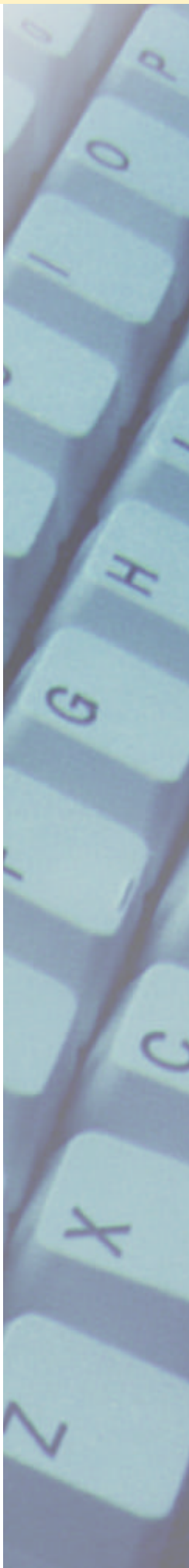
ECI Telecom Employs I-Logix' Rhapsody and UML Graphical Coding Techniques To Develop Embedded Digital Cross Connect Applications

In the telecommunications world, those developing applications in software have much to contend with: ever increasing software complexity, increases in chip performance, migration to off-the-shelf real time operating systems and collapsing time-to-market windows. These issues are creating significant code creation challenges for software engineers - challenges that are not only beyond traditional manual methods of code development but also beyond many of the software development tools and processes available today.

In short, traditional manual techniques no longer suffice and computer aided software engineering (CASE) tools (however new) can be so 'shallow' that they are not effective. Furthermore, they may even be damaging to the code creation process. Nowhere is this more evident than in the conflict between systems planning and design and the often unhealthy but necessary rush to code - where more capable software is demanded in a shorter time and with fewer defects.



Point In Case

It is widely accepted in the telecommunications world that application software is the key to providing market differentiation: and faster, better and cheaper methods of developing applications are critical for meeting time-to-revenue demands. This calls for software engineers to adopt a more iterative approach to application creation - one that ensures precise production quality code implementation while validating system behavior and compressing product development cycles.

ECI Telecom is Israel's largest telecommunications company with over 4000 employees. Split into numerous Strategic Business Unit (SBU), ECI Telecom's Transport Networks SBU is designing/developing the A::DAX, a next generation switching platform, and is employing I-Logix' Rhapsody tool to create feature-rich applications.

Dorit Katz is a software development manager within ECI Telecom's Transport Networks SBU. She says: "Our strategic business unit has been using Rhapsody for over three years. It was initially used on a small project (three software engineers) by way of evaluating the tool, but was soon employed on a much wider scale when we realized how efficient the tool is."

Based on the Unified Modeling Language™ (UML), Rhapsody blends functional decomposition and object methods in a visual programming environment that allows Katz' team to build and deploy complex embedded software applications much faster than with conventional manual programming techniques. The tool - now employed on Katz' project (ten licenses) and three other projects - is designed and optimized for the special needs of the embedded market (and telecommunications systems are extremely embedded) and validates designs in a 'virtual mode', links design documentation and implementation throughout the design process and generates production quality code.

Before using Rhapsody, ECI Telecom had used traditional coding techniques, but was finding it increasingly difficult to efficiently develop real time embedded systems. "With traditional methods, it's extremely difficult to keep your 'design view' and underlying code in sync," comments Katz. "Because Rhapsody works at the design level, engineers can develop systems more rapidly and know that the design view and underlying code are always consistent."

Unlike many CASE-centric visual development products, which are seen by many as nothing short of expensive drawing tools capable of only generating partial or prototype code, Rhapsody employs 'model/code associativity'. This affords engineers total code control over their design because any changes made in the design or implementation are consistently applied across all model representations.

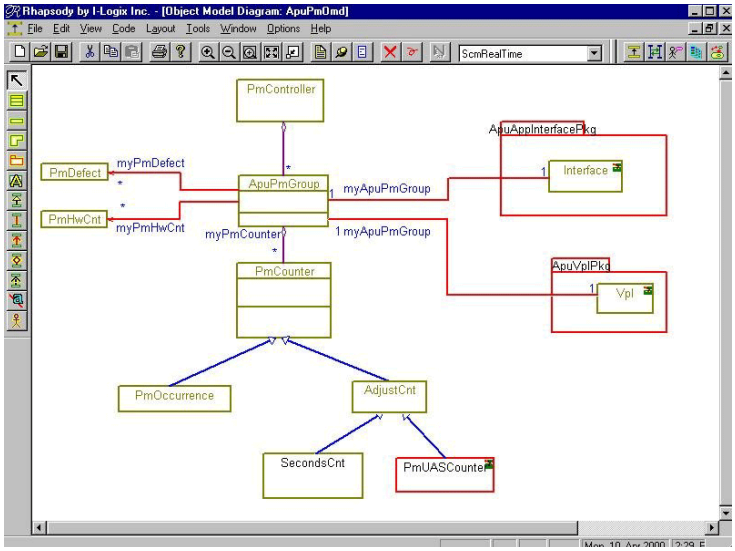


Figure 1: Class Diagram
This class diagram shows the classes and their relationships within one of ECI Telecom's Transport Network SBU's packages.

