality attributes.

3. Specification and Evaluation of HUMS

Our work uses scenarios mainly to accomplish three objectives: (a) to improve reusability of components, (b) to improve maintainability and understandability of software, and (c) to evaluate the system for the desired quality attributes. Scenarios prove to be effective tools in the development of large, complex, real-time systems like HUMS. They also serve to preserve the priorities and the values of the team members, which vary across teams of different disciplines. This section explains how we use scenarios in the specification and the evaluation of HUMS.

There has been no consensus over the correct definition of scenarios and their level of granularity. However, this chaos has only contributed to the creative use of scenarios for various purposes rather than the other way around. Scenarios are generally used as a complimentary form of representation or specification in addition to textual descriptions and diagrammatic models, and take on various forms like stories, prototypes, mockups, etc. [13]. Scenarios could be defined as the narrative description of the interactions between the system and its environment, and are usually couched at two different levels: (a) Application level, which contains information about a specific functionality, and (b) Context level, which contains high-level information [14].

3.1 Development of Scenarios

Any system could be described as the transformations of users' tasks. Therefore, the first step in the development of any system involves capturing of the user interactions (usually as scenarios or use cases). We have evolved a heuristic to develop scenarios that serves to be beneficial

especially in the design of real-time systems. This heuristic is based on Jacobson's work [21].

- Identify all the actors of the system. "Actor is a role of object or objects outside of a system that interacts directly with it as part of coherent work unit [32]".
- Find out all interactions that an actor initiates with the system that involves transfer of data. Investigate the type of data, and its characteristics (e.g., data rate of sensors) in order to elicit as much requirements as possible.
- Identify all interactions that an actor initiates with the system that involves transfer of control.
- Identify all the interactions that an actor initiates with other actors.
- Repeat the steps 2 & 3 to identify all the interactions that the system initiates with the actors.

We followed these steps while we gathered the requirements for HUMS. This heuristic helped us to ensure completeness of the specification developed from requirements gathering.

3.2 Scenarios as Reuse Library Indexing Tool

Different types of scenarios are employed to serve different purposes ranging from recording of work situation overviews or problem descriptions, to the recording of evaluation results at the architecture design stage. These scenarios could serve as a valuable tool for developers to solve several problems in the system during initial development as well as during system maintenance in future.

According to surveys, more than 70% of the problems that technical managers face in new projects are already solved by them in past projects. Therefore, a library of past problems and the solutions adopted could expedite the decision making process in the new development work. We found that such a library could benefit a project like HUMS involving teams from disparate disciplines by serving as a common source of information for all members and at the same time promoting the priorities and the values of the different teams.

Scenarios could be stored in a library catalogued by different sets of indexes. These scenarios could be a combination of design insight scenarios, explanation scenarios, exploration scenarios, work situation overviews, etc. The indexes may be grouped in any order based on the desired target usage of the library. Such a provision would allow users to perform a simple keyword match as well as a complex search using other criteria like approaches to problem solving, design insights, evaluation results, etc. and thus would provide pointers to either effective solutions or potential pitfalls associated with a specific design for a specific problem. These systems could also help new learners to educate themselves from the past work making use of several problems and the solutions developed.

Providing access to a library of reusable work products is a key factor in ensuring successful implementation of library. The existing reuse methods from software engineering may not be appropriate for a complex need such as that outlined above. We found that the scenarios could provide the required flexibility to enable efficient access to the library, thus helping to increase the productivity of software development. We could eventually build case-based reasoning systems to provide us help on developing designs as well as to offer feedback to designs based on the historical knowledge base of all the cases, i.e., scenarios from the past. Thus, the specification of system architecture in terms of scenarios can enable us to improve accessibility to the reusable work products in a library.

3.3 Scenarios in Domain Analysis

One of our project objectives was to perform a domain analysis of HUMS systems and to develop a comprehensive, reusable, open standards-based generic architecture for structural health monitoring of aircrafts. Domain analysis is adopted by the software reuse community in order to increase the productivity of software development [24-25].

Domain study focuses on identifying the commonality and variability of applications in a domain. This understanding can help us to generalize and specialize the design components, making them applicable across applications and hence improving the reusability of components.

