APPLICATION DEVELOPMENT STRATEGIES®
FROM CUTTER INFORMATION CORP.
THE MONTHLY NEWSLETTER ON CASE, CLIENT-SERVER SYSTEMS, AND SOFTWARE DEVELOPMENT TECHNOLOGIES

The Yourdon Analysis

In October 1996, Microsoft and Rational Software Corporation announced a strategic alliance that will have an enormous impact on the object-oriented application development marketplace. Microsoft has licensed some of Rational's visual modeling technology, and Rational has licensed both Visual Basic 5 and the repository technology that Microsoft has been working on with Texas Instruments. Microsoft's actions effectively legitimize the use of formal analysis and design modeling for large development projects.

The new Rational Rose 4.0 product will integrate with Visual Basic 5.0, Visual C++, and Microsoft's Visual J++. In addition, Rational acquired Microsoft's Visual Test product, and then subsequently acquired one of the leading testing vendors, Software Quality Automation.

Meanwhile, Rational has continued developing its Unified Modeling Language, which it will formally submit — together with Microsoft, HP, Oracle, TI, and other vendors — to the Object Management Group in January.

Bottom line: Rational has become the Microsoft of OO CASE vendors.

Ed Yourdon, Editor
Internet:
ed@ yourdon.com
Web site:
www.yourdon.com

RATIONAL SOFTWARE: THE FIRST BILLION-DOLLAR CASE VENDOR?

On several occasions in the past two years, my reports in ***Application Development Strategies*** have discussed the activities and products of Rational Software Corporation. In particular, I discussed Rational's Rose product in the November 1995 ***ADS*** issue on object-oriented CASE tools; and I discussed the Unified Modeling Language (UML), which was developed by Rational, in the March 1996 issue of ***ADS***.

But more recent events during the summer and fall of 1996 have prompted me to take yet another look at the company. As many software developers are probably aware, Rational announced a strategic alliance with Microsoft in the fall of 1996, which (among other things) resulted in the sale of the Microsoft Visual Test package to Rational for $23 million. And then in November 1996, Rational announced the acquisition of Software Quality Automation (SQA), a leading vendor of client-server testing products that I reviewed in the March 1995 issue of ***ADS***. In my March 1996 discussion of UML, I somewhat humorously referred to Rational's methodology as "the Microsoft of OO methodologies"; but with many of the recent developments, it might well be more appropriate to describe Rational as "the Microsoft of CASE vendors."

Whether this characterization is accurate (or even desirable!) remains to be seen; but there is little doubt that Rational has become one of the dominant vendors — if not *the* dominant vendor — in the high-end application development tool market. With this in mind, I decided to visit Rational's headquarters in Santa Clara, California, last month to talk with President Mike Devlin and Senior Vice President Dave Bernstein; I also discussed the new Rational Rose 4.0 CASE tool with Jon Hopkins and Adam Frankl, also of Rational.

I'll begin with a summary of the CASE market; since I covered the same topic in the October 1996 ***ADS***, this review will be quite brief. Instead, I'll concentrate on Rational's vision for application development, and the various tools, products, and services it provides as part of that vision. And of course, any discussion of Rational's plans today has to include some commentary on the company's relationship with Microsoft. I am not privy to any "secret" information in this area, but having discussed the situation with

Devlin and Bernstein, I have a much better understanding of Rational's view of the relationship.

The State of the CASE Market

As I observed in the October 1996 ***ADS***, the CASE market has come a long way since the dark days following the collapse of IBM's AD/Cycle in 1992. But during the "sea change" from mainframe COBOL to client-server GUI applications in the early 1990s, some of the original CASE vendors stumbled badly; large vendors like IBM, Andersen Consulting, Texas Instruments, and KnowledgeWare (now Sterling Software) still have viable application development tools, but they don't provide the degree of intellectual and market leadership they once did.

Indeed, the largest force in the application development marketplace during the past five years has been associated with Visual Basic (VB) — followed closely by PowerBuilder, Delphi, Smalltalk, and various other "visual development" tools. It's important to characterize these products as *tools*, and not just as programming languages; much of the important and productive activities of software development that take place with these products has little or nothing to do with the syntax of a programming language.

As long as the applications developed with these products involved the efforts of only 2-3 people for 2-3 months, the benefits of upper-CASE modeling tools were often hard to identify. But for the past few years, we've begun seeing projects an order of magnitude larger than the first-generation client-server applications — i.e., 20-30 people involved in projects that last for 20-30 months. Even these numbers are rather modest compared to the massive 5-year, 100-person mainframe development projects that we saw throughout the 1960s, 70s, and 80s; nevertheless, they're large enough and complex enough that everyone on the project team tends to agree that some formal modeling, analysis, and design are important.

I find it interesting — given the overall theme of this month's report — that such problems have been particularly evident with Visual Basic projects. Because Smalltalk, Delphi, and PowerBuilder (to name only three of the alternative development environments) are more object-oriented than component-oriented in nature, developers have often been able to manage the complexity of larger projects with well-designed class hierarchies and frameworks. But if one asks a development manager what he or she is most concerned about with Visual Basic, one of the most common answers has been, "it doesn't scale."

I'll discuss object-orientation (OO) and the related concept of *components* in more detail below; but for now, suffice it to say that the convergence of such trends as OO, components, Java, and the Internet — together with the complexities of scaling up small applications to larger ones — has helped to rejuvenate the segment of the application development industry that used to be known as "CASE vendors."

But these trends have not led to uniform success in the industry; indeed, a consolidation has been taking place for the past few years. As I noted in the October 1996 ***ADS***, Cadre and Bachman have merged to form a new company called Cayenne Software. Similarly, Integrated Development Environments (IDE) recently merged with the tools division of France's Thompson AG to form a new company called Aonix. Both of these companies have annual revenues of approximately $65-$70 million, somewhat below Rational's current level of $100 million. In addition, there are several mid-range vendors like Forté and Dynasty, with revenues of $25-$30 million, as well as $5-$10 million "niche" players in the CASE industry like Popkin Systems, Visible Systems, and Evergreen Software (all of which were discussed in the October 1996 issue).

Some of the small niche players are making a nice profit, and may be able to continue prospering in the industry as long as they resist the urge to grow rapidly; on the other hand, as the CASE marketplace matures, they may well find it more and more difficult to compete. Similarly, the mid-range $25-$30 million vendors are likely to find their market squeezed by the consolidation process.

As for the larger vendors: The interesting question is which one is likely to have the momentum and business savvy to become the first billion-dollar CASE vendor. Aonix and Cayenne are both relatively new, and both are excited by the prospect of having achieved a "critical mass" through the merger of their former, smaller constituent companies. Realistically, it's hard to imagine that these mergers would have occurred if the constituent companies (i.e., IDE, Cadre, and Bachman) were flourishing in the marketplace — but it *is* possible that the combined product technology and marketing efforts represented by the new companies could launch them toward success.

Meanwhile, of course, Rational has plans of its own. In addition to its own product development efforts, the SQA acquisition and the $23-million purchase of Microsoft Visual Test are indicative of the company's efforts to accelerate its own growth. I'll be focusing almost entirely on Rational's product development plans in this issue of ***ADS***, but the importance of marketplace "muscle" should not be underestimated.

