INTEGRATED CASE:
Our Experience with teamwork® and Rational

Peter H. Luckey, Robert M. Pittman, and Richard D. Saxton
IBM Federal Sector Division, Route 17C, Owego, NY 13827

1.0 Abstract

Mapping existing software development processes to new development methods, and implementing these methods with new and emerging Computer-Aided Software Engineering (CASE) tools, is now beginning to fulfill the original CASE promises of minimizing the cost and improving the quality of software. In addition to these advantages, the computer-assistance provided by the tools, coupled with the object-oriented approach advocated by current methods, promises both near-term as well as long-term productivity increases.

In response to recent business opportunities, as well as a desire to achieve a significant improvement in development process execution, a team of systems and software engineers at IBM's Owego, NY facility has investigated the quality and productivity improvements available through the application of an integrated software development environment. This environment is comprised of state-of-the-art hardware, CASE tools and an object-oriented software development method. In this paper, the concept of an integrated CASE environment is introduced, as well as Owego's current heterogeneous implementation of such an environment. The current status of the environment is presented, and its advantages and disadvantages are discussed. Future directions of CASE technology and the Owego effort are also presented.

For many years IBM's Owego, NY facility has performed software development and design in a traditional, non-integrated fashion. This approach has served the site well in its development, integration, and production of complex systems for all branches of the military. In the last five years, a number of non-integrated tools have been incorporated into the development process to aid systems and software engineers.

In response to recent business opportunities, in parallel with ongoing research into the software development process being performed by various site and division level working groups, the authors have, in conjunction with members of the site's Systems Engineering Technology (SET) department, undertaken a continuing initiative to examine alternative approaches to system development. Though interested in the entire development lifecycle, the authors have, based on a perceived need, focused their attention primarily toward increasing the quality of software requirements and resultant designs and code. Emphasis has been placed on the productivity and quality gains possible from application of CASE tools and object-oriented development methods to this area of the system development lifecycle.

As a result of this examination, it was reaffirmed that a major problem to be solved was related to developing software requirements, documenting the requirements in Software Requirements Specifications (SRSs) and Interface Requirements Specifications (IRSs), and managing the interface between Systems and Software Engineering organizations. The problems found in the software requirements development process were:

- SRSs are continually plagued with improperly detailed requirements
- Though the Owego Systems Engineering organization possesses a detailed software requirements development approach, lack of strict adherence to this approach has caused software
requirements level and type of detail to be often driven as much by individual engineer personalities and experience as technical need.

- Improperly detailed software requirements containing varied contents have caused some of the resultant specification documents to take on a "life of their own" once delivered to the software developers, 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 development resources, and difficulty in maintaining traceability between the specification and subsequent software design due to the instability of the requirements.

Recognizing this requirements development challenge, exacerbated by the continued increase of software content and complexity in Owego products as well as customer requirements for increased quality and productivity at lower cost, the team of systems and software engineers decided to investigate the incorporation of new methods and tools into the software development process.

2.0 Introduction to Integrated CASE (ICASE) Environments

In order to discuss improved productivity and software quality through utilization of an ICASE environment, before examining the details of a specific implementation it is important to understand the characteristics of a generic environment and understand what is meant by an "integrated" environment.

There are four facets to an ICASE environment:

1. Methods to guide and focus the personnel regarding the use of the software tools
2. Hardware to support the software tools and the tool integration
3. Software Tools to support the chosen methods and aid in the performance of the many and varied software engineering tasks
4. Personnel with the appropriate talents, skills and management support to perform the software engineering tasks using the chosen methods and tools.

There appears to be consensus on two fundamental approaches to CASE integration and three levels of CASE tool integration. The two integration approaches are:

1. Tool-Centered -- Integration of two tools is accomplished by at least one of the tools understanding the format of the data produced by the other tool. Each tool most likely has its own way of representing the data. Direct communication occurs between the tools.