The HUMS domain has several systems built for different target environments like aircrafts, helicopters, spacecrafts, industry machinery, etc. So, domain analysis could help us to identify the commonality and variability of all the systems in the domain and hence enable us to improve reusability. The well-known reusability taxonomy, shown in figure (1), plays a cardinal role in domain analyses.

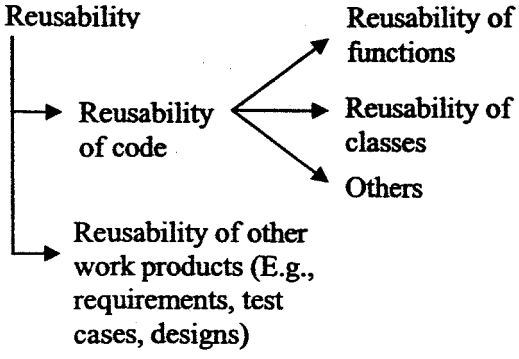

Figure (1) Reusability of Work Products: A Simple Taxonomy

Every system has *specific* objectives. Our domain analysis method dwells on the ability to describe any system in terms of its objectives (and design decisions) alone. Objectives of the system and the various design decisions made are captured using an objectives tree. Every node on the tree represents an objective or a design decision and the arcs in the tree links the objectives or design decisions to sub-objectives or sub-design decisions. Figure (2) presents a model of the objectives tree.

Scenarios play a central role in the development of the objectives tree. As scenarios capture all the possible user interactions, they form the basic list of objectives for the system. For example, "to gather sensor data" is a scenario of interaction with the system and it also represents an objective of the system. As an example of a scenario encapsulating a design decision, consider a user interaction such as "request prognostic analysis", which uses physics-based prognostics.

The objectives in an objectives tree could be mapped to specific classes or objects or sub-system. If an objective has to be implemented in exactly the same ways in multiple applications, then the mechanisms present in those applications that satisfy the objective is the commonality of the applications.

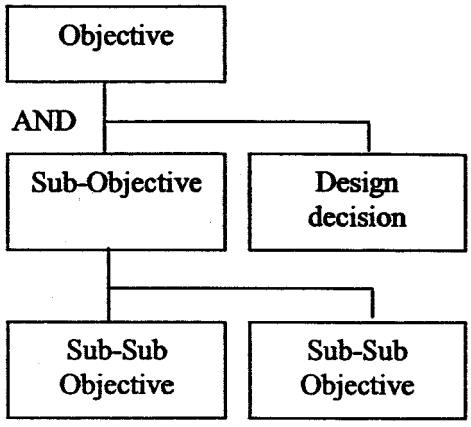

Figure (2) Objectives Tree Model

Table (1) Composition Rules for Objectives

Notation	Description
AND	All the sub-objectives or sub-decisions are used
OR	Any one of the sub-objectives or sub-decisions are used
AND/OR	Any combination of sub-objectives or sub-decisions is used. This is the default.

Variability of applications is not a straightforward issue as commonality among applications since variability could exist at the sub-system level or at the object level.

The mapping between the objectives tree and the object model of the system is utilized for performing the commonality-variability analyses. This method provides several distinct advantages over other methods like [25]. Various advantages are listed below:

- Objectives-based domain analyses aids object-based reuse paradigm,
- Objectives and design decisions form the logical starting point in system development and could be easily generated from scenarios,

- These models can be mapped to classes/objects/modules in the library thereby providing traceability to the realization of generalization or specialization,
- The organization obviates the need for special or complex notations because the objectives and decisions can be expressed hierarchically. See Table (1) for the set of composition rules.
- Hierarchy also allows separation of different issues that can be independently dealt with.

3.4 Scenarios to Improve Maintainability

Maintainability is defined as "the ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment" [26]. According to statistics, maximum time and effort is expended for system maintenance when compared to all other stages of system development lifecycle.

Understanding the design rationales can contribute to simplify and to improve the maintenance activities. We use scenarios to capture the design rationales from two perspectives – problem domain and solution domain – and thus offer a comprehensive solution. Such scenarios also serve as qualitative metrics to evaluate complex systems and thus act as substitutes when quantitative metrics don't exist.

