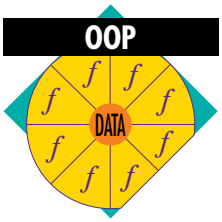


CommonPoint Frameworks Take the Spotlight

By Joyce Uggla

Since its founding in 1992, Taligent, Inc., has been an avid evangelist of object-oriented frameworks. That is because this independent software company—owned by Apple Computer, Inc., Hewlett-Packard Company, and IBM Corporation—is betting its future on this technology by building an object-oriented-based product that promises to deliver dramatic developer productivity and innovation, distributed enterprise application support, and a superior user experience.

Taligent's CommonPoint™ application system is operating system independent and supports the creation and deployment of distributed applications. Application systems combine the common functions (such as text editing, graphics, multimedia, and compound document features) built into today's monolithic applications with common operating system functions (like networking, communications, and data access) to provide developers with a portable foundation to build distributed applications and software components.

"Application systems are an important emerging software technology that will help to re-ignite application innovation," said David Cearley, vice president and director of Workgroup Computing Strategies at META Group™, Inc., a market research firm in Stamford, Connecticut. "Application systems eliminate the redundancies in today's monolithic applications and operating system platforms and provide application portability across operating systems. Taligent's CommonPoint architecture is well positioned to compete in this market."

Earlier in 1995, Taligent shipped a final beta reference release of the CommonPoint system to its investors—Apple, HP, and IBM—and to its early developers. Since then, the company has extended its beta program to a new set of developers and is moving closer to its target ship date of mid-1995.

"Taligent's goal is to establish the CommonPoint application system as the object-oriented standard for the next generation of enterprise applications," said Joe Guglielmi, chairman and CEO of Taligent. "Delivery of the beta reference release is an important milestone in accomplishing this goal because it allows our investors and early partners to continue their development efforts with a feature-complete version of CommonPoint. We plan to move forward aggressively from here to deliver the final reference release this summer."

Why an Application System?

"Application system" is a new term for a class of software that focuses on application development independent of underlying hardware or software. It represents the intersection of several converging trends:

- **Increasing abstraction**. The trend is moving programmer attention upward from hardware details—from disk sectors to files to databases; from memory addresses to variables, arrays, and structures to objects. By working with entities that map one-for-one with elements of the problem, developers become more effective.