2. Data-Centered -- Tool integration is accomplished by each tool accessing a common database. There is a common representation for the different forms of data, and each of the tools that uses the data understands its format.

The three levels of tool integration are:

1. Visual Integration - Refers to a standard user interface, with a consistent mode of invocation and consistent appearance and behavior.

2. Tool Control Integration - Refers to tool-to-tool messaging, event notification, event transportation, event interpretation, and process invocation.

3. Data Integration - Refers to a common data model and direct data sharing between tools via a common database or repository.

These fundamental approaches to integration are typically referenced to the European Computer Manufacturers Association (ECMA) CASE Environment Reference Model. This model, which represents a conceptual CASE architecture as compared to an actual implementation architecture, is illustrated in Figure 1.

To determine the implementation components of our integrated CASE environment, the engineering team examined the software development lifecycle and the specific requirements of near-term business opportunities to select a set of high-level requirements that the environment must meet. The team concluded that the ICASE environment, in order to not only support business opportunities but also improve overall software quality and productivity, must:
Figure 1. ECMA CASE Environment Reference Model

- Support reusable requirements, designs and Ada software
- Provide computer-assisted software development tools across the software development lifecycle
- 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
- Support a lifecycle-encompassing development method, preferably object-oriented, that is well supported by the tools yet is tool independent.

The method must provide a cognitive framework within which the systems and software engineers can use the tools to perform the development lifecycle phase tasks.

- Support graphics-based tools to the greatest extent possible
- Provide a single point interface to the many tools used by each engineer.

Based on these requirements and available technology, the team concluded:

- The principle hardware components should be engineering workstations, which can be easily
networked together. A workstation used as a network file server would be required for software tools.

- The software tools must individually or collectively support requirements analysis and specification, software design, code and test. The tools must also support integration, configuration and project management, quality assurance and document production, as well as prototyping and object-oriented methods that encourages reuse. The appropriate tools, interfacing electronically, must follow either a tool-centered or data-centered integration approach.

- The lifecycle-encompassing method must support an object orientation. Within the method there must be support for reuse at the various levels of development, prototyping, and formalism.

- The personnel pool should be made up of the appropriate mix of domain experts, methodologists, toolsmiths, system and software engineers, programmers, testers, configuration management and quality assurance personnel, and management.

3.0 Instance of an Integrated CASE environment

The selection of the Owego Technical CASE Environment hardware and software tool set and methods was guided by three major influences. First, the selection of the front end, or upper CASE software tools, was defined directly by the near-term business opportunity. Second, as previously mentioned, there existed software development experience in the facility with a variety of hardware, software tools, and methods used in a non-integrated fashion. Third, we were influenced by the IBM FSD Systems Engineering Technology Working Group (SETWG) and the IBM Owego Systems Engineering Technology (SET) Department. While the upper CASE tool requirement pointed to the use of Cadre teamwork® for front-end requirements development, previous Owego Ada software development projects pointed to the Rational Ada Development Environment to support software design, code, test and integration. The SETWG and SET had done a considerable amount of investigation into CASE tools and methods to support the systems engineering aspects of software development; in concert with the Cadre teamwork® tool recommendation, they also recommended the Ward CASE Real-Time Curriculum™ as the development method that would satisfy our requirement to develop reusable, real-time software. As illustrated in Figure 2, the selected tools support our Full Scale Engineering Software Development Process from requirements analysis through software design, code, debug, and test.

To meet the development requirements and achieve the desired software development quality goals, the systems and software engineering team next investigated available hardware and connectivity technologies that would support our chosen CASE tools. The team concluded the most effective and efficient means to implement the environment would be to develop a laboratory consisting of IBM workstations connected via a Token Ring network.

3.1 Hardware

The computing platform selected to host our CASE software was the IBM RISC System/6000™ workstation. The CASE laboratory architecture was implemented with a number of RISC System/6000™ Model 320 and 530 workstations to provide user station and application software server functions, as illustrated in Figure 3. One Model 530 was set up as the Cadre teamwork® server and two Model 530's were set up as disk servers, one each for the two Rational Ada development systems required to support our Ada software development requirements.

