MIT Sloan School of Management

Working Paper 4363-02
April 2002

AUTOMATED DESIGN DATA AND RATIONALE CAPTURE

Amar Gupta, Satwik Seshasai, Jason Yeung, Tara Sainath

Automated Design Data and Rationale Capture

Amar Gupta, Satwik Seshasai, Jason Yeung, Tara Sainath
Massachusetts Institute of Technology
77 Massachusetts Avenue, Rm E60-309
Cambridge, MA 02139
{agupta, satwik, jyeung, tsainath}@mit.edu

Abstract

This paper presents a knowledge-based approach for capturing data and rationale, so that the experience and insights gained while pursuing a major project or endeavor can be utilized when pursuing future endeavors of a similar nature. The proposed approach, embodied in a concept demonstration prototype named SSPARCy, focuses on the means for capturing knowledge about the design process, including the evolving state of the design as well as the rationale behind major design decisions made over time. The system uses a four faceted knowledge-based approach of knowledge acquisition, discovery, management and repository to focus on various areas of functionality to be used in the design process. The proposed approach enables individuals and organizations to benefit from the experiences and lessons learned from previous processes, as well as facilitates the exchange of such knowledge throughout the design process.

1. Introduction

The system described in this paper utilizes a four faceted knowledge-based approach to capture, reuse and better exploit valuable information assets, with the objective of mitigating temporal and spatial barriers in large multi-organizational multi-disciplinary endeavors. **Knowledge Acquisition** is the process of capturing information from various traditional media into computer accessible media. **Knowledge Discovery** involves using emerging techniques to analyze huge amounts of information and to get better insights into such information than is possible using the best human domain experts. **Knowledge Management** deals with mitigating issues relating to heterogeneities in underlying contexts of information coming from disparate sources. **Knowledge Dissemination** is the automated extraction of the most relevant pieces of information from a huge computer based information infrastructure with such extraction being tailored to the needs of different constituencies of users in an organization[1].

This four-faceted technique allows efficient exchange of knowledge vital to collaboration. It is especially relevant and useful for environments where people perform somewhat repetitive work, but do not have the time or interest to document their decisions and the rationale behind those decisions.

This paper uses an example from the domain of space system design. This research is being conducted as part of the Space Systems Policy and Architecture Consortium (SSPARC) at MIT. The purpose of this group is to examine space system design from a variety of perspectives, and specifically, produce optimal methods for choosing between various choices in space system architectures[2]. Current design methods do not provide efficient means for tracking the state and history of design decisions. Improving the capture and transfer of knowledge from one project to another has the potential to greatly reduce the amount of time spent in the design process. To move towards making better higher-level decisions, it becomes especially vital to capture and process as much knowledge about the systems as possible.

2. Motivation

In a number of applications ranging from marketing campaigns of new products to the design of new systems, each campaign or design process is frequently done as a new endeavor. Very little knowledge, if any, is carried over from the previous episode or campaign. This is also true in the design of spacecrafts where each spacecraft is frequently designed as a "piece of art" rather than picked off from a "manufacturing line."

Simulation exercises related to the design of spacecrafts can often be very complex and difficult to debug. Over time, the large number of system functions, data structures, and global variables that become associated with a simulation can lead to a confusing array of problems that are extremely tedious to analyze.

3. Traditional method

In a number of cases, the design team relies on manual entry in Microsoft Excel spreadsheets to keep track of important information contained in the

satellite simulation files. Although these spreadsheet files provide a practical view of the vital project information, they are very tedious to create and therefore can quickly become outdated[3]. Due to the inaccurate data that are often stored in these files, it is usually necessary for members of the team to review the latest source code files in order to find the information they are seeking. Unfortunately, these files can become very long and difficult to decipher, especially to those who are unfamiliar with developing in the particular programming language of the environment.

The way in which the simulations are created also lacks an effective method for analyzing the modifications that have occurred during the development of all the previous systems. As these systems are re-engineered and enhanced over time, it is critical to keep track of the rationale for major design decisions and the reasoning for any change. In order to effectively re-engineer new simulations, the design team needs to be able to understand why certain structures and variables have the specific values they do, and what previous values may have been tried.

