Contents

What is Rapid Application Development (RAD)?

James Martin, in his book first coining the term, wrote, "Rapid Application Development (RAD) is a development lifecycle designed to give much faster development and higher-quality results than those achieved with the traditional lifecycle. It is designed to take the maximum advantage of powerful development software that has evolved recently."

Professor Clifford Kettemborough of Whitehead College, University of Redlands, defines Rapid Application Development as "an approach to building computer systems which combines Computer-Assisted Software Engineering (CASE) tools and techniques, user-driven prototyping, and stringent project delivery time limits into a potent, tested, reliable formula for top-notch quality and productivity. RAD drastically raises the quality of finished systems while reducing the time it takes to build them."

Online Knowledge defines Rapid Application Development as "a methodology that enables organizations to develop strategically important systems faster while reducing development costs and maintaining quality. This is achieved by using a series of proven application development techniques, within a well-defined methodology."

In short, Rapid Application Development is exactly that. It is a process through which the development cycle of an application is expedited. Rapid Application Development thus enables quality products to be developed faster, saving valuable resources. The magnitude of such savings is truly RAD!

Why Do You Need to be RAD?

The Gartner Group writes, "Many of the business processes devised after World War II…have remained essentially the same. Corporations are now finding that work organized stepwise incurs unavoidable delays and errors as paper is handed off from person to person and unit to unit…IT is the single most powerful tool for breaking traditional assumptions and rules about business, and it is the tool that makes new ways of operation possible." The most revolutionary and successful change in IT business practices today is Rapid Application Development.

RAD takes advantage of automated tools and techniques to restructure the process of building information systems. This new process, extrapolated to the entire IS organization, results in a profound transformation of information systems development. RAD replaces hand-design and coding processes, which are dependent upon the skills of isolated individuals, with automated design and coding, which is an inherently more stable process. RAD may thus give an IS organization its first real basis for continuous improvement. In addition to being more stable, Rapid Application Development is a more capable process, as it is much faster and less error prone than hand coding.

Most organizations are faced with a large backlog of new systems to be developed. Over 65% of the typical Information System's budget is spent on the maintenance of existing systems. These systems have little documentation and were developed with programming languages and database systems that are difficult and time consuming to change. These organizations are thus faced with upgrading their aging systems or building new applications. Traditional development lifecycles, however, are too slow and rigid to meet the business demands of today's economy. A new methodology must be implemented, one that allows organizations to build software applications faster, better, and cheaper. RAD enables such development.

The availability of powerful CASE software makes it possible for developers to create systems much faster than ever before. These new integrated CASE toolsets are breaking out of the bubble of traditional software development thought. They take application development beyond generation coding, just as generation, many years ago, surpassed textual coding. These tools enable a developer to drag-and-drop previously generated code, saving that developer the time and effort of individually hand-coding the text of the application. CASE tools also enable a developer to implement Rapid Application Development irrespective of their programming language or platform. CASEMaker's Totem 5.0 brings Rapid Application Development to those coding in COBOL, a traditional, yet far from defunct, programming language.

Stanley Marcus of Neiman Marcus said, "There are only two things of importance. One is the customer, and the other is the product. If you take care of customers, they come back. If you take care of the product, it doesn't come back. It's just that simple. And it's just that difficult." Rapid Application Development, in addition to providing a more quality product in less time, also ensures greater customer satisfaction. By reducing the elapsed time between User Design and Cutover, RAD increases the likelihood that the system will be satisfactory to the users, whose demands are met much quicker than ever before. The RAD process also directly integrates the end-users in the development of the application. Iterative prototyping mandates that the development teams concentrate on delivering a series of fully functional prototypes to designated user experts. Each prototype is tested by those users and returned to the development team for reworking, at which point the cycle repeats. The series of prototypes thus evolves into the final product, giving the users the opportunity to fine-tune the requirements and review the resulting software implementation.

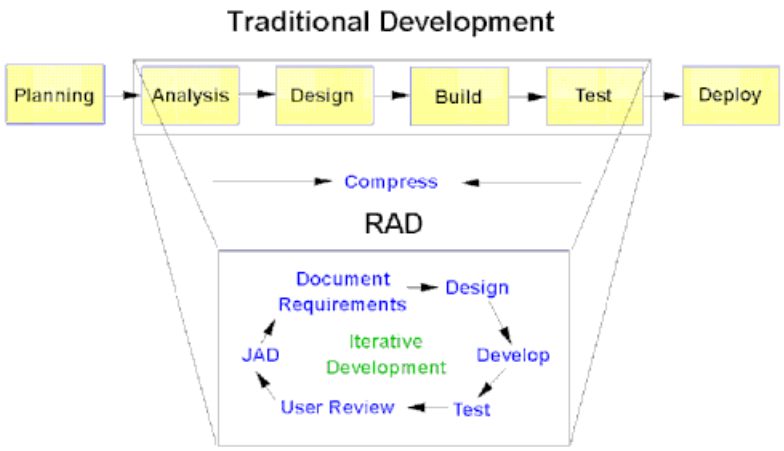
The History of RAD

Traditional lifecycles devised in the 1970s, and still widely used today, are based upon a structured step-by-step approach to developing systems. This rigid sequence of steps forces a user to "sign-off" after the completion of each specification before development can proceed to the next step. The requirements and design are then frozen and the system is coded, tested, and implemented. With such conventional methods, there is a long delay before the customer gets to see any results and the development process can take so long that the customer's business could fundamentally change before the system is even ready for use.

In response to these rigid, cascading, one-way steps of Stagewise or Waterfall Models of development, Barry Boehm, Chief SW Engineer at TRW, introduced his Spiral Model. The Spiral Model is a risk-driven, as opposed to code-driven, approach that uses process modeling rather than methodology phases. Through his model, Boehm first implemented software prototyping as a way of reducing risk. The development process of the Spiral Model separates the product into critical parts or levels while performing risk analyses, prototyping, and the same steps at each of these levels. Similarly, Tom Gilb's Evolutionary Life Cycle is based on an evolutionary prototyping rationale where the prototype is grown and refined into the final product.

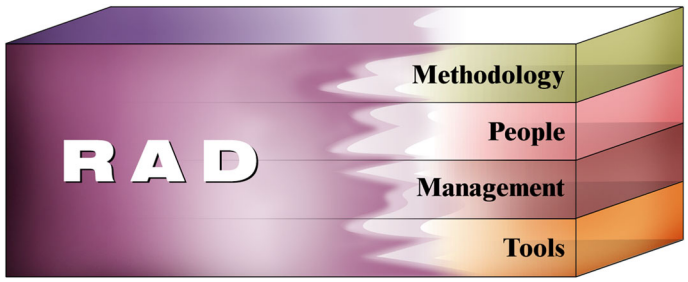
The work of Boehm and Gilb paved the way for the formulation of the methodology called Rapid Iterative Production Prototyping (RIPP) at DuPont in the mid-to-late 1980s. James Martin then extended the work done at DuPont and elsewhere into a larger, more formalized process, which has become known as Rapid Application Development (RAD). RAD compresses the step-by-step development of conventional methods into an iterative process. The RAD approach thus includes developing and refining the data models, process models, and prototype in parallel using an iterative process. User requirements are refined, a solution is designed, the

solution is prototyped, the prototype is reviewed, user input is provided, and the process begins again.



Essential Aspects of RAD

Rapid Application Development has four essential aspects: ***methodology***, ***people***, ***management***, and ***tools***. If any one of these ingredients is inadequate, development will not be high speed. Development lifecycles, which weave these ingredients together as effectively as possible, are of the utmost importance.



METHODOLOGY

The challenges facing software development organizations can be summarized as more, better, and faster. The RAD development path attacks these challenges head-on by providing a means for developing systems faster, while reducing cost and increasing quality. Fundamentals of the RAD methodology thus include:

- Combining the best available techniques and specifying the sequence of tasks that will make those techniques most effective.
- Using evolutionary prototypes that are eventually transformed into the final product.
- Using workshops, instead of interviews, to gather requirements and review design.
- Selecting a set of CASE tools to support modeling, prototyping, and code re-usability, as well as automating many of the combinations of techniques.

- Implementing timeboxed development that allows development teams to quickly build the core of the system and implement refinements in subsequent releases.
- Providing guidelines for success and describing pitfalls to avoid.

Active user involvement throughout the RAD lifecycle ensures that business requirements and user expectations are clearly understood. RAD takes advantage of powerful application development tools to develop high quality applications rapidly. Prototyping is used to help users visualize and request changes to the system as it is being built, allowing applications to evolve iteratively. RAD techniques are also very successful when faced with unstable business requirements or when developing non-traditional systems.