Work situation overviews capture the design rationale information from the problem domain perspective, while the design insights document the same information from the solution domain perspective. This set of scenarios helps in the evaluation of architecture as well as serving as a continuum of understanding between initial developers and those of the future (who primarily indulge in maintenance). Evaluation of architecture, explained in the section 3.5, provides a quantitative account of the system design. The scenarios described here provide qualitative description of different design artifacts and can thus serve as qualitative measures/metrics.

Our work employing scenarios to improve the maintainability of HUMS systems focuses on two primary issues that can simplify performing changes to the system. The issues are: (a) Recording of design insights, and (b) Recording of problem descriptions or work situation overviews.

3.4.1 Scenarios as Design Insights (Solution Domain Perspective)

Scenarios are used to record the design insights gained during the development of HUMS. They act as effective tools due to their flexibility and their ability to gather information that could not be captured entirely in statistical reports, quantitative results, specific incidents, etc.

Generally, designers gain design insights by understanding the problems that exist in the system, by experimenting with different solution approaches, etc., eventually finding an optimal solution. However, these insights are not recorded methodically and hence it becomes difficult for developers who maintain the system later. This is because the future developers lack the background of the initial system study, problems involved, different approaches tried, etc. Therefore, it is essential to carry forward these insights to enable developers in their maintenance work. We use scenarios to fill the gap between the initial developers and the future developers by promoting understanding of the design intricacies involved as well as continuing such understanding throughout development.

Here are a few rules to adhere to in the development of scenarios for design insights. (a) Scenarios must explain the design with reference to a quality attribute such as scalability, performance, etc. Thus, each scenario can serve as a local, qualitative measure used to evaluate the architecture. (b) Scenarios must expound on all the different combinations for the specific interaction under consideration. These combinations might be chosen based on the interests or the consensus of the different stakeholders involved in the system development.

3.4.2 Scenarios as Work Situation Overviews (Problem Domain Perspective)

Work situation overviews serve as rationales for specific designs adopted for the development of the system from problem domain perspective and

hence, they serve as a compliment to the design insights.

Work situation overviews or problem descriptions record how the different functions were performed before the deployment of the system. This information mostly would be obvious to the team involved in the initial development but not quite so for the team responsible for maintenance in future and hence must be recorded.

Scenarios could be used to record the issues that arose and the decisions taken to solve them. These scenarios also form a common ground for all the different teams specialized in different fields, as in HUMS system development.

Following are the set of rules for developing work situation overviews:

- Document information that serve as background or contextual details. For example, while recording information about different faults, it is beneficial to record about different maintenance activity types that exist to detect and correct such faults.
- Record the solution approach (usually the manual scheme) as how it exists before the deployment of the system.
- Record how desired quality attributes are affected without the proposed system in place

3.5 Evaluation of HUMS System for Quality Attributes

Evaluation of architecture even before implementation results in a number of marked benefits like improvement in developer productivity, lowering of development costs, opportunity to try different designs, etc. In our work, we use scenarios to evaluate whether the system satisfies the set of desired quality attributes.

The quality attributes are classified into three major categories by Kazman, et al. [23], as follows: (a) Based on input-output (e.g., correctness, performance, security, etc.), (b) Based on activities of a particular user (e.g. usability, predictability, etc.), and (c) Based on the activities of maintenance team (e.g., maintainability, portability, etc.). We focus on the first category of attributes here.

Real-time systems like HUMS impose a number of constraints on the system, which the design must satisfy. The degree to which the architecture satisfies the constraints is linked to an indication of a measure of the quality attributes of the system. It is possible to measure this aspect by quantitative means for simple to moderately complex systems. We use a mixture of quantitative measures/metrics and qualitative measures to evaluate the architecture developed for HUMS.

The quality attributes of a system affect one another, either in the positive direction, aiding each other, or in the negative direction, conflicting with each other. Therefore, it is essential for system designers to perform tradeoff analyses during design stage to achieve an optimum solution. In this regard, much of the work that is already completed by different attribute communities can be utilized [29,30].