The workstation connectivity provided by the CASE Laboratory Token Ring Local Area Network (LAN) and facility backbone enables users to access the installed CASE tools from anywhere within the facility. This capability has allowed us to offer our CASE environment to those programs requiring access to the tools but not able to work within the confines of the laboratory.

3.2 Software tools

While the RISC System/6000™ Model 320 workstations provide an AIX-based platform for Cadre teamwork®, the Rational R1000 Coprocessors provide an outstanding Ada software design, code,
debug and test environment. DoD-STD-2167A document generation is accomplished from either the teamwork® or the Rational environments, with document production provided by Interleaf Technical Publishing Software™.

Cadre provides a number of software tools that fit well into our integrated environment. The teamwork® tools collect project management information, provide checks for model completeness and balancing, and provide requirements traceability and other capabilities:

- **RqT** is Cadre’s requirements traceability tool. It allows the systems analyst to parse the Statement of Work or other requirements documents into the individual requirements. Each of these requirements are then electronically linked with a structure in the teamwork® database or with a file in the system. Since RqT is a very new product as of this writing, we intend to continue to investigate the applicability of this tool to our requirements traceability applications.

- **teamwork/IM®** is an extensible environment for Information Modeling. It assists systems engineers during requirements analysis with the rapid creation and integration of Entity-Relationship Diagrams and Data Dictionary Definitions.

- **teamwork/SA®** is an extensible environment for Systems Analysis. It promotes rapid creation and revision of Data Flow Diagrams, Process Specifications, and Data Dictionary Definitions.

- **teamwork/RT®** is an extensible environment for real-time modeling. It expands the capabilities of teamwork/SA by supporting real-time (control flow) analysis modeling for mission critical, process control and interactive systems. Teamwork/RT’s expanded Data Flow Diagram editor provides Control Flow and Control Specification connectors. It also includes State Transition Diagram and State Transition Matrix editors to create Decision Tables, State Transition Tables, Process Activation Tables, and State Event Matrices.

- **teamwork/SD®** is an extensible environment for Systems Design. It supports industry standard Structured Design techniques for software development. Designers create and edit...
Figure 3. Integrated Software Development Environment

Structure Charts and Module Specifications for maximum development team productivity.

- **teamwork/Ada®** is an extensible environment for Ada software design. It allows the designer, using Buhr notation, to graphically depict the software architecture in an object-oriented fashion. Ada skeletal code can be generated from Ada Structure Graphs.

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

At software requirements release time, which signals initiation of high-level software design and subsequent low-level 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 Requirements Interface (RRI). This interface provides an automated, paperless transfer of requirements from systems engineering to software developers. The RTI, coupled with the RXI and the graphical nature of the teamwork® tool, provides an important requirements traceability capability across the software development lifecycle.

As originally presented in Figure 2 and further illustrated in Figure 4, requirements traceability is avail-
able starting at the software requirements analysis phase of a project and continues 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 teamwork® 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.

Documentation production capabilities are available with most CASE tools today. At IBM Owego, because of our Federal and Military customers, the documentation that we produce is defined by DoD-STD-2167A. Many CASE vendors now offer 2167A document generation software to address this market.

The Owego Environment also has the capability to generate non-2167A documentation using the Cadre teamwork® and/or Rational tools. The Rational documents are generated from Cadre teamwork® database information extracted by the execution of the RTI software. Documents from teamwork® are generated using a combination of native Document Production Interface (DPI) functions, Interleaf Technical Publishing Software™, and other teamwork®-specific vendor software, such as DocEXPRESS™, an automation tool from Advanced Technologies Applications, Inc. With this variety of documentation development options, the CASE environment provides significant flexibility in document production.

