



VisualAge[®] Generator

**A Powerful New Vision
of Programming™**

Contents

What's new with VAGen?	2
Good for your health	3
Partitioning VisualAge Generator Systems	7
1998 a resounding success for VisualAge Generator	13
Demystifying the VisualAge Smalltalk environment, Part 1: Applying layered architectures	14
1999 VisualAge Generator User Group Meeting	18
Integral Systems, Inc., selects VisualAge Generator as future development platform	19
Diagnosing runtime errors using VisualAge Generator 3.1	20
Using FCWERRA with VisualAge Generator v3.1 C++ programs on CICS	21

What's new with VAGen?

by Gary Johnston, VisualAge Generator Development Manager

Welcome to another information packed issue of the VisualAge Generator Newsletter!

Now, I know what you're thinking. You've noticed that the author of this *VisualAge Generator Newsletter* introduction is yet another unfamiliar name. Being extremely alert, you've also noticed that two out of the past three times that the author changed from issue to issue it's been because VisualAge Generator had a new Product Manager. So, you're asking yourself, "Can it be that VisualAge Generator has undergone still another leadership change? Is there no stability in the world? How will I cope? Help!"

Don't worry. Hayden Lindsey remains at the helm as VisualAge Generator Product Manager. We're simply continuing our informal program of passing authorship of the newsletter introduction around to other people in the organization in order to give you the chance to get a bit more familiar with some of us. So, you see, there was nothing to worry about after all.

Anyway, now that we've gotten that resolved, allow me to introduce myself. I am Gary Johnston, manager of one of the VisualAge Generator development departments. My team is primarily responsible for the development and maintenance of the VisualAge Generator Developer development environment. I joined IBM (as a software engineer in this same department) in mid-1991, and became manager of it last February. So, I've been involved with the development of VisualAge Generator (and its prior incarnations, CSP and VisualGen) for almost 8 years. Whoa! Has it been that long already? Now it's *my* turn to panic!

What's Here?

There is a lot of great information crammed into this issue of the newsletter. We hope you find it useful and interesting. Highlights include:

- A recap of some of the successes that made 1998 VisualAge Generator's best year so far.
- A documentary that describes how VisualAge Generator supported the State of California in moving a critical business application to the Internet.
- A How-to for using the Dynamic Program Partitioning feature of VisualAge Generator's Interactive Test Facility to help you figure out how best to partition your system's application logic.

What Are We Doing?

As you might guess, our entire development organization is in high gear working on VisualAge Generator V4.0, and has been since V3.1 was released last June. We're making great progress, and hope to make this significant release available later this year.

Here is an incomplete list of some of the things we're working on for V4.0:

- Integration of the VisualAge Generator Developer with VisualAge for Java. You'll have the choice of whether to use VisualAge Generator Developer within either the VisualAge for Smalltalk or VisualAge for Java development environment.
- Integrated support for IBM's WebSphere Application Server.
- The first stage of Java generation and support for Enterprise JavaBeans™ (session beans).

Note: IBM's plans are subject to change, especially this year (due to Y2K uncertainties).

Where Are We?

Well, we're on the web, of course. Who isn't? Check out the VisualAge Generator web site at:

www.software.ibm.com/ad/visgen

You'll find plenty of great information about VisualAge Generator there, as well as lots of useful links. From our web site you can:

- Download fixpak 2 for V3.1, or fixpaks for previous versions
- View and download previous issues of this newsletter
- Learn more about how you can use VisualAge Generator in your e-business applications
- Find out about IBM's Application Development Strategy for the 21st Century.

Finally, remember to check out the VisualAge Generator newsgroup! It's at ibm.software.vagen on our news server news.software.ibm.com. Amy Hawkins tells you all the details later in this issue.

If you have articles you would like to share or ideas for topics you'd like to read about, send a note to ahawk@us.ibm.com. ■

VisualAge Generator Training

Need to know more about using VisualAge Generator? VisualAge Generator classes are scheduled for June 21 and September 27 in Research Triangle Park. For more information on these classes, see the IBMLink web site at: <http://www.ibmLink.ibm.com>.

Companies interested in enrolling employees should send email to: websphere_consulting@us.ibm.com.

Good for your health

State of California demonstrates power of e-business to deliver critical health data fast

by Mike Wu, Application Development Customer Reference Manager

In the movie *Outbreak*, scientists in California raced the clock to track a deadly new virus to its source. They had less than 72 hours before the entire population of the United States would be at risk. In this scenario, access to accurate information was not only urgent; it was a matter of life and death.

Outside Hollywood, California is more likely to face an outbreak of salmonellosis or flu, than a killer virus. Still, the film underscores the importance of the early diagnosis and tracking of infectious diseases. That's why, in real life, the State of California Department of Health Services (DHS) is working to speed the testing and reporting process using new intranet technologies.

Since 1981, DHS has used a Microbial Diseases Laboratory (MDL) system to track patients' test samples and results for illnesses related to food and water bacteriology—specifically, diseases caused by bacteria, fungi, and parasites. In 1998 alone, 173,000 cases of infectious diseases were reported in the state.

With the clock ticking to the year 2000, DHS knew it had to update this important legacy system. The public's health and well-being depended on it. The question was, how?

The answer was both unique and innovative. It came through the leadership and executive sponsorship of Bryan Gillgrass, the Chief Information Officer and Byron Roberts, DHS Health Applications Support Unit 1 Chief, working with the IBM Global Services team.

Their strategy: make the MDL system Y2K compliant by redeveloping the system on the Web, while leveraging both the legacy business logic and data. In February, 1999, DHS launched a new intranet version of MDL, built using IBM VisualAge object-oriented development tools.

Now, DHS is preparing to push the envelope even further with a Web-enabled version of MDL. The new application showcases state-of-the-art intranet technologies, including Java™ servlets. At the same time, it allows DHS to make the most of its prior information technology (I/T) investments. In this sense, it provides a model of how to successfully implement an e-business solution.

"We had to take an existing legacy system that was not Year 2000 compliant and bring it into the 21st century," says Roberts. "Compared to our previous paper-based processes, we knew intranet and Internet technologies would offer a much more responsive way to report and track test results. A few months into the project, we decided to partner with IBM to move MDL onto the intranet."

In addition to Year 2000 compliance, DHS's specific goals included:

- Improve data collection quality and timeliness
- Provide a state-of-the-art application to users
- Increase the availability of MDL data to public health professionals through intranet and, down the road, Internet access
- Define an extensible architecture that can be built on and reused for future development projects

"The main benefit of MDL is the timeliness of the information," says Marilyn Capener, Chief of MDL Training and Quality Assurance Section. "As users, we're very impressed with the design. We can capture so much more information than with our previous system. And, we can run reports and check test results whenever we need them. It gives us a lot more control in monitoring potential issues. We can compare outbreak patterns between counties and quickly identify trends."

With the clock ticking to the year 2000, DHS knew it had to update this important legacy system. The public's health and well-being depended on it. The question was, how?



**Choosing the right tools:
a critical first step**

The first critical decision for the MDL project concerned which tool set to use. Roberts explains why DHS chose IBM VisualAge:

“We were already familiar with VisualAge Generator and knew we could use it to quickly rewrite the data access code. Just as important, we knew we would be able to extend the VisualAge Generator code quickly to the Web, using VisualAge for Java. In fact, we were able to reuse all of the VisualAge Generator SQL data access modules for the Web version of MDL. We simply generated Java wrappers to call the DB2 access modules. We didn’t have to change anything.”

Using VisualAge for Java and VisualAge Generator, DHS was able to leverage much of its prior I/T investments, including its S/390 based legacy data. It’s a strategy that makes sense for any organization looking to move into e-business, while preserving the investment already made in enterprise systems. For example, DHS decided to convert its vast stores of historic MDL data from a Natural ADABAS database to new DB2 tables. The development team used VisualAge Generator to write the new data access code. And, they continued to use CICS to manage the transactions between the Web browser and the S/390 Parallel Enterprise Server.

In fact, you could say this kind of architecture brings the 3270 programming model into the 21st century.

“This whole project was centered around leveraging,” says Roberts. “We were able to leverage our VisualAge knowledge . . . leverage our legacy I/T investments and skills. . . and leverage IBM’s products and expertise. The result is a state-of-the-art e-business solution that showcases some very new technologies, including Java servlets.”

Choosing the right people: a large-scale partnership effort

In August, 1998, roughly four months into the development process, DHS teamed up with IBM to begin working with VisualAge for Java. It was a ground-breaking effort, bringing together a number of new technologies on an unprecedented scale—including IBM’s new WebSphere application server. In fact, even the integration of VisualAge Generator and VisualAge for Java together in one application was relatively new.

The answer lay in partnership. The DHS development team worked together with people at a number of IBM development labs, as well as with consultants: namely, Denise Hendriks of IBM AIM Product Affinity Services, Anthony Dailly of IBM Global Services, and Martin Rybczynski of Compete Inc., an IBM Business Partner. Hendriks had previously coauthored an IBM Redbook on the integration of VisualAge Generator and VisualAge for Java.

“Through this large-scale team effort, we delivered a solid, flexible architecture and a combination of new technologies that will help DHS achieve all its critical goals,” says Hendriks, who acted as architect for the MDL project. “It’s designed to work equally well on an intranet or the Internet; its components can also be reused on future development projects; and, it features open, standards-based software. It’s truly a “write once, run anywhere” solution.”

Planning the application architecture: why Java servlets

The new MDL system is based on a three-tier architecture that can be extended to multiple tiers or even used in a two-tier configuration. DHS, for example, plans to move the application from an NT server to an AIX server . . . but it could just as easily run under S/390 UNIX Services.