Of course, there are even bigger fish in the sea: Sterling Software (which acquired KnowledgeWare) has annual revenues of approximately $700 million. And companies like Oracle, with products like Developer 2000, has revenues in the $2 billion range. And there's Sybase (which owns PowerSoft), Computer Associates (CA), IBM, and Microsoft — all of which are in the billion-dollar range.

But these larger companies are not "pure" CASE tool vendors; they market a wide variety of other products and services, in which CASE plays a subordinate role. Sybase, for example, is still primarily a database vendor, and is quite occupied with the intense competition in the DBMS marketplace. Oracle is certainly a successful, aggressive company — but its primary product is databases, its primary competitor (at the moment this article was being written) is Informix, and the passion of its CEO appears to be Java and the Network Computer. CA has a wide range of tools and products, but is not generally regarded as a serious CASE vendor. And while IBM periodically introduces tools that dazzle the marketplace (e.g., VisualAge), and occasionally creates a "vision" of an integrated tool development environment (see, for example, the discussion of IBM's tools in the August 1994 ***ADS***), the company has enough trouble today convincing the marketplace that it is serious about Java and the Internet; it has little or no "mind share" in the CASE marketplace.

If the existing billion-dollar software vendors are focused on issues other than CASE, then why should the smaller companies (including Rational) worry about them? One reason is that the larger software companies control vital technologies that the CASE vendors have to work with very closely — e.g., programming languages, database management systems, and so on. While a PowerSoft, Oracle, or Microsoft might be mildly interested in seeing its 4GL programming language integrated with a CASE tool, it's hard to imagine that a $30-million CASE vendor is going to have much influence on the future technology direction of a billion-dollar software giant.

But sooner or later, there is the possibility that the billion-dollar giants *will* focus on CASE for its own merits, with the same degree of passion and intensity that Cayenne, Rational, Popkin, and all of the other smaller CASE vendors have today. At that point, the "critical mass" for survival, in terms of annual revenues, probably will be around the billion-dollar mark, and most likely *not* the $100-million mark.

None of this is relevant if the aggregate expenditures for CASE-related tools and technology is only a few hundred million dollars; as I pointed out in the November 1996 ***ADS***, for example, the small market situation is still one of the biggest constraints for OODBMS vendors. But Rational believes the CASE market to be much, much larger; citing source data from the Gartner Group, IDC, META Group, and Cutter Information Corp., Rational believes that the market for tools supporting component-based development will grow in the fashion shown in Figure 1. The figures for 1994 and 1995 are fairly well-established, but it remains to be seen whether the growth curve through the year 2000 will indeed follow the pattern

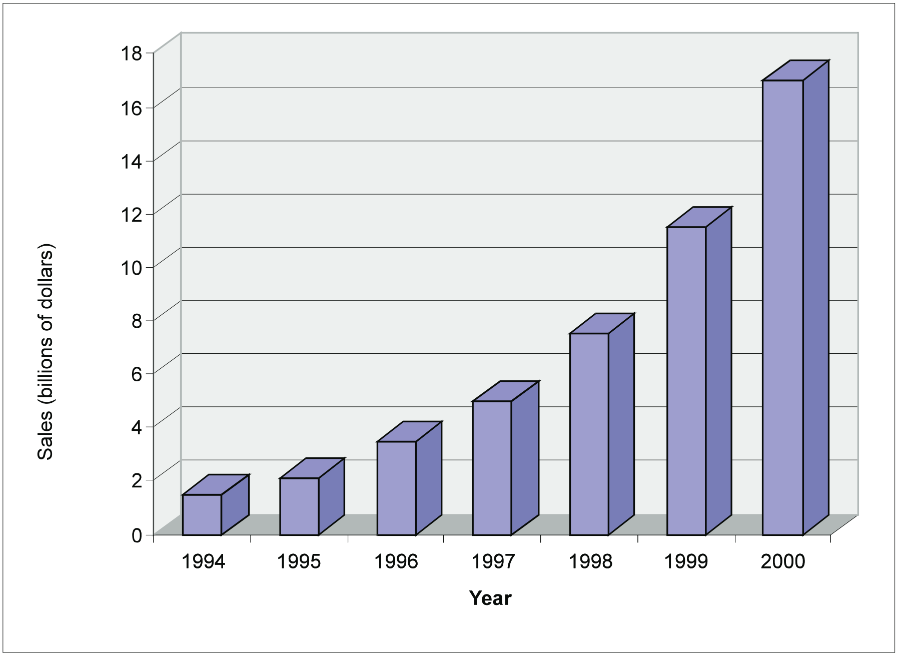


Figure 1: Rational's Estimate of Growth in the Component-Based Tools Market

shown in Figure 1. If it does, though, it certainly *will* attract the attention of the large, billion-dollar players; today's CASE vendor will have to grow to a billion-dollar size to be a serious player.

While this all sounds reasonable, it leaves one question unanswered: What prevents a company like Microsoft from attempting to swallow the entire CASE industry, as it has with so many other parts of the software industry? I'll save that question for later; first, let's look more closely at the Rational situation.

Rational's Vision of Software Development

Rational was formed in 1981, and has been characterized in a variety of ways during its corporate existence. It was originally viewed as a vendor of Ada tools and products; even today, it still supports the Ada market (notwithstanding the likelihood that all of the Ada programmers on earth will probably die of old age before all of the COBOL or FORTRAN programmers suffer a similar fate). More recently, Rational has been known as both a vendor of industrial-strength CASE products and services, and also an intellectual leader in the object-oriented methodology field.

But today, Rational describes its business as "delivering products for component-based development to customers who depend on software for the success of their enterprise." The interesting point here is that more and more companies *understand* that the success of their enterprise depends on software — rather than view software as an expensive form of overhead. Thus, while Rational's traditional customer base has been in market sectors like aerospace and transportation, it has begun acquiring more and more business recently in highly competitive, recently deregulated industries such as telecommunications and financial services.

The other key part of Rational's "mission statement" is the emphasis on *components*. Earlier, I pointed out that Rational has been a leader in OO tools, services, and methodologies — and I think it's fair to say that the OO paradigm still underlies much of the basic Rational technology. But if one regards OO as a "pure" paradigm, as exemplified by Smalltalk, Eiffel, and Delphi, it does not appear to have the same momentum and force in the marketplace as the somewhat bastardized form of the paradigm represented by Visual Basic. And while the commercially successful technologies like Visual Basic may eventually acquire some of the purity of OO, it's more appropriate today to call them "component-based" technologies.

Pure OO technologies generally use the concepts of "class" (roughly equivalent to a "set") and "objects" (or "instances") within the class; component-based technologies like VB typically operate only at the instance level. Pure OO technologies incorporate sophisticated concepts like inheritance and polymorphism; component-based technologies typically do not. Indeed, the only significant things one finds in the component-based technologies are the notions of *encapsulation* (data and process are packaged together) and *interfaces* (explicitly defined protocols for interaction with the component, which serve to hide the details of the component's internal implementation).

Interestingly, one of the reasons that OO represented an advance over the earlier structured methods is that a (pure) object typically represented a larger and intellectually more-useful "building block" for the software developer than the tiny "modules" that resulted from diligent efforts at structured analysis and design. If structured design produced the software equivalent of subatomic particles, then OO produces full-fledged atoms (i.e., in the form of a collection of methods and encapsulated data). And if OO produces atoms, then the component-based approach produces molecules. True, some of the "components" that one sees in a visual programming environment like VB or Delphi are low-level GUI widgets like checkboxes or radio buttons; but the sort of components that one finds when surfing the Net or downloading shareware software products are typically *much* larger — e.g., a Java applet for adding rows and columns in a spreadsheet form, and displaying the results in an appropriate graphical fashion.