The structure of the RAD lifecycle is thus designed to ensure that developers build the systems that the users really need. This lifecycle, through the following four stages, includes all of the activities and tasks required to scope and define business requirements and design, develop, and implement the application system that supports those requirements.

- Requirements Planning

  Also known as the Concept Definition Stage, this stage defines the business functions and data subject areas that the system will support and determines the system's scope.

- User Design

  Also known as the Functional Design Stage, this stage uses workshops to model the system's data and processes and to build a working prototype of critical system components.

- Construction

  Also known as the Development Stage, this stage completes the construction of the physical application system, builds the conversion system, and develops user aids and implementation work plans.

- Implementation

  Also known as the Deployment Stage, this stage includes final user testing and training, data conversion, and the implementation of the application system.

PEOPLE

The success of Rapid Application Development is contingent upon the involvement of people with the right skills and talents. Excellent tools are essential to fast application development, but they do not, by themselves, guarantee success. Fast development relies equally heavily on the people involved. These people must thus be carefully selected, highly trained, and highly motivated. They must be able to use the tools and work together in close-knit teams. Rapid development usually allows each person involved to play several different roles, so a RAD project mandates a great degree of cooperative effort among a relatively small group of people.

Each stage of a rapid development project includes activities that need to move fast. As a result, it is critical that management initiates the project quickly, cutting through any political delays or red tape. At the Requirements Planning and User Design stages, key end users must be available to participate in workshops. While the system is being constructed, the Construction Team, which uses the CASE toolset to accomplish detailed design and code generation, must be poised to move quickly. At the end of the development cycle, the Cutover Team, which handles training and cutover, must also be ready to move quickly.

The key players in a Rapid Application Development project include:

- Sponsor

  A high-level user executive who funds the system and is dedicated to both the value of the new system and to achieving results quickly.

- User Coordinator

  A user appointed by the Sponsor to oversee the project from the user perspective.

- Requirements Planning Team

  A team of high-level users who participate in the Joint Requirements Planning workshop.

- User Design Team

  A team of users who participate in the design workshop. This team should be comprised of both high-level users from the Planning Team and lower-level users with a more detailed knowledge of the system.

- User Review Board

A team of users who review the system after construction and decide whether modifications are necessary before cutover.

- Training Manager

  The person responsible for training users to work with the new system.

- Project Manager

  The person who oversees the development effort.

- Construction (SWAT) Team

  The SWAT (Skilled Workers with Advanced Tools) Team is a small team of two to six developers who are highly trained to work together at high speed. To achieve the fastest possible development, the team members must be highly skilled in the RAD methodology and in using the chosen CASE toolset.

- Workshop Leader

  The specialist who organizes and conducts the workshops for Joint Requirements Planning and Joint Application Design.

MANAGEMENT

Achieving high-speed development is a complex process. Systems will not be developed and deployed rapidly if bureaucracy and political obstacles stand in the way or if users are not appropriately involved. Management must be totally committed to RAD in order to manage the change in culture. They must be prepared to motivate both users and IT staff, select and manage SWAT teams, and demonstrate through the use of performance measurements that RAD does mean speed, quality, and productivity. Good management and dedication to the ideals of Rapid Application Development are thus essential to faster system building.

To successfully introduce rapid development, management must pay careful attention to human motivation. Managers should target those professionals whom they deem as 'Early Adapters.' 'Early Adapters' are those people who see the value of a new methodology and lead the way in making it practical to use. These employees are enthusiastic about the new methodology and they want to make it work well in their environment. Similarly, managers must be aware of the type of motivation that is most effective for each individual employee, whether it be money, pride, prestige, excitement, or some combination thereof.

Because Rapid Application Development is such a sweeping change from the conventional development methods, the best way for a manager to introduce new rapid development techniques is to start small. Original Construction Teams of two to four people should be established and their members should be thoroughly trained in the use of the tools and techniques. As these teams gain experience, they will be able to fine-tune the development lifecycle to improve its effectiveness in their environment. Underlying all of this progress, however, managers must remember the importance of comprehensive and quality training in the use of tools. Good training with tools that are exciting to use can have a profound impact on the attitude of IT professionals, as well as ensure the uninterrupted success of the rapid development project.

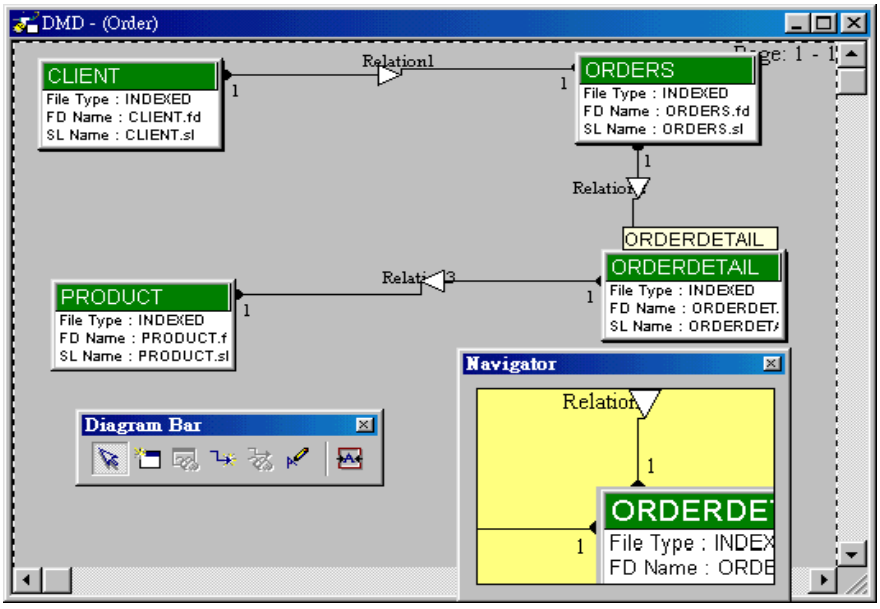
TOOLS

The RAD methodology uses both computerized tools and human techniques to achieve the goals of high-speed and high quality. The above has been primarily concerned with the goals of Rapid Application Development and the role of organizations and the people within those organizations in the achievement of those goals. The success of any Rapid Application Development project is primarily dependent, however, upon the tools used.

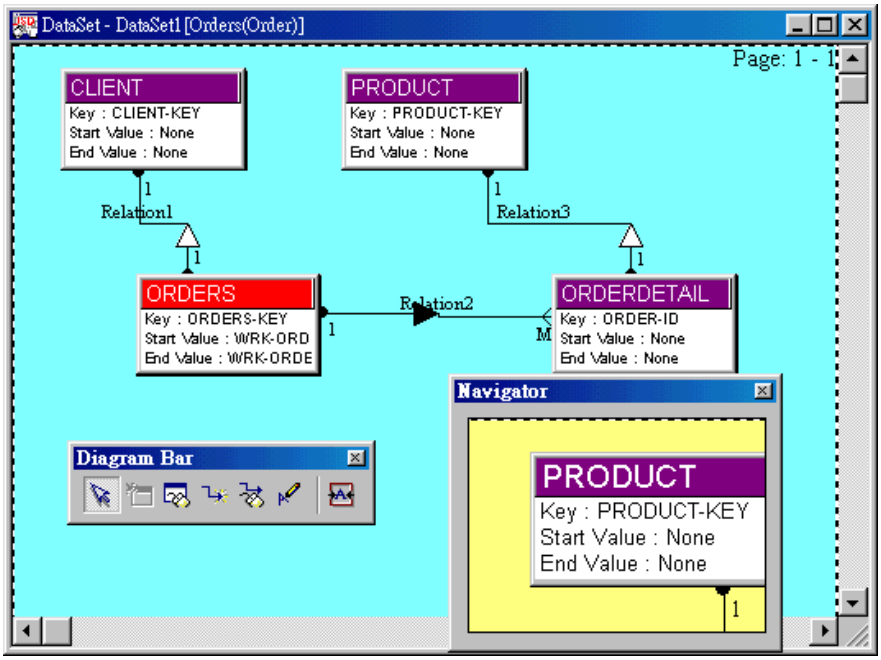
The Wall Street Journal lamented that software is one of the two principle obstacles to economic progress. A former U.S. Pentagon Chief commented, "If anything kills us before the Russians, it will be our software." Power tools are required to overcome the problems involved with software development. These power tools can change the methods of construction by using design-automation techniques, code generation, and computer aided planning and analysis. The power tools utilized in Rapid Application Development are Computer-Aided Systems Engineering (CASE) tools. Such CASE tools have come a long way over the past twenty years, enabling us to overcome many of our software development problems through process innovations such as Rapid Application Development. Powerfully integrated CASE software is now available, as well as software that goes beyond traditional CASE limitations. CASEMaker's Totem 5.0 is one such product.