Joyce Uggla

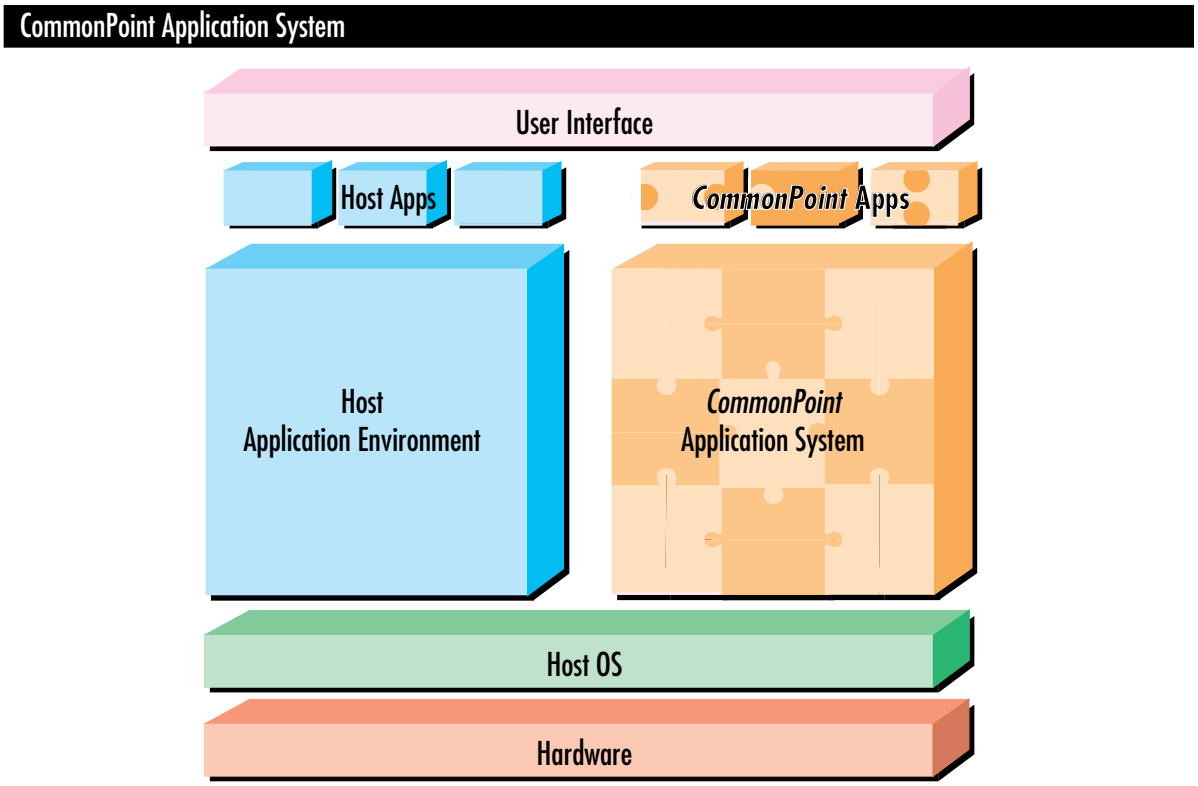


Figure 1. CommonPoint application system

- **Platform independence**. Today UNIX is a cross-platform operating system. SQL is a common query language for relational databases from disparate vendors. Open Database Connectivity (ODBC) is a vendor-neutral database programming interface.

  Platform independence provides rapid porting of applications and support for applications distributed across heterogeneous platforms and networks.

- **Opening the system**. Plug-ins are available for PhotoShop, Excel, and other applications. Open systems allow competition in replaceable parts and innovation in both general-purpose and very focused areas, with special-purpose extensions able to satisfy very specific needs.

- **Accommodating the user**. Graphical window system interfaces initiated a new era in user accommodation. The computer pictured items from the users' world, such as trash cans and folders, and allowed users to manipulate them in familiar ways.