Indeed, this example is a little more subtle than it might seem at first. While the buyer, seller, and end user of this Java applet may think of it as a "component" (i.e., a single instance of a "chunk" of software to carry out a well-defined purpose for a single user), the Java *language* supports the more-sophisticated form of "pure" OO described earlier. Thus, the internal design and construction of the spreadsheet applet/component may well involve classes and inheritance and polymorphism. Further, the server that delivers the spreadsheet applet may treat the entire applet as a class, and provide instantiated "copies" or instances of the class, for each user who needs it on his or her Web browser.

But if OO is the theoretical underpinning, it's components that the user sees and it's components that the marketplace is prepared to buy, sell, trade, and rent. And it's components that one is forced to work with in the architectural framework provided today by Microsoft — i.e., within the framework of OLE, ActiveX, and Visual Basic. Of course, Microsoft is not the only provider of such technology: As noted earlier, Oracle, Java, the Internet, and a dozen other products, companies, and related technologies provide a rich infrastructure within which this kind of component-based "software economy" can operate.

And that's the phenomenon that Rational sees as an emerging tidal wave — especially for those companies that view software as essential to their success. Whether the objective is to increase productivity or quality, reduce risk, or dramatically reduce the "time-to-market" for software-intensive products and services, the increasingly popular strategy is to use a component-based approach. The component thus becomes the focus for project management, development, testing, quality assurance, and all other aspects of software development.

Indeed, this is exactly what many development organizations have been doing for the past few years in component-based development environments like VB, PowerBuilder, and Delphi. But as noted earlier, the problem of "scaling" these projects up to the next level of size and complexity has caused serious problems. While it's theoretically possible to deal with the increased complexity in a pure OO development environment (because of inheritance, class hierarchies, etc.), in practice it's overwhelming to attempt to understand *any* large, complex system by staring at lines of code. That was true a decade ago when we were using procedural programming languages and structured methods; and it's true today with OO and/or component-based languages. Thus, for today's larger systems, *visual modeling* is the key to successful component-based development. And that, of course, is where vendors like Rational, Cayenne, Aonix, and Platinum (with its Paradigm Plus product) step in. Visual modeling tools (a term that sounds much more elegant than "CASE tool") make it possible to produce a picture of a component without looking at the details of the code; and at a higher level of abstraction, visual models provide a picture of *all* the components in an application, so that the developer can see how they fit together.

Traditionally, this has been a way of describing the "analysis" and "design" activities at the beginning of a system development life cycle. But even from the early days of the CASE industry, vendors have been interested in connecting these front-end activities to the critical step of code generation; and more recently, Rational and most of the other vendors in this field have begun focusing on visual models as the central core of a *complete* system development life cycle. Thus, Rational views its Rose product as the mechanism for supporting everything from product requirements to maintenance and version control, as illustrated in Figure 2.

However, it's important to recognize that Rose, by itself, does not automate activities like testing; by itself, it is still primarily a visual modeling tool. But with some of its earlier products, as well as its recent acquisitions, Rational has developed a product suite aimed at supporting component-based development; the suite contains the following products:

- Rational Rose — for visual modeling.
- Rational Apex — taken from its earlier Ada tools, Apex provides automated support for architectural planning, component, and "build" management.
- Rational Summit — a tool for project and process management.
- Rational Visual Test — to provide automated support for testing. Initially, this consists of the former Microsoft Visual Test product; in the near future, it is expected to incorporate the products from SQA, which Rational acquired in November 1996.
- SoDA — also taken from its earlier Ada tool suite, SoDA provides automated support for document generation. While this may have been originally for Defense Department Ada projects, it's becoming more and more important for organizations today who need to generate user manuals, technical manuals, and various other

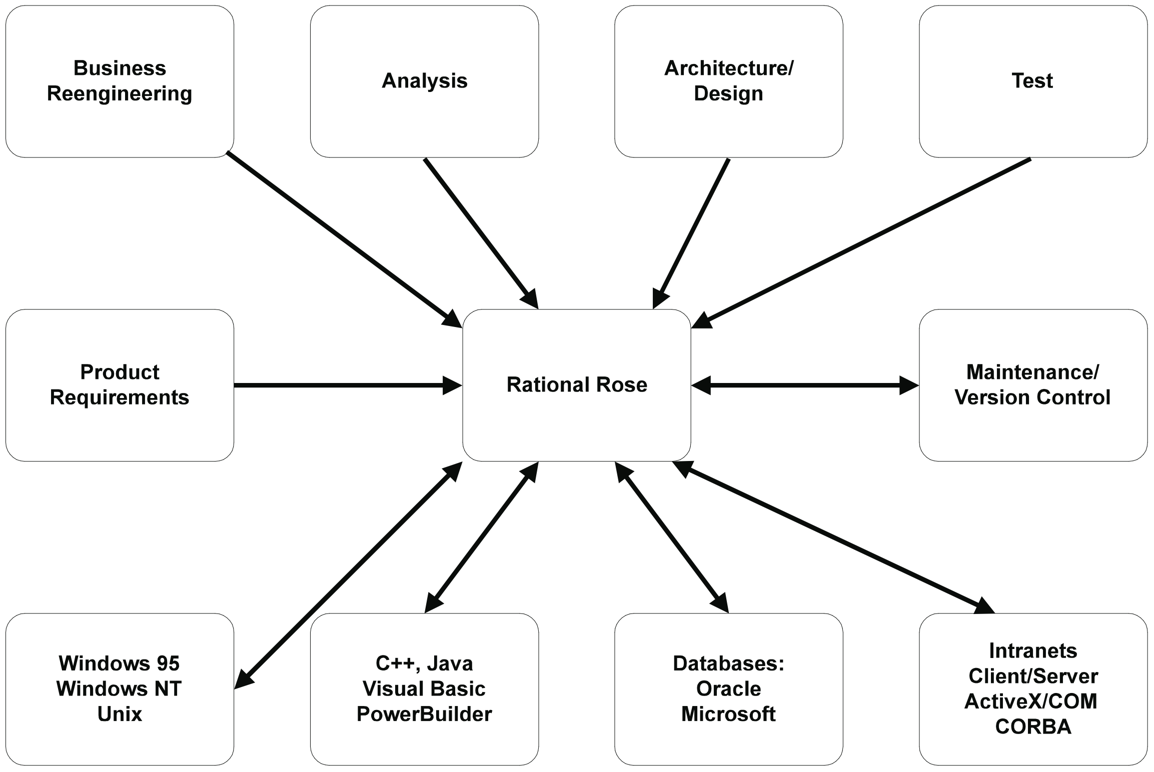

Business Reengineering
Analysis
Architecture/ Design
Test
Product Requirements
Rational Rose
Maintenance/ Version Control
Windows 95 Windows NT Unix
C++, Java Visual Basic PowerBuilder
Databases: Oracle Microsoft
Intranets Client/Server ActiveX/COM CORBA

Figure 2: The Role of Rational Rose in the Application Development Life Cycle

forms of documentation (including HTML) from the visual model created in Rose.

