IMPROVING SOFTWARE QUALITY UTILIZING AN INTEGRATED CASE ENVIRONMENT

Peter H. Luckey and Robert M. Pittman

IBM Federal Sector Division, Route 17C, Owego, NY 13827

New methods in software development, coupled with Computer-Aided Software Engineering (CASE) tools, make it possible to minimize the cost and greatly improve the quality of Ada software through the application of an integrated CASE environment. In response to a recent trainer system contract opportunity, the Trainer and Simulation department located at IBM's Owego, NY facility is currently in the process of transitioning from a more traditional software development approach to an integrated environment comprised of state-of-the-art hardware, tools and a selected software development methodology. The pre-integration development environment is discussed. The chosen environment implementation is also discussed, including the CASE Real-Time Curriculum™ developed by Paul Ward. This Curriculum describes a software requirements and design development method that is being applied to develop reusable, object-oriented domain models. Technical considerations driving the implementation of the heterogeneous environment comprised of IBM and vendor products are addressed. The current status of the CASE environment, with a look towards its future capabilities, is also discussed.

For many years IBM's Owego, NY facility has performed software requirements development and design in a non-integrated, highly manual, text-oriented fashion. In response to a recent trainer system contract bid, in parallel with ongoing research into the software development process, the facility's Trainer and Simulation department had the opportunity to examine alternative development approaches aimed at increasing the quality of its Software Requirements Specifications (SRS) and subsequent software design and code.

As part of this examination, the following trend in SRS development was discovered:

- SRSs are continually plagued with improperly detailed requirements
- Though the Systems Engineering organization possesses a detailed software requirements development approach, lack of strict adherence to this approach has caused SRS level and type of detail to be driven more often by individual engineer personalities and experience rather than technical need
- Improperly detailed SRSs containing varied contents have caused some of these specifications to take on a "life of their own", yielding a large number of unpredictable changes throughout the development cycle. This situation has destabilized projects due to negative impact on system development schedule and cost, Systems Engineering and Software resources, and difficulty in maintaining traceability between the specification and subsequent software design due to the instability of the specification.

Recognizing this trend, exacerbated by the trainer system contract demand for application of CASE tools and the Trainer and Simulation department's overall desire for higher software quality and productivity, a team of systems and software engineers was formed to investigate the incorporation of new methods and tools into the software development process.

INTRODUCTION OF AN INTEGRATED CASE ENVIRONMENT

In order to discuss improvement of software quality through utilization of an integrated CASE environment, it is important to understand the characteristics of a generic environment before examining the details of a specific implementation.

CH3007-2/91/0000-00665 $1.00 © 1991 IEEE

To determine the implementation components of our integrated CASE environment, the engineering team examined the software development lifecycle and the specific requirements of the trainer system contract to select a set of high-level requirements that the environment must meet. The team concluded that the integrated CASE environment, in order to not only support the contract but also improve overall software quality and productivity, must:

- Support reusable requirements, designs and Ada software
- Provide an electronic interface between tools such that various hardware components can electronically communicate and allow data to be entered into the environment only once during the development process
- Provide computer-assisted software development tools across the software development lifecycle
- Support a lifecycle-encompassing development method that is fully supported by the tools yet is tool independent. The method must provide a cognitive framework within which the software engineers can use the software tools to perform the development lifecycle phase tasks.

Based on available technology, the team concluded that our principle hardware components, the user workstations, could easily be networked together. The team also determined that a host or server would be required for software tools. This electronic interface, which could be implemented in an AD/Cycle Repository approach or by a common data format, would allow data to flow from one tool to another without human intervention.

The selection of the CASE tools was influenced by contract constraints as well as previous software development experience. The contract specified the use of Cadre team*work*® for requirements development, while previous software development projects pointed to the Rational Ada Development Environment to support software design, code, test and integration.

Since a driving contract requirement was to develop reusable, real-time software, the team selected the combined methods described in The CASE Real-Time Curriculum™ as the software development method. As illustrated in Figure 1-1 on page 3, the selected tools support the development method from front-end system requirements analysis through software design.

