1999

A Computer Tool for Modeling C4I Applications

Luqi

Monterey, California. Naval Postgraduate School

þÿ A Computer Tool for Modeling C4I Applications , with Z. Guan, Proc. of
Command and Control Research and Technology Symposium, Washington, DC. (no page numbers).

http://hdl.handle.net/10945/37870

A Computer Tool for Modeling C4I Applications

Luqi and Jennifer Z. Guan
Software Engineering Automation Center, Naval Postgraduate School
833 Dyer Road, Monterey, CA, 93940
{luqi, zguan}@nps.navy.mil

ABSTRACT

The large-scale nature of C4I applications makes it difficult to formulate accessible requirements before putting lots of effort into development. Rapid modeling/prototyping has been proved to be efficient for requirement validation and verification by providing a mini scale software product. The latest updated software modeling/prototyping tool is an advanced tool for software prototyping and modeling via a unified graphical environment. To support complex requirement specification and elicitation, this tool is designed as a user centered modeling environment that represents requirements in multiple levels, supports project management, reduces modeling/prototyping effort, maintains model consistency and helps error prevention and elimination. This tool is demonstrated to be useful for modeling C4I applications.

Keywords: C4I applications, Software Tools, Computer Aided Prototyping, Modeling and Simulation, Requirement Elucidation and User Centered Design.

1. INTRODUCTION

During software system development, especially in clarification of the requirements, considering the human as a part of whole system is important for three major reasons. (1) Humans are ultimately responsible for command and control activities and in many cases the commander relies upon or delegates portions of his command and control responsibilities to other humans [5]. (2)There is often no explicit description about what should be done by humans and what should be done by the software in the original software requirement documentation [6]. Making a clear separation between the specifications of the human activities and the software activities is very hard to achieve simply by writing requirement documents. (3) Operations and performance of commanders in C4I applications are flexibly changed following the rapid change of information and date in the war. Given the constraints outlined above, clearly defining the operator's role in a specific C4I system[1] is very difficult at the beginning of the development of the combat system. A tool and methods to support the clarification of the requirements of C4I with considering the role of the operator is needed.

Rapid modeling/prototyping, when utilized during the early stages of the development life cycle, enables validation of the requirements, specification and initial design before valuable time and effort are expended on implementation software. This iterative process has been found to be an effective technique for clarifying requirements by providing mini scale modeled prototypes of the software product. It consists of fast and frequent iterations between approximate designs and concept development on the

[1] We refer a specific instance of C4I applications as a C4I system

one hand and user interviews and corrective feedback on the other hand. This iterative procedure leads to better designs more quickly and for less cost than traditional high fidelity prototyping. Bernstein estimates that for every dollar invested in prototyping, one can expect a $1.4 return within the life cycle of the system development. [2]

The Requirement Document based Modeling/Prototyping Tool under development at the Software Engineering Automation Center, Naval Postgraduate School, replaces the traditional software life cycle with an enhanced two-phase cycle that consists of rapid modeling and automatic program generation. This tool is updated from CAPS (Computer Aided Prototyping System), which runs on Sun Workstations [16,17,18]. CAPS-PC[2] is an integrated development environment that generates source programs directly from high-level requirement specifications in PSDL, Prototyping Specification Description Language, which is a language for analysis, modeling and prototyping of systems [19,20]. Not only for small applications, PSDL supports the modeling of large combat systems by providing a simple computational model that is close to the designer's view of real-time systems and easily understood by people whose professional expertise is in areas other than software design.

CAPS-PC makes the construction and modification of application models rapid, accurate and cheap. It involves the final operators of the system in the development process by demonstrating the simulated system to them. The collected feedback from operators forms the basis to further refine or modify system requirements. Furthermore, human factors are considered in the design and implementation of CAPS-PC. It provides several enhancements to facilitate the design of C4I applications to satisfy the characteristics of the software operators/users. The closed loop between the designers and the final users will improve the reliability of the designed software and increase the satisfaction of the users with the product version of the software.

2. C4I APPLICATIONS AND COMPUTER-AIDED PROTOTYPING/MODELING

C4I applications help military officers understand tactical situations. They include the following characteristic features [26,27]

- Their use in strategic, operational, and tactical defense applications makes correctness and reliability critical.
- Systems are influenced by many people, by organizations and by policies, so their requirements are complex and difficult to determine.
- Their design depends on techniques to guarantee that hard real-time constraints will be met both in large distributed systems connected by long-haul networks and in local distributed systems with many hardware structures.
- Their complex, dynamic interfaces make it almost impossible to deal with changes in requirements.

