IBM Rational Rhapsody and Rational Rose
White paper
June 2009

Rational. software

Make the transition to IBM Rhapsody software.

A guide to increased productivity and quality

Scott Niemann, IBM

Contents

Executive summary

In the past decade, modeling with the Unified Modeling Language (UML) has evolved as a powerful way to describe systems architecture and design intent. With IBM® Rational® Rose® software, IBM emerged as a key player in bringing UML to the systems market. Rational Rose software not only addressed the design aspects of a system through UML, it also included an architectural framework that helped enable it to integrate with other lifecycle development tools to create a complete solution from concept to target system. Although still a useful tool, updates in technology and development standards have paved the way for more modern solutions such as IBM Rational Rhapsody® software to offer new capabilities.

The Rational Rose application aimed to reduce the complexity of designing systems and software architectures and to improve collaboration by leveraging UML as a common language that could be easily adopted and understood by engineers. While this approach has enjoyed great success, keeping pace with technological evolution and increasing system complexity requires a solution that addresses the gaps that exist within Computer Aided System Engineering (CASE) technology. Model-driven development (MDD) is a technology that addresses testing and implementation while allowing you to easily take advantage of your existing intellectual property in the form of Rational Rose models or even source code.

IBM Rational Rose set the stage for incorporating UML into the systems design marketplace. IBM Rational Rhapsody offers a model-driven development (MDD) solution with additional capabilities to handle new and emerging technology and development standards.

MDD versus CASE

Model-driven development uses UML 2 models to specify not only the design, but also requirements and implementation. As such, these specifications of various parts of the design process are executable—they can be validated as the system is being developed. The CASE technology upon which the Rational Rose solution is built is used primarily for documenting the system architecture and specifying how the implementation will proceed. However, because the document is static, CASE technology can't provide an optimal way to validate whether a specification is correct or how a system will behave. Therefore, if an error is introduced into the system in the early design phase, it may not be caught until implementation is complete.

Highlights

Figure 1 illustrates how costly it can be to miss errors early in the lifecycle. Many errors are introduced in the early stages of high-level (HL) and software (SW) design, primarily because of misinterpreted requirements. Catching errors before the implementation phase can save both money and time. As the lifecycle progresses and architecture and implementation features are added, the cost of finding and correcting design errors can be catastrophic.

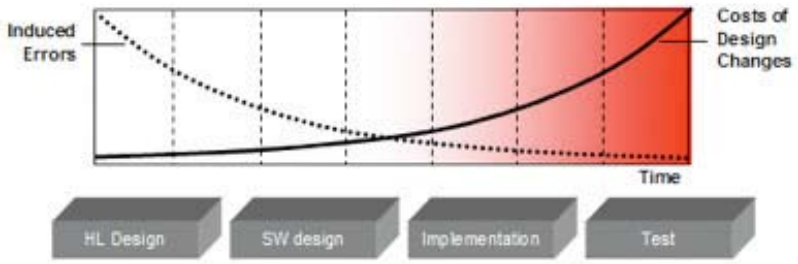

Figure 1. Cost of errors in the design phase

Rhapsody helps you find and fix errors before they are implemented, saving you time and money.

The goal of an MDD approach with IBM Rational Rhapsody is to catch the defects early in HL design because the cost of change at this point is minimal. Fixing errors in the design phase lowers not only the cost, but also the time spent in development.

IBM Ration Rhapsody and IBM Rational Rose

IBM Rational Rose users can easily migrate their data to Rhapsody software to take advantage of its advanced MDD capabilities.

At the most basic level, Rational Rose is a subset of IBM Rational Rhapsody because the Rhapsody software not only supports the latest UML standard, it also supports standard, domain-specific profiles such as the Systems Modeling Language (SysML). This flexibility enables developers to use the design language most appropriate for describing their problem domain. IBM Rational Rhapsody, an MDD platform, is used primarily for the aspects of the environment that go beyond simple modeling. We will discuss these aspects later in this white paper. The migration process presented allows users of Rational Rose software to take advantage of the advanced capabilities of IBM Rational Rhapsody after they migrate their data.