Another major disadvantage of the traditional method of simulation design is the lack of useful error checking for the projects. As the code gets passed on from year to year, some of it becomes redundant and unnecessary variables and functions continue to stay hidden inside the code. Unfortunately, this extraneous information only makes the code more difficult to read and understand, therefore making the re-engineering process even more complex.

In order to provide better inspection and analysis of the future simulation files, a new approach was developed to provide the following functionality:

- Permit the viewing and storage of important information (name, value, rationale, author, etc) relating to the *MATLAB functions* that are used in the project
- Enable the viewing and storage of important information (name, value, rationale, author, etc) relating to the *global constants* that are used in the project
- Facilitate the viewing and storage of important information (name, value, rationale, author, etc) relating to the *design variables* that are used in the project
- Permit storage of the history of the above data as it changes over time in order to enable the user to have information regarding what information has been updated since the last system design
- Provide for error checking of the current system design to inform the user as to where possible errors may exist in the MATLAB files and how those errors may be fixed.

4. Proposed Architecture

The IT Team, a part of SSPARC, has developed a software integration support application that can assist in the re-engineering process for satellite simulations. SSPARCy is a tool that can facilitate design and development steps over time by ensuring that vital design decisions and rationale have been encapsulated. By providing the user with useful system analysis, history reviews, and error checking, SSPARCy attempts to address the void that currently exists for methods to automatically capture crucial information with no or little human intervention.

Each simulation exercise can be encapsulated in one major object, which is referred to as a project. The Project object contains all the necessary objects and variables that represent the information stored in the application, such as functions, constants, design variables, and errors. The Function objects refer to actual functions in the simulation source code. Similarly, the Constant and Design Variable objects represent each global constant and design variable that is defined in the system. Lastly, the Error object refers any possible error in the simulation design that can lead to redundant, unused, or misrepresented code in the code files. A graphical view of the architecture of the data model is shown below in Figure 1.

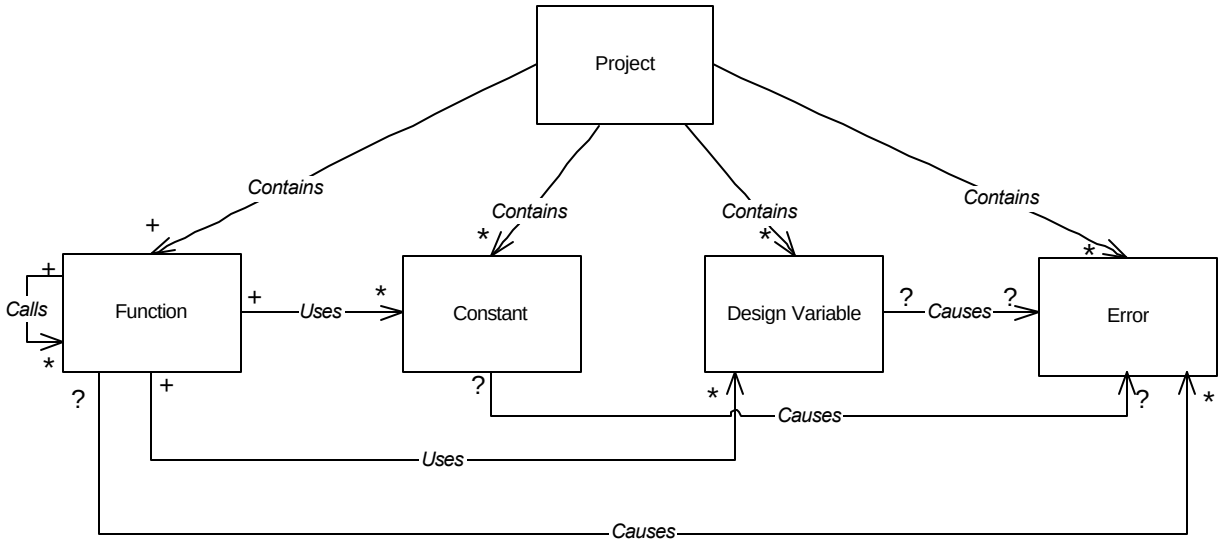

Figure 1: A graphical representation of SSPARCY Data Model which shows how major system objects can interact over time[4].