To build large scale combat systems, requirement certification before putting full effort into the whole system development is very useful to lower the cost and achieve a reasonable schedule. Modeling/prototyping is an economic way to build scale models and prototype versions of most systems, which has been proved to be an efficient and effective methodology to evaluate proposed systems if acceptance by the customer or the feasibility of development is in doubt [12,15]. Developing a modeling prototype early in the software process has several benefits for improving the quality of the final product

[2] CAPS-PC: Updated from CAPS for Personal Computer Usage

and enhancing the communication between designers and end users [14]. By providing a conceptual mini system, misunderstandings between software developers and users may be identified as the system functions are demonstrated. When the executable modeling is demonstrated, missing user services may be detected. Difficult-to-use or confusing user services may be identified and refined. Modeling of applications should focus on the issues that are the least well understood, such as new services that have not been implemented in previous versions or that are being integrated for the first time by combining stand-alone systems.

The design and implementation of complex C4I applications tends to produce various kinds of errors at every stage. Incomplete and inconsistent requirements are common. Software modeling/prototyping can help software developers to find incomplete and/or inconsistent requirements issues when the prototyping model is developed. The feasibility and usefulness of the application will be demonstrated to management by a working, albeit limited, system.

To fully support the software development process, software prototyping models have been used as the concrete basic skeleton from which the comprehensive system can be evolved and by which continued evolution of the software can be supported. From this point of view, modeling is tending to be more than a simple set of techniques for software projects. Modeling of applications, building of prototypes, evaluation of constraints, and automated program generation are integrated as the basis for modeling/prototyping methodology. Modeling is necessary for capturing the architecture and the behavior of the system to be developed. Quality, functional or non-functional constraints of the model are measured and evaluated through simulation or testing. An accurate image of the system can be illustrated by the software automatically generated from the model without doing a long-term and costly coding phase. Besides the issues mentioned above, having a user-friendly interface and utilizing advanced interaction technologies to support the intensive interactions between the designers and the users are considered to be a prerequisite for an integrated tools, such as CAPS-PC, for larger scale usage of modeling in software development [3,4,8,18].

3. CAPS-PC FOR RAPID MODELING OF C4I APPLICATIONS

CAPS-PC has been successfully used as a research tool in modeling a large war-fighter control system and patriot missile defense systems. It provides a graphic interface to help the designer to draw data-flow graphics, compose the formal specification, and generate the software architecture from a reuse software base. CAPS-PC has demonstrated its capability to support the development of large complex embedded software.

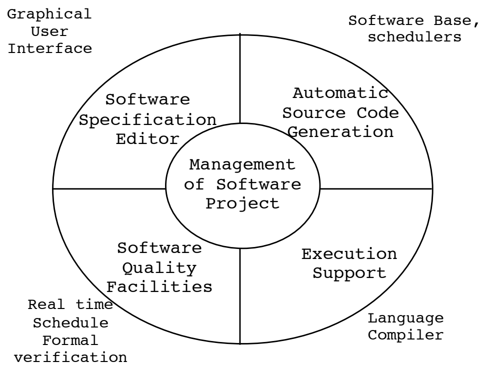

Graphical User Interface
Software Base, schedulers
Software Specification Editor
Automatic Source Code Generation
Management of Software Project
Software Quality Facilities
Execution Support
Real time Schedule Formal verification
Language Compiler

Figure 1 The Conceptual Model of CAPS-PC

The modeling of a software system is the process to build the system from the skeleton to detailed entities. A software project begins with requirement documents from the customers, which are normally in natural language. Mapping the natural language of requirements to the software system is one of the most important parts in the system lifecycle, which in most cases, is the main responsibility of system designers. CAPS-PC provides a method for tracking the natural language description and helps a designer to specify the requirement from a narrative depiction.

3.1 Language for Analysis, Modeling and Prototyping of Systems

Formal specification of requirements can help the system designer to make a clear and complete interpretation of the customer's intention. The language PSDL for analysis, modeling and prototyping of systems supports rapid modeling/prototyping based on abstractions and reusable software components. PSDL supports operator, data and control abstractions, and encourages hierarchical decompositions based on both data flow and control flow [13,19]. Explicitly declared timing constraints support time-critical operations typical of C4I applications.

The computational model of PSDL contains OPERATORS that communicate via DATA STREAMS. Each data stream carries values of a fixed abstract data type [19], which could be the built-in type EXCEPTION. The triggering mechanism of an operator can be input data driven or periodic time driven. The computational model of PSDL can be formally depicted as an augmented graph

$G = (V, E, T(v), C(v))$

Where V is the set of vertices, E is the set of edges, $T(v)$ is the set of timing constraints for each vertex v, and $C(v)$ is the set of control constraints for each vertex v. Each vertex is an operator and each edge is a data stream.