Our method is based on ATAM[SM] in principle but it differs in the way the evaluation is carried out, thus offering an alternative evaluation method. ATAM[SM] considers the suggested design, the set of constraints and the assumptions to develop an analytical model. Our method focuses on each step of scenario in the analysis and hence operates at a detailed level. Investigation of constraints information at each step of a scenario also helps to elicit the system requirements better.

Following are the steps involved in evaluation of any architecture using scenarios:

- Build a constraints matrix (2 x 2) with rows mapped to system scenarios and columns mapped to desired quality attributes,
- Flesh out the matrix with constraints information. These constraints affect achievement of quality attributes.
- Construct a system representation that facilitates building of analytical models for evaluation. E.g. UML sequence diagrams to evaluate performance
- Use the constraints matrix and the available quantitative data to refine the model
- Use any of the methods – custom developed or those developed by the

different attribute communities - to evaluate the architecture

- Perform trade-off analysis

An architecture style contains information about the components, their topology and the advantages and the disadvantages for using the style. Such information can be used in the selection of a candidate architecture style out of many possible styles. This step can be simplified by making use of the constraints matrix. Different constraints of the environment determine the quality attributes that are essential for a system. Diagrams similar to the fish-eye diagram [23] could be built based on the constraints matrix. This exercise brings into focus the most important attributes to a system and hence, could be used in initial selection of the candidate architecture styles.

These quantitative evaluations could make use of several quantitative evaluation metrics such as performance metrics, scalability metrics, etc. [27].

4. Use of Scenarios in HUMS Development: Demonstration

This section demonstrates the concepts introduced in the previous section through several examples from the development of HUMS systems. First, an overview of the system is provided, followed by the examples and explanations.

4.1 System Overview

HUMS systems are online, continuous health monitoring systems that are deployed on aircrafts in order to improve the accuracy of fault detection as well as to simplify the maintenance operations by automating most of them. Usually, the aircraft carriers institute several maintenance programs in order to upkeep the health of different equipments on board, where most of the maintenance activities are manual and thus are error-prone. HUMS system monitors the health of equipments by employing sensors that are mounted on critical parts of the equipments on the aircraft and by using several processing systems distributed throughout the aircraft to process the sensor data in order to detect any abnormalities with the system. HUMS can intimate the maintenance personnel once a fault has been detected, thus initiating corrective actions on time. These systems are also capable of predicting when faults occur in future (prognostics) as well as diagnose the causes of faults (diagnostics).

4.2 Requirements Gathering

We have identified six different actors that interact with the system. They are:

- Maintenance personnel: People responsible for mounting the sensors, maintain them, carry out visual inspections, etc.
- Operators: End-users of the HUMS systems like pilots, Managers, etc. who plan or make decisions based on the system results
- System implementers & testers: People responsible for deploying the HUMS system.
- Ground staff: Experts on the ground station providing diagnostic information and who help in decision making,
- Other systems: External (sub-domain) systems like OS, DBMS, etc.
- Sensors
- End user: Super actor of Maintenance personnel, Operators, System implementers, and Ground staff

Table (2) shows a set of scenarios generated for these actors.

4.3 Enhancing Reusability

Domain analysis was performed for HUMS domain involving systems developed for different target environments like aircrafts, helicopters, industry machinery, naval systems, etc. and scenarios were gathered as shown in Table (2). Objectives and design decisions were isolated from these scenarios and were organized as an objectives tree based on the model shown in figure (2). As an example, the objectives and design decisions involved in the development of prognostics support for HUMS system for aircrafts is shown in figure (3). As can be seen from figure (3), the prognostic system might be equipped with either condition prognostics, or failure prognostics, or both. When considering these objectives, different applications in the domain might implement part or all the objectives specified in the diagram.

Once the objectives tree is built, classes and objects in the object diagram can be mapped to different nodes on the tree. A subsystem itself can also be mapped. The next step is to perform commonality and variability analysis. For example, the presence of branch 'A' under 2 different nodes might indicate commonality.

Table (2) provides a Sample of a Subset of Scenarios Developed for HUMS.