A fundamental principle of RAD tools is that diagrams are employed whenever possible as an aid to clear thinking. Diagrams are used to represent planning information, overview of systems, data models, process models, detailed designs, and program structures. In addition, RAD tools must generate executable code. In Totem 5.0, the Data Modeling Diagram represents the FD files and their relations at the project level so you can more completely understand your project's data and real file contents. The Data Set Diagrams, similarly, represent the FD files and their relations at the program level so you can redefine file relations and Totem will automatically convert those relations into code.

Totem's Data Modeling Diagram (DMD) graphically illustrates your file relations in the format of an Entity Relationship Diagram (ERD). ERDs define each entity, or file, within a rectangular box and connect those boxes with arrows that represent the relationship between those entities. This graphically organized style is certainly an aid to clear thinking. Not only does the DMD provide a simple and comprehensive outline of your files and their relations, but it also allows you, the developer, to modify an entity's definition or to define a relationship between entities within the diagram. Without requiring you to hand write any code, Totem will then automatically generate the code that supports those newly modified definitions or defined relations.

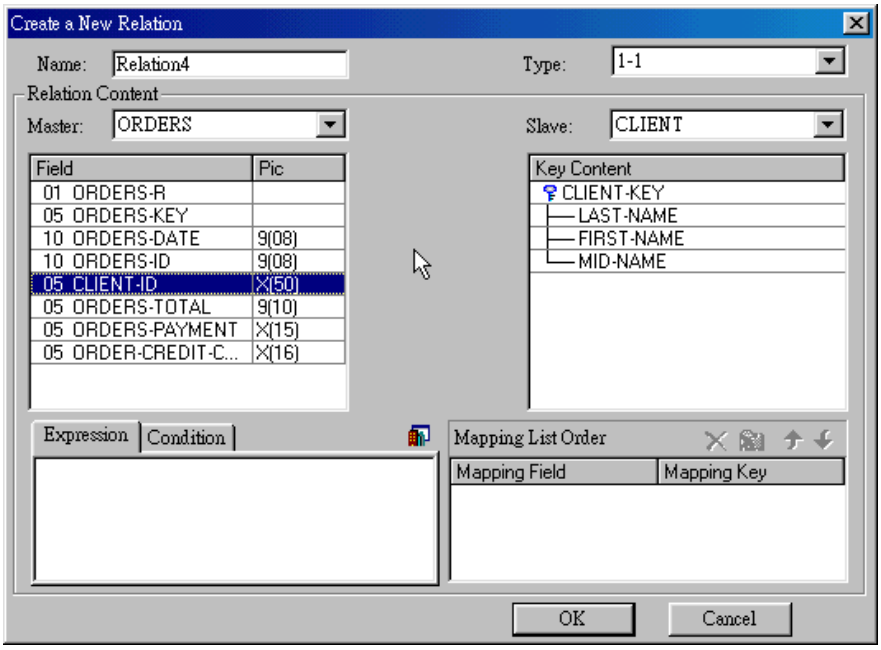


Similarly, Totem's Data Set Diagrams (DSD) also represent your FD files and their relations in a graphical Entity Relationship Diagram (ERD). A DSD, however, represents files and their relationships at the program level. While the DMD is used to represent all of your files, a DSD defines a specific access to a subgroup of those files. A DSD can inherit file relations that have been defined in a DMD, but the DSD allows you to redefine those relations as needed by your application. A DSD thus allows you to organize your data in a variety of ways. Once you specify relations between master and slave files, Totem will automatically convert these new relations into code; again, saving you the time it would otherwise take to hand code these new relations.



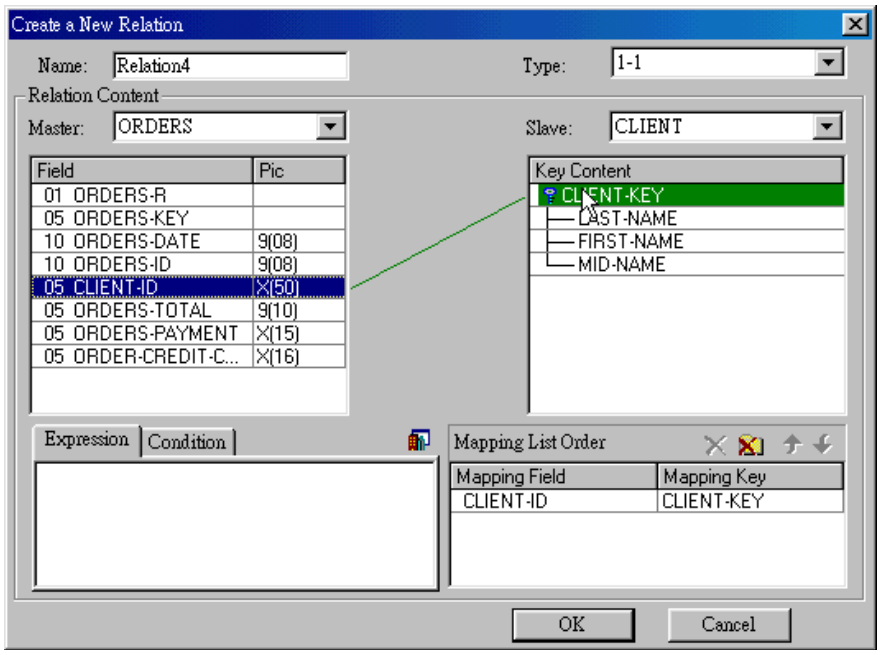
This is what establishes Totem as RAD. These diagrams, which enable a user to more clearly view what is being constructed, are automatically translated into code. The developer no longer has to generate that code by hand. Throughout Totem, the Application Wizard is responsible for converting these file relations into code. It rapidly generates code while guiding the user through the step-by-step process of developing Graphical User Interface (GUI) COBOL applications.

Using Totem, creating a new relation between two files can be as easy as dragging and dropping fields from the master file to the slave file in the Create New Relation dialog box.



STEP 1) Simply select the data item you want in the field column.

STEP 2) Click-and-drag the data item from the Field column to a slave key in the Key Content column.



STEP 3) Drop the data item and a green line appears joining the two data items. This line represents the relation that you have just defined simply by dragging a data item over to a new field column.

The tight integration of analysis and design tools with a code generator allows much higher productivity than would be possible if the tools were not coupled. In this manner, the user may view the screen as it would appear and not simply code the structure behind it. The coding process itself is also simply made easier by the ability to point-and-click or drag-and-drop many new features, without writing the actual code text for each function.

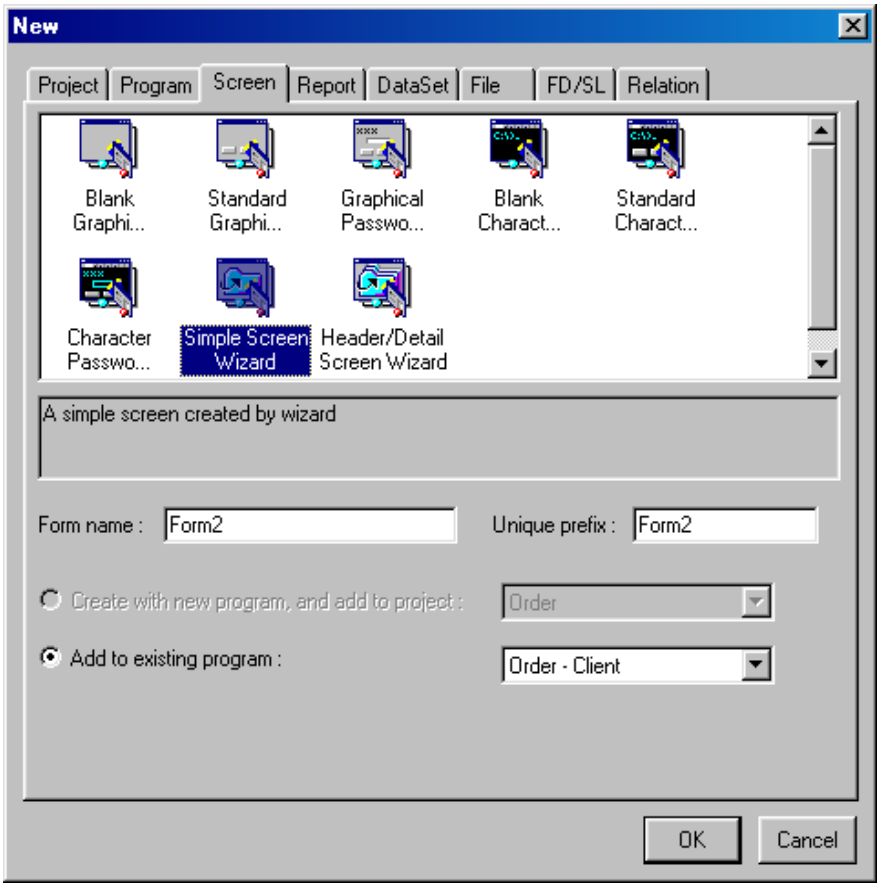
The heart of much of this efficient coding is the repository, which stores the meanings of diagrams and their explanatory text. The repository, as is the case with Totem's repository, steadily accumulates information related to the planning, analysis, design, construction, and maintenance of systems. Such data storage is critical to rapid development lifecycles. Information is stored, and can thus be extracted, at every stage in the lifecycle. The repository also stores reusable templates, structures, models, and designs, which are essential to rapid development.