3.2 Interface of CAPS-PC

CAPS-PC is composed of five parts: Software Project Management, the Software Specification Editors [25], Automatic Code Generation, Software Quality Facilities, and Software Execution Support. Each part is supported by extended facilities. Software project management, as the core of software development, provides a platform to support building of new software projects, retrieval of former projects, retracing of software development process and software version control.

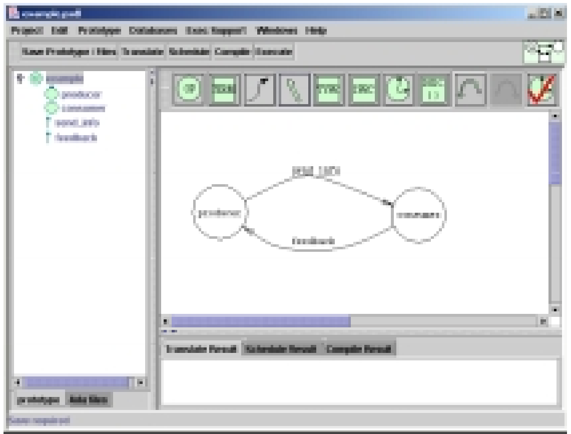


Figure 2. Environment Interface of the CAPS-PC

In CAPS-PC, a unified internal knowledge representation of software requirements is formalized in terms of PSDL definitions, which are designed for supporting automatic materialization of multiple views for different purposes. To the extent that the processes supported by documentation are performed manually, its representation should be understandable by people. To the extent the processes are performed by tools, the representation should be tractable by software. CAPS-PC provides both kinds of views. The tool provides a system model editor for users to create and modify their system models defined by the PSDL modeling language [17, 19], a translator to check the syntax/semantics of the system model and to generate glue and wrapper codes to realize the design for the target system architecture, and a scheduler to analyze the timing constraints and to generate code to realize these constraints in the target architecture [21,22]. The tool interface also provides menus for users to manage their projects and compile source code into an executable model [17]. The CAPS-PC development environment is represented in Figure 2.

4. FEATURES OF CAPS-PC FOR MODELING C4I APPLICATIONS

4.1 Features for Supporting Large Scale C4I System Design

As mentioned in section 2, C4I applications are naturally large scale systems. They involve lots of hardware and software to support commanders in decision making, routine program processing, data computing, etc. C4I computer applications are too large, complex, dedicated, intractable and mutable to meet mission needs under the development circumstances [27]. As with any large system, their development is costly, and the current low productivity of software development aggravates the problem. CAPS-PC can relieve this information explosion problem by providing multi-level information representation and well-designed project management to reduce the complexity of the development.

A. Multi-level Information Representation

CAPS-PC provides multi-level points of view of the whole system design. The construction of the model begins with a top-level definition, followed with a number of levels of refinement based on functional decomposition; the whole model can be built with different granularities for different aspects, depending on the focus of the effort. For components with complex control functions that have many data transitions, the design can be detailed to trace all the process of data computation. For components with simple data handling, although the data may be larger scale, it can be designed as a single operator to handle all the incoming information with the help of reusable data types tailored to the C4I domain.

The hierarchical design of models helps to organize the requirement specification in a way that can be tracked throughout the system's development according to abstraction level and responsible functionality. The clear and precise illustrations and diagrams accompanying the documentation make it easy for a designer to check the consistency with text. Each operator defined in the diagram refers to the requirement item number in narrative documents, which makes it easy to find cross-references in the whole design model [10].

B. Project Management

CAPS-PC provides two kinds of implicit management of a project. From the point of view of development lifecycle, CAPS-PC has a strong capability to support the process of software development from the starting requirement to operational prototyping model. CAPS-PC helps a designer to:

1. Elicit requirements from natural language descriptions
2. Formalize identified requirements
3. Formalize the specification
4. Depict the model in graphic representation
5. Generate the system skeleton and control code
6. Generate the packaged destination code for each designed module
7. Schedule the built model to assess feasibility of the real time constraints

8. Compile the model to be executable, and let the users comment on demonstrated aspects of the system

From the system evolution point of view, CAPS-PC also supports the refinement of functionalities and whole system evolution by providing a pre-structured systemized architecture and template-defined function modules. Further detailed functions and controls can be realized based on this well-structured system. Version control of system development is also provided by CAPS-PC. A higher version of system can be generated based on the models of former versions of system and integrated with the enhanced models, which are designed with higher refinement. The module codes and system architecture can be inherited based on a mature former version of the system so that the functions in former version and higher version can be kept consistent.