Highlights

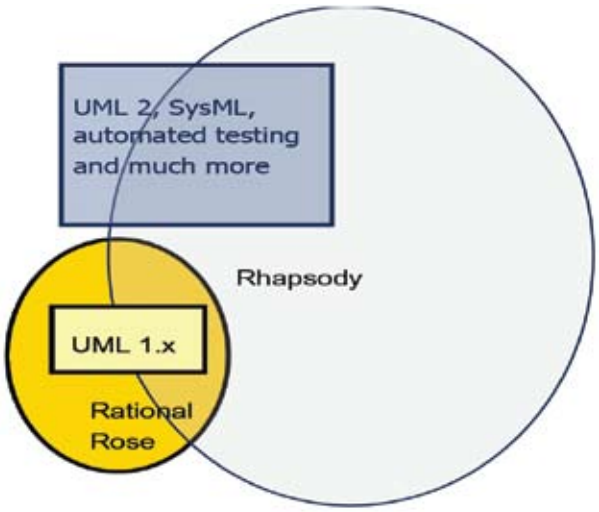

Figure 2. IBM Rational Rose and Rational Rhapsody

IBM Rational Rhapsody differentiates itself from its competitors with its robust model-based testing and debugging capabilities.

Model-based testing

The biggest differentiator for Rational Rhapsody software is its testing capabilities, specifically model-based testing. With this feature, engineers can validate the architecture as it is being created, helping to eliminate defects early in the development process, before a single line of code is written. This technology helps accelerate development at a lower cost because less time is spent fixing errors late in the design phase. Additionally, testing can be automated on the design so that regression suites will execute every night.

The two key aspects of model-based testing are model-level debugging and automated requirements-driven testing.

Model-level debugging

With model-level debugging, you can test the system as it is built and reduce defects as they are introduced into the system. There are multiple levels of granularity to this debugging, the systems level and the source level for implementation.

Highlights

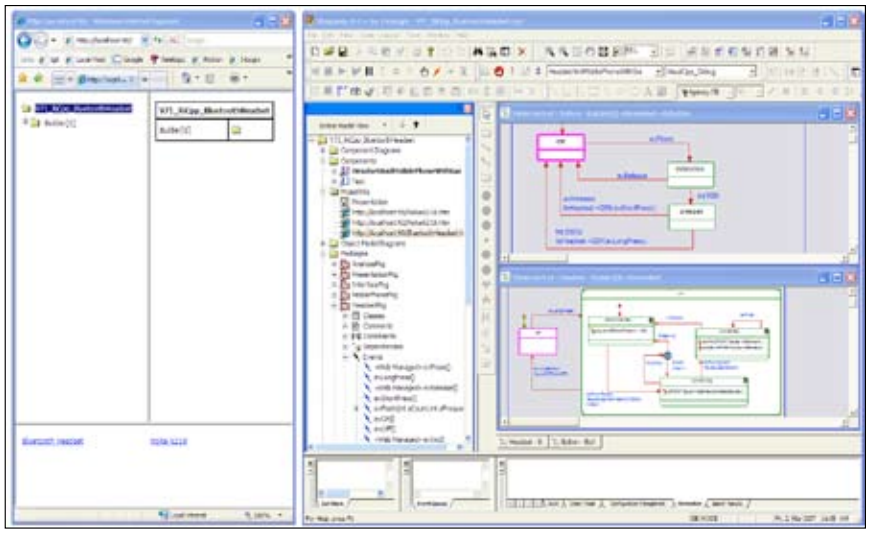

Figure 3. Model-level debugging

Visual, model-level debugging enables you to see how your integrated system will operate at implementation, even before the components are built.

In Figure 3, the design on the right-hand side highlights as the system executes. This allows you to see exactly where you are within a certain component of the system, be it a subsystem or even an actual class. The HTML panel on the left can be used to inject a stimulus into the design. This panel is automatically created by the Rational Rhapsody application so you can more easily test the system by clicking on events on the panel. If the system is a device under development, such as a mobile phone, the panel can be extended to more realistically represent the system. This approach better communicates the intent of the design to customers, marketers and engineers.