Taken together, all of these tools can be viewed in the architectural structure shown in Figure 3 below. Note that there are separate versions of Rational Rose for Visual Basic, Visual C++, and Visual J++ (Microsoft's visual Java development tool). Note also the strong implication that a typical application development environment is likely to include a word-processing package (e.g., Microsoft Word), a project-management package (e.g., Microsoft Project), and a spreadsheet (e.g., Excel).

For a Unix environment, Rational envisions an overall architecture similar to that shown in Figure 3, but with appropriate modifications: FrameBuilder in place of Word, Testmate in place of Microsoft's Visual Test, Apex Ada and Apex C++ in place of the Microsoft programming languages, and so on. Rational does realize that both Unix and Windows are growth markets, and that many of their customers may mix Unix and Windows together in many of their projects. Thus, the objective is to ensure that the core technologies (especially in the form of Rational Rose) are reusable across various platforms.

"Roundtrip" engineering is shown in Figure 3, and while it is now a familiar concept to many software developers, it deserves some mention. Earlier generations of CASE tools tended to assume that all software development was accomplished in a "forward engineering" sense — i.e., an analysis and design model would be created first, and then code would be generated (manually or automatically) from the model. In theory, any subsequent changes or enhancements should be accomplished by changing the model, and then generating new/revised code. But in practice, it usually doesn't work this way: Many of the changes that are made during the testing, debugging, and maintenance phases of a typical project are accomplished at the code level. After all, if you were a programmer, debugging the final version of the software at midnight on the day before the deadline, it's far more likely that you would be working with code than with a model.

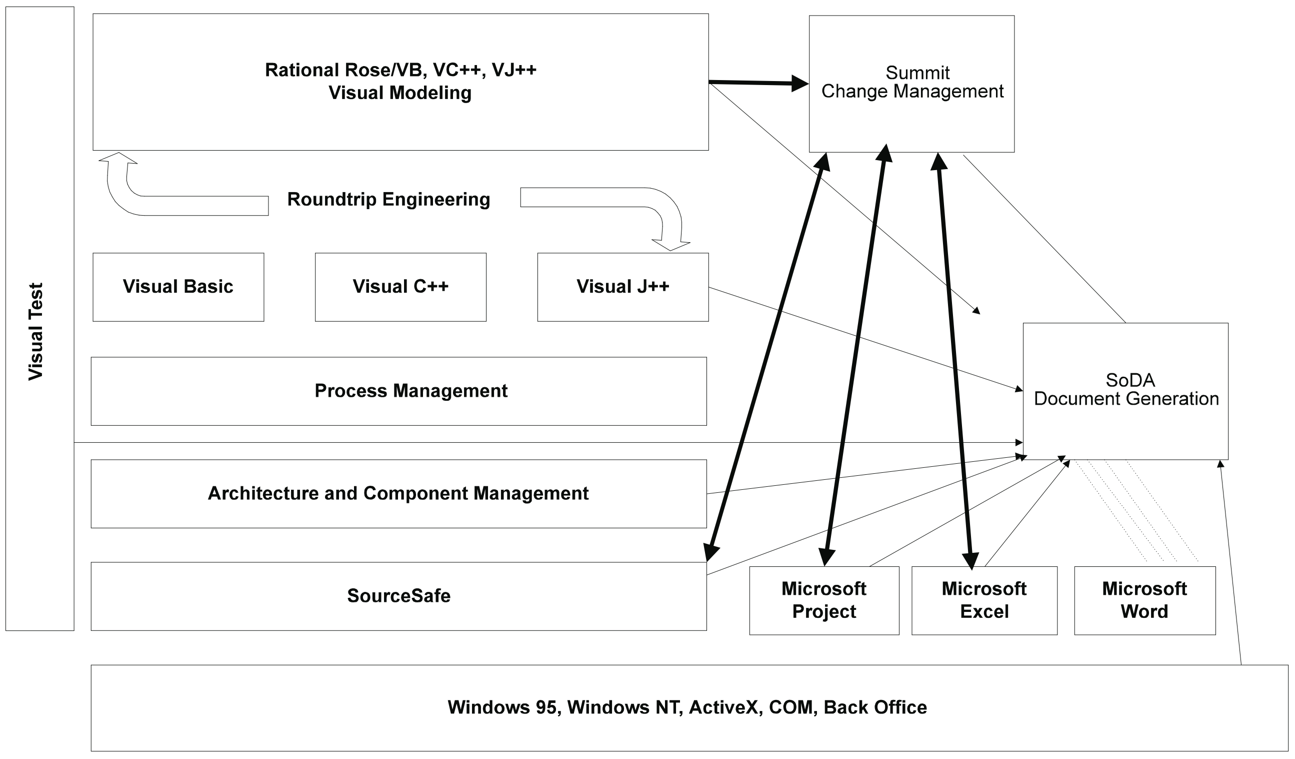


Figure 3: The Architecture of Rational's Component-Based Windows Development Tools

Thus, most of the current CASE tools support the idea of forward engineering, as well as an incremental form of reverse engineering: If changes are made to the code, those changes can be propagated back to the analysis and design models, thus ensuring that the two are kept in a synchronized state.

The "roundtrip" engineering shown in Figure 3 looks as if it involves only the interactions between the visual-modeling tool and the language-level tool in which the code is written (as well as the code generated automatically by the tool). After all, that's the only part of the life cycle that the typical programmer is concerned about synchronizing — assuming that he or she is concerned at all! But ultimately, the roundtrip concept needs to be applied from the very beginning of the life cycle (e.g., business process reengineering, or perhaps the definition of product/systems requirements) all the way to the very end of the life cycle (which might include defect tracking, software maintenance, and version control). If the programmer is making a change to the code, does it involve a change to the design — or perhaps a change to the very requirements of the system? And if an end user calls the hotline support desk to complain about a bug, should the entry made into the defect-tracking system indicate a code-level bug, or perhaps a need to change the requirements at the front end of the life cycle?

I don't believe that Rational, or any of the other major vendors, has yet managed to implement roundtrip engineering in quite such a comprehensive fashion. One of the reasons for this is that the automated tools that support the various life-cycle activities mentioned above have been developed by independent tool vendors, each of which has its own database/repository for storing information. Thus, though Rational Rose can maintain a single repository to synchronize changes to the analysis and design model and changes to the code, the changes that might be associated with testing will cause some difficulty — because Rational has only recently acquired Microsoft Visual Test, and is still in the process of completing the acquisition of SQA and its TeamTest tools.

While it's theoretically possible to imagine an integrated environment with a single repository containing information pertinent to *all* of the tools, such an approach is only likely to be practical if all the tools are developed by the same vendor at the same time; IBM's efforts to accomplish a single repository for AD/Cycle, for example, never did succeed. For Rational, and for most of the other tool vendors, the reality is a core group of tools will share a common repository; the remaining "peripheral" tools will each have their own "private" repository with a mechanism for communicating with the central core. Rational's "core" toolset is, of course, Rose; the mechanism for communicating with other tools varies from a simple, passive import-export to an active, or "hot" API that allows either Rose or the external tool to interrogate or update repository information in the other tool.

The distinction between these two approaches is crucial, and is another indication of the overall trend toward component-based construction of the tools themselves. Consider the integration of two tools, as shown in Figure 4; if tool A exports its repository information to an intermediate file, and then tool B imports the information in order to update its repository, then there will be a period of time (which may range from seconds to hours to days) when the two repositories are unsynchronized. Indeed, just because tool A exports its information, there is no guarantee that the application developer (or the project manager, or whoever is responsible for such things) will remember to import the information into tool B's repository; thus, the synchronization problem could be far worse.