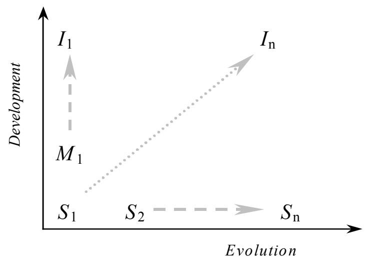


Figure 3 Project Management Diagram (Development and Evolution)

S: Specification M: Modeling I: Implementation

4.2 Features for Supporting Communications in C4I System Development

Present day and future joint warfare will increasingly demand rapid and accurate processing and transfer of data for Command and Control. The requirements for each standalone system will be rapidly adjusted accordingly. Improved capabilities to map requirement changes into system development are critical. A better way of integrating operations, and providing consistent, tactically relevant, and accurate information across commands and services is needed [7]. CAPS-PC provides information communication services by embodying the system information into a unified document representation. The unified documentation can enhance the evolution of the software to satisfy rapidly changed requirements. By considering the various possible information required by combat commanders or software designers, different views can be generated according to specific usage circumstances (rigorous representation for the software developers or natural documents for combat commanders).

A. Unified Document Representation and Multi-view Presentations

CAPS-PC utilizes the concept of building a unified document representation to maintain the information consistency and support information evolution. It also employs multiple information presentations to support the information communications between different stakeholders and tools by providing different information representations to different clients based on the same unified core knowledge artifacts. Highly central documentation knowledge and diverse distributed representation generalization are useful in the maintaining software consistency and improving the flexibility of information understanding.

Multiple views of the documents for different purposes can solve the confliction problem in information presentation. Views intended for human consumption should be tailored to users' role (e.g. developer views should be different from commander views), and those for tools should be tailored to the appropriate API's or tool input language.

A common internal representation of the software knowledge is used to maintain consistency among information presented to both humans and computer tools. Its underlying unified semantic modeling of software documents in the document repository will provide a basis for the mapping of meanings between different terminologies endorsed by different tools. This will provide a principled basis for deducing the relationships between interfaces, data and other elements used in the software process models and tools to facilitate the interoperability of diverse good software "point solution" tools.

B. User-centered Design

CAPS-PC not only has strong capabilities for building models, but also provides high usability for model designers by considering human factors in its own software design. CAPC-PC can improve the quality of the systems developed, reduce the time and cost of software development, enhance the developer's satisfaction and productivity, and make the software development a more delightful and exciting task. Ease-of-use (usability) and ease-of-learning (learnability) can further enhance the software development activities supported by the integrated development environment.

By involving human factors considerations in the design of CAPS-PC, we help the CAPS-PC user to correctly understand the entire range of functionalities offered by CAPS-PC, learn how to apply them, and use them efficiently in a specific context of use or for a specific project. Several types of interaction principles are designed and partially implemented in the CAPS-PC tool:

(1) Highlight most important functionalities in the prototyping effort, such as modeling, translating, compiling, etc.
(2) Make visible program artifacts and CAPS-PC functionalities when they are relevant and required. Irrelevant or rarely needed artifacts and functionalities compete with the relevant ones and diminish their relative visibility.
(3) Provide contextualized feedback and appropriate messages to developers at the time of happening, which efficiently inform the developer about the system status and hints for possible consequential results.
(4) Keep the developer informed of the CAPS-PC status and the model being designed to support the continuity of the design activities.

For a development environment, it would be an effective facility to prevent problems from occurring in the first place by providing human error resistance strategies to prevent or limit the consequence of human error. Error resistance can be achieved by means of two strategies: error prevention and error handling. Error prevention can result from forcing functions, operator's selection, or training, which will not get rid of all human error occurrences. The error handling is intended to catch the remaining errors in case they occur, and control the evolution of errors to minimize the error effect. To increase the usability of specifying and designing of system requirement models, CAPS-PC employs a backward error recovery strategy to reset the system state when an error or mistake occurs. Also, keeping of design histories and making them retraceable can remind the application modeler the process of their past design steps, which can provide helpful information for developer to find the most effective recover approach.

4.3 Features for Supporting Quick Developed C4I System Delivery

A. Automated Code Generation

To facilitate the testing and demonstration of designed prototyping models, the modeling tool provides the user with an execution support system that consists of a translator, scheduler, and compiler. The translator/scheduler generates the glue code needed for timely delivery of information between subsystems across the target network. For C4I applications that require sophisticated graphic user interfaces, an interface editor is provided to interactively sculpt the interface in Unix system by using the TAE+ Workbench [24] and automatically generate corresponding code.

5. DESIGN AND IMPLEMENTATION OF MISSILE DEFENSE SYSTEM

A C4I system called Missile Defense (MD) has been developed by CAPS-PC [27]. The MD system provides defense functions to a specified area or a nation such that it can be extended to a TMD(Theater Missile Defense) system and a NMD(National Missile Defense) system. TMD and NMD systems are extremely complicated; however, we need to know how system requirements are obtained. The first model MD is very simple and immature, but it gradually gets more refined and feasible as the MD system goes through the evolution process several times.