Often, developers need to perform functional testing of the system before hardware is available. Model-level debugging helps reduce the amount of time spent in integration and system testing because many defects are eliminated by these phases in the lifecycle. When the hardware becomes available, the application can be built for the target platform, and the model-level debugging can be performed on the actual target, provided there is a link from the target to the host platform.

Highlights

Linking test scenarios to your requirements database enables you easily trace and show how each test validates a customer requirement and how each requirement can and will be tested to ensure compliance.

Requirements-driven testing

Requirements-driven testing allows you to verify that your systems satisfy the customers' requirements, which may be maintained in a requirements database such as IBM Rational DOORS®, IBM Rational RequisitePro®, or in a Microsoft® Word or Excel document. These requirements can be depicted as model elements in the MDD environment and linked to scenarios that describe how the system fulfills customer requirements. These scenarios, in turn, become test vectors that can be built and automatically executed for regression testing. The scenario simply describes the expected execution of the system based on an external stimulus. The linkage between the scenarios and the customer requirements can be synchronized back to the requirements database as well to provide traceability information. This process is illustrated in Figure 4 which shows a requirements database first imported into Rhapsody software, then linked to design-level test vectors.

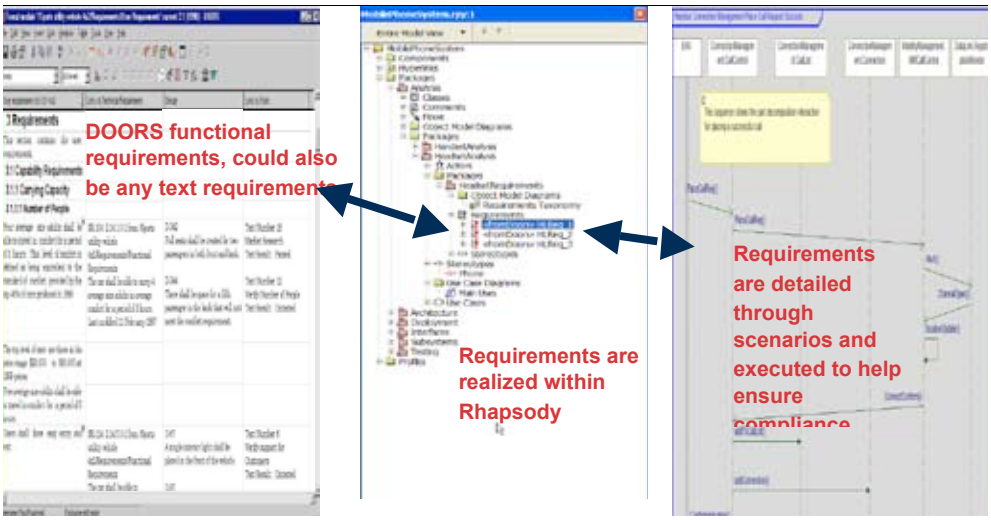

Figure 4. Requirements to design to test linkage.

Highlights

Combining the power of UML with domain-specific profiles enable organizations to create models with a clear, industry-specific, view of their design intent.

You can use the execution results of the MDD environment to determine if a requirement has been satisfied by the modeled communication systems. This means the test vector on the right of Figure 5 can be used as the benchmark of a pass/fail result. The executing model can show what the system is actually doing and automatically compare it against the benchmark. The traceability helps enable you to easily keep track of which customer requirements are being satisfied by the resulting executing system and where you still need compliance. This approach more easily allows you to produce systems that meet customer expectations.

Domain-specific modeling

UML is a powerful language for modeling, but it only provides a generic template of information for systems. Modeling closer to the relevant domain reduces ambiguity and provides a clearer view of the design intent for colleagues and customers. IBM Rational Rhapsody supports the latest version of SysML to enable systems engineers to use a functional decomposition approach for building high-level system components and modeling subsequent subsystems.