But suppose that tools A and B have an API that allows direct, albeit controlled, access to their respective repositories. Both tools might be running concurrently in some environments; or a developer might be

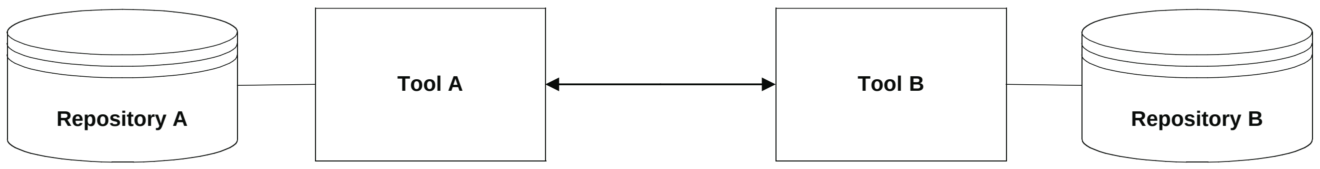


Figure 4: Synchronization of Repository Information Between Two CASE Tools

working with tool A and make a change that causes tool A to invoke tool B and send it some information to update B's repository. Thus, the discovery that a test script has failed might lead the programmer to fix a bug in his or her program; the tool environment might query the analysis and design model to see whether the change has any impact on its repository; and a change to the code *or* the analysis and design visual model could involve a change to the product requirements.

There are various ways of implementing such a tool architecture, of course; but if most of the tools are already operating within the Microsoft environment, the "obvious" approach is based on OLE technology. Thus, just as it's possible for any of the tools shown in Figure 3 to invoke Microsoft's Word, Excel, and Project products as OLE containers, it should also be possible to invoke any of the CASE tools as an OLE container.

The Microsoft Relationship

After a flurry of rumors in the late summer of 1996, Rational and Microsoft made three formal announcements together that will, in my opinion, have a significant impact on the entire application development industry, as well as Rational's future.

The first announcement, dated October 2, 1996, confirmed that Rational had acquired Microsoft Visual Test for $23 million. The announcement indicated that Microsoft had shipped some 75,000 copies of Visual Test since its release in 1972 — a tiny number compared to the number of Visual Basic licenses, but nevertheless a nice "base" for Rational to begin with. Current customers of Visual Test were told that they would continue receiving the same level of technical support for six months after the acquisition, and that paid support would continue for six months beyond that. Future upgrades and support will be available from Rational, with the first upgrade scheduled for early 1997.

Why would the two companies make such an agreement? The October announcement said that the agreement would allow

> Microsoft to focus on development environments such as the Visual Basic programming system, Visual C++ development system, and Visual J++ development tool and Rational to focus on life-cycle tools for analysis and design, testing, documentation, and change management.

Rational's interest in testing was, of course, further confirmed by the independent announcement of its own a month later that it was acquiring Massachusetts-based SQA. Meanwhile, Rational and Microsoft had a second announcement on October 2, the same day as the first announcement: The two companies agreed to a joint development and technology cross-licensing alliance for enterprise and Internet development tools. The alliance consists of technology cross-licensing, joint development projects, joint marketing programs, and Rational's acquisition of Visual Test.

Probably the most significant example of technology licensing was announced a month later, on November 11: Rational announced that it had licensed Microsoft's Visual Basic 5.0 for inclusion in future Rational products — most notably, Rose 4.0. In addition, Rational licensed the Microsoft Developer Studio, an integrated development environment that will become the basic framework in which Rational's component-based tools will operate. The plan is to fully integrate its visual-modeling tools into Developer Studio, providing seamless integration with the full range of Microsoft development tools.

Equally important, Microsoft has agreed to include technology from Rational's visual-modeling technology in future versions of its enterprise-level visual-development tools; in plain English, this means you should expect to see a subset of Rational Rose integrated into future versions of VB, Visual C++, and Visual J++. Details were not provided in the October or the November press release, nor was I given any further details when I visited Rational in December; indeed, I have the impression that even Microsoft has not decided exactly what aspects of Rose they'll integrate, or when the integrated products will appear. I don't think we'll see it in the initial release of VB 5.0, but perhaps in a mid-1997 release of the development environments.

Rational has also licensed the rights to the future Microsoft repository, and will extend Visual SourceSafe (Microsoft's version-control system), by adding facilities for change management and process automation. Again, no further details were provided in the press release or in my visit to Rational; but I have been aware of Microsoft's repository project for the past couple of years, and have recently heard rumors that it would appear as a limited-functionality product in early 1997. The repository project was initially launched as a joint effort with Texas Instruments (TI), based on the IEF repository that TI developed. It could well be the "core" repository for all of Microsoft's tools, and it wouldn't surprise me if Rational decided to replace its own repository with the Microsoft product eventually. Over the next two to three years, this could mean that the amount of "active API" interfaces between "private" tool repositories will decrease, though I believe we'll continue to see a variety of independent, third-party tools that continue to communicate with the Rational/Microsoft repository in the fashion suggested in Figure 4.

Beyond the specifics of the public announcements, what does all of this really mean? The most significant thing, in my opinion, is that Microsoft has officially endorsed the concept of visual modeling (which appears to be the "politically correct" way of saying CASE, or high-level analysis and design). As Denis Gilbert, general manager of visual languages at Microsoft, said in the public announcement, "We want to raise the bar for enterprise application development. Standards-based visual-modeling support in such mainstream products as Visual Basic, Visual C++, and Visual J++ will significantly grow the market for modeling tools." This is certainly true, and it obviously works to Rational's benefit. But what Gilbert didn't say is that the alliance with Rational also helps to legitimize Microsoft's tools within the enterprise-level application development marketplace. As I noted earlier, a common perception of Visual Basic has been, "a nice toy for small client-server projects, but it doesn't scale well, and it can't be used for 'serious' enterprise-wide projects." That kind of assessment is likely to be reevaluated in many large organizations as Rational and Microsoft begin to deliver their integrated products in 1997. And it could enhance Microsoft's efforts with VB 5.0 in the Internet/intranet marketplace.

Obviously, application development organizations will find all of this news intriguing; but the other CASE vendors and toolmakers have been equally intrigued. Privately, many of them are incredibly discouraged by Rational's ability to woo Microsoft; but publicly, most of them are trying to put a positive "spin" on the news. Indeed, if Microsoft has legitimized CASE and visual modeling, it *will* be good for all of the vendors — but Rational obviously stands to gain most of the benefit, and the other vendors are likely to be left with the crumbs. While the Rational/Microsoft alliance has obvious consequences for the other mid-size CASE vendors (the ones that would presumably also like to be billion-dollar players in the CASE marketplace), it could be even more devastating to the smaller vendors. If a developer gets a reasonable subset of Rose with the enterprise-level version of VB (which typically has a price tag of about $1,500), then why buy a separate $1,500 CASE tool? For that matter, why buy PowerBuilder, Delphi, or the various Java development environments? Of course, the vendors of these other environments will offer plausible answers to such a question, but it does put them on the defensive.

Among the more interesting comments I've already heard from several of the other CASE vendors are, "I hope Rational knows what it's doing," and "I don't think Rational has any idea of what Microsoft is going to do to them a year from now." Such comments are based partly on the popular image of Microsoft as the "Evil Empire," but there have been examples in recent years of Microsoft releasing products that effectively cripple a smaller company with which it previously had a "strategic alliance." What's to prevent the same thing from happening to Rational in 1997 or 1998?