Four kinds of power tools help make software development faster, cheaper, and of higher quality:

- Non–procedural languages allow developers to program by expressing the result that they want, rather than by detailing steps to achieve it.
- Prototyping tools permit iterative development, through which an application prototype is successively tested and refined.
- RAD tools allow plans, models, and designs to be expressed graphically.
- Code generators generate source code from high level constructs.

Integrated toolsets, which link these four capabilities in one facility, are the foundation of truly successful Rapid Application Development. Totem is such an integrated RAD tool. Using Totem's wizards, you may construct projects and programs without any hand coding.

For example, use the Screen Wizard to create a screen for the existing program.



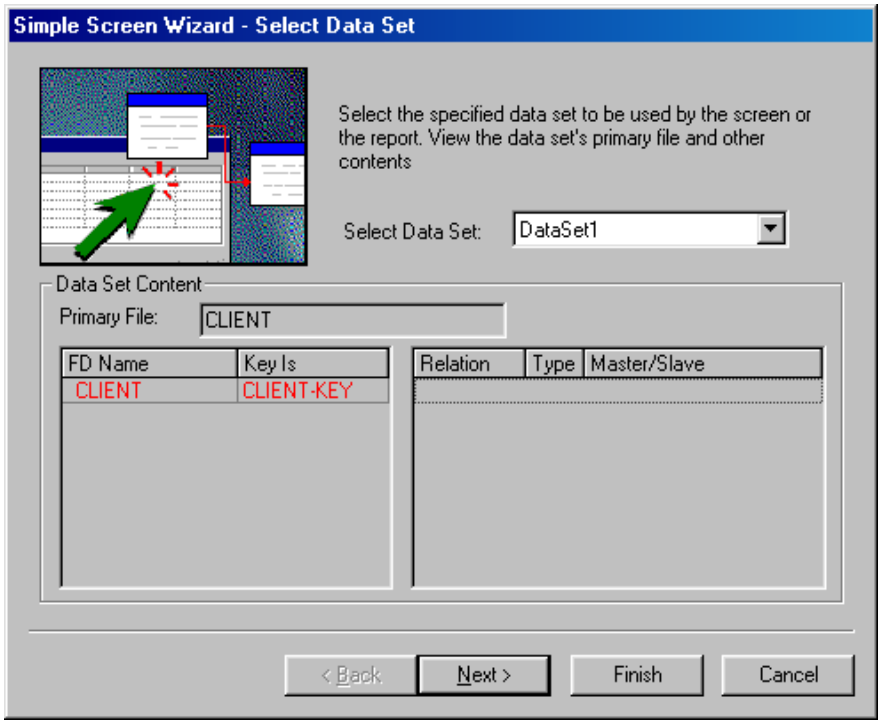
STEP 1) Choose File > New. The New dialog box appears.

STEP 2) Click the Screen tab. The Screen page is displayed.

STEP 3) Select the Simple Screen Wizard icon.

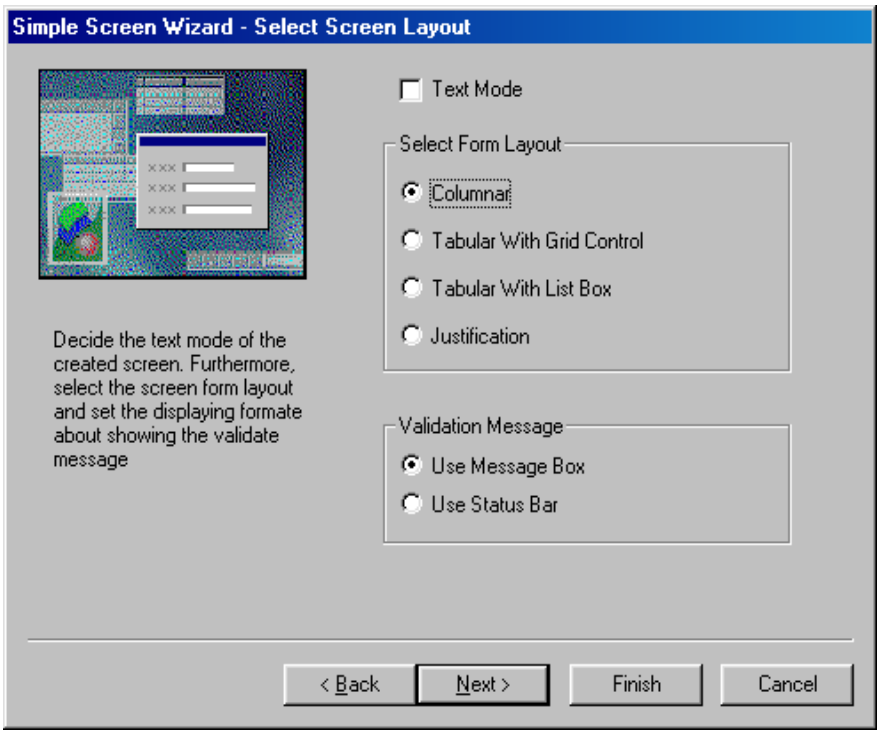
STEP 4) Enter a screen name in the Form Name text box.

STEP 5) Click OK. The Select Data Set dialog box in the Simple Screen Wizard appears.



STEP 6) Select a data set from the Select Data Set list.

STEP 7) Click Next. The Select Screen Layout dialog box in the Simple Screen Wizard appears.



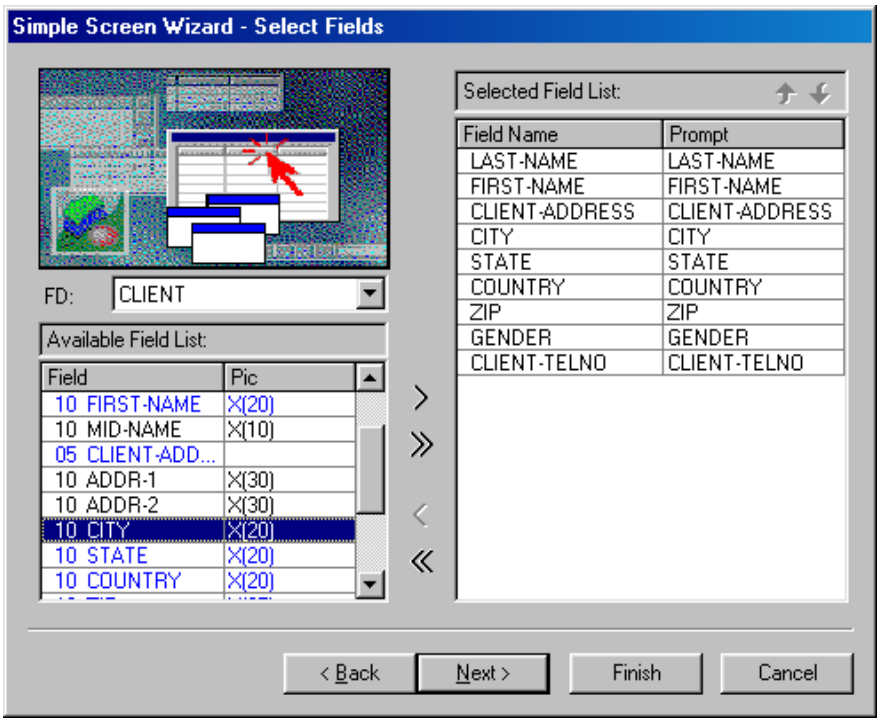
STEP 8) Select a screen layout:

- To choose a screen layout in graphics mode, click a screen layout option button in the Select Form Layout area. To help you choose a screen design, the diagram box displays an example of the selected screen.
- To create a text mode screen, select the Text Mode check box.