"In fact," adds Katz, "the model and the code are simply different views of the same information. This ensures that the design and code remain in sync and consistent throughout the entire development process."

Rhapsody is, effectively, a concept-to-code solution and is designed to make code developers far more productive - enabling them to build application software in an iterative manner. ECI Telecom uses the tool to ensure that product behavior is validated early in the development process.

Rhapsody maps well to ECI Telecom's design flow - which is how the company wishes to work, as opposed to being constrained by how other tools might force them to work. Through its iterative approach Rhapsody links every facet of the application software design process, including analysis, behavior validation, documentation and automatic code generation and test. To this end, Rhapsody associates changes made to either the design or the code, always ensuring their consistency.

Five Views

Today, most telecommunication system products are developed in a team environment and ECI Telecom's A::DAX is no exception. Katz's team is, like most embedded systems development teams, split into four distinct groups: systems, hardware, software and testing. Of these, the software team is further divided to reflect key sub-systems.

"Rhapsody supports the way we work in teams, and our engineers can design individually and still integrate their work with other developers within the same visual environment," notes Katz. Rhapsody users share design components through a collaborative project repository. Rhapsody stores designs as a hierarchy of packages and components within individual project workspaces, providing application management and partitioning support. The tool also provides software programmers with the ability to analyze, design, and implement their application while making use of five different yet integrated graphical design diagrams:

- Use Cases
- Object Model Diagrams
- Sequence Diagrams
- Statecharts
- Activity Diagrams

Each type of diagram allows engineers to focus on a different design perspective while building upon a common model, simplifying the overall development process.

Five Views Seeing Is Believing

Use Cases are particularly important. They capture the major functions of an application and represents specific features of the planned system. They are, essentially, an excellent way for programmers to communicate specific functional aspects of the system and the Use Case diagram can be easily understood by the technical and non-technical alike, consequently forming the central roadmap of the system's functional requirements. Katz says: "The 'uses cases' are really system specifications and allow designers to define - at a high level - the system's features."

Also of fundamental importance, and key to how Rhapsody is helping ECI Telecom's software engineers efficiently develop good code, are Object Model and Sequence Diagrams.

"Object Model Diagrams show all the objects that exist in your system," continues Katz, "where objects can be things like cross connects, or line cards. You can work at a very high level of abstraction because you can be defining how the objects will link/communicate long before defining exactly what's going to be in the objects."

The Object Model Diagram provides a graphical means of capturing the software elements and their corresponding relationships. Elements, such as objects and packages, are placed on the 'canvas', while relationships are established by selecting

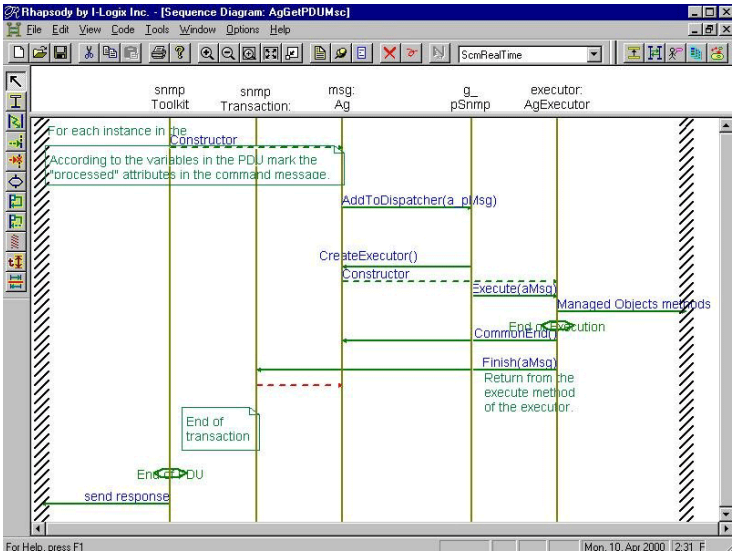


Figure 2: Sequence Diagram
This diagram is a description of an SNMP (Simple Network Management Protocol).