Two main subsystems included in the MD system model are a separate Scud missile launch and tracking subsystem. The system will also provide a separate missile defense subsystem that would detect Scud missile attacks, determine threat, launch an interceptor Defense missile, and track/guide it to destroy the enemy Scud.

- Scud System Inputs: The system model shall provide a console for launching a simulated Scud to support training and system demonstration/testing.
- Scud System Outputs: The system shall display current Scud position in ground range and altitude.
- Scud Missile Characteristics: Scuds fly on a simple ballistic trajectory based on initial launch angle and velocity.
- Defense Missile System Inputs: The patriot has a radar that searches for and tracks Scuds, producing radar returns, 20 times per second. Radar returns include slant range and elevation. The radar also tracks the Defense missile when launched, to impact.
- Defense Missile System Outputs: The tactical display shows the location of the Scud currently being tracked (if any) and the Missile interceptor (if any) in terms of ground range and altitude. The tactical display shows the predicted impact point in terms of distance in side the defended area. The tactical display tells if/when the Defense Missile destroys the Scud.
- Initial Defense Missile Characteristics: Defense Missile thrust is constant and sufficient to overcome gravitational and atmospheric effects. Average velocity is 5,000mph. A Defense Missile is controlled from the ground, such that it's direction is always toward the last known position of the Scud, with no perceptible change in velocity. The Defense Missile warhead is detonated when it is within 20 meters of the Scud.

Assumptions can be generalized and specialized according to requirements from users. Users provide criticisms by their own view with different generalization and specialization ideas. The issue analysts collect criticisms into different generation and specialization issues. The requirement analysts provide some information about requirements, such as cost benefit analysis and resource constraints, to mangers. Model demonstrations and measurements provide additional information, such as effectiveness evaluations. The decision made by managers to new requirements of next generation model decides whether assumption are generalized or specialized.

During the design and model processes by using CAPS-PC, several implementation issues are provided to improve the reliability of the result model and the usability of the design activities.

5.1 Reduce the Workload of Designer by Providing Contextual Information

Contextualized feedback and appropriate messages can help the designer realize the system design and meet the schedule for the design tasks. Providing relevant information to the developer at the time when specific kinds of knowledge about the design are needed is helpful to make appropriate system designs. CAPS-PC attempts to provide this type of facility to let the user know the current status of system (editing / compiling / execution) and the identification of the model designed (name / version / requirement coverage). Giving relevant information for user to keep track of the design process will reduce the interruption in the design process.

In the current tool implementation, several kinds of information about the tool are updated and displayed at run time to inform the developer, which include the name of current model project and its version, the decomposition level and the name, type, and attributes of the decomposed component, the history of the system backup and current timeline of the model.

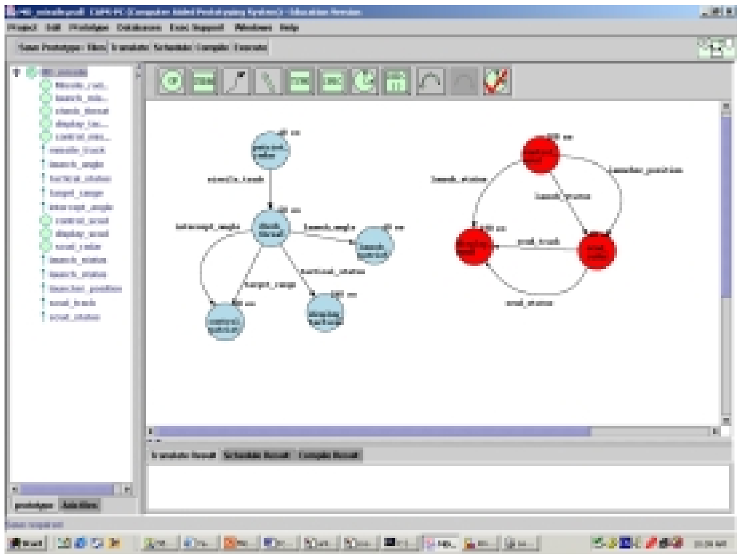
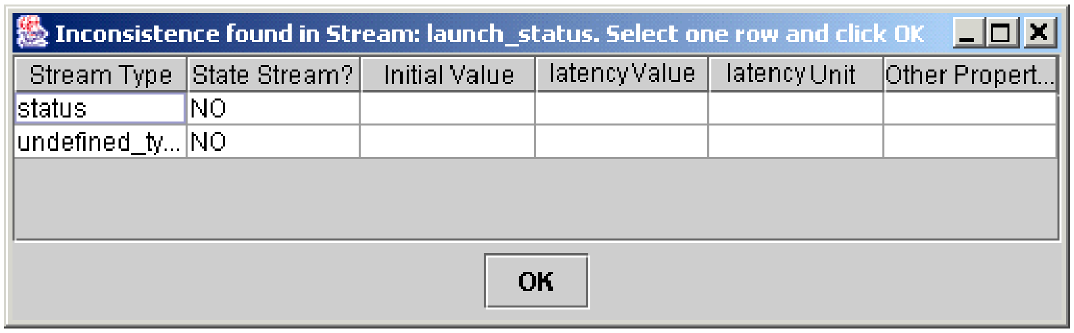