APPLICATION OF THE INTEGRATED CASE ENVIRONMENT

To meet the requirements of the trainer system contract as well as achieve the desired software development quality goals, the systems and software engineering team next investigated available technology that would meet the requirements of the integrated CASE environment. Upon conclusion of this investigation, the team decided to implement the environment by applying a mix of IBM workstations, networks, and graphics and text-based CASE tools to support Ward's CASE Real-Time Method™.

Hardware and tools

As illustrated in Figure 1-2 on page 4, the Integrated CASE Environment hardware complement consists of RISC System/6000™ Model 320 workstations connected to a central RISC System/6000™ Model 320 server on an IBM Token Ring network. The Model 320 workstations, available to both systems and software engineers, provide common access to the software requirements through the Cadre team*work*® graphics-oriented CASE tool. The Model 320 Server, to be upgraded to a Model 530 processor in the near future, houses the team*work*® control information as well as Cadre team*work*®/Rqt, a supporting requirements traceability tool. IBM PS/2 personal computers as well as IBM X-terminals provide additional access to requirements information, while the Owego Backbone Ring and a Model 9370 processor provide access to host application software, electronic mail, scheduling and other support services.

The Rational Ada development environment

While the RISC System/6000™ Model 320 workstations provide a platform for Cadre team*work*®, the Rational R1000 Coprocessors provide a superb Ada software design, code, debug and test environment.

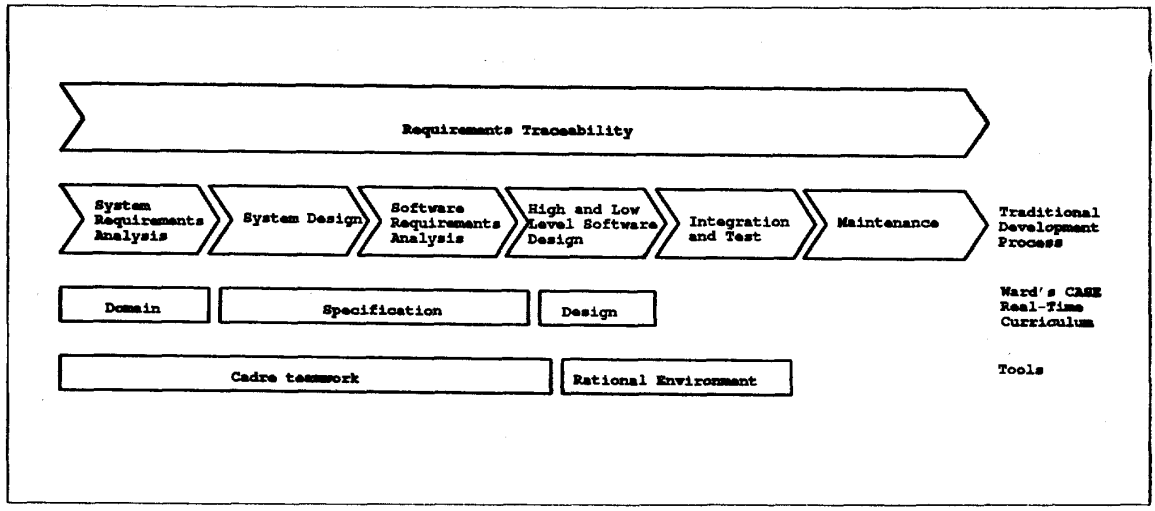

FIGURE 1-1. Map of Traditional System Development Process to Integrated CASE Environment Tools and Method.

The Rational Environment is comprised of an integrated set of tools used to design, code, test, integrate, and manage an Ada system. An X-Windows interface allows the software developer to access the Rational environment from his RISC System/6000™ workstation.

At software requirements release time, which signals initiation of high level software design and subsequent software design, the graphical requirements and associated database developed on the RISC System/6000™ workstations are electronically transferred to the Rational Coprocessors using the Rational Teamwork Interface (RTI). This interface provides an automated, paperless requirements transfer from systems engineering to software developers. The RTI, coupled with the Rational X-Windows interface and the graphical nature of the team*work*® tool, provides an important requirements traceability capability across the software development lifecycle.