1.	Gather sensor input	Sensors, Maintenance personnel	• System accepts the sensor data based on the data acquisition policy. • System performs data validation to reduce the false alarm rates • If data validation proves that the sensor is malfunctioning, manufacture data to keep up the supply & intimate maintenance personnel • System performs data correction to normalize the values (if required) • System stores the data either temporarily or permanently for further analyses.
2.	Install new sensor grid	System implementer & tester/Maintenance personnel	• Maintenance personnel/ system Implementer enter information about the new sensor grid • He/she enters sensor[s] settings information and save it • System saves the information
3.	Change sensor settings	Maintenance personnel	• Select the sensor or set of sensors at any appropriate level • Change the settings information • System updates the changes
4.	Request Fatigue detection analysis	End User	• End user specifies a chronological period for input data range. • End user specifies the analysis method(s) for structural fault detection • The system performs the analyses based on the inputs • The system displays the results

- *The service is designed to be implemented as autonomous units in different clusters. Accordingly, even when the number of sensors and/or the number of structural components to be covered is scaled up, the load can be handled by establishing additional service units.*
- *Since each of the buffering service units can incorporate multiple data servers supporting buffers and file systems, it is easy to make a unit more scalable by increasing the number of data servers. If the buffer controller is found to be a bottleneck, using the relationship services, it is easy to reassign the sensors (and sensor-controllers) to different buffering services.*
- *Since the buffer controller, the buffers, and the file servers are not closely tied in terms of geographical proximity, there is much more flexibility in assigning resources when a system is scaled up*

Figure (4) Scalability Issues in Buffering Services Module

Thus, this tree could help us to determine how to improve reusability of components across applications and to build generic components that have wide applicability in the domain.

Another advantage is that this model could help the designers to identify *similar* functionality (commonality) within an application itself. This promotes modular design of the system and so, once the reusable components are identified, we could use a set of modularity metrics like degree of coupling, degree of cohesion, etc. to validate the design.

4.4 Enhancing Maintainability

In this section, examples for the scenarios that improve maintainability are provided. These two examples are not related but indicate the essential aspects of the scenarios. But in practice, we have two matching scenarios for problem description and design insights.

Figure (4) presents a sample design insight scenario that serve as qualitative evaluation of scalability of HUMS. Buffering services is one of the five key modules that are part of HUMS kernel. (See [28] for more details).

The scenario is oriented towards attainment of a quality attribute (scalability) and captures different combinations of the interaction like increase in number of sensors (in future), geographical expansion of the area of the system, and the presence of bottlenecks in the system, thus satisfying both the rules for design insight scenarios. The scenario specifies how different designs help to overcome these issues. Thus, scenarios containing information on design insights could act as design rationales and help in better maintenance of the system

Scenarios for problem descriptions or work situation overviews also provide design rationales. Figure (5) presents a work situation overview scenario of unscheduled maintenance.