The Rational Design Facility (RDF), another component of the Rational Environment, works with RTI to automate the production of the DoD-STD-2167A documentation. An extensive Configuration Management and Version Control (CMVC) system in the Rational environment can control configuration management for the teamwork® requirements database as well as the Rational-based design information, and is accessible on the LAN. In essence, the Rational Environ-
3.3.1 Personnel

The concept of improvement in software productivity is not, of course, aimed at improving the speed of machines or code, but aimed squarely at allowing people to create high quality, easy to maintain software in a minimal amount of time. Though we would like to imagine today’s software development environment to be highly automated, it is still the engineers and programmers that produce the product.

In the Owego CASE Environment, we have found it necessary to introduce some new personnel as well as change the role of some existing personnel:

- Domain experts

As explained in Figure 2, in the traditional development approach, front-end System Requirements Analysis and System Design precedes actual Software Requirements Analysis and the subsequent creation of software requirements. In order to best accomplish generation of software requirements without getting our engineers too wrapped into tools and methods distractions, we have found it advantageous to employ Domain experts to analyze system level requirements. Though not a truly new concept, we have conscientiously isolated our Domain Experts, primarily at the front end of a program, from involvement in tools, methods, and the mechanics of generating requirements. These
experts verify the quality of system level requirements, assist in levying the system level requirements to system segments, and consult on requirements allocation to individual hardware and software configuration items.

**System and Software Development Methodologist**

The system development lifecycle illustrated in Figure 2 is perilous if not mapped out in advance. In order to get a mapping of the Development Process (i.e. DoD-STD-2167A) to the selected Development Method (i.e. Ward), and guide the application of the tools towards implementation of the Method, we have found that employing the talents of a methodologist is helpful. The methodologist helps ensure that the overall system development approach, including hardware and software design, integration, and test, flows smoothly and the work products developed in one stage of development are useful in a following stage.

**Toolsmiths**

Our CASE Environment tools exist to implement the selected system development method. To guarantee that the best tools are selected, purchased, installed, and maintained, it is necessary to employ one or more toolsmiths to oversee that aspect of Environment operations. The toolsmiths and methodologist cooperate to verify that the tools support the chosen method.

**Software Systems Engineers**

In an integrated, electronic software development environment, with instant access to software requirements and online Software Requirements Specifications, designs and code, the roles of the Systems Engineers and Software Architects are changing. Rather than task Systems Engineers strictly with developing SRSs and IRSs, and tasking Software Architects with translating the requirements to designs, we believe it is now possible to apply Concurrent Engineering principles in order for these two groups to work in a much more cooperative manner. We are currently investigating teaming together Systems Engineers and Software Architects, with the ultimate goal of growing personnel who can perform in both roles. With this approach, we are considering making this team responsible for not only the final requirements specifications but also the high level software design. Systems Engineers will still be responsible for overall system design, architecture and performance, as well as a high level, preliminary SRS. Programmers will still be responsible for the actual code and test, but the intermediate activities, including software architecture and performance, prototyping, and detailed software requirements, will be the responsibility of the Software Systems Engineers.

As indicated in Figure 5, this team, with their electronic window into the software requirements and design, can eliminate their previous paper interface. We believe this approach will free systems engineering to develop better quality system level requirements, improve the quality of the software requirements, and yield a better software product.

**IVV, CM, SQA**

The software products produced by this CASE Environment still require the talents of Independent Verification and Validation (IVV), Configuration Management (CM), and Software Quality Assurance (SQA) personnel. However, the computer assistance provided by the CASE tools, especially the Rational, provide support to these disciplines that was lacking in previous development environments.

**System Administrator**

Though teamwork® and Rational provide computer-assisted Configuration Management support for requirements, designs, code and tests, a System Administrator is still required to maintain the configurations and perform desired backups. Though the manual aspects of this task have been reduced, they have not been eliminated.