STEP 9) Select a format for a validation message.

- To choose a message box format, click Use Message in the Validation Message area.
- To choose a status bar format, click Use Status Bar in the Validation Message area.

STEP 10) Click Next. The Select Fields dialog box appears.



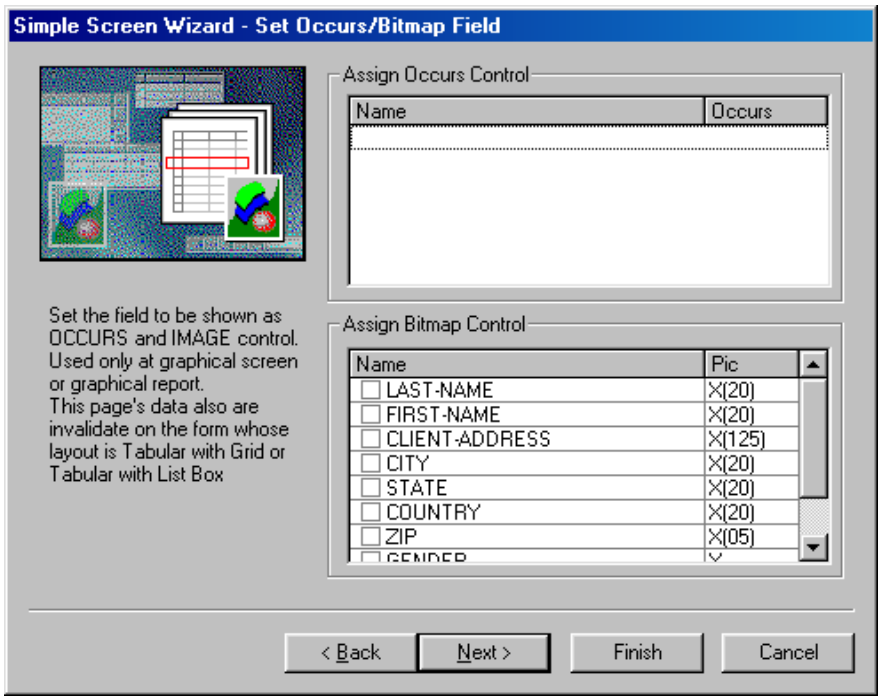
STEP 11) Select the file you want to use for the screen from the FD list.

STEP 12) Choose fields from the Available Field List.

STEP 13) Select a field(s) in the Available Field List.

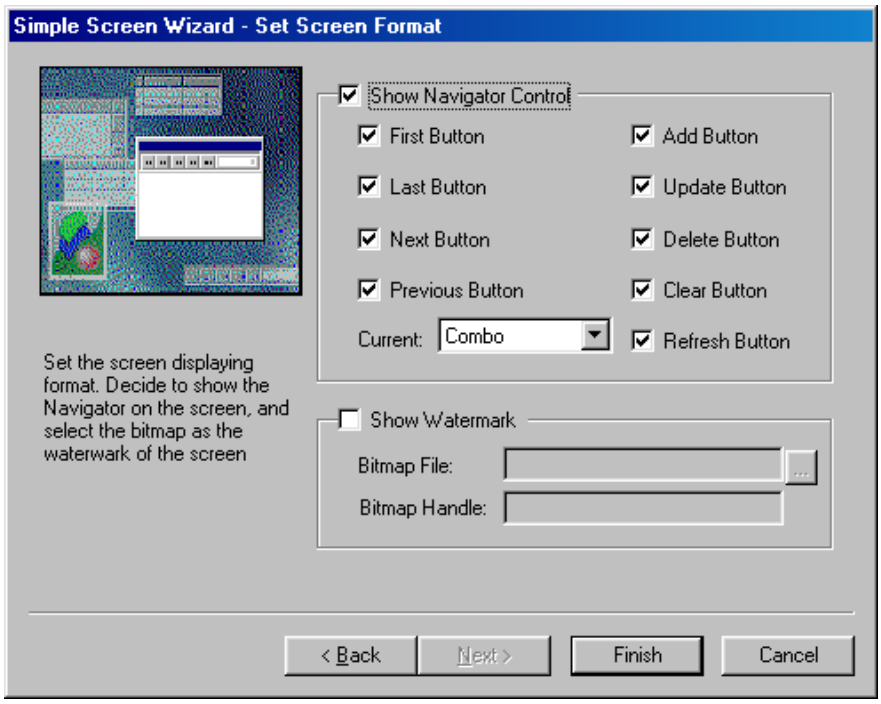
STEP 14) Click the single right-pointing arrow.

STEP 15) Click Next. The Set Occurs/Bitmap Field dialog box appears.



STEP 16) To assign occurs control or field bitmap control, in the Name column, select the check box next to the field name.

STEP 17) Click Next. The Set Screen Format dialog box appears.



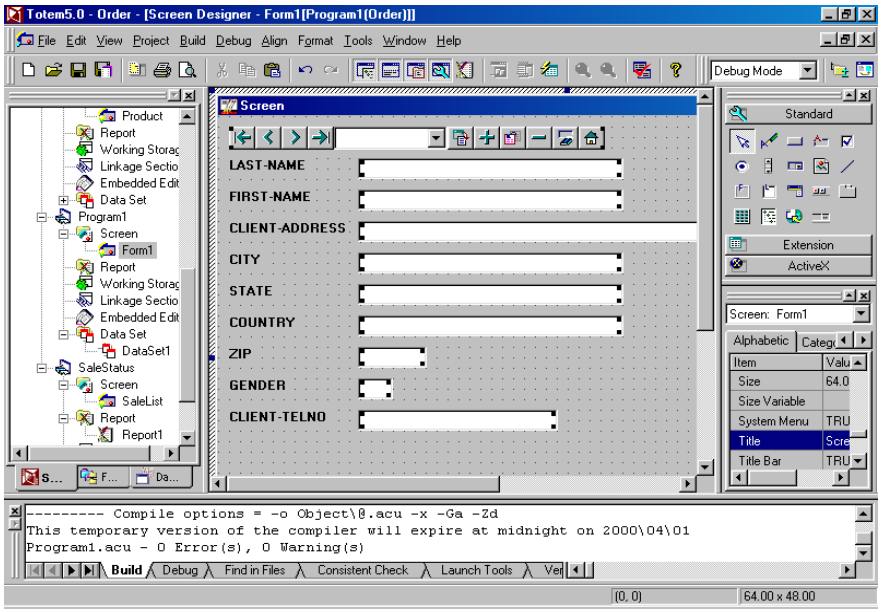
STEP 18) Select the Show Navigator Control check box to add a Data Navigator control to a screen.

STEP 19) If you don't want to include any of the navigator controls in your screen, clear the desired check boxes.

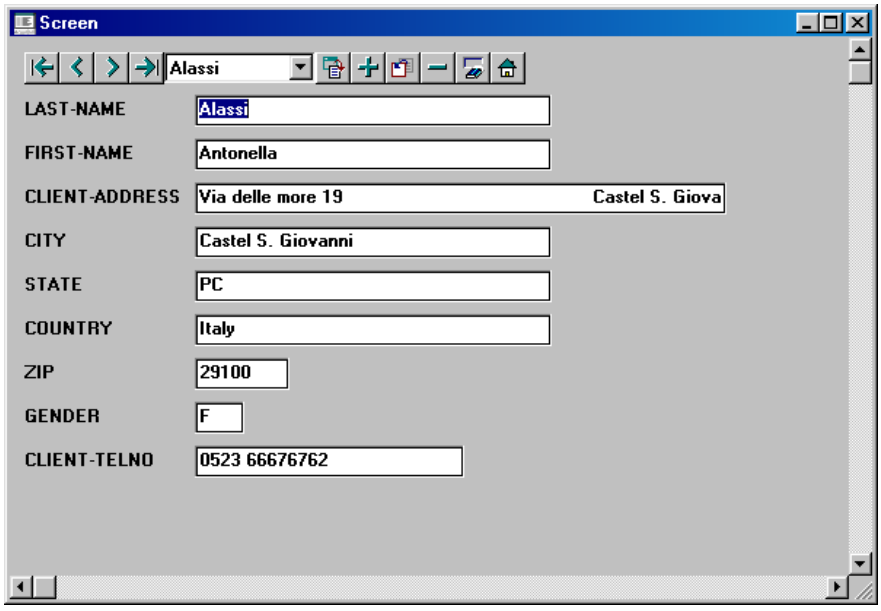
STEP 20) Select the Show Watermark check box to add a watermark.