Figure 4. Model Design of MD System

5.2 Maintain Consistency

CAPS-PC provides several methods to support the monitoring of the syntax consistency during the modeling effort. The syntax consistency between components design is especially important for the safety reliability properties of MD system. Before saving PSDL codes to file, the PSDL codes are verified for correct syntax. The grammar error information is displayed if some errors are found. This enhances the quality of the resulting software model. When saving the psdl file, the psdl data flow diagram will be checked to ensure that the data flow diagram has a valid structure. Furthermore, CAPS-PC also performs instant checks during every user input to prevent human error, and an error message is displayed if an error is detected.

During the design of a software model, the interactions between the modules will be defined, which include their data communication, input constraints and output constraints. All of that information is defined in the psdl edge annotation. Edge inconsistency problems may occur when the properties of a model edge are changed in one of its instances [25]. CAPS-PC provides automatic consistency maintenance to check the differences between the interaction edge's name, date type, and related time properties. The interaction monitor will dynamically detect when an edge's name has changed and automatically fill in the edge properties if it has the same name as an existing edge. When any edge's property has changed and the OK button is clicked, all the edges are searched to find every edge with the same name but different properties. To assist the selection of the correct interlink definition, the monitor will show an inconsistency table that displays the detailed information about the conflict edges, and will provide evaluation hints for user to select a correct property.


Inconsistence found in Stream: launch_status. Select one row and click OK
OK

Stream Type	State Stream?	Initial Value	latencyValue	latencyUnit	Other Propert...
status	NO				
undefined_ty...	NO				

Figure 5. Check table for edge inconsistency properties

5.3 Documentation Generation Supporting Customer's Feedbacks

In software development, the content and style of the manipulated documentation depend on the stage of the development and the role of readers. For example, requirement, specification, design, etc, would be in different specification forms. In the requirement specification stage, the document of the software specification could be in a specification language, such as PSDL, or in another specific problem specification language, such as temporal logic, depending on the actions underway, in this case formulation or analysis.

Even in the same stage, different personnel involved in a specific development stage would desire different illustration styles according their task assignment, responsibility and preference [9]. For example, a requirement specifier may prefer to see requirement specifications as a whole in a textual page because they are responsible to make each item clear, insure consistency between dependent items and make whole structure reasonable. A quality insurer may prefer to see information about a specific kind of software constraint, such as timing constraints, in a precisely defined formula and symbols, such as temporal logic formula. This would make them concentrate on their part very clearly and precisely. The different document styles for different stages and different persons will provide comprehensive support for software evolution and improve the flexibility of the development.

In CAPS-PC, our initial effort to unify the software knowledge representation is to define the software functionalities, time constraints and control constraints by using PSDL. Computational models will encapsulate all the information related to the development process. The analysis of the software requirement constraints, such as real time constraints, can be done by performing computation scheduling with the internal knowledge representation. The automatic model-based prototype generation can also be completed by internally analyzing semantic meanings of the software specifications and doing software retrieval and adaptation based on matching of the specifications of the desired component and reusable component candidates.

The information resident in the internal software knowledge representation can be transformed into a graphical information representation to illustrate the structures of the software design. The graphical display of the software structure in CAPS-PC provides the designer an easy way to define the functionality and software constraints. Meanwhile, it also makes it easy for the sponsor to get knowledge of the rough software product.

A dynamic executable model with a user-friendly interface is another efficient information presentation style, which we treated as another kind of software documentation. The model generated by CAPS-PC provides another way to show designers their design results and helps them to find the defects and incompleteness, which also provides a vivid information display to show sponsors and let them check if it meets their imagination of the product. Engineers can instrument the model with gauges that measure and display runtime properties relevant to the design, such as the longest observed running time for a time-critical component.