As originally presented in Figure 1-1 and further illustrated in Figure 1-3 on page 5, requirements traceability is available starting at the Requirements Analysis phase of the project and continuing through the software test phase. Traceability is especially important at software requirements release time, when requirements are translated to High Level Design (HLD). Upon completion of requirements transfer across the RTI, the Rational Environment provides traceability from the Rational-resident design information back to the team*work*® requirements. This capability allows systems and software engineers to simultaneously view lower level textual design information and its associated higher level graphical requirements at their RISC System/6000™ workstations. The multimedia presentation of requirements and design information enhances communications between systems engineers and software designers.

The Rational Design Facility (RDF), another component of the Rational Environment, works with RTI to automate the production of DoD-STD-2167A documentation. An extensive Configuration Management and Version Control (CMVC) system in the Rational environment can control configuration management for the team*work*® requirements database as well as the Rational-based design information, and is accessible on the Local Area Network. In essence, the Rational Environment not only strongly supports project level Configuration Management, but also automates many tasks normally associated with Software Quality Assurance (SQA) organizations. In addition, there is an Ada-sensitive syntax and semantics-directed editor and program traversal tools, as well as facilities to support remote or cross development

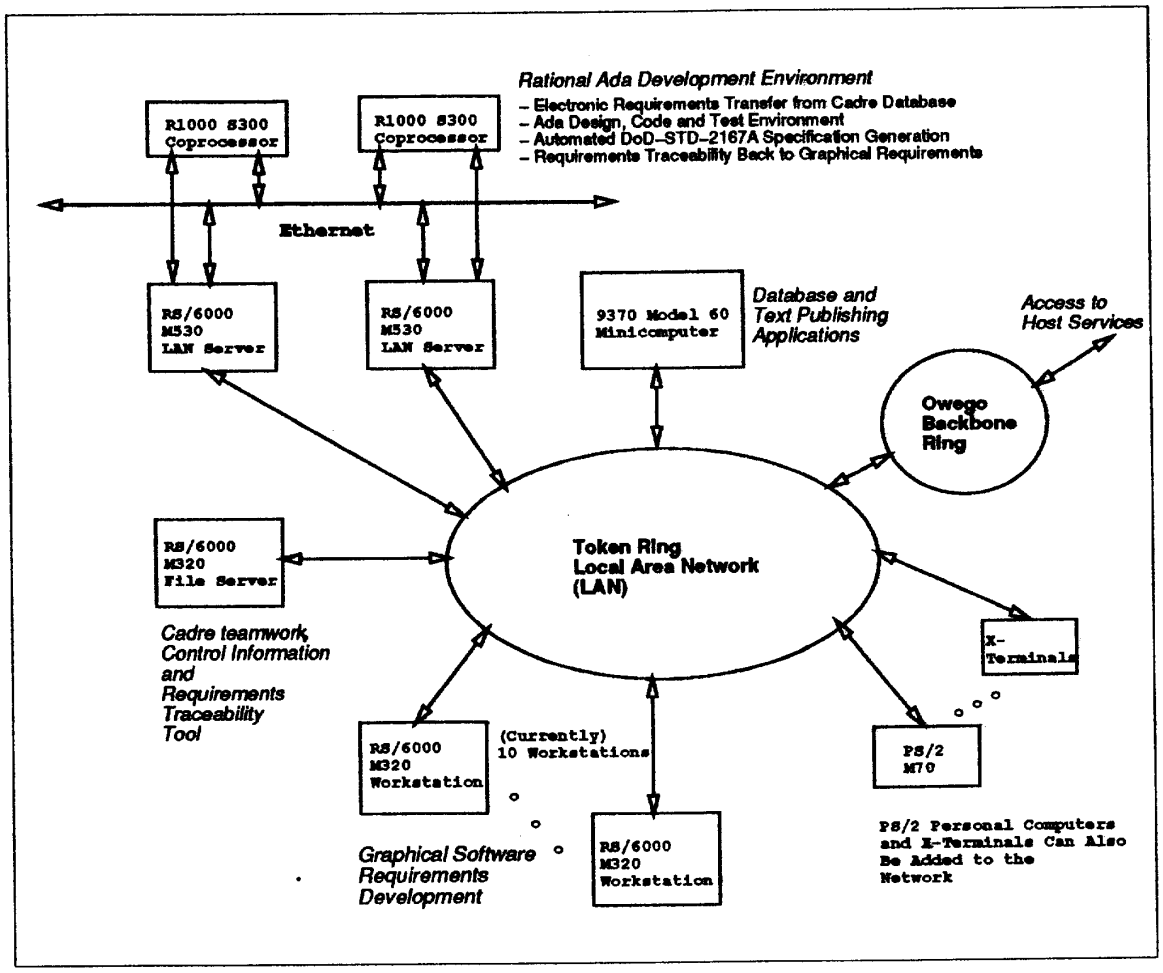

FIGURE 1-2. Integrated Software Development Environment

for an Ada system during Integration and Test (I&T) activities.

Software development method

This integrated set of hardware and software CASE tools is capable of implementing several different development methods, including The CASE Real-Time Method™. Based on the concepts of Objects, Classes and Inheritance, this Method is a structured, disciplined approach that yields object-oriented requirements and design through the performance of Domain Analysis and various modeling techniques.

Domain Analysis seeks to guarantee reuse of requirements, design and software by capturing knowledge about a family of systems within a given application domain versus developing an individual system in isolation. The product of domain analysis is a Domain Model, which contains a complex set of both static and dynamic aspects of family of systems under examination. However, due to the broad scope of the Domain Model, and its lack of implementation detail, Specification and Design models must be extracted to guide creation of individual systems.

The Specification model transforms the set of family information to a specification for a single system.