Beyond horizontal disciplines, such as software and systems engineering, Rhapsody software supports domain-specific profiles for the automotive, aerospace, defense and communications industries. Profiles can also be created for your organization to help you better specify the true intent of your designs. Figure 5 is an example of the Department of Defense Architecture Framework (DoDAF), a framework used in the aerospace and defense market, represented in the Rhapsody solution.

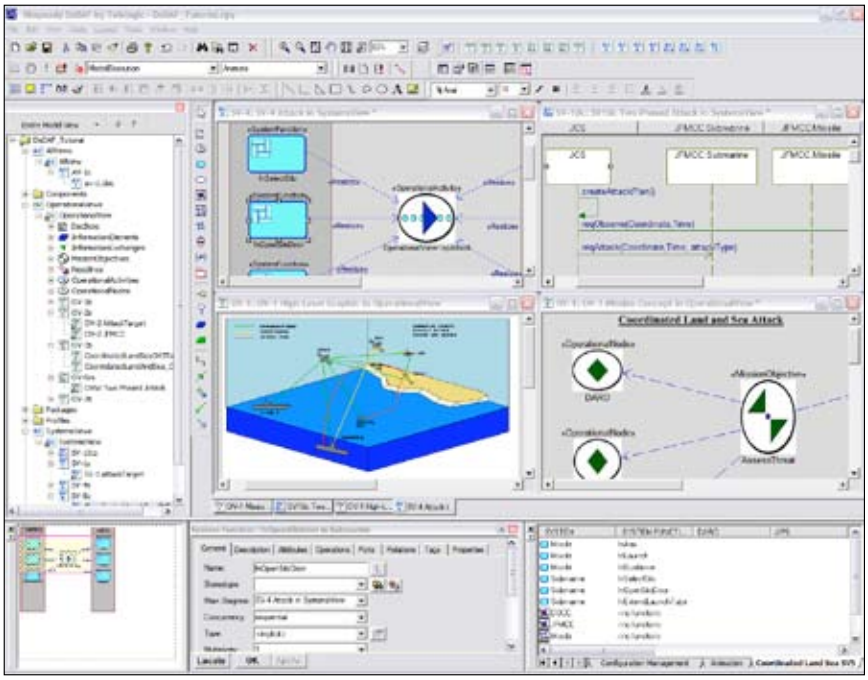

Figure 5. Example of an industry-specific profile—DoDAF.

Profiles can also be created for organization-specific purposes. Profiles are extensions of UML 2, so they are compatible with XML Metadata Interchange (XMI), which is used for model interchange. A systems designer can model a profile in Rhapsody software to use as a modeling template and distribute the template to systems developers throughout the organization. This approach can better enable your organization to adhere to design best practices and guidelines.

Highlights

Rhapsody software facilitates a flexible coding environment that integrates with multiple configuration management tools and enables collaborative development.

Collaboration

Another area where Rational Rhapsody benefits are clear is in development teams, where the focus is on producing robust software. Rational Rhapsody software provides an environment where the model and the code are integrated. If you simply want to create an operation in Rational Rhapsody and handwrite the code in C, you can do it. The information is stored in the model. Rather than maintaining the model and code separately, you have one model element that you check in and out of your configuration management (CM) repository, which can greatly reduce and simplify parallel development.

Graphical differencing and merging

The graphical differencing and merging capability of Rhapsody software makes the integration of model and code possible. As shown in Figure 7, Rhapsody software not only comes with its own auto-merging facility, it also integrates with CM tools such as IBM Rational ClearCase® software. For example, you can simply run the auto-merge facility in Rational ClearCase software, and the Rhapsody solution differencing and merging tool is designed to automatically merge the design (both model and code). If there are any conflicts, systems developers can resolve them by viewing the graphical differences in IBM Rational Rhapsody. Model and code merging helps simplify and drastically reduce integration time.