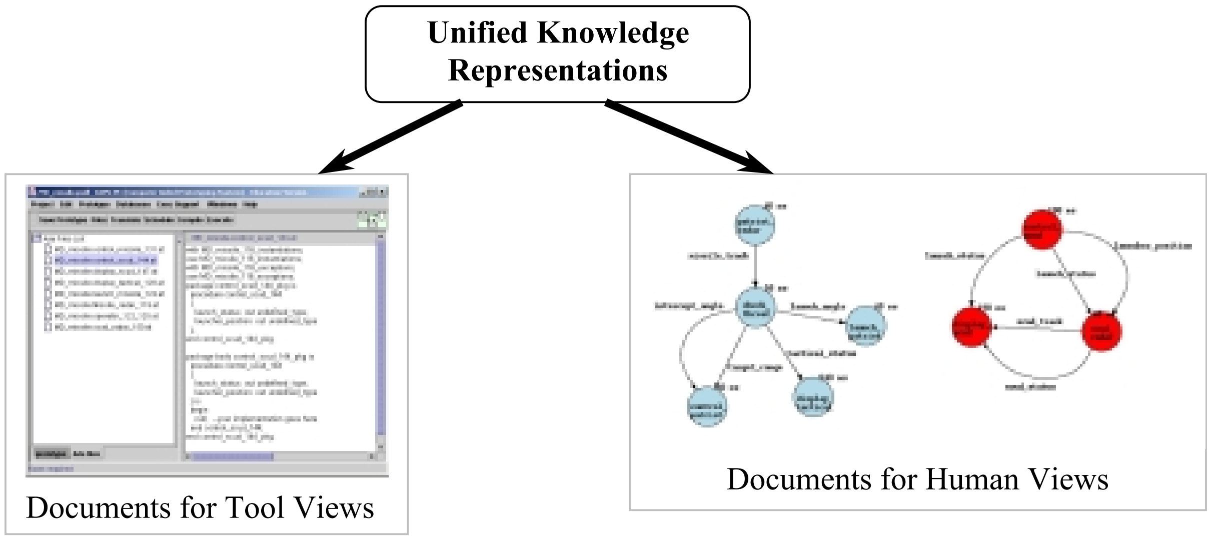


Figure 6 Document presentations for human and tools in different information style

An example of information presentations for different tools in software development process and different stakeholders involved in the development activities can be roughly perceived in Figure 6. The left side presents document presentations for software tools, while the right side presents document presentations for person’s review and revision.

6. BENEFITS OF C4I MODELING VIA CAPS-PC

Based on the requirement document, a model is built and provides the relevant alternatives for derivation of documents for the specification, design, implementation, and even testing of the system development.

The consistency maintained by the tool between these documents provides a solid baseline throughout the continuous development effort. The document generation function provided by the tool makes it easy for the customer, user, and sponsor to understand, handle, and review the system during development.

The specification of requirements can be generated with completeness and consistency checking, according to the system functionalities and constraints. The graphical design process maintains the syntax of requirement specification, and a further translation process ensures the semantic consistency of the specification document.

Furthermore, the version control documentation for requirements specifications and model design can be maintained by CAPS-PC. If any changes are made to the requirements during the remaining phases of development, the changes can be tracked from the requirements document, through the design process, all the way to the test procedures.

7. CONCLUSION

The nature large scale of C4I applications make their development extremely complex. The quality and reliability of the developed system is difficult to ensure and maintain. CAPS-PC has been demonstrated to be a useful comptuer tool to model large complex control systems, such as MD system discussed in this paper. CAPS-PC formulates and validates requirements via executable model demonstration and user feedback, assesses feasibility of real-time system designs, enables early testing and integration of completed subsystems, supports evolutionary system development, integration and testing, reduces maintenance costs through systematic code generation, produces high quality, reliable and flexible software, and avoids schedule overruns.

Our update efforts on CAPS-PC make it easier to be used by end users, such as commanders of MD systems, who know little about formal specification. It sets a step forward for requirement document validation, and provides the basis for further generation of the software documents based on the usage of model specification.

8. ACKNOWLEDGEMENT

This work is supported in part by the U.S. Army Research Office under grant number 40473-MA-SP. We would like to thank all of the reviewers for their support.

9. REFERENCES

[1] Seffah, A.; Rilling, J., "Investigating the relationship between usability and conceptual gaps for human-centric CASE tools", *Proceedings IEEE Symposium on Human-Centric Computing Languages and Environments*, 2001, pp. 226 –231.

[2] L. Bernstein, Forward, "Importance of Software Prototyping", *Journal of Systems Integration – special issue on Computer Aided Prototyping*, 6(1), 1996, pp. 9-14.

[3] R. Bonk, "Prototyping – the Effective Use of CASE Technology", *Perentice Hall*, 1989.

[4] Andrew Brooks, Louise Scott, "Constraints in CASE tools: results from curiosity driven research", *Proceedings of International Conference on Software Engineering*, 2001, pp. 285-293.

[5] Susan G. Hutchins, William G. kemple, Gary R. Porter, Michael G. Sovereign, "Evaluating Human Performance in Command and Control Environment", *Proceeding 1999 Command and Control Research and Technology Symposium*, U.S. Naval War College, Newport, RI, 29 June – 1 July 1999, pp. 50-52.