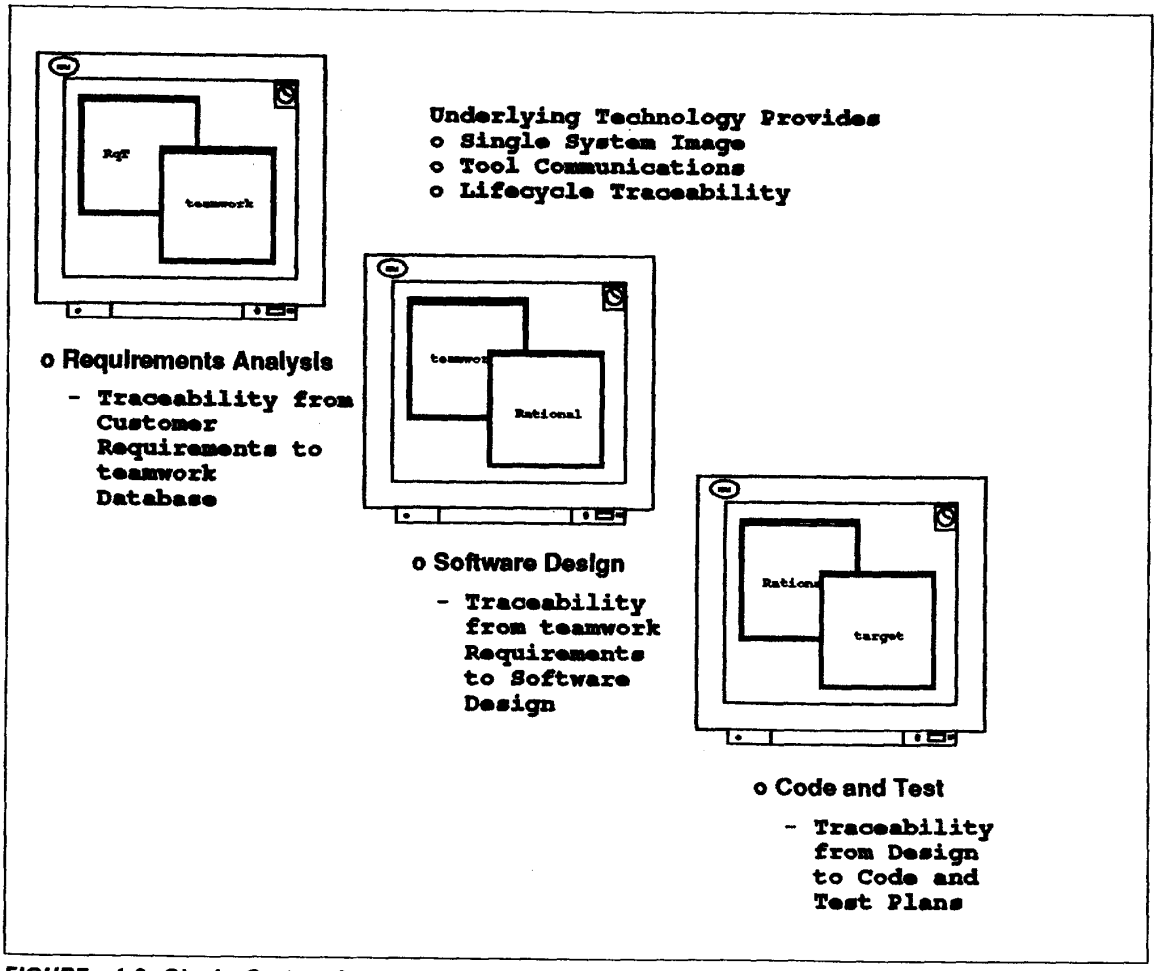

FIGURE 1-3. Single System Image and Full Requirements Traceability Is Provided Across Software Development Lifecycle

The Specification model defines the implementation boundary, indicating the hardware and software configuration items to be developed. The Design model then translates the Specification model to a design, repackaging the information to reflect the implementation's particular hardware and software environment and setting the stage for detailed software design.

Upon completion of these successive models, the developer will have examined the software from Operational, Processing and Physical Views, and captured the time sequencing of the software, as well as the functionality of the software and how it is to be constructed.

INTEGRATED CASE ENVIRONMENT ADVANTAGES AND DISADVANTAGES

As summarized in Table 1-1 on page 6, the Trainer and Simulation department's current implementation of an integrated CASE environment carries a distinct set of software development advantages and disadvantages. Starting with advantages, it is clear that an integrated set of tools implementing a selected software development method provides a setting that enhances communications between requirements and software developers. Over the development lifecycle, the integrated environment preserves concurrency between requirements and design while minimizing

the need for manual paper transfer of software requirements between systems and software organizations. In total, more time and energy will be available to the developers to worry about important system and architecture problems as compared to documentation generation problems.

On the negative side, though CASE products have been the focus of much debate for some time in technical publications, the tools are still somewhat immature, with limited language support. The buyer must beware of marketing hype that oversells a tool's true capabilities. As truly integrated tools are not yet available, attempting to interconnect equipment and software from different vendors continues to be problematic and should be investigated carefully before assuming that such tools will work together as advertised.