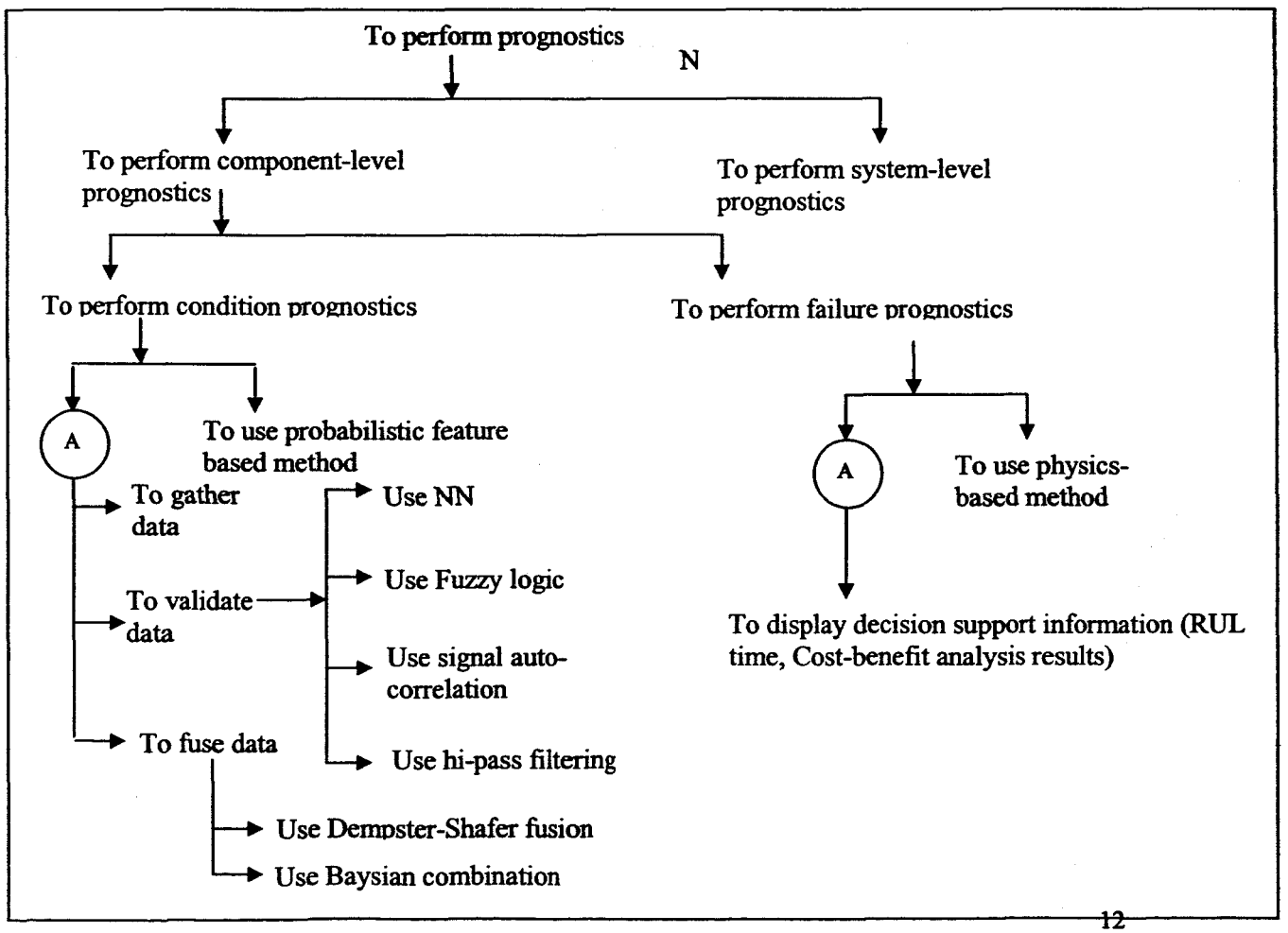

Figure (3) Objectives Tree for Developing Prognostic Support in HUMS

Unscheduled corrective maintenance is usually performed when damage, defects, or degradation are discovered during operational inspections and checks by aircrew, maintenance, or support personnel (e.g., pre- and post-flight inspections and service checks). In most cases, the problem will be immediately corrected under an engineering order or action. Such unscheduled corrective maintenance activities are normally accomplished by air carrier or contractor maintenance technicians following the calibration, repair, and overhaul procedures published in the airline maintenance manual, aircraft structural repair manuals, and work cards. Whenever possible, minor maintenance and repairs are performed on the flight line (i.e, without returning the aircraft or component to the maintenance shops). Unscheduled maintenance requirements always have the potential to cause costly departure delays.

Figure (5) Work Situation Overview: Unscheduled Maintenance ([3])

Scenario	**Performance**	**Reliability**	**Scalability**	**Storage**
Gather sensor input	Latency = (0.5-1) ns	Reliability of communication (network/wireless)	Support N sensors	Space to store 100 hours of sensor data before offloading
Install new sensor grid	N/D	Protection against disk crashes	N/A	Available disk space
Change sensor settings	N/D		N/A	Available disk space
Request fatigue detection analyses	Latency = $M*T*C_i$		Support N sensors	TPS capacity of disk
Request prognostic analyses	Latency = $M*T*C_i$		Support N sensors	TPS capacity of disk
Request diagnostic analyses	Latency = $M*T*C_i$		Support N sensors	TPS capacity of disk
Request diagnostic help	Latency = f (TT)	Reliability of communication (wireless)	Support N sensors	N/D

Figure (6) Constraints Matrix Example from HUMS ([3,31])