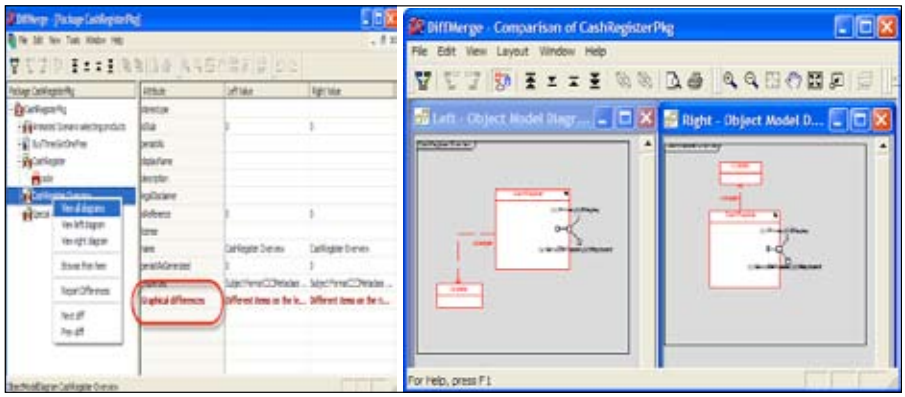

Figure 6. Rhapsody software differencing and merging

Highlights

The window on the left of Figure 6 shows the model data and indicates if there are differences that require merging. If some of that model data is code, then developers—through the Rational Rhapsody solution's integration with common text differencing tools— can use the differencing tool of their choice, including those available in Rational ClearCase software and other CM tools. But this is only necessary if the change could not be merged automatically.

In the case of graphics, the diagram on the right of Figure 6 shows how changes that cannot be resolved are displayed. You can step through these changes to create a merge candidate.

This functionality is integrated with the Rational ClearCase application through its type manager facility, so the invocation of auto-merge and Rational Rhapsody software differentiate/merge is actually done through a Rational ClearCase software command line or through a version tree.

Code-centric workflow

The synchronization capabilities of Rational Rhapsody help ensure that code changes are reflected in the model and model changes are reflected in the code.

The ability to code efficiently and effectively is extremely important to most developers. The Rational Rhapsody solution offers a series of powerful tools that can make coding easier or even allow you to design models that automatically output the code you might otherwise write by hand. No matter which method you choose, Rational Rhapsody software helps ensure that the code and model are always synchronized and the design documentation is an exact representation of the implementation. The code-centric workflow is broken down into synchronization as well as component reuse through visualization.

Highlights

IBM Rhapsody helps Rational Rose users reuse their intellectual property while taking advantage of Rhapsody's enhanced modeling and testing capabilities.

Code synchronization

The code synchronization capabilities of Rational Rhapsody enable you to produce code in whichever way is most familiar and comfortable, while still ensuring that the design and model remain synchronized. As you type code in the editing environment, the system automatically updates the model. The "code respect" feature ensures that the placement of information within the source file will be respected in the modeled environment. That is, if you add data and operations in a particular place in a file, that information will remain there through future engineering tasks.

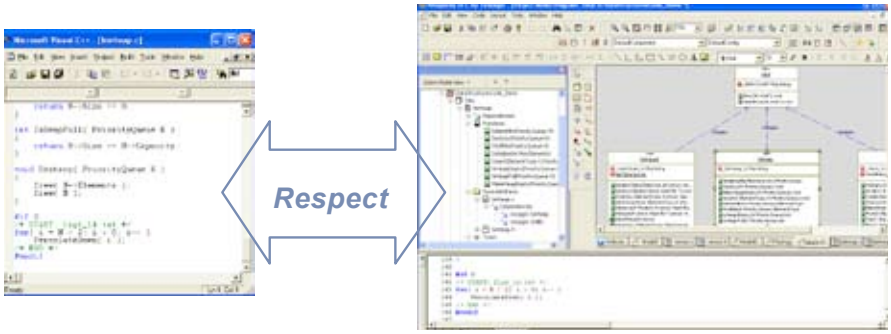

Figure 7. Code synchronization: Whether you update the code or the model, both are synchronized and code placement is respected in the Rhapsody application environment.

Modeling helps developers visualize their code, which contributes to their overall understanding of a system's architecture.