“We can run MDL on any server platform or client, and we won’t have to change the application,” says Mike Virga, Technical Project Leader. “That’s the kind of flexibility VisualAge Generator and VisualAge for Java provide.”

Knowing the application would one day run on the Internet, DHS chose to base the architecture on Java servlets, rather than applets—even though servlets are considered fairly new territory in I/T today. Where applets must be executed on the client PC, Java servlets execute on the Web server. The MDL servlets process the business logic for the client by accessing DHS’s legacy system. The result: an ultra thin client, offering improved response times for intranet applications.

“The servlet model takes us away from distributed computing, back to a more centralized model,” notes Virga. “Yet, it’s truly platform independent. It makes no difference what anyone uses on the client side—not even which browser. If they have intranet or Internet access and the proper authorization, they can access the MDL system.”



“This approach allows DHS to create a fairly graceful, light solution,” adds Hendriks. “It avoids overloading the network with real-time transaction processing. Everything takes place on the server, instead of the client. In that sense, it’s a lot like 3270 programming model. In fact, you could say this kind of architecture brings the 3270 programming model into the 21st century.”

Managing the learning curve: how mentors make the difference

As the Java development began, a team of six DHS developers received two weeks worth of education on VisualAge for Java and its integration with VisualAge Generator. Within two weeks, they were busy writing code. With assistance from IBM Global Services’ Anthony Dailly, it took them roughly four months to complete the intranet version of the MDL system.

“As with any project, no matter how good the tools are, it’s the methodology that makes it successful,” says Virga. “The key to our success with the MDL project is that we got just-in-time training, combined with strong mentoring from IBM and Compete to guide us through the issues and obstacles. Through them, we also had great connections with the IBM labs. Their input helped ensure we did things the right way.”

“Without our mentors, we couldn’t have done it—not even if we had twice as much time,” says Roberts. “Nor would we have delivered as good a product or infrastructure.”

Once the developers were ready to go, Hendriks helped them divide up the classes of objects into a logical order and assign owners. Initially, the focus was on building the user interface. The team’s motto was, start simple and build from there.

For example, they were able to split the routing architecture out from the other screen elements. That meant developers could start with a relatively easy challenge: building the screen fields and buttons.

“In a couple of weeks, our developers produced more than a dozen maintenance applications,” recalls Virga. “They literally just whipped together a large chunk of the application code. It was a great way to build up the team’s confidence before we moved on to the more complex functional areas of the main system, such as architecting the routing structure.”

Building on the principle of reuse: a boost for developer productivity

According to Virga, reuse was an important aspect of the overall design methodology for the project. Their approach to building the user interface offers a case in point. “We created a standardized approach to give our screens a common look and feel,” explains Virga. “We were then able to reuse objects, which accelerated our development cycle significantly.”

For example, the DHS team divided up the class structures in order to isolate all the field validations. That meant the objects could be used to validate multiple fields, such as date, social security number and so on. In addition, the team was able to reuse the same header and footer objects on each screen.

“We estimate we reused more than 30 percent of our code,” says Virga. “Each screen is unique, but behind it there are reusable objects, including business logic, graphical objects and resource beans.”

“With every new VisualAge project, the development cycle just gets faster and faster,” he adds. “That’s because our basic methodology is to build basic templates and core programs, which can be reused among projects. In fact, we were able to reuse some VisualAge Generator programs from previous application development projects to build MDL.”

As an added benefit, developers could respond swiftly to user requests for changes. That’s because the VisualAge code is broken up into small, stand-alone objects that developers can quickly modify—without changing the rest of the application or recompiling. This supported an iterative development process, where developers and users worked closely together to optimize the application design. When users requested revisions to the prototype, they were available quickly—usually within days.

Testing for quality code: a critical part of the process

Rapid turnaround to users made testing that much more critical to the development process. “Testing should become the emphasis in project management today,” says Virga. “Your developers can whip an application together quickly, but it needs volume testing. If you don’t have resources and people dedicated to this, you can create bottlenecks fairly quickly.”

“The testing environment is critical,” concurs Roberts. “It’s another infrastructure unto itself. We had to set up separate regions for programming, for acceptance testing and for production—each with a separate database and a separate group ownership.”



**The prior MDL system:
a host-centric legacy solution**

**The new MDL System:
a distributed, Web-enabled
and Year 2000 compliant solution
featuring Java servlets**

MVS/VM-based system	S/390 CICS and DB2-based system
100+ ADABAS Natural and COBOL VSAM transactions	150+ CICS transactions 100+ SQL queries 30+ DB2 tables
Approximately 30 "green screen" maps (3270) in REXX and Natural	30+ Java Servlets and associated HTML
Business logic and rules	Business logic and rules

The ultimate test of technology: improving the quality of healthcare

Today, in its initial rollout, the new MDL system is running on a secure intranet. Primarily, users include local public health officers at some 40 public health laboratories throughout the State. They use MDL to notify DHS when samples for testing are about to be shipped. They can also dial up and see DHS test results as soon as they are posted. And, through better access to statistical reports, they can study trends and identify potential outbreaks much more quickly.

The next step for MDL is to move onto the Internet. Then, clinics and doctors will also be able to add patient data and retrieve test results from the system. That means statistical data will be collected from the earliest possible moments.

"It will give us much earlier clues that something is happening in a specific geographic area, which will in turn help us respond faster with the appropriate health services," says Roberts. "That's the ultimate reason for what we're doing: we're using leading-edge technology to improve the quality of healthcare for everyone in the State of California. With MDL, we have a Year 2000-compliant certified solution that will take us into the 21st century of healthcare."

"And," he adds, "the beauty of it all is that we did it through leveraging our legacy business logic and data." ■

VisualAge Generator Web Pages

The VisualAge Generator web address is:
www.software.ibm.com/ad/visgen

For IBM's predecessor 4GL, Cross System Product, the web address is:
www.software.ibm.com/ad/visgen/csp

Partitioning VisualAge Generator Systems

by Beth Lindsey, VisualAge Generator Development; Doug Kimelman, IBM Thomas J. Watson Research Center; and Amy Hawkins, VisualAge Generator Information Development

One of VisualAge Generator's greatest strengths is that it enables developers to quickly write high-quality, complex systems containing many different software components. In client-server or multitier environments, these components are often spread across multiple machines. The systems are often object-based with many fine-grained components handling the user interface, data access, and logic portions of the system.

While this kind of freedom is a developer's dream, it can also be fuel for performance nightmares. Placing too much of a system's logic on the client side or on the wrong tier can result in poor performance due to many remote procedure calls or remote database accesses.

Ideally, distributed application systems are partitioned to minimize communications between machines, which helps to achieve optimal performance.

VisualAge Generator's Interactive Test Facility (ITF) has a feature called Dynamic Program Partitioning (DPP) that can help you solve partitioning problems. This feature is derived from technology that researchers at IBM's Thomas J. Watson Research Center have been refining for several years.

Prior to generation and deployment, DPP helps you partition the logic of your systems onto the correct platforms. By identifying potential partitioning problems during the debugging phase, you can build VisualAge Generator systems that perform well—before they go to System Test.

The DPP feature of ITF uses a Clustering View window to visually display all the components of the VisualAge Generator system, including the target system machines. During a test run under ITF, the Clustering View window is updated dynamically to provide you with a real-time, visual presentation of what is happening as the system runs. Icons and colors are used to show interaction between system components, making potential problems easy to spot.

Components that communicate frequently move toward each other on the diagram while components that communicate infrequently or not at all repel each other. After you run a variety of test cases, you should see components forming clusters. These clusters indicate groups of components that should be kept together during program placement for optimal performance.

You can either place components on target machines yourself or allow DPP to automatically place them for you. Once you are satisfied with the component placement, you can generate programs from the Clustering View window to ensure that your components are targeted for the right platform. The next section provides an overview of DPP. Following that is an example of DPP use in practice.



Dynamic Program Partitioning Overview

DPP is enabled when you select **View Clustering** from the Test Monitor tool bar or when you select the **Partition** submenu from the Test Monitor **Tools** menu. In the Clustering View window you can watch the various components of your system as your test runs and use DPP options to determine optimal partitioning.

There are two categories of icons that appear on the Clustering View window. First of all, you can describe your runtime topology by dropping icons, called Target Machines, to represent the following machines:



Client



Database Server



Logic Server

You can assign meaningful names to these icons and select the target operating system platform. The second category of icons, called Programs, represent each type of component in your system.



Client icons—User interface components (GUI clients, 3270 maps, etc.)



Database icons—Components that access data (process data I/O options)



Logic icons—components that perform neither UI nor data access

Partitioning VisualAge Generator Systems, continued

Program icons can be placed on specific target machines and they will remain on their target machine during the entire test run. Program icons can also be left unplaced, which allows them to float on the diagram during the test run. These floating icons will gravitate toward their optimal placement on the Clustering View window.

Icons are added to the Clustering View window when one system component calls another. When the call occurs, appropriate icons for both components appear on the Clustering View window connected by a colored line. As the test runs, DPP continually analyzes the amount of communication that occurs between system components. Both the number of times two components communicate and the amount of data passed on the call are factored into the communication algorithm. The more two components communicate, the more likely they are to move toward each other to optimize performance. If one or both of the program icons are free to float, you will see the icons physically move closer to each other as the test continues.

The color of the lines connecting icons is important as well. Lines approach red as communication between the two components across a network link increases, or when one component is being pulled towards two different machines. Both situations indicate a potential performance problem. The color spectrum along the bottom of the Clustering View window shows that blue is used to depict 'cold' connections, and that red is used to depict 'hot' connections.