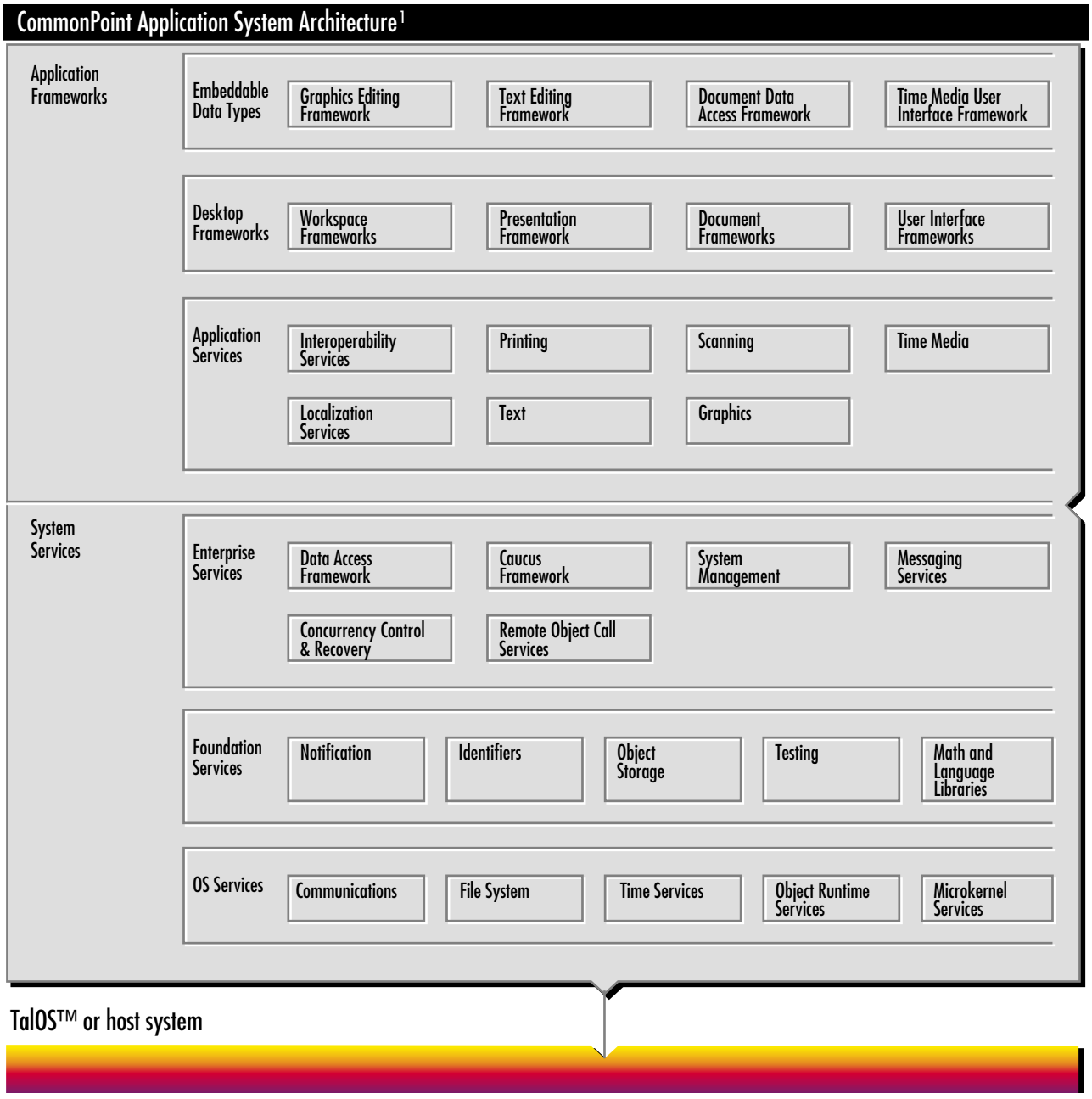
  Computers are getting easier to use because developers are understanding that successful products accommodate the user rather than the reverse.

Application systems bring these trends together and provide developers with high-level, platform-independent abstractions that cover all common application functions. An application system defines a user model and provides a user interface that embodies the model. The best user models let the user work in the user's way, rather than requiring the learning of special computer techniques.

A comprehensive application system is rich in functionality, but maintains consistency both internally and with third-party and application code.

Although other products have elements of an application system, the CommonPoint product is the first designed specifically as an application system (shown in Figure 1). It consists of a rich set of extensible frameworks, utilities, and an innovative enterprise graphical user interface.

Specifically, more than 100 object-oriented frameworks (700,000 lines of C++ code, and just under 2,000 classes) make up the CommonPoint system, which provides rich functionality for compound documents, collaboration, multimedia,



[1]Features and functions shown are subject to change by Taligent without notice. Some functions are not in the first release.

Figure 2. Architecture for CommonPoint Application System

2-D and 3-D graphics, data access, communications, and distributed objects. That means developers can leverage this functionality, so they do not have to build it into each application.

Initially, the CommonPoint application will run on top of existing 32-bit operating systems such as AIX, OS/2, HP-UX®, and a future version of Mac™ OS. Taligent also has plans to offer a version of the CommonPoint system for Windows NT and Windows '95.

CommonPoint Architecture

The CommonPoint application system provides the developer with both breadth and depth of programming functionality. Figure 2 offers an overview of the key facilities available in the system.

At its highest level, the CommonPoint application system can be thought of as providing two distinct sets of services: a set of application frameworks that can be used to create powerful,

interactive applications; and a set of system services that can be used to manipulate data, communicate with other computers, and interface to the underlying operating system.