Visualization

Software reuse is often especially important to Rational Rose users because, in most cases, software has been written to comply with the design specification created in the Rational Rose application. Though it's sometimes necessary to recreate new system components from scratch, in many cases Rhapsody software facilitates the reuse of proven intellectual property, offering greater advantage over the competition.

Highlights

The component-based design features in Rational Rhapsody software facilitate the creation of reusable components to help accelerate design and foster collaboration.

Often, this intellectual property is code. In embedded systems, the code is usually written in the C++, C or Java™ language. Using a model, you can visualize the code, which helps improve your understanding of the architecture. When necessary, you can even reverse-engineer the code into the model for intellectual property capture and refine it as part of the ongoing design. The model also provides automatic design documentation so you can spend more time in design and implementation and less time documenting.

Component-based design

Component-based design uses modeling methodologies to create plug-and-play development environments where software may be created as reusable components. Rational Rhapsody facilitates this kind of development approach by offering the following:

- *UML 2 compatibility*
- *The ability to save individual components such as classes, subsystems or other software units*
- *The ability to export or reference components to other projects*

The combination of these technologies helps enable teams to accelerate the design phase by improving collaboration and reuse.

Another benefit of component-based design is the ability to create complete applications, rather than code skeletons, for virtually any realtime target. For build processes that incorporate components from third parties or other groups, Rhapsody software provides a rich command line interface that can be easily incorporated into build scripts. Make files, which sometimes present a challenge, can be automatically generated and you can also dictate precise specifications about how to generate build files.

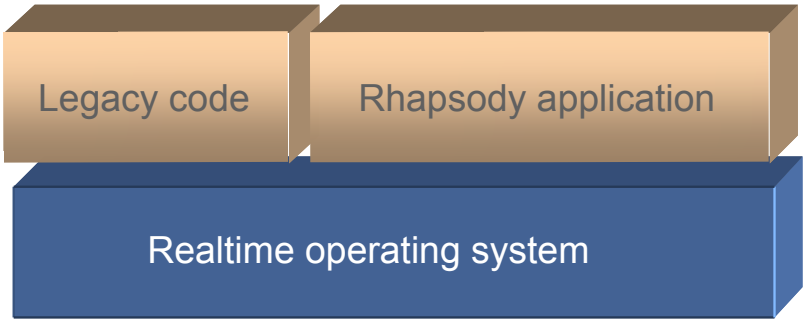

Figure 8. Rhapsody application architecture

Figure 8 depicts a simplified Rhapsody application. The Rhapsody application represents the UML 2 or other domain-specific profile used to model the system. With Rhapsody software, you can completely model code behavior, including state-machine and activity diagram behaviors. You can also easily incorporate legacy code using the visualization tools such as graphical component diagrams. You can then rapidly retarget from the host environment to actual target systems, such as Linux®, Wind River VxWorks, 16-bit operating systems (OS) or even no OS at all.

Highlights

Rational Rhapsody helps enable an MDD approach which can help companies develop higher quality systems faster and more efficiently.

The Rational Rhapsody application is designed to be target independent of the underlying operating system. Although it uses the services of the underlying operating system at all times, the Rhapsody application introduces no additional ones. Rational Rhapsody achieves this through a thin abstraction layer that enables the application to be created generically for a particular OS. The concrete operating system is easily selected using the Rational Rhapsody environment's menus. This mechanism can be extended to virtually any OS.

Conclusion

Transitioning to Rational Rhapsody and bringing the latest capabilities of MDD into your development environment can help you improve productivity, system performance and time to market while reducing cost. The primary benefits of an MDD approach can include:

- *Reduce errors early in the development lifecycle with model-based testing.*
- *Improve communication within your organization through domain-specific modeling based on UML 2.*
- *Simplify and automate documentation.*
- *Improve complex systems integration and team collaboration.*
- *Reuse existing IP in current software development processes with a code-centric workflow.*
- *Accelerate application development with a plug-and-play approach for virtually any platform and existing tool infrastructure.*

For more information

To learn more about IBM Rational Rhapsody, please contact your IBM representative or IBM Business Partner, or visit:

ibm.com/software/rational

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
June 2009

RAW14047-USEN-01