Lines approach red as communication between the two components across a network link increases

So, as your test runs, you can use DPP to dynamically see the optimal partitioning of your systems and spot potential problem areas. Components that have gravitated toward each other are good candidates to consider placing on the same machine. You should examine components that are connected by red lines to determine whether modifications could further optimize performance.

When you know the details about your actual system configuration, you may want to add target machine icons yourself that accurately reflect your runtime system. For example, if you have a Windows NT client and an AIX server machine, you can add those target machines to the Clustering View window yourself. As with any type of debugging, you will probably need to run the test numerous times to get the partitioning correct. In order to ensure that your partitioning is optimal, we recommend viewing several representative runs of your system, complete with a representative mix of transactions. You may also want to experiment with several different machine configurations to see which one provides you with the best system performance.

Instead of adding your system target machines each time, you can save this information to a file and reuse it the next time you test. The target machine information is called the **Clustering View Topology**. Once you have the target machines set up correctly, with both correct names and target environments selected, you can save the topology. To reuse the topology during the next run, open an existing Clustering View window instead of creating a new one. You will then be prompted for the name of the file from which to read the topology information.

There are three component placement options that are provided by DPP:

- **Clustering only**
- **Manual partitioning**
- **Default partitioning (default)**

These placement options allow you to tailor the way **Target Machine** and **Program** icons are added to the Clustering View window.

With the **Clustering only** option, DPP simply shows the component icons and the connections between them. No target machine icons are added automatically and no placement of programs onto target machines is attempted.

With the **Manual partitioning** option, you are prompted to specify the target machine to place each program on as it is added to the Clustering View window. This option puts the responsibility of determining initial machine location on the user.

With the **Default partitioning** option, **Client** and **Database Server** target machine icons are added automatically on the Clustering View window when a component with user interface or data access is added. **Client** programs are placed by default on the Client target machine and **Database** programs are placed by default on **Database Server** target machines. **Logic** programs, those that do not perform user interface or data access functions, are not automatically placed on any target machine. You can choose to be prompted for target machine placement any time a **Logic** program is about to be added to the Clustering View window or you can allow the **Logic** programs to be added unplaced (free to float) during the test run so that they naturally gravitate to their ideal machine placement.

After you've used DPP successfully to determine the optimal partitioning, you can actually drive the generation of the system components from the Clustering View window. Set up the Clustering View window with your target machines, run the test, and get all the programs placed on the correct machines. You can then select all the program components and choose **Generate** from the context menu of one of the programs. Generation windows for each target machine are displayed, listing all the program components that should be generated for each platform.


Using Dynamic Program Partitioning—A Real Example

Before you can effectively use the DPP feature, you need to have your VisualAge Generator system at least partially completed. You don't have to have all the business logic written, but the structure of the program components, communication between them (including data that will be passed), whether they perform user interface, data access, or pure business logic should already be defined. At this level, you can use DPP to partition your system.

In the following section we describe a small sample VisualAge Generator system that we will use throughout this task-oriented example. The sample is representative of actual VisualAge Generator application systems and allows us to show the usefulness of DPP. The user interface is a VisualAge GUI client called CUSTGUI that controls user requests and calls numerous back-end programs to handle the data access and business logic. There will be three platforms in the client-server system: Windows NT, OS/400, and MVS CICS. This three-tier architecture is typical of many VisualAge Generator systems.

1. Remember that DPP is a feature of the Interactive Test Facility and is activated from the Test Monitor window. Since our starting point is a VisualAge GUI client, we need to set the **Break on event entry** option so that the Test Monitor will be brought to the foreground on the first event. Then we can activate DPP before the first VisualAge Generator CALL is made.

To set the **Break on event entry** option:

- a. From the VisualAge Organizer window, select the **Preferences** button. 
- b. On the VAGen-Test General page, set **Break on event entry for GUI clients**.
- c. Select **OK**.

2. Start the test run by selecting CUSTGUI from the VisualAge Organizer window and selecting the Test button. The Manage Customers client window is displayed. Select Retrieve Customers to display the Test Monitor.
3. Look at the current DPP options by selecting **Tools**, then **Partition**, then **Setup**. The Partitioning Setup (defaults) dialog is displayed. You can specify the partitioning **Placement Options**, described above, which type of connections to show, and an animation delay that can be used to slow down the graphics on the Clustering View window.

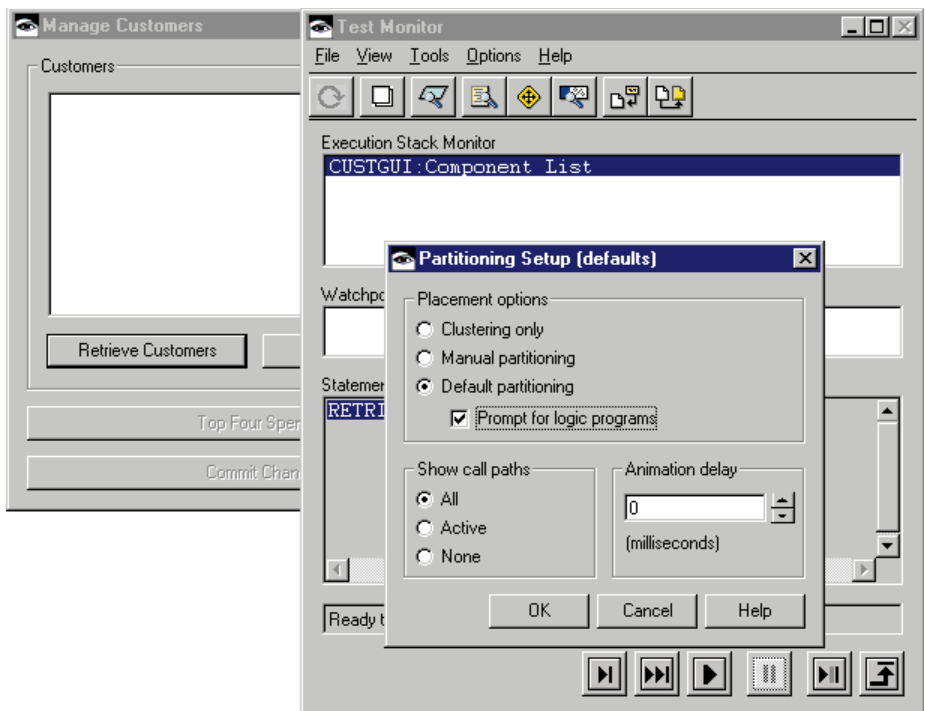



Figure 1. Test Monitor, Manage Customers, and Partitioning Setup Windows

Partitioning VisualAge Generator Systems, continued

The options that you select help you tailor what you see in the Clustering View window and are saved across images so you only need to set them once and they will apply to all DPP test runs. If you want to modify the options for one particular DPP run, you can do so by selecting **Options**, then **Setup** from the Clustering View window to display the Partitioning Setup dialog. We will select **Prompt for logic programs** for this example.

- Set up your topology information so that it can be reused on subsequent test runs. From the Test Monitor, select **Tools**, then **Partition**, then **View Clustering**, and then **New** or select the **View Clustering** button  from the tool bar. An empty Clustering View window is displayed.

Select **Options**, then **Add a Target Machine**, and then **Client** to add the target machine icon that will represent the client Windows NT system. On the **New Target** dialog that appears, replace CLIENT1 with NT_Client and select Windows NT from the Target environment drop-down list. Select **OK** to drop the target machine on the Clustering View window. Repeat the procedure to drop the logic server machine, by selecting **Options**, then **Add a Target Machine**, and then **Logic Server**, and specifying OS/400 as the target environment. Repeat the steps to add the other target machine, by selecting **Options**, then **Add a Target Machine**, and then **Database Server**, and specifying MVS CICS as the target environment. You can then move the target machine icons to desirable locations in the Clustering View window.

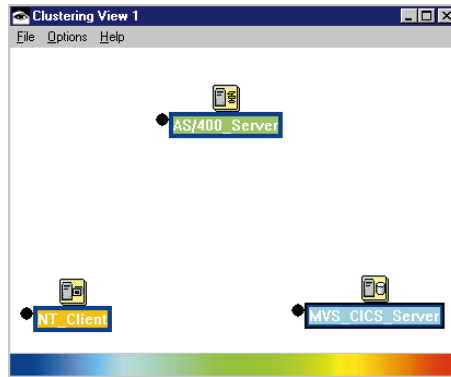


Figure 2. Clustering View Window with Target Machines

Now that you have your target machines set up correctly, you want to save them away for future reuse. To do this, select **File**, then **Save Topology As**, and specify a file name. When you want to use the topology to initialize the Clustering View window for future test runs, instead of opening a new Clustering View window like we did in this example, you can open an existing one. You will be prompted for the name of the topology file to restore.

- With the Clustering View window options set and the target machines initialized, you are ready to actually run the test. Select the **Run** push button on the Test Monitor. The Clustering View window changes as the test runs and looks like Figure 3 when it finishes.

The first VisualAge Generator logic part to run is a statement group named RETRIEVE-CUSTOMERS. This statement group contains a loop that controls calling the server programs to do the data access. As soon as you execute the first CALL statement, program icons are placed on the Clustering View window. Since we are using the default partitioning placement option, a **Client** program icon, named CUSTGUI, is placed on the **NT_Client** target machine. This icon is placed there because it performs

user interface functions. The second icon, named CUSRCC1, represents the called program. Since it is not part of the user interface and performs no database access functions directly, it is represented by a **Logic** program icon and we are prompted for its placement. Select the **AS/400_Server** target machine. A line between the two program icons represents the call.