The application frameworks include the following:

- **Embeddable data types** support viewing and editing of complex data types.
- **Desktop frameworks** provide support for the CommonPoint application model, including its user interface policy, its "look and feel," and its compound document architecture.
- **Application services** support media and data handling services needed to create industrial-strength interactive applications.

The depth of the system can be illustrated by the text frameworks, which include frameworks for line layout, paragraph styles, text styles, text and style storage management, and character sets. The graphics frameworks include 2-D and 3-D graphics, colors, font support, sprites, pixel buffers, graphic devices, and displays.

The system services includes the following components:

- **Enterprise services** is a set of services that allow the CommonPoint application system to interoperate with other computers distributed within an enterprise.
- **Foundation services** is a fundamental set of object services that make it easier to write object-oriented programs.
- **OS services** provide basic support for creating programs that work across a wide variety of host operating systems and hardware platforms.

Taligent's early developers are already recognizing the benefits of the system. "CommonPoint is our product strategy choice because it enables us to build a new class of collaborative applications," said Stephan Adams, president of Adamation, Inc., a developer of commercial and custom enterprise applications based upon object-oriented platforms. "This functionality makes CommonPoint distinctive in the marketplace, thus providing unique value to customers and new opportunities for developers. The collective frameworks allow us to concentrate on the components of our applications rather than the low-level functions."

Developer Productivity with Frameworks

Frameworks and systems based on frameworks, such as the Taligent environment, empower developers to fully realize the potential of improved design and code reuse, including reduced development requirements, reduced maintenance, and higher reliability.

What are Frameworks?

A *framework* is a set of cooperating classes that constitute a generic design solution that can be adapted to a variety of specific problems within a given domain.

In the same way that classes are aggregates of functions and data, frameworks are aggregates of classes—but frameworks are not mere collections of classes. Frameworks are architectural; they provide infrastructure, which reduces the amount of code that you must write and thereby alleviates many productivity problems. Frameworks represent the next level of abstraction beyond class libraries, just as classes represent the next level of abstraction beyond functions and data.

Frameworks and systems based on frameworks empower developers to fully realize the potential of improved design and code reuse.

Capturing Expertise

A framework represents a generic design solution. It is a meta-solution that represents the set of all possible solutions within a particular problem domain, not any one solution.

A framework abstracts the essential entities, state, and behavior in its problem domain. It provides key mechanisms, defines the interaction protocols for key scenarios, and encapsulates and enforces fundamental invariants. It has strong "wired-in" connections among its objects that capture design decisions common to its problem domain.

In this manner, a framework embodies the domain expertise of the designer of the framework—it captures the programming expertise necessary to solve a particular kind of problem. See Figure 3.

Using Frameworks

A framework dictates the architecture of applications based on it. It defines an application's overall structure, its partitioning into classes and objects, the key responsibilities, how the classes and objects collaborate, and the flow of control. A framework also defines and enforces the responsibilities of a developer who wants to make use of the framework, as well as the degrees of freedom available to a developer who wants to customize the framework.