Lastly, regarding Ward's CASE Real-Time Method™, it is our experience that this approach, as well as others, is "front-end loaded" in a number of areas. First of all, its users must receive up-front education on how to use the method and obtain results that can be applied in the user situation. Also, management support is required to allow adequate schedule at program startup for the front-end domain analysis; it is important to note that, typically, this schedule is not allocated by a customer at program start. However, without the domain analysis the entire method is weakened and results cannot be predicted.

Once actual system development begins, a brief period of follow-up methods consultation is also recommended to achieve maximum advantage of the method's impact on the development process.

Table 1-1. Advantages and Disadvantages of Current Integrated CASE Environment

CATEGORY	DESCRIPTION	ADVANTAGES	DISADVANTAGES
HARDWARE	• RISC System/6000™ • Token Ring Network • Owego BackBone Ring and 9370 Processor	• Interconnectivity Provides Single Point Access to Tools and Host Services • Powerful Commercial Workstation	• Heterogeneous Hardware and Operating Systems Mix Complicates Integration and use.
TOOLS	• Cadre teamwork® • Rational Development Environment	• Graphical Orientation • Superb Ada Development Environment • Computer-Assisted Requirements Traceability and Transfer to Software Developers • Computer-Assisted Documentation • Automated Method Support	• Immature Tools • Tool Interfaces Not Fully Integrated • Tool Expectations Different From Reality • Uni-language support
METHOD	• Ward's CASE Real-Time Method™	• Provides Software Design Framework • Structured Process • Supports Reuse and Prototyping • Object-Oriented	• Front-End Loading Must Be Factored Into Development Schedule and Cost • Requires Personnel Training • Requires Startup Consultation for Inexperienced Users