As the test continues, select **AS/400_Server** each time you are prompted to place logic programs. Watch what happens on the Clustering View window. Numerous other program icons are added to the diagram and DPP continually evaluates the communications between all the components that make up the program system. Over time, several of the icons are connected on both sides by very red lines pulling them to different target machines. This indicates a potential problem area and prompts reevaluation of the program partitioning.

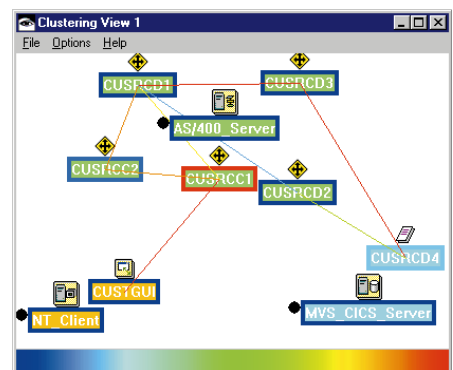


Figure 3. Application System Requiring Repartitioning

6. So that DPP can help us alleviate this problem, let's release several logic programs that we explicitly placed on the **AS/400_Server** target machine. Select the icons representing CUSRCC1, CUSRCC2, and CUSRCD3 and choose **Release Programs** from the context menu. This gives DPP the freedom to determine the optimal target machine placement for these logic programs.

7. You can now select **Adjust Configuration** from the context menu of the Clustering View window to allow the DPP system to relieve some of the tension in the diagrams (the red lines). Now free to move, the program icons move in the direction that will lessen this tension. **Adjust Configuration** can be used repeatedly. In most cases, a stable program system configuration can be reached, and movement between programs stops. However, it is possible that the programs continue to move each time **Adjust Configuration** is selected. If the system does not reach a stable configuration, programs should be broken into smaller pieces of logic.

After you select **Adjust Configuration**, the Clustering View window will look like Figure 4. Since **Adjust Configuration** simulates a separate test run, different colors have been selected by chance for the target machines. Notice that the released logic programs, shown in dark blue, have moved away from the **AS/400_Server** target machine toward the other machines. The red lines pulling them in different directions have changed color. This shows a more optimal partitioning: the logic programs that are called often from the user interface are on the **NT_Client** client machine, the logic program that interacts closely with the data access program is on the **MVS_CICS_Server** database server machine, and the other logic

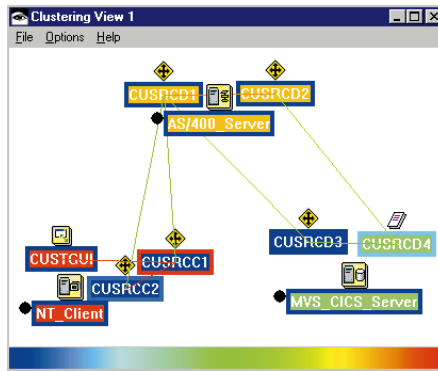


Figure 4. Correctly Partitioned Application System

programs are left on the middle **AS/400_Server** logic server machine.

8. Once the system configuration is stable, we can run **Automatic Placement** to allow DPP to place the floating **Logic** programs on the target machine that provides the optimal performance. To do this, select **Options**, and then **Run Automatic Placement**. An information window, shown in Figure 5, lists which programs have been placed on each target machines. If you don't like the placement that DPP selects by default, select **Options**, then **Undo Automatic Placement** and manually place the programs on the target machine you want to use.

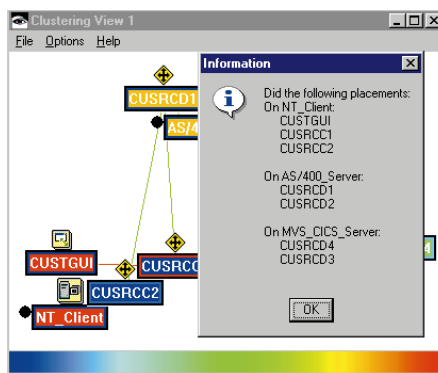


Figure 5. Clustering View Window with Placement Information

For this application system, automatic placement improved performance by a factor of 15. The time to complete a RETRIEVE-CUSTOMERS benchmark decreased from 75 seconds to approximately 5 seconds. These times were measured for generated C++ code running on a small testbed based on a 16 MB token-ring network. Modest Windows NT 4.0 workstations served as client machine, logic server, and database server.

9. Since the system is now partitioned correctly and the test is successful, you can generate the VisualAge Generator programs from the Clustering View window. Use the mouse to draw a box around all the program icons. A black line will surround the name box below each icon, indicating that it is selected. From the context menu of the Clustering View window, select **Generate**.

Note: You will get an information message that non-VisualAge Generator components cannot be generated from this window. This is because we have the icon representing the VisualAge GUI view selected. Since the "generation" of that part is controlled by Smalltalk packaging, it isn't handled here. Client icons could also represent web pages or other user interfaces, which must be handled in other ways as well. Generation will continue for all other selected VisualAge Generator programs.

For this application system, automatic placement improved performance by a factor of 15.

Partitioning VisualAge Generator Systems, continued

Three separate Generate windows will be displayed, as shown in Figure 6: one for each target system. The programs that should be generated for each system will be listed on the appropriate Generate window and the target system will be preselected. You should now create or modify the necessary generation option and linkage table files, select **Set Options** to specify Generation options for each Generate window, and then select **Generate** to generate the VisualAge Generator programs for the right target platforms.

If your system configuration changes in the future, you can repeat the dynamic partitioning and regenerate for the new platforms. ■

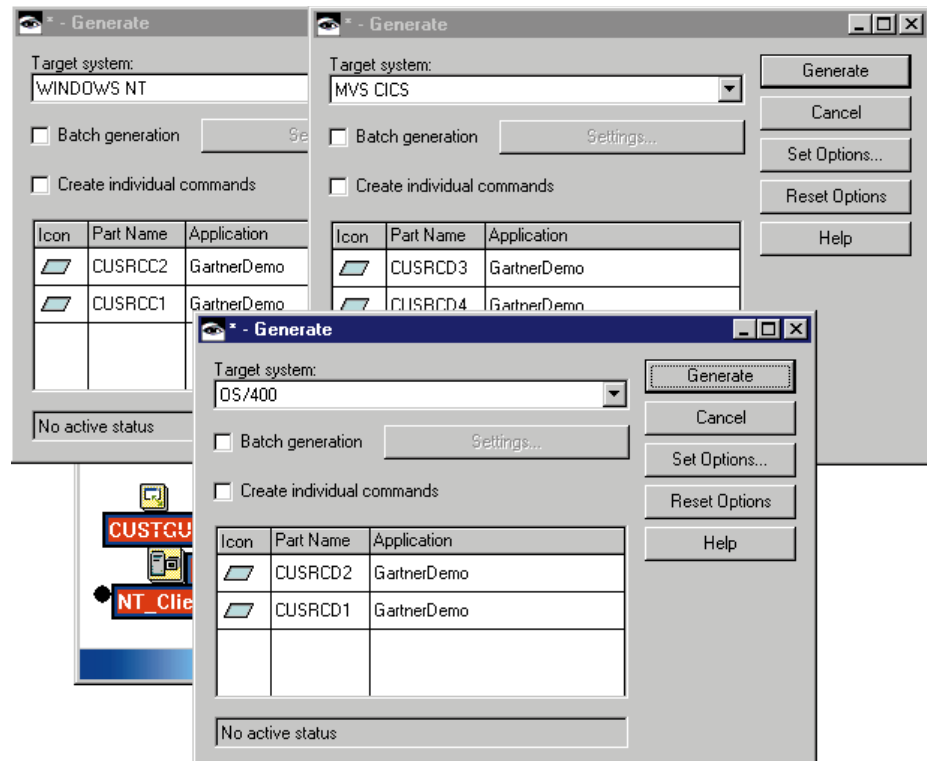


Figure 6. Program Generation from the Clustering View Window

An invitation for you

We are currently seeking customers to partner with the lab and with IBM research to help us identify any changes that would make the tool even more valuable to you. To participate, you will need to host at least one IBM researcher for approximately a week, provide access to your VisualAge Generator client-server development systems so that the DPP analysis can be performed on them, and provide at least one VisualAge Generator developer to work with the researcher.

The value to you will be hands-on experience to learn how to effectively utilize the DPP feature, the ability to greatly influence the direction of future enhancements, and a better understanding of the performance of your systems. If you would like to participate or just get more information, visit our web site at

<http://www6.software.ibm.com/reg/vast/vagrcf1-r>

and select **other** and indicate your interest in "Dynamic Program Partitioning." ■



1998 a resounding success for VisualAge Generator !!!

by Rusty Edmister, VisualAge Generator Sales Support

The year just completed was by far the most successful year that VisualAge Generator has enjoyed since it became generally available in 1994! Not only did many new companies begin using this increasingly powerful product, but many current customers increased their usage of the product as well.

Among new users for VisualAge Generator were one of the world's largest consulting firms, one of the world's largest credit card companies, and one of the largest banks in the US. All came to the product because of its productivity, its versatility, its scalability, and its portability. On the other end of the scale, some of the new users included organizations with only a handful of developers and one with only one programmer! Whether in the hands of a single developer or in the hands of hundreds, VisualAge Generator continued to prove its worth as a tool allowing development of applications that range from batch programs to web-enabled applications and everything in between!