STEP 21) Click the button next to the Bitmap File field. The Open dialog box is displayed. Choose a bmp file from a desired location.

STEP 22) Click Finish. The screen is complete.

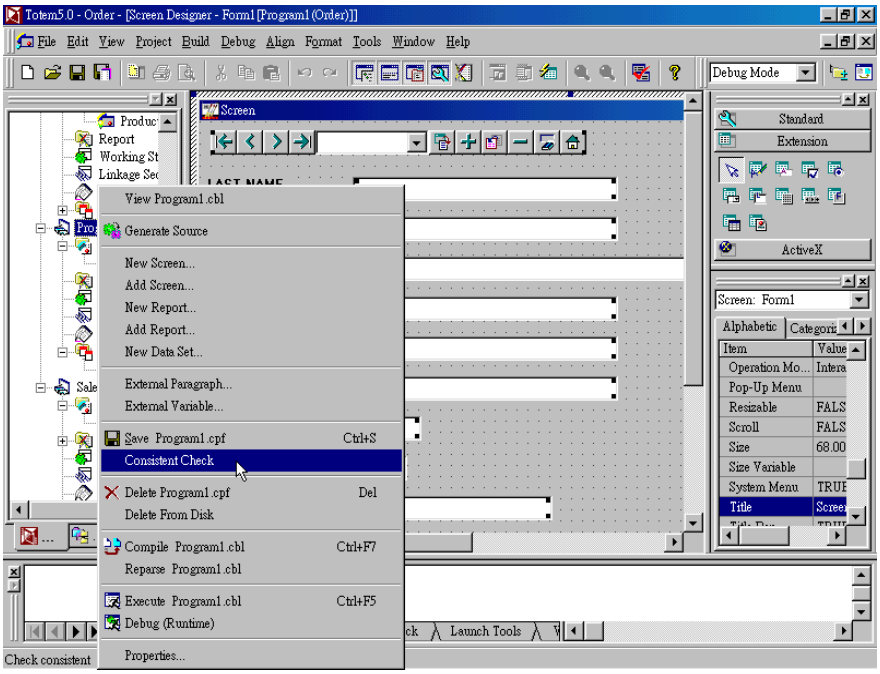


STEP 23) To execute the program and view the screen result, choose Build > Execute.



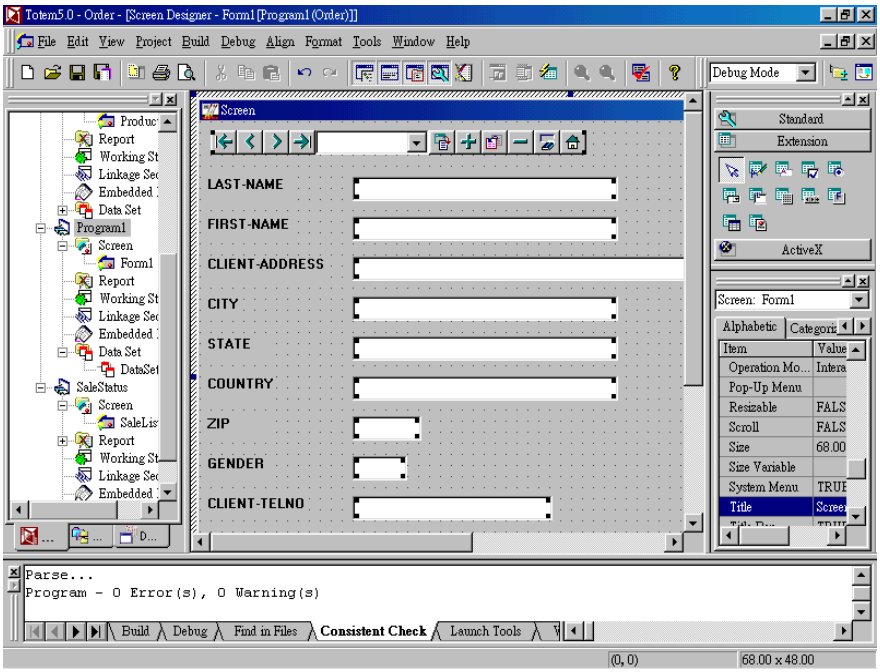
Upon completing a screen, Totem's Consistency Check enables you to ensure that all of the objects in the project—your screens, reports, and controls—remain consistent and bug-free.

STEP 1) To perform a consistency check on the screen just created, right click the program name in the workspace. A pop-up menu is displayed.



STEP 2) Make sure that the Output Window is displayed. To show the Output window, click View > Output Window.

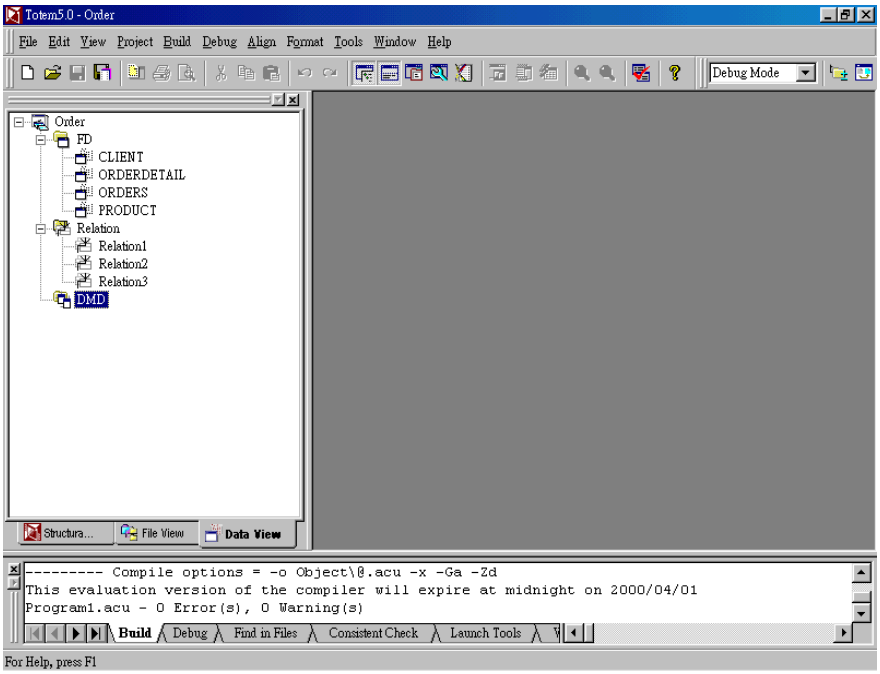
STEP 3) Select Consistent Check. The Consistent Check result is displayed in the Output Window, which is located at the bottom of the screen.



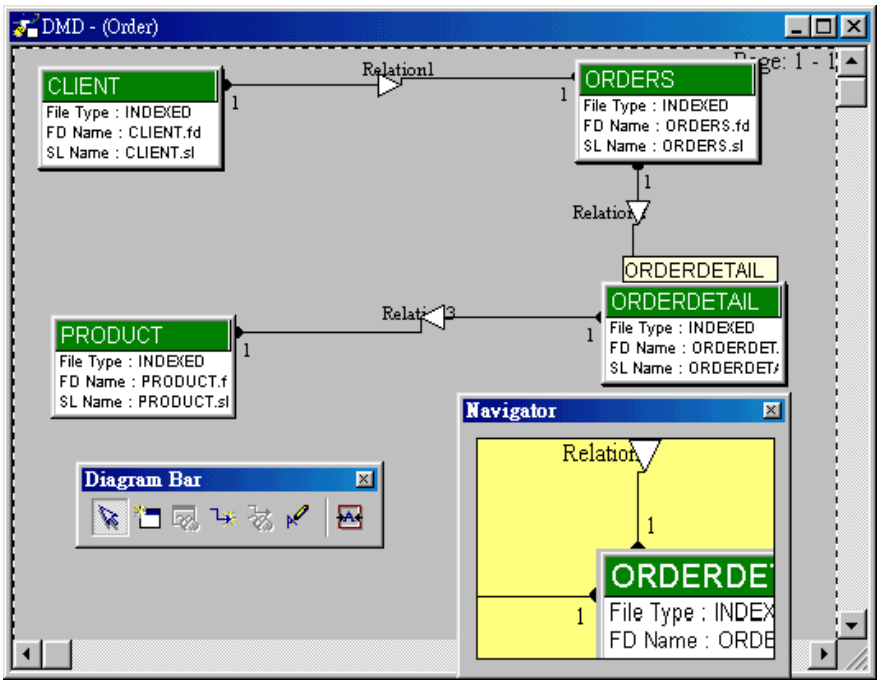
STEP 4) After performing the Consistent Check of your new program, if you observe no Errors or Warnings in the Output window, your project is consistent and bug-free.