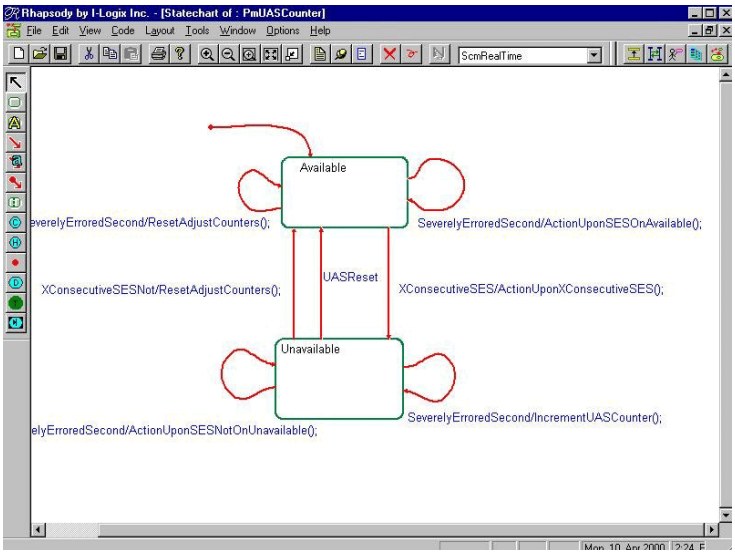


Figure 3: Statechart
This Statechart describes the states of a counter.

a relationship segment from the 'palette' and connecting the desired objects/packages. "This graphical representation is automatically converted into code which saves a great deal of time," adds Katz. Class Diagrams, similar to Object Model Diagrams, are also available through Rhapsody and Katz' team employs these extensively (*see Figure 1*).

Sequence Diagrams, on the other hand, describe the interactions between design components and the environment. Instances of these design components appear on the horizontal axis of the diagram, with interactions such as events, operations, and time-outs placed chronologically on the vertical axis.

"Fundamentally, Object Model Diagrams are static and Sequence Diagrams are dynamic," continues Katz. "Both play crucial roles in the development of our applications and the power of Sequence diagrams, in particular, is of great benefit." The Sequence Diagrams (*see Figure 2*) enable ECI Telecom's programmers to gain a clear insight into the dynamics of the implementation that is simply not available with traditional methods.

On Best Behavior

The graphical modeling of system behavior enables ECI Telecom's engineers to tackle system design based on those activities the switching platform is intended to perform. This is a powerful design approach because it focuses on what the system is supposed to do, making it possible to map design requirements directly into the design.

ECI Telecom is using Rhapsody to define and set the behavior of objects using (see Figure 3) Statecharts that provide a simple yet formal means of modeling the complex event-driven behavior common to telecommunications systems. All semantics necessary to express behavior - states, historical properties, timing, transitions, and compound transitional connectors - are available on the Statechart palette. "Rhapsody Statecharts contain features such as timing, concurrency, and inheritance that allow a clear and complete description of our systems," says Katz.

Rhapsody also features Activity Diagrams. These may be used to describe the process flow or algorithmic behavior of real-time software components and operations. This diagrammatic view closely resembles a flow chart and is thus appropriate for describing algorithmic design components. Used in conjunction with the other views of the design, Activity diagrams are a very powerful aid.

Five Views Design To Implementation

Rhapsody affords programmers the ability to generate production-ready code at any point in development, which is very much in keeping with ECI Telecom's philosophy. The capability is enabled by an open and extensible real-time execution framework that takes care of the tedious, error-prone coding required to move from design to implementation. This framework, serving as an abstraction of the underlying operating systems is entirely user-customizable. The framework is provided to programmers in source code form as well.

In addition to this framework, Rhapsody automatically generates the required source code, makefiles, and even invokes the compiler and linker, enabling the graphical design to be built into a complete library or executable.

"What's good about this, is it's not necessary to leave the design environment to compile and build the system," comments Katz. "This means you're not wasting time taking code over to the compiler - which is the case with design-centric tools." In addition, Rhapsody is integrated with compilers to speed up any transition from code generation, to compile time, and back. In terms of the flexibility, Rhapsody's code generation environment provides over 200 properties to customize code generation, resulting in high code fidelity. Comprehensive, tight, flexible code generation enables shorter development cycles and improved designs.