And speaking of the Web, the State of California - Department of Health Services, an award-winning development organization in Sacramento, developed and delivered its first web application called Microbial Disease Laboratory System (MDL) in record time using VisualAge for Java and VisualAge Generator in late 1998! Details of the application appear in another story in this edition of the newsletter.

1998 was also a very good year for VisualAge Generator with industry consultants. The Gartner Group placed VisualAge Generator and some of its VisualAge peer products in its "Leadership Quadrant," a distinction reserved for products that it deems will be the leaders in the area of application development in the coming decade! Other industry consultants, including Ovum and Meta Group, wrote very positive articles about the product last year as well.

1998 was the year of User Group Meetings. In July, a North American User Group meeting was held in RTP, the home of VAGen, and was attended by more than 40 companies represented by more than 100 people. A similar meeting was held in Stuttgart, Germany. That event was attended by nearly 160 people from more than 70 companies that are either using or considering using the product. The 1999 North American meeting is described in another article in this edition of the newsletter and will be held in Chapel Hill, NC, on June 15 & 16. Please see the article mentioned or visit VAGen's web page for details.

1998 was also the year the VisualAge Generator User Forum moved to the web at the request of a multitude of its users from around the world. It is a newsgroup and is accessible from the VAGen web page at www.software.ibm.com/ad/visgen. It provides users a way to ask questions about the use of the product. Answers come from both RTP developers and VAGen customers.

1999 and the years ahead promise to be even more successful and exciting for the product and its users. Early this year, new customers in Peru and Qatar began using VAGen. That brings to 52 the number of countries in which it is being used. Product plans for the future include new production environments, an enhanced scripting language, and interoperability with VisualAge for Java! With all of that said, the Development Team in RTP understands that the product cannot be successful and has no future without its customers and thanks you very much for your continued use of and confidence in VisualAge Generator! ■

1998

Demystifying the VisualAge Smalltalk environment, Part 1: Applying layered architectures

by Tony Cianchetta, VisualAge Information Development Manager, Generator, Java and Smalltalk

This article was previously published in VisualAge Magazine (<http://www.vamagazine.com>).

Editor's Note: This is the first in a series of articles geared toward making VisualAge Generator users more comfortable in the VAST environment. You don't have to learn Smalltalk to develop systems with VisualAge Generator, but a grounding in the underlying concepts, especially those outlined in this series, will certainly help you build better application systems. The author of this article is a longtime VisualAge Smalltalk instructor. He has taught users from all backgrounds, so whether you're new to the VA environment or you've been nervously ignoring it for a while, this article will help you clean up your connections and get plugged into the power of reuse.

When applications get more sophisticated, the value of visual programming begins to diminish in a "green haze" of tangled connections.

It is widely accepted that visual programming is a viable tool for building user interfaces and small applications quickly. Reusable components, instance collaboration through connections, and a point-and-click development interface all contribute to meet the need for rapid application development. But when applications get more sophisticated, part responsibilities become more complex, and the value of visual programming begins to diminish in a "green haze" of tangled connections. These circumstances raise a serious question: "Can the visual programming metaphor provide a practical solution for building large scale enterprise applications?"

The answer is "Yes." But the visual tool must allow you to move easily between visual programming and writing specialized code. With VisualAge Smalltalk, the two coexist and you can work with either at will. Event-to-script connections allow you to communicate "things" that happen at the user interface to actions implemented in Smalltalk code. But the need remains for a level of discipline to avoid that "green haze."

By adopting an architecture appropriate to the scale of your application[1], you can successfully use all the features of VisualAge Smalltalk to meet your development needs. When parts are separated into layers by responsibility or service, the increased complexity associated with large applications is spread across those layers.

By applying a "divide-and-conquer" approach to complexity, the total number of connections per part can be kept reasonable. Components, in turn, become more reusable and certainly more understandable.

The result may be more components in the solution space, but this is not necessarily a bad thing. The more reusable components your team develops, the less they'll have to do later in the development process.

This article explores the various ways a layered architecture can be applied to visual programming and compares the complexity level of layered versus non-layered component development.

Getting back to basics

Think back to the early days of OO technology. There were many discussions, and volumes written about, layered applications, multitiered architectures, and the separation of the Model, View, and Controller (MVC)[2]. If we take those concepts and apply them to a VisualAge Smalltalk project today, some pretty extraordinary things happen; we find it can reduce the complexity of our composite parts and make them reusable.

Layered or multitiered architecture concepts map well into the world of VisualAge Smalltalk. With a team development environment that enables us to logically separate the parts in each layer, all we need to do is work with an available object model and define those layers. Let's define a simple domain and step through the concept.

Simple objects

Let's keep it simple and start with a model that contains name, address, and person objects. Because we can prototype this quickly, let's use the VisualAge Organizer to define the classes for these domain objects as Smalltalk classes:

VastName (Smalltalk class) Attributes firstName (String) middleName (String) lastName (String) nameString* (String) Actions Events *no setter needed	VastAddress (Smalltalk class) Attributes street (String) city (String) state (String) zipCode (string) addressString* (String) Actions Events *no setter needed
--	--

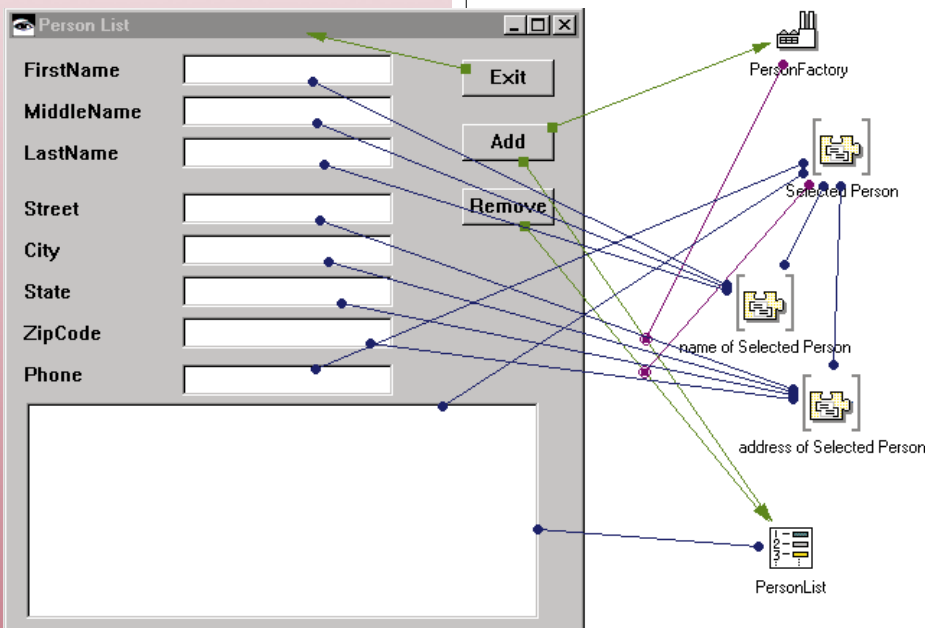
VastPerson (Smalltalk class) Attributes name (VastName) address (VastAddress) phone (String) mailingLabel* (String) nameString* (String) Actions Events *no setter needed
--

Let's also say that the following list is a subset of the behavior we need to provide for these objects:

- Create new instances
- Collect all the instances into a manageable group
- Work with the instances as a group
- Select individual instances from the collection

- Display attributes of an instance as read-only
- Display attributes of an instance to allow value update

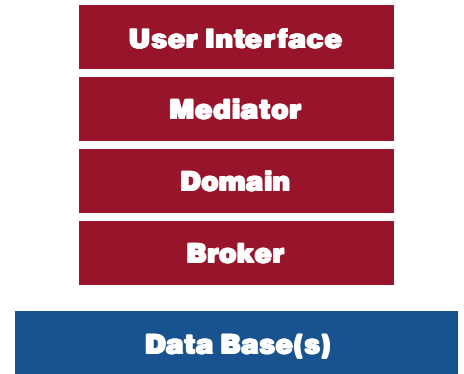
There's nothing astounding about the list. It includes routine functionality, some of which we take for granted in Smalltalk because of class behavior. If we were to visually build a part to satisfy these requirements, the result might look like this:



OK- so this is, perhaps, an exaggeration. But even this simple example shows that the visual metaphor tends to orient development towards a visual solution (...when you think your only tool is a default window, every specification looks like a UI problem...). It's just so easy to drag, drop, and connect that the overall design process often gets ignored or, at the very least, compressed.

There's nothing wrong with this style for simple parts. However, when you build parts to satisfy the need for more complex behavior, separating parts into layers will serve to reduce the complexity of implementing that behavior.

If we back up and look at this model from the perspective of separating parts by responsibility, we can work to fit the implementation into layers. Here's one potential view of separation:



Defining layers

If we work our way up from any existing database to the user interface, we can define each layer in the diagram as follows:

Broker layer

This layer is responsible for translating whatever is returned from the database into collections of domain objects. If a MultiRow Query part is used to query a relational table that contains person information, a collection of dictionaries representing person records is returned. A broker part is responsible for creating a VastPerson instance for each person record returned from the database and setting the appropriate attributes from the corresponding record values. Each VastPerson instance should then be added to an OrderedCollection for use within the application domain.

By applying a “divide-and-conquer” approach to complexity, the total number of connections per part can be kept reasonable.

Demystifying the VisualAge Smalltalk environment, continued

Caching, to improve performance, and transaction processing, to preserve data integrity as well as enable rollback, can be added in this layer. Brokers also provide the opportunity to implement update notifications for concurrency protection and collaboration with other broker parts when an object requires data from multiple tables that have their own brokers. Broker parts are also responsible for transposing domain part instances into a form suitable for saving into the database. We can either build this layer from nonvisual and database parts or it can be implemented by add-on features like ObjectExtender.