**4.0 Advantages and disadvantages**

As summarized in Table 1, Owego's current instantiation of an ICASE Environment carries a distinct set of advantages and disadvantages, as discussed in the following sections.
### Increasing Level of Refinement

<table>
<thead>
<tr>
<th>Systems Engineering (SE)</th>
<th>Software Systems Engineering (SSE)</th>
<th>Programmers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Responsibilities</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- System Design</td>
<td>- Software Architecture and Performance</td>
<td></td>
</tr>
<tr>
<td>- System Architecture and Performance</td>
<td>- Software Trade Studies</td>
<td></td>
</tr>
<tr>
<td>- System Trade Studies</td>
<td>- Prototyping</td>
<td></td>
</tr>
<tr>
<td>- Project Technical Management</td>
<td>- Detailed Software Requirements</td>
<td></td>
</tr>
<tr>
<td>- System Operational Scenario</td>
<td>- Systems/Software Team That Understands</td>
<td></td>
</tr>
<tr>
<td>- Interface to Integration and Test</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Deliver: High Level SRS to SSE</td>
<td>- High Level SW Reqts/Languages/Hardware</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Interface to IVV</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Deliver: Low Level SW Requirement (SRS) to SW</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- High Level Design / Detailed Design</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Code and Unit Test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Module Test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Produce SDD and IOD</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5. Software/Systems Engineering Concept.**

### 4.1 Hardware advantages and disadvantages

The authors' experiences clearly indicate that a set of high performance workstations interconnected on a network offers requirements developers and designers enhanced communications. Over the development lifecycle, this arrangement offers simultaneous, electronic access to requirements and resultant designs and code. Unfortunately, the current lack of tool integration forces the user into a heterogeneous mix of hardware and operating systems, which complicates hardware integration and use.

### 4.2 Software tools advantages and disadvantages

Current tools represent an evolutionary status point on the overall CASE tool transition from stand-alone tools with unique user interfaces and databases to a truly integrated CASE environment. As such, tools like Cadre teamwork® and the Rational Ada Development Environment provide, respectively, desirable graphics-based software requirements development and superb Ada design, code and test capabilities. The computer-assisted transfer of software requirements information from systems engineers to software developers provided by the Rational Teamwork Interface (RTI) is extremely powerful. The computer-assisted DoD-STD-2167A documentation capability is also a positive step, as it moves the often-compelling document generation focus to system development, where it belongs. The focus shift towards development of requirements, designs and code rather than the paper representation of this information allows the developers to concentrate on technical problems rather than documentation generation problems.

The majority of today's CASE tools are also missing an important set of cross-lifecycle support tools. These include requirements traceability, program and process management, and the automated handling of change requests and problem reports. These capabilities must be present in development projects of any size, but most CASE tool users must apply existing systems or tools in
conjunction with CASE tools to fully provide these support activities.

4.3 Methods advantages and disadvantages

The authors’ experience has demonstrated that a set of tools, implementing a selected software development method, provides a setting for disciplined software development. Application of a structured process with real-time extensions allows all of the desired views of a system specification, the behavioral, processing and physical views, to be represented. Application of an object-oriented method introduces yet a fourth view, the information view, that introduces the concept of reuse and prototyping at the front-end of the specification effort.

Concentration at the front-end requirements phase of software development and addition of the information view, however, illustrates a disadvantage of such a development method. Addition of these front-end activities introduces cost and schedule factors that today are not typically factored into development efforts. Also, since most development methods are non-trivial to apply, there is a front-end investment in personnel training. 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.

Lastly, it must be realized that application of CASE technology requires a level of management commitment. Capital investment for tools and software, time for tool and method training, allocation of support resources, and the “front-end” analysis load on project cost and schedule must be accepted. Management and customer support to allow adequate time at program startup for the front-end domain analysis must be addressed prior to the start of a project.

4.4 Personnel advantages and disadvantages

In discussing the Owego CASE experiences, the tools especially have come under criticism. However, it is the computer-assistance in executing the development method that offers one aspect of the expected productivity gains that make CASE technology attractive. Given that a user wants to develop applications according to a development process such as DoD-STD-2167A, attempting to execute a detailed, disciplined development method without CASE tools is not worth consideration.