I posed this question to Mike Devlin and Dave Bernstein when I visited Rational in December, and what impressed me was how realistic their assessment of the situation was. "Microsoft is in business to enhance Microsoft's interests, not Rational's," Devlin said candidly. "They don't wake up in the morning and ask themselves what they can do to help Rational. As long as we understand that, we'll be fine." And indeed, that's exactly what *any* small company must do when it finds itself in a relationship with a much larger organization. To use a metaphor: I spend my summers in Montana, where grizzly bears still roam the mountains and kill a few unwary tourists each year. The local folks in my little town ask a rhetorical question: “How long do you dance with a grizzly bear?” To which the answer is: “As long as the bear wants to.”

And what would cause a grizzly bear named Microsoft to want to stop dancing with Rational? Again, there was a candid assessment of the situation from the Rational executives: Their belief is that new technologies have a tendency to become “commoditized” over time, and thus gravitate down to the “foundation-level” environment on a computing platform. In plain English: Visual-modeling tools could eventually become part of the operating system. Microsoft owns the operating system “space,” and therefore, Microsoft could end up controlling the basic technology associated with visual modeling.

If this sounds too extreme, consider what has happened with several other technologies in the past few years. In addition to such things as incorporating disk compression technology into Windows 95 (which led to a lawsuit that Microsoft lost), a more interesting example is the gradual incorporation of Visual Basic into Microsoft Office and various other tools and products. Similarly, Microsoft has indicated that it plans to make Java accessible from the operating system as part of its “embrace-and-extend” strategy in the Internet marketplace. Thus, if a subset of Rational Rose is going to appear in the enterprise-level edition of Microsoft’s programming tools in 1997, it wouldn’t be surprising to see it appear in the “professional” tools (at $999 per copy) in 1998, and the “personal” tools (at $499 per copy) in 1999.

Where does this leave Rational? Quite simply: If the company sits still and assumes that it will continue making money forever from its existing Rose technology, it is likely to be in a lot of trouble. But you don’t have to be a rocket scientist to understand this; and if you’re a realistic business person (as opposed to an idealistic entrepreneur running a young startup company), you’re also likely to accept the inevitability of Microsoft’s strategy and the overall industry trends. And an interesting indication of this awareness came from Devlin: “The demo version of Rose that we give away free on our Web site,” he said, “has more functionality than the commercial version that we sold in the marketplace two years ago.” That’s a common trend, too: Technology changes so quickly that a savvy vendor has to assume that today’s product will become such a widely available commodity within two or three years that the best thing to do with it is give it away as a marketing strategy. Obviously, this means that the R&D “pipeline” has to be stocked with new products that will be available when today’s technology has lost its commercial value.

The trick, of course, will be to ensure that Rational can keep the “crown jewels” of its new, advanced technology carefully protected so that Microsoft doesn’t take advantage of it in some harmful fashion. But a dose of reality is important here, too: Microsoft is going to move into this area whether Rational (and the other CASE vendors) like it or not. Microsoft is either a partner, collaborator, or competitor ... or any combination of the three. From this perspective, it’s a good idea to remember the old adage: Keep your friends close, but keep your enemies closer.

The State of UML

I discussed UML in the March 1996 issue of ***ADS***, and by now, most developers who have any involvement in object-oriented development are aware of it. My March 1996 discussion dealt with draft version 0.8 of UML; you can now download version 0.9 from the Rational Web site at http://www.rational.com.

UML represents the integration of the methodologies that had previously been developed by the “three amigos” at Rational: Grady Booch, James Rumbaugh, and Ivar Jacobson. While it is understandably associated with Rational and its three methodology gurus, it’s important to realize that UML also represents a consortium effort. That’s because UML will be submitted to the Object Management Group (OMG) in January 1997 as a proposed standard for a visual-modeling language to represent object-oriented and component-based systems. Consequently, Rational has been working with a number of organizations to fill in the details and ensure that UML addresses the needs of a wide range of development tools, organizations, and technologies. For example, UML has been updated in 1996 to include appropriate diagramming notation to represent OLE components and the various mechanisms for interacting with OLE components.

The inclusion of OLE is particularly significant in light of the strategic alliance between Rational and

Microsoft. Quite simply, Microsoft wants to ensure that applications developed with Visual Basic 5 and ActiveX controls can be modeled with UML and implemented with the version of Rational Rose that will coexist with the Microsoft tools.

Of course, Microsoft is not the only company interested in UML, nor is it the only vendor that wants to ensure that its interests are represented. Indeed, Rational has been put in the interesting position of serving as an intermediary where otherwise bitter competitors can lobby for compromises to ensure that UML does indeed address everyone's needs.

As of mid-December 1996, the following organizations were working with Rational to fine-tune UML; those listed as a "submitter" will be part of the joint group that collectively submits the final draft of UML to the OMG:

- CDIF
- Digital Equipment Corporation
- Ericsson
- Hewlett Packard (submitter)
- i-Logix
- IntelliCorp (submitter)
- IBM
- ICON Computing (submitter)
- MCI Systemhouse (submitter)
- Microsoft (submitter)
- James Odell
- Oracle (submitter)
- Rational Software (submitter)
- Texas Instruments (submitter)
- Unisys (submitter)

There are several advanced notational features in UML that I didn't cover in my March 1996 discussion. Two of them are worthy of mention now, partly because of the Microsoft/Rational alliance, and partly because of the growing influence of Internet- and intranet-based applications:

- *Stereotypes* — a "metaclassification" of an element within UML, i.e., something used to define a *type* of element. For example, UML contains predefined class stereotypes such as Event, Exception, Interface, MetaClass, and Utility; predefined task stereotypes such as Process and Thread have also been predefined. The advantage of this concept is that a developer can refer to a type of an element in a familiar way, by saying, "The XYZ class is a Utility class." In addition, stereotypes make UML extensible; any user of UML can define additional stereotypes.
- *Packages* — a group of inherently related, or cohesive entities. During the analysis and design activities, the "class model" documented in UML can be organized into packages that represent various areas of concern such as the user interface, database management, and so on. And the implemented code can be packaged into subsystems that represent deployed software components. This, of course, is convenient for many of the component-based applications that one is likely to see in a Microsoft environment. The notation for a package — an icon that looks like a tabbed folder — is also conveniently familiar to anyone working in a Microsoft environment.
- *Deployment diagrams* — a mechanism for showing the organization of hardware, and the binding of software to the physical devices. Deployment diagrams show hardware devices and their physical interfaces; the type of hardware device is given by its stereotype, such as Disk, Processor, etc.

It should be noted that UML will not be the only methodology submitted to the OMG. The most significant and well-developed alternative is the OPEN methodology developed by a group of researchers, consultants, and vendors coordinated by Professor Brian Henderson-Sellers. The final version (1.0) of the OPEN Modeling Language (OML), as well as a comparison between OPEN and UML, is available at http://www.csse.swin.edu.au/cotar/OPEN/OPEN.html.

OPEN is certainly a worthy alternative to UML, and it does have a lot of interest and support from individuals and organizations all over the world. However, I think the heavy-duty industry support behind UML virtually guarantees its dominance; Indeed, it has already become a de facto standard throughout the industry, and I expect that the OMG will endorse it sometime in 1997.