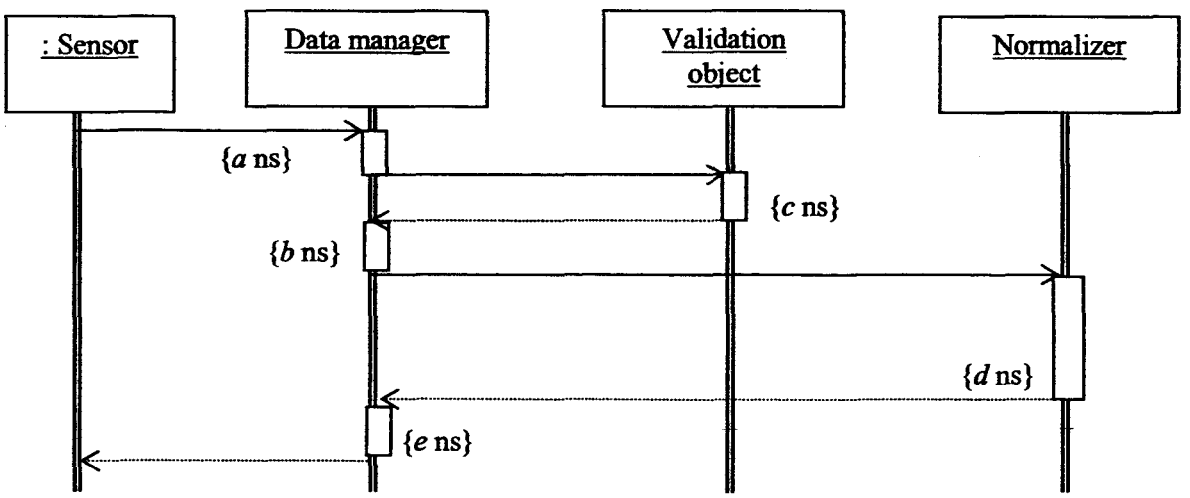

Figure (7) UML Sequence Diagram for "gather sensor data" Scenario

This scenario presents the background or contextual information, the method adopted as solution before the deployment of HUMS, as well as quality factors affected. For example, the presence of different maintenance tasks like pre- and post-flight inspections and service checks, who performs the maintenance, where the maintenance is performed, its effects, etc. may provide enough background for the future developers to understand the problems and to better relate the designs (solutions) to the problems that exist.

4.5 Evaluation of HUMS for Quality Attributes

Step 1: Using scenarios from table (2), a constraints matrix was constructed as shown in figure (6). Four quality attributes are chosen for demonstration, which constitute all the columns.

Step 2: Constraints information is filled in each cell based on the corresponding scenario and the quality attribute. For example, latency of processes and communication reliability are factors that affect quality attributes, performance and reliability respectively.

Step 3: As an example, a sequence diagram is built for the scenario "Gather sensor data" as shown in figure (7). Sensor, data manager, validation object, and normalizer are entities that participate in the interaction

Step 4: Using the constraints matrix, quantitative data may be assigned to the representation. For example, the duration of execution may be provided on the diagram.

Step 5: Analyze the model for the effect on a specific attribute. For e.g., using the data from step 4, we could calculate the total latency involved in the operation and check to see if the chosen design satisfies the latency constraint of 0.5 to 1 ns.

Step 6: We could use different representations or the same representation as in step 3 to analyze a different attribute such as reliability, robustness, etc. We could then analyze the tradeoff between the attributes based on the quantitative information obtained so far.

5. Conclusion

Scenarios have mostly been used for human-computer interaction (HCI) designs. Recently, they have been successfully adopted for many of the software engineering work. This paper documents a few applications of scenarios in the design and development of real-time systems like HUMS. Use of scenarios in enhancing reusability, improving maintainability and in evaluation of software architecture was demonstrated.