The learning curve and additional training levied on personnel was discussed previously; another twist on the personnel issue is the demand for new talents in the area of Domain Experts, Methodologists, and Software Systems Engineers. This initial difficulty was eased in Owego by our SET Department, which passed on a good deal of previously acquired CASE tool and development method knowledge. However, transition to CASE technology, and introducing relatively unproven ideas such as Software Systems Engineers, is still challenging. We see a need for more training and experience, especially in production contracts, before the CASE approach is more widely adapted across our facility and more personnel become knowledgeable in the required areas.

Table 1. Advantages and Disadvantages of Owego’s Current CASE Environment

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>DESCRIPTION</th>
<th>ADVANTAGES</th>
<th>DISADVANTAGES</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDWARE</td>
<td>RISC System/6000™</td>
<td>High Performance Workstation</td>
<td>Heterogeneous Hardware and Operating Systems Mix Complicates Integration and Use</td>
</tr>
<tr>
<td></td>
<td>Token Ring Network</td>
<td>Interconnectivity Provides Single Point Access to Tools and Services</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Owego Backbone Ring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOFTWARE TOOLS</td>
<td>CodeWright®</td>
<td>Graphical Orientation</td>
<td>Tool Interfaces Not Fully Integrated</td>
</tr>
<tr>
<td></td>
<td>Rational Ada Development Environment</td>
<td>Super Ada Development Environment</td>
<td>Tool Expectations Different From Reality</td>
</tr>
<tr>
<td></td>
<td>Computer-Assisted Transfer Between Systems Engineers and Software Developers</td>
<td></td>
<td>Unilingual Ada support</td>
</tr>
<tr>
<td>Method</td>
<td>Ward’s CASE Real-Time Method™</td>
<td>Disciplined Software Development Framework</td>
<td>Front-End AnalysisLoading Must Be Factored Into Cost and Schedule</td>
</tr>
<tr>
<td></td>
<td>Structured Process with Real-Time Extensions</td>
<td>Supports Reuse of Prototyping</td>
<td>Requires Startup Consultation for Inexperienced Users</td>
</tr>
<tr>
<td></td>
<td>Object Oriented</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

381
5.0 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 development environment covering the entire lifecycle. In other words, the Data-Centered approach to tool integration will be commonplace.

The ICASE tools will be integrated at all three levels; visual, control, and data. 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 the correctness of requirements, 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. We are also participating in IBM's AIX CASE Internal Customer Advisory Council (ICAC). By doing so, we have the opportunity to stay abreast of the AIX CASE strategy and influence its direction.

6.0 Conclusion

This paper has discussed a number of issues regarding the development process currently practiced by IBM Owego Systems and Software Engineers, and the CASE methods and technologies that exist today to resolve these issues.

Two major problems were considered; the first was the Systems Engineering/Software Engineering development process. There exists the classical interface difficulties between matrixed organizations, systems and software being just one such example.

To realize the productivity that would come from a cooperative process, the team needs to apply Concurrent Engineering principles to develop software requirements and work together through software High Level Design. Until recently, there have not been common methods or tools available to support the concurrent activities of systems and software. We see that the object-oriented methods and integrated CASE environments available today are beginning to deliver on the promised benefits of CASE technologies.

The second problem, the generation of a Software Requirements Specification acceptable to and usable by all users of that document, has existed for years. The difficulties stem in part from the fact that procedures practiced by Systems Engineers provide no defined end-point to SRS development, and software designers often have difficulty defining the level and type of detail they need. Consequently, SRS's become too detailed and are subject to the style of individual authors. We have discussed how the discipline imposed by today's object-oriented methods can contribute to the generation of a well-balanced SRS, and that CASE tools are capable of producing DoD-STD-2167A documentation directly from model databases. Further, it was discussed that today's CASE tools are beginning to provide the capability to trace requirements across the development life cycle, from the beginning of an SRS through to code.