Each Function, Constant, Design Variable, and Error object can be referred to as a general Variable object. The architecture has been designed so that each Variable object can contain valuable information regarding the Variable's name, value, units, valid range, author, date of creation, and possible aliases which can refer to it in the project. Also, each Variable stores its rationale, so the user can record design decisions and changes that relate to each object in the system. The data that are stored in the object model provide significant possibilities for greater functionality in the graphical user interface.

5. Concept Demonstration Prototype

Over the summer of 1999, the SSPARC project progressed from the B-TOS design project and moved into the C-TOS phase. With increasing collaborations among the participating universities (Massachusetts Institute of Technology, Stanford University, California Institute of Technology, Naval War College), new tools were introduced to handle the linkages of various subsystems. For example, Excel replaced MATLAB as the dominant environment for design sessions. As a result of these changes, SSPARCy was modified to accommodate these new approaches. It was expanded to work with multiple file types and can now handle both MATLAB- and Excel-based source files. To ensure an easy transition for users, we have kept all of SSPARCy's original features and maintained backward compatibility. In addition, the application's interface remains similar to its previous versions, while new features have been added in a consistent format.

6. The Knowledge Based Approach

6.1 Knowledge Acquisition

The graphical user interface has been designed to assist in knowledge acquisition during intelligent capture of simulation exercises. For example, the history table feature visually conveys the degree of stability of parameters in the system. Users can easily identify the volatilities of different parameters across iterative design sessions. Also, by storing Error objects in the data model, the user can analyze potential system problems that are currently very difficult to analyze or even recognize.

Even though the functionality of the system is continuously evolving, several of the major components can be seen through the graphical user interface. Currently, SSPARCy is compatible with both MATLAB and Excel based simulation files. SSPARCy facilitates project management by allowing multiple projects of different formats to be opened, viewed, and saved simultaneously. The system also enables the user to view any Variable object in the current project, and to see all the vital information fields that are stored along with that Variable. As the system design evolves over time, variables may be added, removed, or changed from

the current status. The graphical user interface provides a way for the user to view the way in which the project has changed over time, in terms of the variables that represent the project. Finally, the current system provides functionality for useful error checking so users can see what possible errors might exist in the project and where those errors might have occurred.

6.2 Knowledge Management

6.2.1 Current Variable Data

As mentioned before, it is essential that the user is provided with an easy and effective way to view the current state of any Variable object in the system. Whether the user wants to see the value and the rationale of a global constant in the project or just the author of a specific design variable, SSPARCy provides appropriate display options. By using tables to display a collection of Variable objects, the application allows for quick review of essential data in the source files and the rationale behind their existence and their respective values.

A typical variable data table would show the name of each constant in the project listed in the first column in alphabetical order. Next to the name, the second column lists the subsystems in which these parameters belong to, followed by the current values of these parameters. This setup represents the display at the default situation. Note that only the parameters in use are kept in this table. If a parameter had been removed in the latest design sessions, it would not show up on this list but would appear on the history table instead.

6.2.2 Variable History Review

Just as it is important to review the current state of specific Variable objects in the system, it is also important to see how they have changed over time. For every Variable object in the system, the user is able to quickly review how that variable has changed over time and what variables have been added or removed from the current project. As seen below in Figure 2, a table is once again used to display such information.

In this example, the history of the project's design variables is presented to the user. The first column of the table lists the names of every design variable that has existed in the project since it was created. The rest of the columns in the table represent the state of the project over time. Each column gives the value of the design variables at that time, or leaves the cell blank if that design variable was not present in the project at that point in time.

By adding color codes to the table, the graphical user interface gives the user an easy-to-read look at how the history has changed. As seen below, the coloring of a cell in a specific row x and column y means that something has changed for the design variable in row x at time y. If the background of the cell is colored green, it indicates that the design variable was added to the project at this time. In the same manner, a dark gray cell indicates that the specific design variable in row x was removed from the project at time y. Finally, a red background notifies the user that the design variable has been neither added nor removed, just that its value has changed from time y-1 to time y.