Domain layer

This one's easy. Here we build Smalltalk classes that model the real-world objects of our business domain problem space. These objects are the result of OOA&D and are defined in the resulting object model.

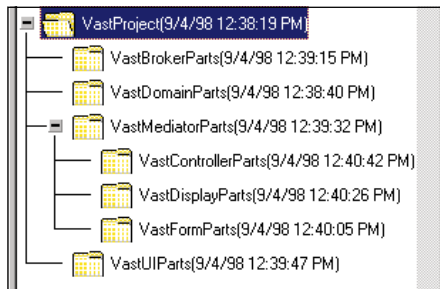
Mediator layer[3]

There's more going on here than you may think. This layer prepares domain objects for collaboration with the user interface. We can define sub-layers to handle form parts, display parts, and controller parts.

Form parts are reusable views of part attributes that enable the end user to update or modify attribute values (sort of like read/write access to instance attributes). Display parts are reusable views that show attribute values in a manner that is read-only (like a label). Controller parts enable us to create, modify and remove domain instances from their respective collections during run time. By utilizing VisualAge's event handling capabilities, controllers give us a vehicle for implementing behavior associated with the Observer pattern[4].

UI Layer

The user interface is made up of the navigation windows and visual parts that enable the end user to see, manipulate, and work with business domain information. From a VisualAge Smalltalk application perspective, we can make the relationship of these layers more intuitive (visually) by creating the following hierarchy:



We then define and implement the following parts in each layer:

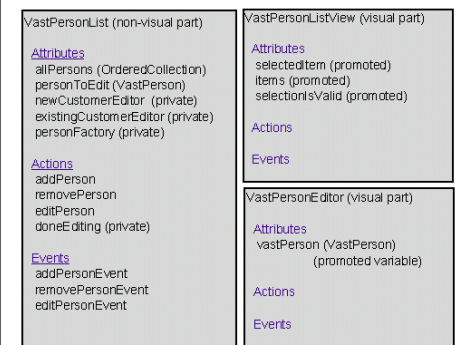
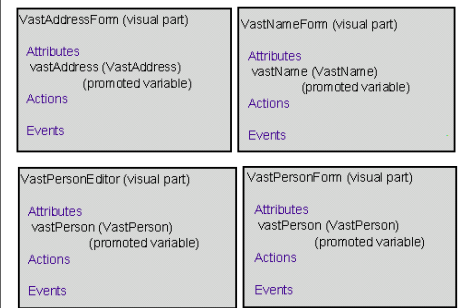


By building the parts in the above list, a "vertical slice" through the domain space is implemented. That vertical slice enables our application to process from the database up through to the user interface for a particular domain object or set of domain objects. The implementation of all these classes, and an explanation of how they fit in a

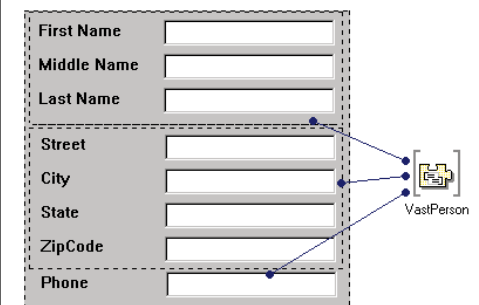
layered architecture, is best explained in a series of hints & tips papers. These part definitions are key to this example.

Leveraging layers

By building reusable form and display parts as subcomponents (like VastNameForm or VastAddressForm) we can create more complex reusable components like the VastPersonForm. This technique serves to reduce the

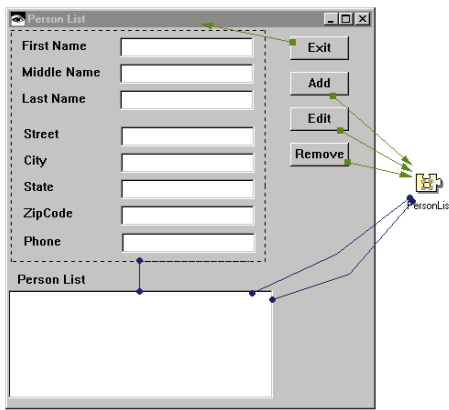


number of connections required at each level of component complexity, as shown in the following VastPersonForm example:



If the name and address forms were not used, this particular view would require a minimum of eight connections to represent the attributes of a person instance.

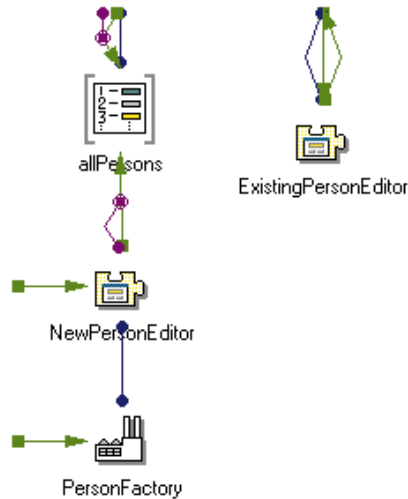
The difference becomes even more pronounced by the time we get to the user interface.



This is the same part that was implemented earlier as a single view. The number of connections required to represent the name, address, and phone attributes of a person reduce to one on this visual part. Earlier, it required eight connections and two tear-off attributes to get what we needed. In fact, the total number of connections was reduced from 16 to 7, and this part implements an edit function.

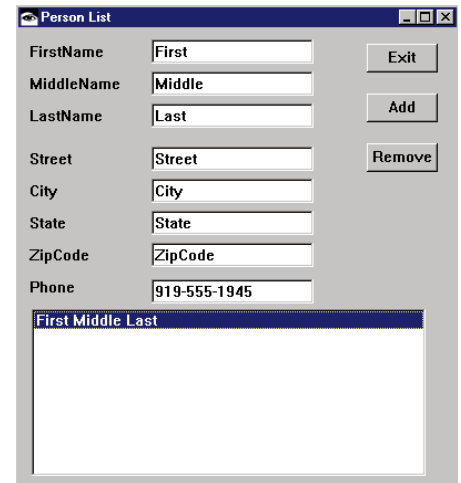
The implementation of a controller part (VastPersonList) eliminates the need for parameter-passing connections to satisfy the add: and remove: keyword connections. Controllers also provide the opportunity to open editor views on new and existing person instances to allow the end user the chance to modify attribute values.

The following figure shows the completed VastPersonList controller. When this controller receives the addPerson message several things happen. It's PersonFactory subcomponent creates a new instance of VastPerson and passes it to a NewPersonEditor view through an attribute-to-attribute connection. The NewPersonEditor is then opened by the same addPerson message. When the end user closes the NewPersonEditor, the instance is then added to the allPersons collection.



Instances of VastPerson are removed directly from the allPersons collection, or edited in another instance of the person editor called ExistingPersonEditor. The result is a very flexible controller that is capable of working with a group of VastPerson instances or a single instance. It receives three different messages and implements each of them internally, separating the complexity of that implementation from collaborating parts. If this implementation comes too close to "green haze", then the behavior implemented here can be split out into separate controller parts.

The simple nature of the example has no bearing on the scalability of its concept. It's just as easy to implement layers for policies, accounts, schedules, set tops, or any other domain object.



Conclusion

In the early days of object technology, we struggled with a new paradigm that was defined by concepts that were considered, at that time, abstract at best. We struggled because we lacked the tools to make those new concepts work for us. After a lot of hard work there are some highly productive tools in our hands but, as developers, we seem to forget why those tools were created. By not applying layering techniques to software development with VisualAge Smalltalk, we miss something fundamental. Call a hardware support representative with a problem and their first question will most likely be "Is it plugged in?" Maybe it's time to ask ourselves that same question about the way we use VisualAge Smalltalk.

References

- [1] Kyle Brown. "Remembrance of Things Past: Layered Architectures for Smalltalk Applications", The Smalltalk Report, 4:9, pp. 4-7, July-August 1995.
- [2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, pp. 4-6, Addison-Wesley, Reading, MA, 1995.
- [3] *ibid.* pp. 273-282.
- [4] *ibid.* pp. 293-303. ■

1999 VisualAge Generator User Group Meeting

The 1999 North American VisualAge Generator User Group Meeting has been announced. It will be held in the McColl Building of the Kenan-Flagler Business School on the campus of the University of North Carolina in Chapel Hill, NC. The dates of the meeting are June 15 & 16, 1999.

The two-day event begins at 9am EDT on Tuesday, June 15, and will conclude no later than 4pm EDT on Wednesday, June 16. Topics will include a variety of subjects of interest to current and prospective users of the product.

Speakers will include current customers, IBM VisualAge Generator Architects and Developers from the RTP, NC Development Lab, and IBM Business partners. A detailed agenda may be found on the VisualAge Generator web page at

www.software.ibm.com/ad/visgen.

There will be a \$50 fee charged to each attendee to cover the cost of meals and other expenses of the event. Travel and living expenses are the responsibility of the attendee. Enrollment details for the event may be found on the Web.

A special program number has been established with SATO Travel, an agency with which IBM does much of its travel business. This firm MAY be able to help attendees save money on airline tickets due to IBM corporate contracts with some of the airlines serving Raleigh-Durham International Airport. SATO can also help with hotel reservations at a number of hotels in the Chapel Hill area. SATO's telephone number and the program number for this event can be found in the event description at the web site mentioned above. Last year's meeting was judged a great success by those who attended! There were more than 47 companies represented by more than 110 people at the July, 1998, event. More are expected when the meeting commences on the third Tuesday in June this year. Those who come can expect to enjoy informative speaker presentations as well as opportunities to share information with fellow attendees.