Although not explicitly discussed up to this point in this paper, the team had high initial expectations of the CASE environment:

- We expected that engineering workload would decrease by having a graphical interface front-end requirements development tool. To some extent this was correct, but we soon realized that the engineers still needed to spend quality time thinking through the domain analysis and subsequent design steps; tools are not a substitute for human engineering skills.
- We expected the electronic interface between our upper and lower CASE tools, would have been easier to use. Our experience to-date indicates that, although the tools can be interfaced together, the interface has required much more manual attention and intellectual energy than we originally anticipated. It still requires additional useability refinements.
• We had anticipated and expected to be able to resolve document change control management issues using our CASE environment. Today, we still use an Owego-developed, VM-based tool for document change management. We have not yet been successful in "filling this hole" in our CASE environment.

• As with document change control, we had anticipated and expected to resolve full-lifecycle requirements traceability issues using our CASE environment. Today we use an Owego-developed, VM-based tool for requirements traceability. We are still working to develop a full lifecycle solution.

• We expected that we could have easily overcome the difficulties discussed in the Disadvantages section of this paper.

Once we began to use the tools within our environment to perform some of the basic activities of our Owego systems and software engineering processes, we realized that although CASE tools offer many important capabilities and should be used, in general the CASE industry is still in the embryonic stage of development. The tools have some growing up to do.

Our current and planned future activities, then, are focused on using our current technical CASE environment as a learning tool and stepping stone towards a full-lifecycle environment that does not require VM-based pre- or post-processing functions.

Lastly, we recognize that 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.

Although 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 the front-end impact.

7.0 Acknowledgments

We wish to acknowledge the contributions of Mr. Kelly Lee for his ideas pertaining to the systems and software engineering workload partitioning. We would also like to acknowledge the Owego System Engineering Technology (SET) group for their assistance in evaluating various CASE methods and tools.

References


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 Sector 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, and has written a number of papers on these topics. Most recently a staff programmer in the Mission Avionics Software Development Department, 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 at Binghamton, NY, teaching Software Engineering with Ada.

283
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 Rensselaer 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 in the Systems Engineering Technology area, Mr. Pittman is currently focused on the application of CASE technology and advanced development methods to Owego programs, including Federal Systems Integration opportunities. Mr. Pittman is a member of the Institute of Electrical and Electronics Engineers, and in 1989 was Chairman of its Binghamton, NY Section. In 1989-1990 he was an adjunct professor in Computer Science at the State University of New York in Binghamton, NY, and continues to teach computer-related subjects in the Binghamton area.

Richard D. Saxton IBM Federal Sector Division, Rt. 17C, Owego, NY 13827. Mr. Saxton received the B.A. degree in Physics and Mathematics from Marist College, Poughkeepsie, NY, in 1977 and has pursued graduate studies in Physics, Mathematics, and Electrical Engineering at Ohio State and Syracuse Universities from 1977 through 1987. Since 1980 he has performed Systems Engineering for the IBM Federal Sector Division in Owego, NY. He has developed requirements specifications, system design, and system implementation plans for a number of IBM Owego avionics programs. He developed significant portions of the IBM FSD Systems Engineering Methodology, Standards, and Software Requirements Workshop. He participated in early attempts to design and implement reusable avionics Controls and Displays operational specifications and software using object-oriented development techniques. Since 1987, Rich has provided UNIX operating system and workstation expertise to IBM Owego including IBM AIX on the IBM RT, PS/2 and RISC System/6000 Workstations. In 1990, he designed and built an integrated CASE Laboratory comprised of networked IBM RISC System/6000 Workstations connected to the facility Token Ring to support Owego Ada development programs. Most recently an Advisory Systems Engineer in the Avionics Advanced Programs department, Rich has experience on such programs as the Special Operations Forces Aircrew Training System (SOF ATS), V-22, Combat Talon II, and the Light Airborne Multi-Purpose System (LAMPS).