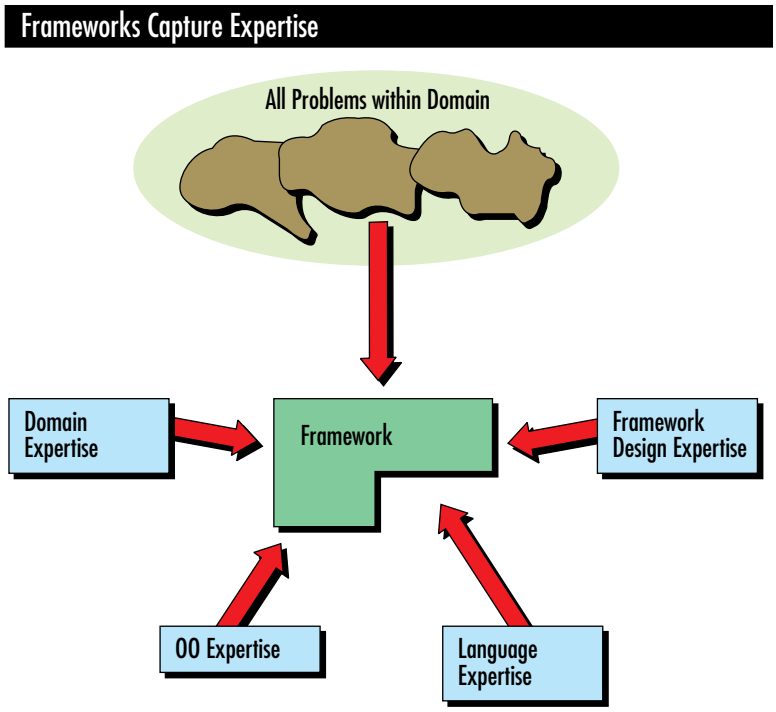


Figure 3. Frameworks capture expertise

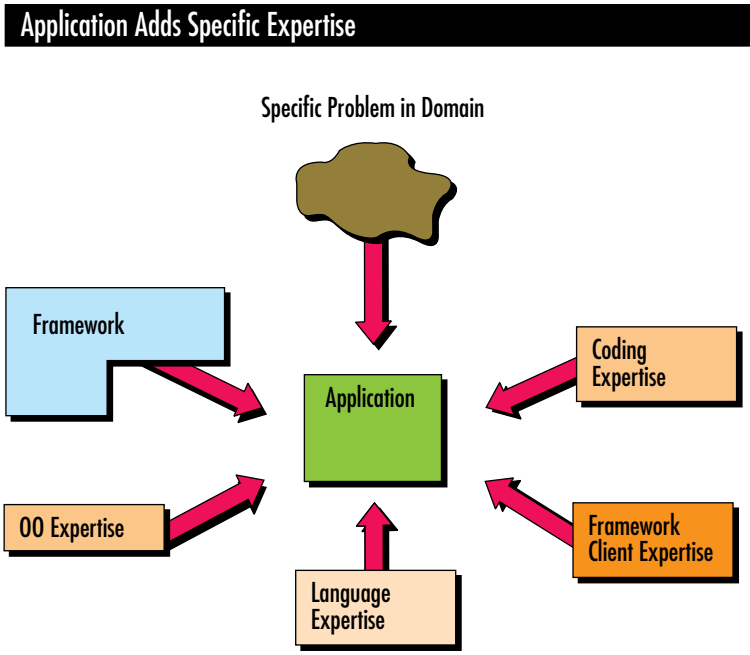


Figure 4. Application adds specific expertise

Working within the constraints imposed by the framework, you supply code to tailor the framework to solve your particular problem. In this manner, you turn the abstract solution represented by the framework into a concrete instance of an application, as shown in Figure 4.

Note that while you need to know how to use the framework in order to solve the problem, you do not need to be a domain expert in order to solve the problem. This is because, through use of the framework, you reuse the design captured by the framework. In effect, you inherit the domain expertise and problem-solving ability of the designer of the framework. A framework is a form of expert system.

Multiple Frameworks

So far, we have discussed applications and frameworks in terms of a one-to-one relationship: one application per framework, one framework per application. In practice, an application typically is implemented using multiple frameworks, shown in Figure 5.

When you require multiple frameworks for an application, the application consists of code for each framework.

Kinds of Framework Use

Frameworks are very flexible programming constructs, more so than procedural or class libraries. You can use frameworks in three different and complementary ways:

- **Use as is.** Use the framework as is, like a specialized class library. Some frameworks provide sufficient default behavior that they can be used out-of-the-box. (This is the simplest kind of use and does not preclude more sophisticated uses of the framework—it just means that the more sophisticated kinds of use are optional, rather than required.)
- **Complete**. Add code to the framework to implement specific capabilities. It is important to keep in mind that a framework represents a generic design solution, not any one solution. A framework does not have to exhibit complete default behavior—it may be only partially filled in. And a framework might not even be able to execute as delivered (it might be abstract, requiring developer-supplied code to make it concrete).
- **Customize**. Replace parts of the framework implementation. This is the most sophisticated and radical way to use a framework. You can

replace a little or a lot, or even the entire implementation. You can even implement some or all of the code in hardware (such as a graphics accelerator). In all cases the same interface to the framework is preserved, and programs that ask the framework for services do not need to know about the changes that you have made to the framework's underlying implementation.