Indeed, it's interesting that several of the other OO CASE vendors in the field (including Platinum, Popkin, and IDE/Aonix) announced support for UML when it was still at draft 0.8; by contrast, up to the time this report was written in mid-December, Rational's own visual-modeling tool did *not* support UML. But that's all about to change with the forthcoming release of Rose 4.0.

Rose 4.0

Rose, as I've already discussed, is Rational's flagship product for object-oriented modeling. I saw a demonstration of Rose 4.0 during my visit to Rational; the Windows version is scheduled for release by the end of December, and the Unix version should be available in January 1997.

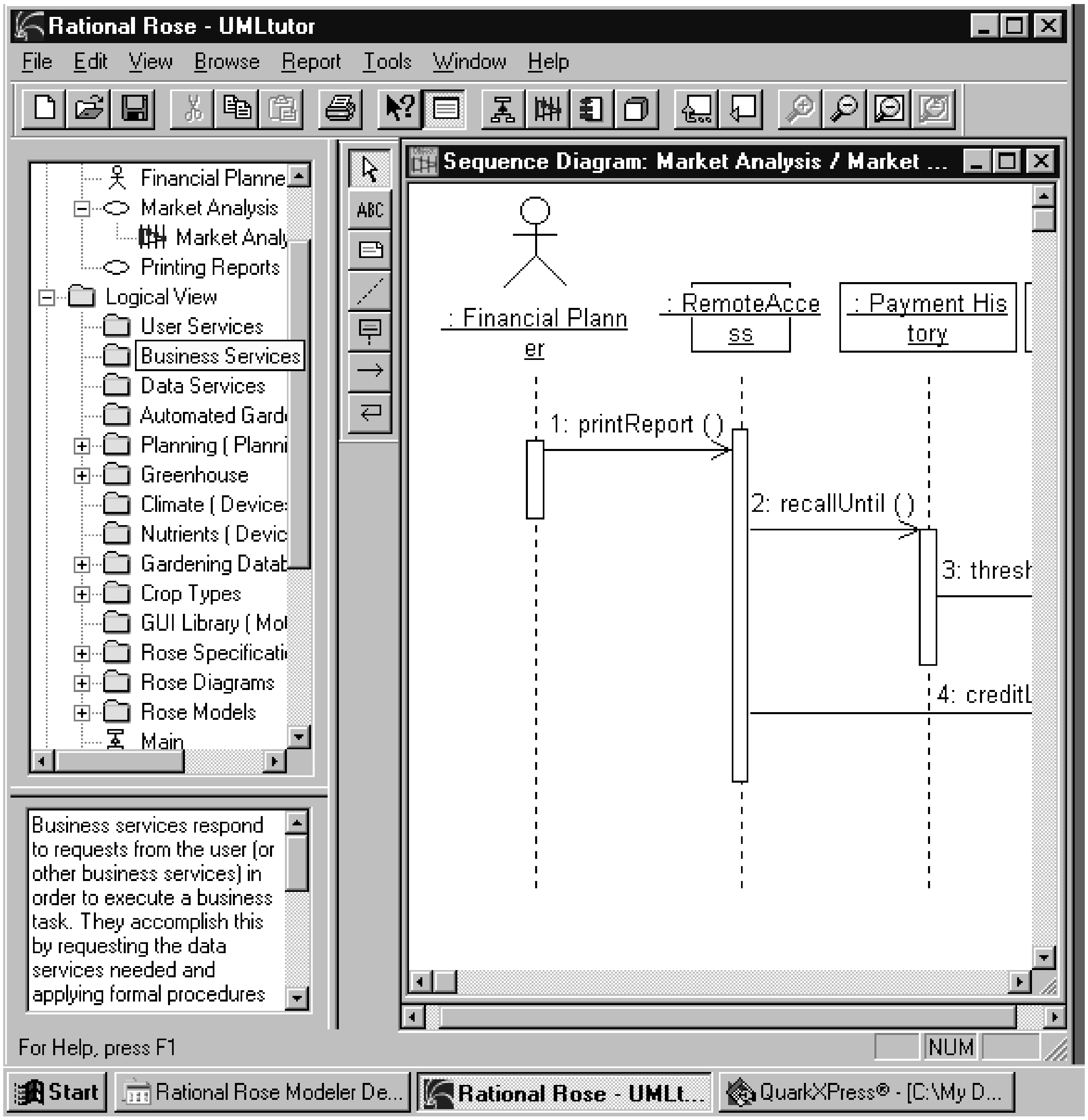
One of the most significant changes from the previous version, of course, is the incorporation of UML. This includes not only the class diagrams and all of the other advanced features discussed earlier, but also the fundamentally important concept of *use cases* for modeling the interactions between an "actor" (a human user or an external system) and the system being developed. Several other CASE tools have already begun providing use-case support, and while Rational decided not to "patch in" a quick-and-dirty form of UML-compliant use-case support in its earlier versions of Rose, it's definitely there in Rose 4.0.

Rational has also worked hard to incorporate the Windows 95 "look and feel" in Rose 4.0; an overall feeling of the user interface is suggested by the layout shown in Figure 5. As with most current Windows 95 applications, the various "panes" within the overall window are individually resizable; and the organization of different diagrams within a Rose model are presented in the familiar Windows 95 outline style. An interesting part of the Rational/Microsoft strategic alliance was Microsoft's suggestion that Rational take advantage of the extensive usability lab that Microsoft uses for all of its own products. This led to a number of subtle, but nevertheless important, changes in the user interface of Rose 4.0 — changes that might not be important or even necessary for the "power-user" marketplace to which Rational has traditionally aimed its products, but which will be extremely important if it is to piggyback onto the Microsoft development tools and appeal to a larger community of novice programmers and casual users. A good example of this is the principle that an end user should never have to retype any information that has already been typed into the tool before; wherever possible, the user is presented with a browsable list of choices and defaults, in order to speed up interactions with the tool.

One of the objectives of this focus on the user interface in Rose 4.0 was to make it highly compatible with the user interface for Microsoft's implementation of VB, Visual C++, and Visual J++. Indeed, the ultimate goal is to make all of these user interfaces so similar that the user/developer no longer knows or cares which environment he or she is working in at any particular moment.

In any case, I spent a couple of hours manipulating the demo version of Rose 4.0 (which is downloadable from the Rational Web site) and found it quite comfortable to use. Like any sophisticated PC product, you can't *really* tell if the user interface is acceptable until you've spent a couple of months using it on a nontrivial project. It's only been a month since I abandoned my Macintosh forever and made a complete transition to the Windows 95 environment, and I'm still discovering that a variety of tools and programs that looked fine during the first hour of use have turned out to be completely unwieldy for "serious" use. But with that caveat, my initial impression is that Rose 4.0 has been very well-engineered for human use.

An interesting aspect of Rose 4.0 is its reengineering capability. As mentioned earlier, this capability is particularly useful for roundtrip engineering; but it can also be used for more conventional reverse engineering — i.e., when the developer has nothing but code or an existing database, and wants to produce an object model from it. A particularly interesting form of reverse engineering involves Java: Applets are created by a developer in a source language that looks like C++, but are then compiled into machine-neutral "bytecodes" for distribution over the Internet.