As stated earlier, any FD files, relations, and designs in Totem are expressed graphically in the form of Entity Relationship Diagrams.

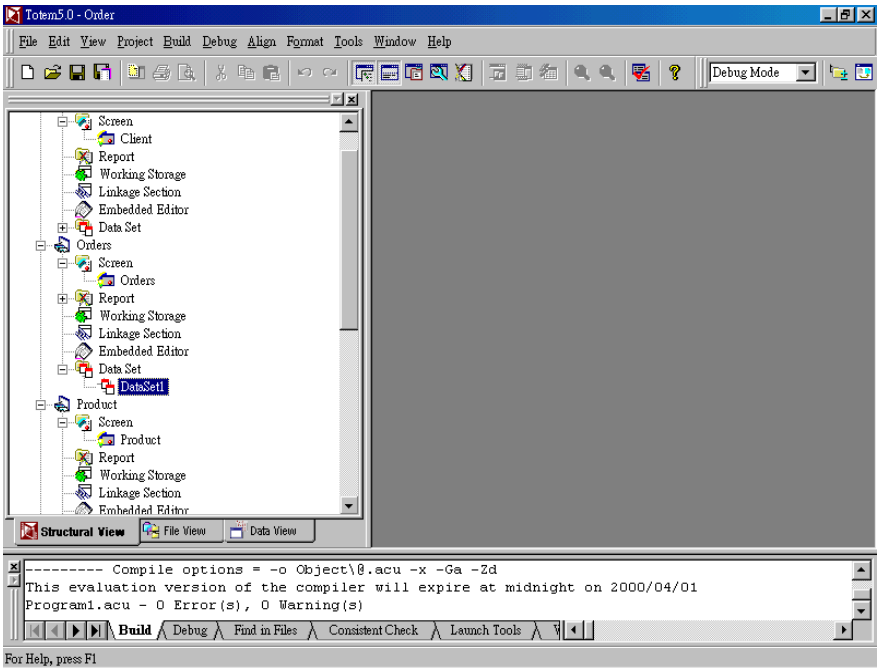
STEP 1) To view the Data Modeling Diagram of your project, click the Data View tab in the Workspace. The Data View page is displayed.



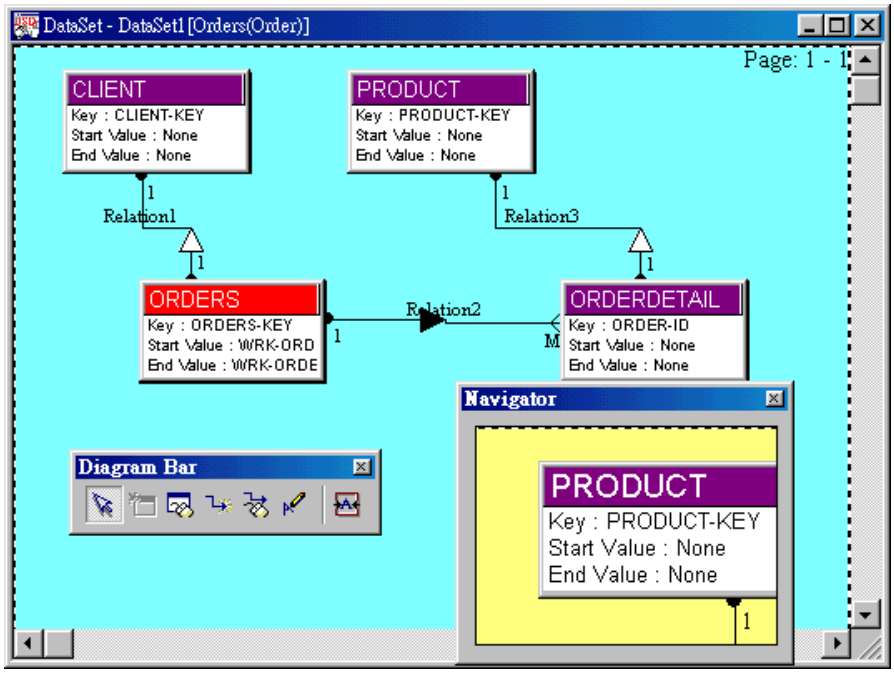
STEP 2) Double click the DMD icon. The Data Modeling Diagram appears.



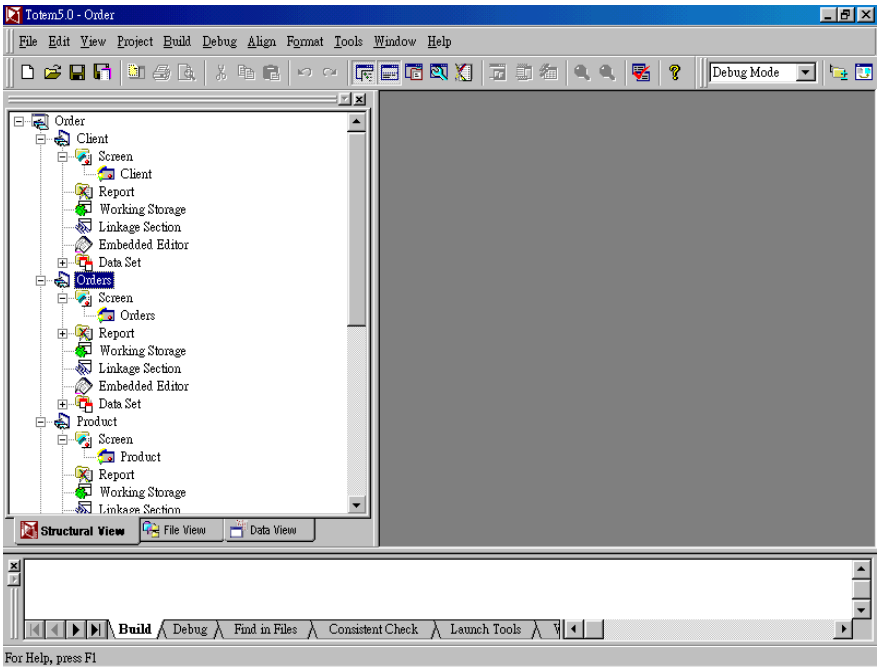
STEP 3) To view a Data Set Diagram, click the Structural View tab in the Workspace.



STEP 4) Double click the Data Set icon. The Data Set Diagram appears.

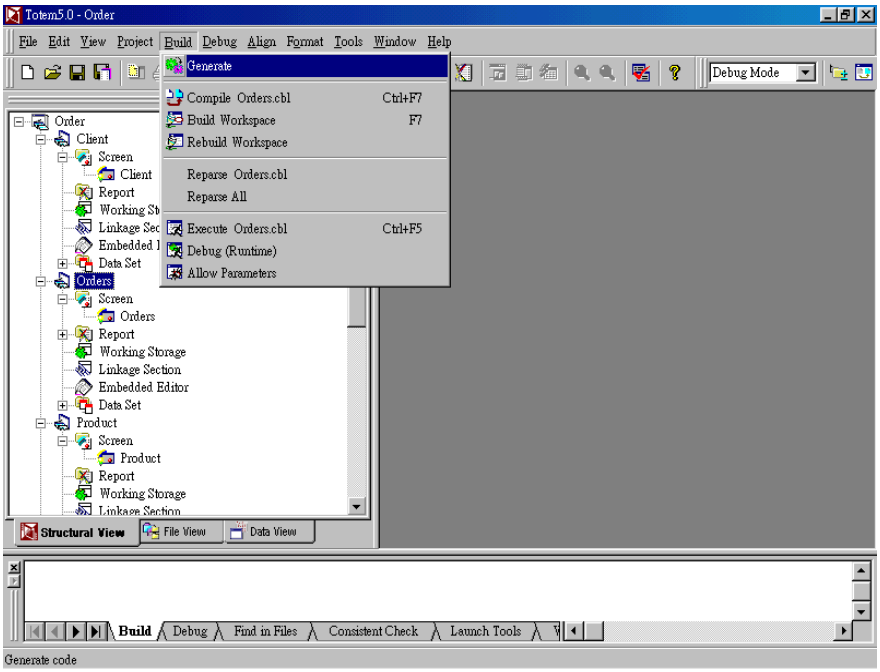


After your files and their relations have been illustrated graphically, Totem's Code Generator automatically generates the source code with precision.