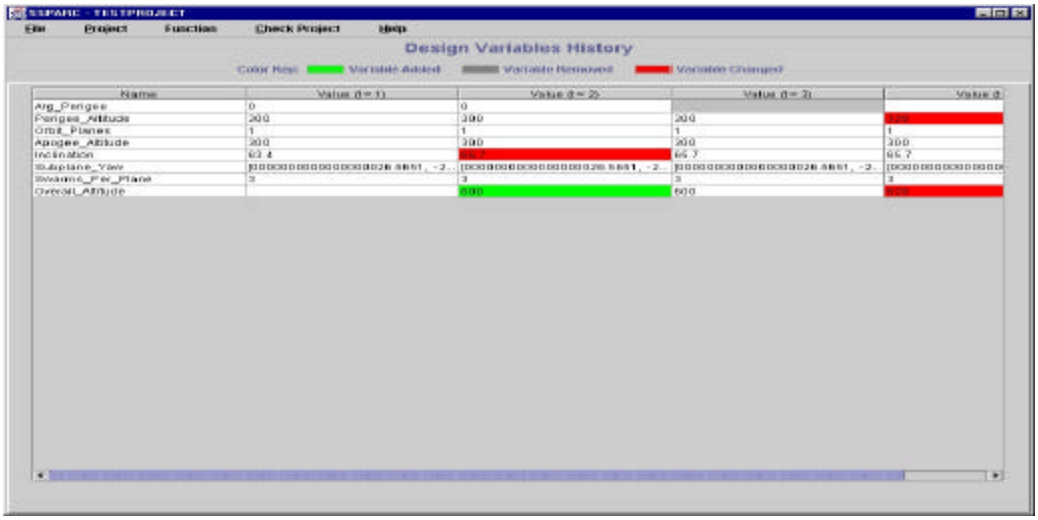

Figure 2: The history of Variable values over time can be viewed in a color-coded table

By allowing the user to view the history of any Variable object in the system, SSPARCy provides an extensible tool for comprehensive analysis of successive simulation exercises.

6.3 Knowledge Discovery

As the design rationale capture should require minimal amount of work from the designers, SSPARCy automatically parses parameters' values, units, comments, and timestamps from their source files. However, if a designer wishes to enter additional details regarding a parameter, he or she can do so by selecting the parameter from the table and then clicking the button "Edit Info." New pop-up windows will appear, and the user can enter supplementary information such as rationale and URL references (see figure 3 below). Other users can later access these information by choosing the button "View Rationale" or "View URL."

A table can be generated for any of the Variable objects in the system. Therefore, with just one selection from the menu bars at the top of the screen, the user can be presented with a table that displays all the MATLAB functions used in the simulation, the global constants that exist, and the design variables that are used throughout the project.

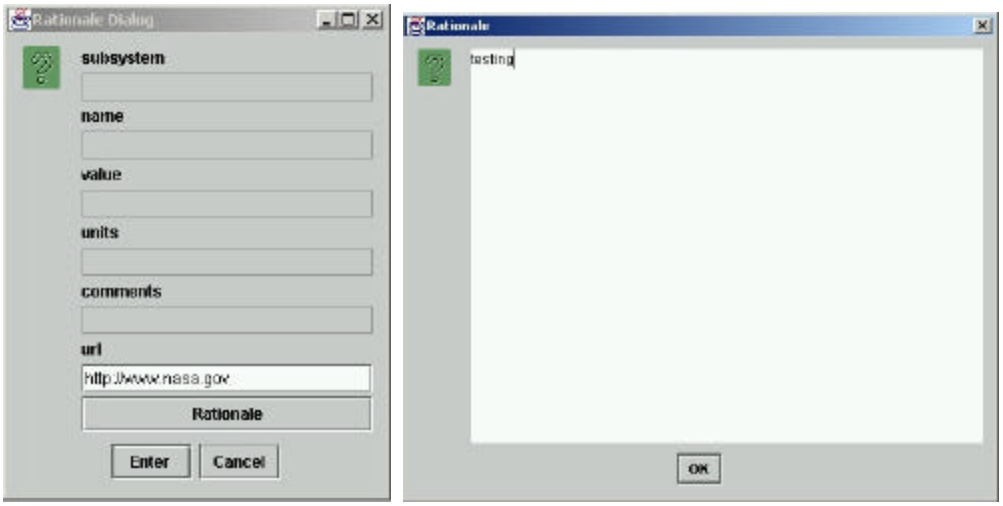

Figure 3: Rationale Dialog Windows allow users to input additional details regarding specific parameters.