[6] David Noble, "High Leverage Command and Control Functions with Critical Human Roles", *Proceeding 1999 Command and Control Research and Technology Symposium*, U.S. Naval War College, Newport, RI, 29 June – 1 July 1999, pp. 1324-1337.

[7] Craig S. Anken, "Intelligent C2 Information Technology", *Proceeding 1999 Command and Control Research and Technology Symposium*, U.S. Naval War College, Newport, RI, 29 June – 1 July 1999, pp. 1010-1017.

[8] David Finnigan, Elizabeth A. Kemp, and Daniela Mhandjiska, "Towards an ideal CASE tool", *Proceedings of the International Conference on Software Methods and Tools (SMT 2000)*, 2000, pp. 189-197.

[9] Zhiwei Guan, “Study on User Intention based Human Computer Interaction”, *Ph. D Dissertation*, Chinese Academy of Science, 2001

[10] Jennifer Z. Guan, Luqi, “A Software Prototyping Framework and Methods for Supporting Human’s Software Development Activities”, *Workshop on Bridging the Gaps Between Software Engineering and Human –Computer Interaction, International Conference on Software Engineering 2003*, Portland, Oregon, May 3-11, 2003, pp. 114-121.

[11] G. Holzmann, “The Spin Model Checker”, *IEEE Transaction on Software Engineering*, Vol. 23, No. 5, 1997, pp. 279-295.

[12] Fabrice Kordon, Luqi, “An introduction to Rapid System Prototyping”, *IEEE Transactions on Software Engineering*, Vol 28, No. 9, 2002, pp. 817 –821.

[13] B. Kraemer, Luqi and V. Berzins, “Compositional Semantics of a Real-Time Prototyping Language”, *IEEE Transactions on Software Engineering*, May, Vol. 19, No. 5, 1993, pp. 453-477.

[14] M. Kuhl, B. Spitzer, K. Muller-Glaser, and U. Dambacher, “Universal Object-Oriented Modeling for Rapid-Prototyping of Embedded Electronic Systems”, *Proc. 12th IEEE Int’l Workshop Rapid System Prototyping*, 2001, pp. 90-96.

[15] Luqi, “System Engineering and Computer-Aided Prototyping”, *Journal of Systems Integration, special issues on Computer Aided Prototyping*, Vol. 6, No. 1, 1996, pp. 15-17.

[16] Luqi, “Computer-Aided Prototyping for a Command-and-Control System Using CAPS”, *IEEE Software*, Vol. 9, No. 1, January 1992, pp. 56-67.

[17] Luqi and M. Shing, “CAPS – A Tool for Real-Time System Development and Acquisition”, *Naval Research Reviews*, Vol. XLIV(44) ,1992, pp.12-16.

[18] Luqi, P D Barnes, and M Zyda, “Graphical tool for computer-aided prototyping”, *Informaiton and Software Technology*, Vol. 32 No. 3, April 1990, pp. 199-206.

[19] Luqi, V. Berzins, R. Yeh, “A prototyping language for real time software”, *IEEE Transactions on Software Engineering*, Vol. 14, No. 10, 1988, pp. 1409-1423.

[20] Luqi, “Real-Time Constraints in a Rapid Prototyping Language”, *Journal of Computer Languages*, Vol. 18, No. 2, Spring 1993, pp.77-103.

[21] Luqi, V. Berzins, Man-tak Shing, J. Puett, Z. Guan, et al., “Infusion pump”, *Technical Report*, #NPS-SW-02-004, NPS Monterey, CA: September 2002.

[22] Luqi, M. Shing, “Real-time scheduling for software prototyping”, *Journal of Systems Integration - Special Issue on Computer Aided Prototyping*, Vol. 6, No. 1, 1996, pp. 41-72.

[23] Jarzabek, S., Huang, R. “The Case for User-Centered CASE Tools”, *Communications of the ACM*, 41(8), 1998, pp. 93-99.

[24] “TAE Plus Programmer’s Manual (Version 5.1)”. Prepared for: NASA Goddard Space Flight Center, Greenbelt, Maryland. *Prepared by: Century Computing, Inc.,* Laural, MD, April 1991.

[25] Shen-Yi Tao, “Design and Implementation of a Platform Independent Prototype Specification Editor”, *Masters Thesis*, Naval Postgraduate School.

[26] Luqi, “Computer-Aided Prototyping for A Command-and-Control System using CAPS”, *IEEE Software*, January, 1992, pp. 56-66

[27] M. Harn, V. Berzins, Luqi, and W. Kemple, “Evolution of C4I Systems”, *Proceeding 1999 Command and Control Research and Technology Symposium*, U.S. Naval War College, Newport, RI, 29 June – 1 July 1999, pp. 1361-1380.