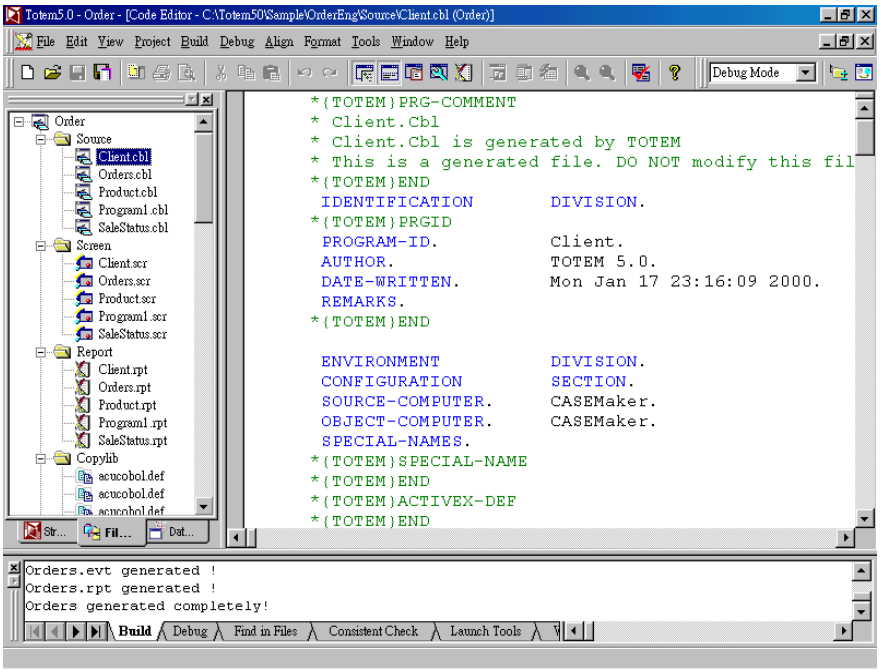


STEP 1) Click Structural View tab in the Workspace.

STEP 2) Select a program to generate the source code from.



STEP 3) Choose Build > Generate to generate the source code of the selected program.



STEP 4) To view the source code, click the File View tab in the Workspace.

STEP 5) Double click the source code file under the Source folder node. The source code appears.

RAD and COBOL?

Rapid Application Development is far more than a simple management strategy or methodology. With the implementation of powerful software tools, RAD becomes a practical and realistic method for faster and more efficient software development. The spillover effects of such development may influence other aspects of the corporate culture within an information systems organization, but the chosen tools are the fundamental basis for successful Rapid Application Development. One of the most neglected and yet most important areas in any RAD tool, however, is the programming language, because it ties many of the components together. A powerful programming language that will foster Rapid Application Development must:

- Provide important functionality such as Exception Handling, pointers, and Object-Oriented Programming, which enables customers to easily reuse and maintain code.
- Not need another programming language to successfully deploy an application.
- Increase productivity by decreasing development effort through use of powerful constructs such as pointers, Object-Oriented Programming, etc.

Given the powerful nature of most CASE tools today, any programming language may be used in Rapid Application Development. John Silver of QBS Software said, "You can't do RAD with COBOL (or at least not without strong medication)." Anti-COBOL dogmatists, such as John Silver, continue to pitch that COBOL is a computing language whose time has come and gone. To such a claim, Dave Babson, President of Burl Technologies, succinctly retorts, "The new technology providers bash COBOL because they must create a 'throw away - build from scratch mentality' in order for their products to succeed." Thus, contrary to some popular beliefs, COBOL is, in fact, still an excellent fit with the needs of large-scale, complex, long-lived business computing and application development. We, at CASEMaker, thus come not to bury COBOL, but to praise it!

COBOL is an English, readable, self-documenting programming language. It encourages a simple, straightforward programming style. It is the most portable programming language, as it can be developed, ported, downsized, upsized, rehosted, reused, and rightsized to every operating system on every hardware platform. In addition, COBOL is efficient, scalable, universal, open, complete, mature, and, most importantly, reliable and proven.

----------COBOL Language Elements and Properties----------

Business Application Paradigm Realities and Requirements	Self-Documenting	Simple	Portable	Efficient	Scalable	Universal	Open	Complete	Mature	Reliable
Large Applications	X	X	X	X	X	X	X	X	X	X
Complex Applications	X	X	X	X	X	X	X	X	X	X
Long-lived Applications	X	X	X	X	X	X	X	X	X	X
Dynamic Applications	X	X	X	X	X	X	X	X	X	X
Critical Applications	X	X	X	X	X	X	X	X	X	X
Evolving Architectures	X	X	X	X	X	X	X	X	X	X
Business-orientation	X	X	X	X	X	X	X	X	X	X
Project understaffing	X	X	X	X	X	X	X	X	X	X
Bell-shaped curve of talent	X	X	X	X	X	X	X	X	X	X
Every shop is unique	X	X	X	X	X	X	X	X	X	X
Maintenance/ Production supp.	X	X	X	X	X	X	X	X	X	X

And now, with the emergence of CASEMaker's Totem 5.0, COBOL can also be used in Rapid Application Development. Rapid Application Development separates the user from the code. The user may concentrate on the core of the application—

the business rules and processing flow—while the tool, in this case Totem 5.0, handles the rest. As such, it does not matter what language the code is in. The most recognized Rapid Application Development programming language is Visual Basic, a language whose structure and simplicity is very similar to COBOL. There is no reason to believe, therefore, that COBOL cannot be used for the implementation of RAD.

Totem's Screen Painter saves you time and gives you flexibility during the initial design phase by providing various default forms and allows you to link your screen's elements and graphic controls to the related physical data fields without having to do any hand-coding. The Data Modeling Diagram (DMD) and Data Set Diagrams (DSD) allow you to easily define and represent FD files and their relations on both the project and program levels. Logical View provides a hierarchical view of the program's structure and enables you to directly modify the program's structure through the embedded editor. The Screen Painter provides a friendly user interface to create and design visual graphical reports in both HTML and text format. Throughout each of these functions, hand coding is kept to a minimum. Totem automatically generates the vast majority of the code allowing for Rapid Application Development, and it writes that code in COBOL. As a result, many of the traditional restraints of COBOL, including graphical printing, are bypassed while all of the benefits may still be reaped.

Traditionally, COBOL has been just that, traditional. Now, however, with the use of Totem 5.0, it can be RAD!

Conclusion

Michael Hammer writes, "Radical surgery is needed in IS processes. One of the first ideas that will have to go is the whole notion of traditional development lifecycles." Rapid Application Development is that surgery. RAD, an outgrowth of prototyping methods and conceptual work by Barry Boehm and Tom Gilb, has appeal in an environment where getting products out quickly has changed from a competitive advantage to a competitive necessity.

When one compares a RAD organization to a traditional IS organization, one can clearly see an organization that is optimized for rapid development and an organization that is optimized for maintenance. Maintenance looks to the past, as it is threatened by change. Rapid development, however, looks toward the future, embracing that change. Professor Clifford Kettemborough thus states, "It is believed that the dominant trend of our era—in technology no less than anywhere else in our businesses—is unrelenting, accelerating change, and we expect that trend to continue for the foreseeable future. If we are correct, [organizations] that fail to adopt RAD…will simply be left behind."

Rapid Application Development, the development of higher-quality, cost-efficient software in a fraction of the time, has thus become a necessity as we strive to meet the new demands of the software industry. Ed Yourdon acknowledges that "information technology is now a consumer commodity" and software developers must embrace this fact by challenging themselves to adopt new, innovative means of meeting consumer demands. Rapid Application Development, and its use of powerful CASE tools, is such a means. It is a dynamic process that emphasizes throughput (getting things out the door) while de-emphasizing control (blocking the door). It overcomes the challenges of more, better, and faster. It provides both a framework and the functional tools for achieving successful, accelerated software development. Rapid Application Development is, quite simply, RAD.