6. References

[1] Kent, Renee and Dennis Murphy, Jan 2000, *Health Monitoring System Technology Assessments – Cost Benefits Analysis*, Technical Report, NASA/CR-2000-209848

[2] Munns, Thomas, et al., Dec 2000, *Analysis of regulatory guidance for health monitoring*, Technical report, NASA/CR-2000-210643

[3] Munns, Thomas, et al., Feb 2002, *Health Monitoring for Airframe Structural Characterization*, Technical Report, NASA/CR-2002-211428

[4] Tumer, Irem & Anupa Bajwa, 1999, *A survey of aircraft engine health monitoring systems*, AIAA-99-2528

[5] Orsagh, Rolf, Mike Roemer, & Ben Atkinson, May 2000, *An Internet-based Machinery Health Monitoring system*, Impact Technologies, MFPI Committee Meeting, VA Beach,

[6] Kacprzynski & M.J. Roemer, Dec 2000, *Health Monitoring Strategies for 21st Century condition-based maintenance systems*, Intl' COMADEM Congress, Houston, TX,

[7] Vergroesen, et al., May 1998, *An automatic in-flight Data acquisition system for the RNLN Lynx Helicopter*, The 19th Intl' Symposium on Aircraft Integrated Monitoring systems

[8] Patterson-Hine, Ann, et al., *A model-based health monitoring and diagnostic system for the UH-60 Helicopter*, May 9-11,2001, Proceedings of the AHS International 57th Annual Forum and Technology Display, Washington, DC

[9] Galie, Thomas, et al., March 2001, *Prognostic Enhancements to Diagnostic Systems for improved Condition-based maintenance*, IEEE, Big Sky, MT

[10] Bartelds, Sept 1997, *Aircraft structural health monitoring, prospects for smart solutions from a European standpoint*, NLR TP 97489, The Intl' workshop on structural health monitoring, Stanford, USA

[11] _____, Open system Alliance for Condition-based maintenance web site, http://osacbm.org/

[12] _____, Machinery Information Management Open Systems Alliance, http://www.mimosa.org/

[13] Carroll, John, 1995, *The scenario perspective on system development*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[14] Kuutti, Kari, 1995, *Work processes: Scenarios as a preliminary vocabulary*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[15] Erickson, Thomas, 1995, *Notes on design practice: Stories and prototypes as catalysts for communication*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[16] Nielson, Jakob, 1995, *Scenarios in discount usability engineering*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[17] Kyng, Morten, 1995, *Creating contexts for design*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[18] Karat, John, 1995, *Scenario use in the design of a speech recognition system*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[19] Muller, et al., 1995, *Bifocal tools for scenarios and representations in participatory activities with users*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[20] Carey, Tom & Mak rusli, 1995, *Usage representations for reuse of design insights: A case study of access to on-line books*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[21] Jacobson, Ivar, 1995, *The Use-case construct in object-oriented software engineering*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[22] S.P.Robertson, 1995, *Generating object-oriented design representations via scenario queries*, Scenario-based design: Envisioning work and technology in system development, John Wiley & Sons, Inc.

[23] Kazman, et al., Nov 1999, *Scenario-based analysis of software architecture*, Vol. 13, No. 6, IEEE Software, pp.47-56

[24] G. Arango, April 1994, *A Brief introduction to Domain Analysis*, ACM symposium on applied computing

[25] Kang, Kyo, et al., Nov 1990, *Feature-Oriented Domain Analysis: Feasibility Study*, Technical Report, CMU/SEI-90-TR-21

[26] _____, 1990, IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries New York, NY: 1990

[27] Mukkamala, Ravi, 2000, *Distributed scalable architectures for health monitoring of aerospace structures*, 19th Digital Avionics Systems conference

[28] Mukkamala, Ravi, et al., 2001, *Design and analysis of a scalable kernel for health management of aerospace structures*, 20th Digital Avionics Systems conference

[29] Barbacci M., et al., May 1998, *Steps in an architecture tradeoff analysis method: Quality Attribute Models and Analysis*, Technical Report, CMU/SEI-97-TR-029

[30] Klein, Mark and Rick Kazman, Oct 1999, *Attribute-based Architectural Styles*, Technical Report, CMU/SEI-99-TR-022

[31] Hoss, Robert & Edward Lacy, 1993, *Fiber Optics*, 2nd ed., Prentice Hall

[32]_____, Sept 1997, *UML Notation Guide ver 1.1*, Rational Software Corporation, http://www.rational.com/uml