FUTURE DIRECTIONS

Future trends in CASE hardware, tools and methods indicate that product offerings will build on current advantages and close the gap on current disadvantages. Ultimately, the current functionality provided by multiple standalone tools will be integrated into vendor team solutions that will provide a seamless requirements, design and code environment. The integrated tools will provide human-engineered graphical front-ends. Once graphical requirements are created, the tools will allow simulation of these requirements, providing designers with early feedback on the validity of their requirements. Once the simulation verifies requirements correctness, the requirements will be translated to levels of design and, ultimately, to the finished software product. Full requirements traceability will be preserved at all levels of detail, and automatic documentation production will be provided.

This trend, though tinted with a futuristic flavor, is beginning to be seen in the latest CASE tool offerings. It is our intent to monitor new product releases and, as opportunities arise, upgrade our existing CASE environment to strive for as highly automated a software development process as possible.

CONCLUDING REMARKS

Changes in a traditional software development approach are not made without justification, as any change requires retraining and adds some level of risk to future projects.

The Training and Simulation department's goal, primarily driven by a training system contract opportunity, was to create an integrated CASE environment that would take advantage of emerging technology and software development methods and yield higher software quality and productivity. Though the front-end personnel training and domain analysis add additional costs, we believe these can be amortized over the development lifecycle and future contracts. By automating the currently manual transition from software specification to design and code, removing the schedule impact and interpretation problems associated with these transitions, and striving to make the day-to-day handling of paper requirements specifications obsolete, we believe the resultant software development productivity and quality will more than offset this front-end impact.

ACKNOWLEDGMENTS

We wish to acknowledge the contributions of Mr. Richard Saxton for his efforts in implementing the Owego Integrated CASE Environment.

References

1. Paul T. Ward, "The CASE Real-Time Curriculum™", Course Notes, Software Development Concepts, New York, NY (1989).

2. David P. Wood, William G. Wood, "Comparative Evaluations of Four Specification Methods for Real-Time Systems", Carnegie-Mellon University, Software Engineering Institute Technical Report, 1989.

3. Rational Design Facility: Cadre Teamwork Interface, Users Manual (1989).

4. Cadre team*work*® CORE Product Training Workbook (1989).

5. J.M. Sagawa, "Repository Manager Technology", IBM Systems Journal 29, No. 2, pp.209-227 (1990).

Peter H. Luckey IBM Federal Sector Division, Rt. 17C, Owego, NY 13827. Mr. Luckey received the M.S. degree in Computer Science from Purdue University, West Lafayette, IN, in 1983 and the B.S. degree from Houghton College, Houghton, NY, in 1974. Since 1983 he has been with the Software Engineering organization of IBM's Federal Systems Division in Owego, NY. Throughout his tenure with IBM, Peter has been involved with the insertion of software engineering technologies, (including Ada, Object-Oriented Design, Software Reuse, and CASE), into Owego's software development process. Most recently a software engineer on the Special Operations Forces Aircrew Training System (SOF ATS) project, he is a member of the Association for Computing Machinery and SIGAda. He is also an adjunct professor at the State University of New York in Binghamton, NY, teaching Software Engineering.

Robert M. Pittman IBM Federal Sector Division, Rt. 17C, Owego, NY 13827. Mr. Pittman received the B.S degree in Electrical Engineering from Purdue University, West Lafayette, IN, in 1978 and the M.S. degree in electrical engineering from Renssellaer Polytechnic Institute, Troy, NY, in 1982. Since 1983 he has performed Systems Engineering for the IBM Federal Sector Division in Owego, NY. He has been involved in a full range of Systems Engineering activities, from front-end business acquisition to contract performance, on projects ranging from classified military communications systems to automated manufacturing systems. Most recently an advisory engineer on the Special Operations Forces Aircrew Training System (SOF ATS) project, he is a member of the Institute of Electrical and Electronics Engineers and an adjunct professor at the State University of New York in Binghamton, NY.