7. Related Work

WAVE is an algorithm to learn information extraction rules[5]. Since it is intended to be an algorithm, it does not offer the broad functionality that is available in SSPARCy. However, the incorporation of the WAVE algorithm into SSPARCy would augment the latter's flexibility and ability to adapt to changes in the syntax of the designer code. Further, this could potentially enable the application to analyze other programming language by allowing SSPARCy to learn the information extraction rules for a new language over time.

The existing field of design rationale capture tools spans the spectrum from fully unstructured rationale to completely modeled rationale. Meeting minutes represent an unstructured, time delineated capture. QuestMap[6] and DRAMA[7] are examples of the next step – they provide basic structural elements and enable the user to devise a useful structure. At the other end of the spectrum from meeting minutes is DRIM[8], which is a completely specified model for the rationale underlying the design process. As a design rationale capture tool, SSPARCy lies somewhere on the spectrum between QuestMap and DRIM. SSPARCy creates a simple structure for design rationale by associating rationale with each simulation entity. Additionally, SSPARCy captures this rationale over time. Since this rationale can evolve at any level from project to variable over time, a minimal logical structure is provided for the user to specify rationale in the manner and the level he or she perceives as being most beneficial. SSPARCy does not impose the rigid conceptual structure that DRIM proposes. Therefore, SSPARCy is a compromise in terms of design rationale capture between inflexible structure and amorphous disorder. Furthermore, the basic structure it provides is most appropriate for the domain-specific design process it endeavors to capture.

5. Conclusion

Large scale periodic endeavors are generally performed on an *ab initio* basis. Whether one is desgining a spacecraft or a dam, or launching a sales campaign, each endeavor tends to have a life of its own. In virtually all cases, the effort requires more effort than was originally envisaged. As delays occur, one tends to focus on the main deliverable, with the understanding that one will come back to do the documentation later. In practice, the latter rarely happens. Accordingly, there is either no documentation or there is some unstructured text which is very difficult to use by individuals performing the same or similar endeavor months or years later. SSPARCy represents a new approach in which the raw information is captured from the keystrokes entered by the user while performing the primary activity. Such information is distilled to produce knowledge for later use, without imposing additional burden on human users involved in the first endeavor or the succeeding endeavors.

6. Acknowledgements

This work has benefited from fellow team members of the Space Systems, Policy and Architecture Research Consortium. Professors Daniel Hastings, Hugh McManus and Joyce Warmkessel have provided guidance and feedback as these approaches have been developed. Quincy Scott, Shane Cruz and Presley Canady assisted in developing the software for the SSPARCy approach.

7. References

[1] Gupta, Amar. "A Four-Faceted Knowledge-Based Approach for Surmounting Borders", *Journal of Knowledge Management,* Vol. 5, No. 4, December 2001.

[2] McManus, Hugh and J. Warmkessel. "Creating Advanced Architectures for Space Systems: Product and Process", August 2001.

[3] Scott, Quincy R. "SSPARCy: A Software Integration Support and Design Rationale Capture System." Master's thesis, MIT, July 11, 2001.

[4] Cruz, Shane. 'SSPARCY: A Software Integration Support System for Satellite Simulation" Advanced Undergraduate Project, MIT, May 21, 2000.

[5] J. Aseltine. "WAVE: An Incremental Algorithm for Information Extraction." *In Proceedings of the AAAI 1999 Workshop on Machine Learning for Information Extraction*. 1999.

[6] QuestMap v3.12. The Soft Bicycle Company, 2000.

[7] A. Brice and B. Johns. "Improving Process Design by Improving the Design Process." *A DRAMA white paper*. QuantiSci, Oct. 1998.

[8] F. Pena-Mora, R. Sriram, and R. Logcher. "Conflict Mitigation System for Collaborative Engineering." *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*. pp. 101-124, 1995.