Business services respond to requests from the user (or other business services) in order to execute a business task. They accomplish this by requesting the data services needed and applying formal procedures

Figure 5: A Typical Rational Rose Visual Model

If a user downloads a Java applet to his client machine, all he can "see" is the bytecodes; similarly, if a developer acquires a Java applet from a third-party vendor, he might only have access to the bytecodes. But with Rose 4.0, the bytecodes can be reverse-engineered into class models; this won't recover the original source code (though I've heard rumors of other reverse-engineering tools that *will* reproduce the source code), but it does help the developer understand, with formal documentation, the structure and interfaces of the Java applet.

Another important aspect of Rose 4.0 is a scripting language that's included as part of the Rose

Developers Kit. Just as Visual Basic for Applications can be used to "drive" Microsoft Word and Excel, and just as Java Script can be used to embed "macros" within a Web page without the need for compiling them into Java bytecodes, the Rose scripting language can be used to "drive" the behavior of the Rose tool. The technology is further extended with a number of "wizards," which are written in Visual Basic script, and which help in the creation of Rose scripts.

The scripting language is compatible with Visual Basic, and it effectively "exposes" the object model contained within the Rose repository. The same technology is used by Rational for its code generation, and thus could be used by third-party vendors to generate code for languages that Rational has decided not to support. I doubt very much that the average Rose user will find a need to use the scripting language, but it will be a useful tool indeed for ISVs and VARs — e.g., vendors like Iona, Next, and Forté — who want to interface Rose with their own tools.

The official Windows 95 release of Rose 4.0 will provide code-generation support for Visual Basic 5.0, C++, and Java. It won't support VB 4.0, which took me a little by surprise. However, I suppose one could make the argument that VB 4.0 is part of the "old world," along with VB 3.0 and Windows 3.1. In any case, if you *do* want to use Rational Rose together with an earlier version of VB, then you'll need to use one of the older versions of Rose.

After the initial release, a second release of Rose 4.0 — which is tentatively scheduled for the second quarter of 1997 — will support PowerBuilder, Forté, IDL/DDL (and thus CORBA 2.0), and Smalltalk. Rational is working with one of its partners to produce a version for Centura (formerly Gupta, the vendor of SQL/Windows), and is talking to third-party sources about a Delphi version.

The single-copy price of Rose 4.0 is $2,400 for any of the languages mentioned. Each programming language effectively represents a different version of the Rose product; Rational charges an incremental upgrade price for additional language coverage. For the organizations that don't need code generation (e.g., those who are using Rose for business process reengineering or other forms of process modeling), the "basic" version of Rose 4.0 will be $1,695. While these are reasonable prices for an enterprise-oriented tool, I was curious about the possibility of a "personal" version of the product; the folks at Rational reminded me that the single-user "Rose Solo" product for Rose version 3.0 has been available for $495, and told me that they have similar plans for Rose 4.0.

This may prove to be irrelevant if the alliance with Microsoft proceeds as planned. While Microsoft's initial plan may be to include a subset of Rose with the enterprise-level version of its programming tools, there's a good chance that the same subset (or perhaps a slightly more watered-down version) will be available with the "personal" editions of VB, Visual C++, and Visual J++ within another couple of years.

Rational As a Company

While products and technology are the most important things I focus on in these ***ADS*** reports, some other aspects of Rational are worth mentioning, too. As noted earlier, the company has been in existence — with its current management team — since 1981; thus, it's a mature company in almost any industry, and *very* mature in the computer industry.

The company has approximately 500 employees, with 18 sales offices in North America and 9 international sales offices; in addition, it has international partners in Japan, Singapore, Israel, and China.

Corporate headquarters are in Santa Clara, California, but it's interesting that software development is distributed throughout North America and overseas. For example, much of the Rose development work takes place in Milwaukee, Wisconsin; work associated with the company's Objectory products is carried out in Kista, Sweden; and there's even development work going on in Bangalore, India.

I was curious about the rationale for distributed software development — especially as Rational's new strategic partner has just the opposite strategy for its software development. The answer I received from Rational's management team was interesting:

- Some of the development work takes place in surprising parts of the world (Milwaukee and Kista) because of acquisitions. It's unlikely that Rational would have started up a new development center in Kista, Sweden — but that's where Ivar Jacobson's company happened to be located when Rational acquired it in 1995.

- The existence of a global, distributed development environment mimics what many of Rational's large customers are doing. As a result, Rational can better understand what kind of tools and processes are needed to support this kind of effort; Microsoft refers to this as "eating your own dog food." Here is a small example, which has nothing to do with visual modeling: When I visited Rational in California, I had a conversation with the Rose product manager, Jon Hopkins, in Milwaukee via videoconference. Of course, videoconferencing is a familiar technology; but it's not readily available or widely used in most of the application development organizations that I visit. By contrast, the Rational employees regarded it as something no more unusual than e-mail or a fax machine.
- A distributed development environment is becoming more and more necessary in order to recruit talented software developers in today's marketplace. It's hard to hire hotshot Java programmers in the Santa Clara area where the cost of living is high and the competition from other Silicon Valley companies is fierce; it's not quite so hard in Philadelphia, Pennsylvania, or Huntsville, Alabama, or some of the other places where Rational now operates. Indeed, the relatively new office in Bangalore may be one of its smartest moves.

One of the interesting consequences of the Microsoft strategic alliance is a dramatic increase in the number of end-user licenses and customers that Rational can now call its own. From March 31, 1995 to March 31, 1996 the number of end-user licenses doubled, from 15,000 to 30,000. These are impressive numbers, but not substantially different from the kind of numbers one hears from Cayenne and Aonix and even the smaller vendors like Popkin. However, between March 31, 1996 and November 30, 1996 the number of end-user licenses increased to 115,000. This includes the customers associated with the Visual Test product, which is estimated at 75,000; similarly, Rational's increase in "cumulative customers" increased from 1,500 to 10,000 during the same period, largely because of the acquisition of Visual Test. Thus, even though the number of "traditional" Rose licenses and customers has been increasing at its usual impressive base, the number of customers who now look to Rational for product support has increased by approximately a factor of five.

And keep in mind that this is only the beginning. Microsoft currently has some two million licensed Visual Basic customers, and a reasonably large number of Visual C++ licenses. As it ramps up its marketing efforts for VB5 and Visual J++, it will be carrying Rational along with it. Companies and developers who had never heard of Rational *or* visual modeling are likely to hear a lot about the company and its products in 1997.

Summary

Two years ago, my assessment of Rational would have been, "Great company with a great CASE tool — but too heavily immersed in the Unix marketplace, and stuck with an albatross called Ada." I'm not sure when the epiphany struck Rational, but it seems to me that at some point in the last two years, the key decision makers woke up one morning and said in unison, "Microsoft wins!" Of course, this doesn't mean that Rational's Unix tools have been abandoned, or that the company has stopped marketing its Apex Ada tools. To the extent that the Apex technology can be retrofitted into the exploding Windows component-based marketplace, that's fine; and to the extent that some Department of Defense and aerospace clients still want to buy Ada technology, that's fine. It's right to continue supporting one's clients, much the way IBM still supports its PL/I customers. But Rational has clearly shifted its energies toward the Windows environment and has accomplished what no other upper-CASE tool vendor has managed to accomplish: endorsement from Microsoft, together with ownership of the "intellectual space" represented by UML.

This doesn't guarantee that Rational will succeed; after all, when you're dancing with an 800-pound grizzly bear, anything can happen. But just as the battle for the desktop computing environment is over, I think the battle for dominance of the object-oriented visual-modeling marketplace is over: Rational wins.