Tuesday, June 15

- VisualAge Generator V4– Presentation & Demo
- VisualAge Generator Templates V4– Presentation & Demo
- State of California MDL Project– A Management Perspective (Customer Speaker)
- State of California MDL Project– A Developer's Perspective
- Meet the VisualAge Generator Development Team– VAGen Managers and Architects User Dinner

Wednesday, June 16

- VisualAge Generator User Group Business Customer Testimonial– Customer Speaker ENVY Tips & Techniques
- VisualAge Generator Assist– Presentation & Demo
- VisualAge Generator/VA for Java/ WebSphere– Your Path to the Web

While it is not on the agenda, one of the most valuable parts of the meeting is NETWORKING, NETWORKING, NETWORKING! Before the meeting starts each day, during the breaks, at

lunch, and over dinner, customers and prospective customers get the chance to talk with one another about their experiences with VAGen. Networking alone is reason enough to attend.

Registration

By going to the VAGen web page, anyone interested in attending the meeting may register. The web page is www.software.ibm.com/ad/visgen. Once there, you will find a link to information about the meeting. From that point, another click will take the reader to a registration form. That registration will come immediately to RTP. Our confirmation will include information about how the attendee might save money on both the airline fare and the hotel. The confirmation will be returned to the registrant within 48 hours of receipt in RTP. If you do not have access to the web, you can also register by calling Kaleigh Horn at 919-254-1788 or by sending her a note at khorn@us.ibm.com. Contact Rusty Edmister at 919-254-1706 for more information and assistance. Y'all come!

Visiting the VisualAge Generator Newsgroup

Setting up your web browser to access the VisualAge Generator newsgroup is pretty easy. If your browser is not already set up to use a manual proxy, contact your system administrator for configuration instructions before you get started.

If you are using Netscape(R) Communicator 4.5 to access the newsgroup:

1. Start your browser.
2. From the Edit menu, select Preferences.
3. Expand the Advanced category and select Proxies.
4. Select the Manual radio button, then select View.
5. When your manual proxy has been set, select OK on the View window and on the Preferences window.
6. From the Communicator menu, select Newsgroups.
7. From the Netscape Communications Services window, select File, then Subscribe.
8. On the All tab, select the Add Server button.

9. On the Add Server window, select News Server and click Continue.
10. In the Server field on the Communicator: Subscribe to Newsgroups window, type news.software.ibm.com and select OK. The new server appears in the Netscape Communications Services window and a collapsed list of newsgroups is displayed on the All tab in the Communicator: Subscribe to Newsgroups window. Depending on system speeds, this step may take a few minutes.
11. In the Newsgroup field on the Communicator: Subscribe to Newsgroups window, type ibm.software.vagen and select the Subscribe button.
12. Select OK.
13. In the Netscape Communications Services window, click on ibm.software.vagen.
14. In the Download headers window, select Download to view posts to the newsgroup.

If these instructions do not work for you, see your system administrator for additional information.

Integral Systems, Inc., Selects VisualAge Generator as Future Development Platform

by Mike Rhoads, Manager, VisualAge Generator, Smalltalk Sales and Rusty Edmister, VisualAge Generator Sales Support

Integral Systems, Inc., a long time user of IBM's 4GL product called Cross System Product, announced to its users that it plans to use VisualAge Generator as its development platform of choice for future software development. This announcement was made at its annual Users' Conference in San Diego, CA, in August, 1998, and restated in its Fall/Winter newsletter, INSIGHTS, last fall.

Cheryl Coleman, at the time Integral Vice President of Product Development and Professional Service, said in the newsletter article, "We're very pleased to join together with IBM and move forward with our future development plans. This will enable us to advance the product into the 21st century both technologically and functionally."

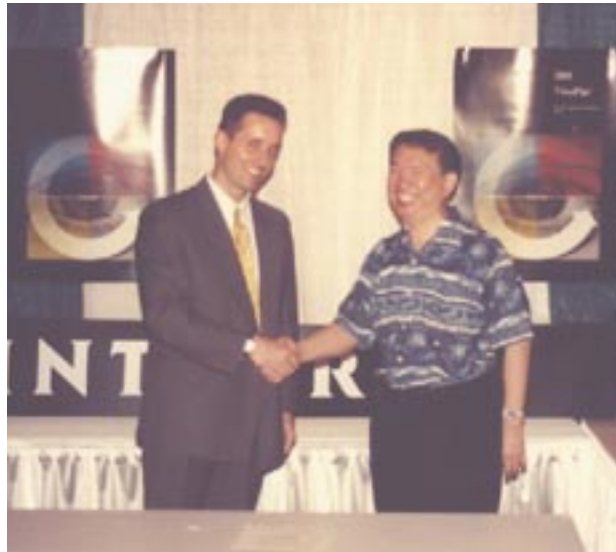
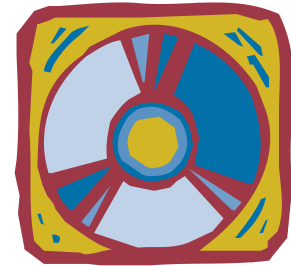
Integral's stated rollout strategy at the San Diego conference was to deliver a VAGen version of its product by mid-year 1999. Its next opportunity to release such a product would be 2Q-3Q, 2000. This first release would use a text interface only but would allow Integral and its customers to take advantage of VAGen graphical and browser interfaces and other product features in its future releases.

JAT Computer Consulting, Inc., is a business partner of both Integral and IBM and will be assisting Integral with the migration to VisualAge Generator. Mary Carr Sarracino, JAT western regional vice president, was quoted in *Insights* as saying, "We are in the process of training our technical staff in VAGen to assist CSP clients of both IBM and Integral with their migration from CSP to VAGen."

According to the newsletter article, Integral had been leaning toward VisualAge Generator for some time, "Other options abounded, but each lacked the comprehensive range of features and strong points provided by VAGen. Among Integral's criteria in making the decision to go with VAGen were:

- familiarity with IBM products, staff, and standards in the Integral user community
- the capability to support text, graphical and browser interfaces
- feedback from Integral customers

Additionally, the product supports VSAM and VSE. The future possibilities for our online system are infinite." ■



Mike Rhoads, VisualAge Generator Sales and Kelvin Kwok, then Integral's interim CEO

Diagnosing runtime errors using VisualAge Generator 3.1

by Jim Eberwein and Jim McVea, VisualAge Generator Support Team

Customers have recently reported walkbacks when executing their packaged images. These errors did not occur while their programs were running under the interactive test facility. Their WALKBACK.LOG files showed the error message: "AbtNLSCoordinator class does not understand defaultReplacementRules".

Due to a defect with VisualAge Generator 3.1, the proper runtime error messages are not always issued when a runtime error is detected. This defect will be fixed in fixpak 3 of VisualAge Generator when it is available.

For the short term, the support team has written this article to help you with problem source identification by documenting the runtime error messages and explaining how to decipher the runtime error in the walkback.log. Customers should be able to follow these instructions to resolve most of their runtime problems without opening a problem record with the IBM Support Center.

When a runtime error occurs, VisualAge writes a stack trace to the file WALKBACK.LOG. If VisualAge Generator did not issue the correct error message, the first entry in the file following the date/time stamp would be the message "AbtNLSCoordinator class does not understand defaultReplacementRules." To detect the actual runtime error, browse the file and search for the text string "CxRunMsg." Immediately following this string, you will see an integer that correlates to the runtime message that VisualAge Generator is trying to display. For example, in the case that a call to an executable failed, you should see the following text in the WALKBACK.LOG:

```
AbtMRI>>#hptRules receiver =  
[CxRunMsg.8 ] CxRunMsg is not a  
registered message group. temp1 = nil
```

To determine the runtime message associated with the integer value, refer to the following table:

1	The number of columns in table contents file %2 does not match the number of columns in table %1.
2	Record this information and contact your system administrator.
3	Could not call a remote program because the client/server communications support is not installed correctly.
4	Client/server communications error %1 was detected: %2
5	Could not abort the transaction in GUI client %1.
6	Client initialization for GUI client %1 failed.
7	%1 A value stored in the EZE data word is not valid. %2 cannot hold %3.
8	A call to %1 failed because the executable could not be found.
9	Service request from GUI application %1 to remote client %2 failed.
10	Transaction initiation for GUI client %1 failed.
11	An internal error occurred while calling %1 in DLL %2. The error code is %3.
12	One or more columns in table contents file %2 do not match the columns in table %1.
13	An internal error related to argument %4 occurred while calling %1 in DLL %2. The error code is %3.
14	%1 %2 contains invalid numeric data.
15	%1 User variable overflow occurred. %2 cannot hold %3.
16	An error occurred in remote application %1 date %2 time %3 error %4.
17	Subscript value %1 is not valid for data item %2.
19	An error occurred while calling %1 in DLL %2. The error code is %3.
20	Could not commit transaction in GUI client %1.
21	%1 Maximum value overflow occurred.
22	The table contents file named %2 for table %1 could not be opened. The error code is %3.
23	Client finalization for GUI client %1 failed.
24	%1 can not run because its VAGen runtime code has not been generated.

*18 is not a registered message. This is not a misprint and was omitted intentionally.

If the message contains placeholders (indicated by a '%' sign), then the value for the placeholder(s) can also be located within the walkback. The method preceding the AbtMRI>>#hptRules method (either AbtMRI>>hptAsAnnotatedStringReplace: or AbtMRI>>hptAsStringReplace:) will have one argument which is indicated by "arg1 =" in the walkback.