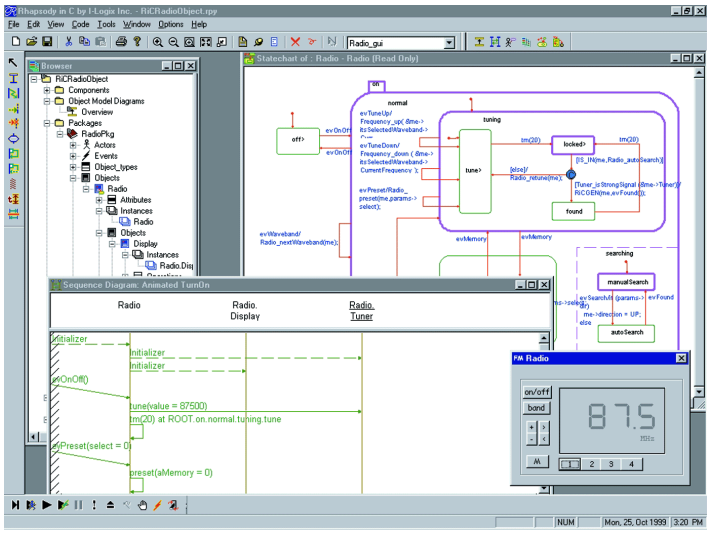
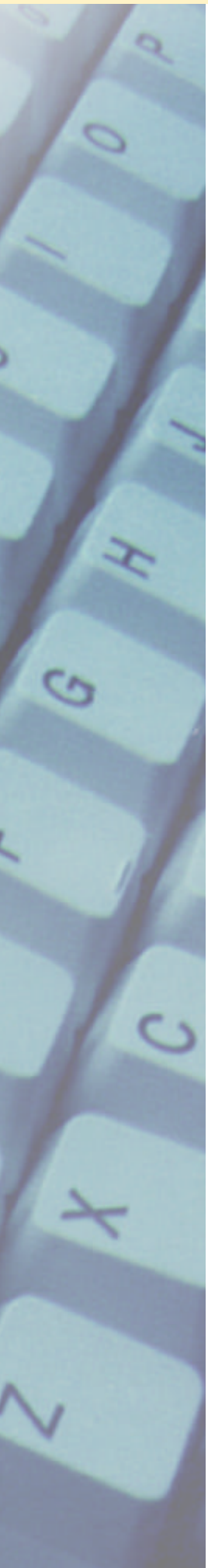


Figure 4: Rhapsody's Browser provides full navigation and shows the entire design content.
The Browser provides a single and central environment, partitioning, modifying or enhancing the entire design.



Automatic Code Generation

In the past, many unsubstantiated claims have been made about the benefits of automatic code generation because the quality of code is (as programmers know all to well) subject to many factors including available space, memory implications, hardware costs, and execution speed requirements.

With Rhapsody, the target application evolves as programmers complete and validate the graphical model. More importantly, programmers are always in touch with the code and hence able to make changes where necessary, and validate the performance and behavioral characteristics of the executable application.

"We're very impressed with this feature of Rhapsody, and the automatically generated code is production quality," notes Katz.

This code is used in several validation steps well before final deployment, and ECI Telecom's programmers use it extensively to exercise the model so they can debug at the design level, using the same diagrams to debug the application as it is designed. In fact, Katz' team uses most of the automatically generated code to drive and populate Sequence Diagrams and trace files so that real system performance and behavior can be checked against the project's initial intentions.

In this way, the code is run on the target where the team's programmers can see it executing. Unlike any other system (CASE tools for example), this can be done because the code is real and not just a simulated model of the application. Hence, programmers have total code control throughout all phases of the development process. The code is the model and the model is the code.

Conclusion

Rhapsody has elevated (and continues to elevate) ECI Telecom's code generation, debugging, and validation to the graphical design level. The result: better designs, higher quality implementations and improved time-to-market. Katz adds that the use of Rhapsody has reduced the projects design and coding cycles by at least 30%.

"We were one of I-Logix' earliest customers in Israel," concludes Katz, "in fact, we were a Beta Site, and we're receiving excellent support from the tool vendor. Employing UML coding techniques and standardizing on I-Logix' Rhapsody has, and is continuing to, really improve the way in which we're developing our telecommunications products."

About I-Logix

Founded in 1987, I-Logix is a venture backed software company that provides enterprise solutions for real-time embedded applications development in the growing pervasive computing market. I-Logix solutions significantly compress systems and software development cycles while improving product quality. These products allow engineers to graphically model the behavior and functionality of their embedded systems, analyze and validate the system and automatically generate production quality code in a variety of languages. I-Logix uniquely integrates and associates the entire design flow from concept to code across an enterprise using both conventional and collaborative Web-enabled technology.

I-Logix is a member of the Object Management Group™ (OMG™), the Bluetooth SIG, the International Council of Systems Engineers (INCOSE), a founding member of the Embedded Linux Consortium and a co-author of the Unified Modeling Language™ (UML™). The company is headquartered in Andover, Mass., and has sales offices and distributors throughout the USA, Europe and the Far East. I-Logix can be found on the Internet at www.ilogix.com

I-Logix

I-Logix Inc.
3 Riverside Drive
Andover, MA 01810
Tel: 978-682-2100
Toll Free: 888-845-6449
Fax: 978-682-5995
E-mail: info@ilogix.com
http//www.ilogix.com

European Headquarters
I-Logix UK Ltd.
1 Cornbrash Park
Bumpers Way
Chippenham
Wiltshire SN14 6RA
England
Tel: +44 1249 467-600
Fax: +44 1249 467-610
E-mail: info_euro@ilogix.com