The value for arg1 will contain the values for the placeholders in the error message in the table above. To determine what the complete text of the error message should be, simply replace the value(s) in arg1 for each corresponding placeholder.

From the “missing executable” example:

```
AbtMRI>>#hptRules
receiver = [CxRunMsg.8 ] CxRunMsg is not a registered message group.
templ = nil

AbtMRI>>#hptAsStringReplace:
receiver = [CxRunMsg.8 ] CxRunMsg is not a registered message group.
arg1 = ('MYDLL')
```

Using the information above from the walkback, we can determine that our error message is: “A call to MYDLL failed because the executable could not be found.”

For users who feel comfortable with filing in Smalltalk code, the following code will fix the current defect:

```
!AbtMRI publicMethods !

hptRules
  “Broken out from the rules calculation code in #displayButtonType:replace:.”

  | grp |
  (grp := self groupInstance) == nil
    ifTrue: [AbtNLSCoordinator current defaultReplacementRules]
    ifFalse: [grp replacementRules]! !
```

In order to file in the fix, you will need to type the text above into a new workspace, select all of the text, then choose File In... from the Edit menu. After filing in the fix, you will be prompted to make a scratch edition of the HptBaseMessageSupport application. Click OK and be sure to save your image. ■

Using FCWERRA with VisualAge Generator V3.1 C++ programs on CICS

by Chuck Proffer, VisualAge Generator Development

VisualAge Generator Server for Windows NT and AIX ships a sample error transaction, FCWERRA, for the CICS environments. FCWERRA displays a screen whenever a VisualAge Generator program terminates showing the most recent runtime errors. To activate FCWERRA, the following steps need to be performed by the CICS administrator.

Place the following line in the CICS environment file (*/var/cics_regions/RegionName/environment*):
FCWOPT=2 Run script fcwcicsinstall.

It is located in directory */usr/lpp/vgwg31/bin* on AIX and directory *x:\vgserver\exe* on Windows NT. This script will add the necessary CICS definitions for FCWERRA as well as other VisualAge Generator utility programs. Make sure that the CICSREGION environment variable contains the name of the region where you want to activate FCWERRA. Restart the CICS region.

FCWERRA is a VisualAge Generator program. The ESF for FCWERRA is located in directory */usr/lpp/vgwg31/samples* on AIX and directory *x:\vgserver* on Windows NT. Minor customization changes can be made to the maps; however, the input parameter list should not be modified. ■

Acronyms

3GL	third-generation language
4GL	fourth-generation language
AIX	Advanced Interactive Executive
API	Application Programming Interface
AS/400	Application System/400
CAE/2	Client Application Enabler/2
CASE	Computer-aided Software Engineering
CICS	Customer Information Control System
CICS OS2	Customer Information Control System Operating System/2
CPU	central processing unit
CSP	Cross System Product
DB2	Database 2
DBCS	double-byte character set
DBMS	database management system
DCE	distributed computing environment
DRDA	distributed relational database architecture
EMEA	Europe/Middle East/Africa
GUI	graphical user interface
IBM	International Business Machines
IMS	Information Management System
LAN	Local Area Network
MSL	member specifications library
MVS	Multiple Virtual Storage
NT	Microsoft Windows NT
OS/2	Operating System/2
OS/390	Operating System/390
OS/400	Operating System/400
RAD	rapid application development
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
VM	Virtual Machine
VSE	Virtual Storage Extended
WWW	World Wide Web

VA Generator Assist(tm) Provides VisualAge® Generator Users with Powerful ENVY Management and GUI Building Capabilities

By Eric Clayberg, Vice President of Product Development, Instantiations, Inc.

Instantiations, Inc. is very pleased to announce the release of VA Generator Assist™.

Gen Assist is a new product that adds major new capabilities to IBM's popular VisualAge® Generator for Windows and OS/2. Gen Assist provides an extensive set of new power tools for ENVY management, GUI building, and code development. Gen Assist integrates seamlessly into the familiar VisualAge Composition Editor, adding hundreds of new features and commands.

"We are delighted that Instantiations, Inc. has developed VA Generator Assist," said Hayden Lindsey, IBM STSM & Product Manager, VisualAge Generator. "From the powerful ENVY

administration aides to the fastpath GUI building capabilities, I have found Gen Assist to be a valuable add-on to VA Generator. Check it out!"

Some of the ENVY management features include:

- Powerful change propagation tools—modify an app and update all affected maps in one step.
- Advanced Configuration Map tools—release latest applications and required maps, identify maps needing to be updated, etc.
- Recursive application, subapplication and class versioning—version an entire application in a single step.
- Prerequisite and dependent path determination—find out why an application is listed as a prerequisite or dependent.
- Super user features—manipulate editions and versions regardless of ownership.
- Version renaming commands—baseline app names with ease!
- Browse Changes Including Required Maps—browse the changes between any two config map editions including any changes in their required maps.
- String Search—find arbitrary strings within classes or applications.

Some of the GUI building features include:

- Attribute tools—apply attribute changes to multiple widgets simultaneously.
- Attachment styles—create complex layouts with ease.
- Link filtering—concentrate on just the event or attribute links you are interested in.

Some of the code development features include:

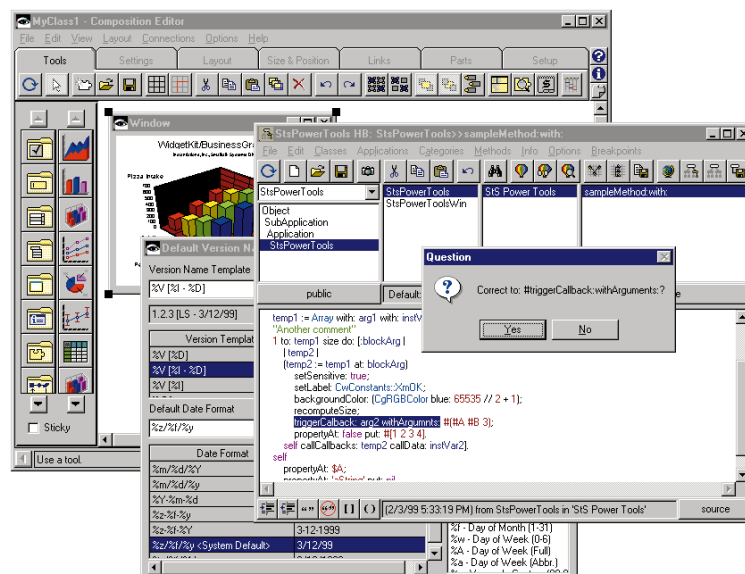
- Dynamic & Batch-oriented spell checking—catch spelling errors and common typos instantly.
- Fully customizable color syntax highlighting.
- Customizable toolbars for all code browsers.
- Advanced editing features—comment/uncomment, indent/unindent, auto indent, insert matching brackets, etc.
- Customizable Application and Configuration filtering—hide the apps and configs you aren't interested in.

In summary, Gen Assist lets new and experienced Generator developers easily harness the new power that ENVY and Smalltalk bring to the environment.

Users will find that Gen Assist encourages them to use a much broader spectrum of Generator's capabilities, increasing their productivity and ease of use.

Instantiations is a premier member of IBM's Object Connection Program and has been providing products and services to VisualAge customers for years. Thousands of developers are currently using Instantiations' VisualAge productivity products.

You can download a free, fully functional 30-day evaluation copy of Gen Assist at Instantiations' Smalltalk Systems web site (<http://www.smalltalksystems.com>). For more information, contact Instantiations at 800-808-3737.





The VisualAge Generator Newsletter

This newsletter is published by the IBM Software Solutions Division, Research Triangle Park Development Laboratory. Letters to the editor are welcome. Please address correspondence to:

The VisualAge Generator Newsletter

Managing Editor
IBM Corporation
Dept. TF6B/062
P.O. Box 12195
3039 Cornwallis Road
RTP, NC 27709-2195
USA
FAX: (919) 254-0206

© Copyright International Business Machines Corporation 1998. All rights reserved. Printed in U.S.A.

The following terms used in this publication are trademarks or service marks of the IBM Corporation in the United States or other countries or both: AIX, AS/400, CICS, CICS OS2, COBOL, Database 2, DataJoiner, DB2, DB2/2, DB2/400, DB2/6000, IBM, IMS, LE/370, MQSeries, MVS, VM, VSE, Operating System/2, OS/2, OS/390, OS/400, RISC System/6000, SQL/DS, VisualAge, and VisualGen.

The following terms and phrases used in this publication are trademarks or service marks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U. S. and other countries.

Texaco is a trademark of Texaco, Inc.

Informix is a trademark of the Informix Corporation.

Oracle is a trademark of Oracle Corporation.

Nikon is a trademark of Nikon Corporation.

ENVY is a trademark of Object Technology International, Inc.

HP is a trademark of Hewlett-Packard Company.

Microsoft, Windows, Windows NT, the Windows 95 logo, Visual Basic, and ActiveX are trademarks or registered trademarks of Microsoft Corporation.

Other company, product, and service names may be trademarks or service marks of others.

IBM has made reasonable efforts to ensure the accuracy of the information contained in this publication. However, this publication is presented "as is" and IBM makes no warranties of any kind with respect to the contents hereof, the products listed herein, or the completeness or accuracy of this publication. Customer experiences may be different from those described here. IBM does not warrant any non-IBM programs or products which are described in this newsletter. These articles are for information only, and you should contact the stated company with your questions.

G242-0315-11