Benefits of Frameworks

Frameworks return a number of benefits to developers. Some of these benefits are simply attributable to the fact that frameworks support design and code reuse; these benefits are also present when using well-designed class libraries, although to a lesser extent. Other benefits are unique to frameworks, and are advantages that frameworks have over both procedural and class libraries:

- **Less code to design and implement**. Much of the program's design and structure, as well as its code, already exists in the framework. By providing the infrastructure, the framework dramatically decreases the amount of standard code that you must design, code, test, and debug. You write only the code that extends or specifies the framework behavior to suit the program's requirements.

  Because the infrastructure of the framework is already in place, you write code only as required by the framework or to override some default behavior of the framework that is inappropriate for the application (this is sometimes called *programming by differences*). Typically, the amount of code required for an implementation is a small fraction of the code required to write the same application from scratch. This provides a corresponding decrease in the effort, time, and cost required to implement the functionality.

- **More focus on areas of expertise**. You can concentrate on your particular problem. You do not have to be an expert at writing user interfaces, networking, or printing to use the frameworks that provide those services. You can focus on your true value-added area. Just as standard programming interfaces insulate software routines from system dependencies and standard utilities facilitate development, frameworks provide standard solutions. This frees developers who are not necessarily experts in a certain area from the complexity of the underlying details. In this manner, frameworks create an environment in which solving domain problems—not programming problems—is possible. All you need to know (at least initially) is how to use the basic parts of the framework to get the job done.

Multiple Frameworks Used in Applications

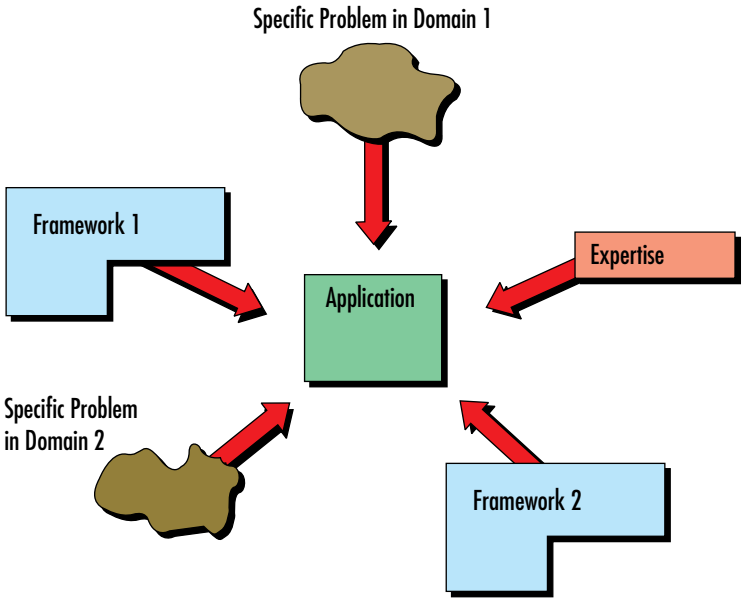


Figure 5. Applications can use multiple frameworks

- **Proliferation of expertise**. Good software design in a particular area requires domain knowledge that you typically acquire only by experience. Corporate and commercial development organizations as well as systems integrators have this acquired experience in particular areas, such as manufacturing, accounting, insurance, or financial applications. Frameworks allow organizations to package the common characteristics of their expertise.

  Frameworks and the embodied expertise behind the frameworks are a strategic asset for internal use and for those organizations that see business opportunities for reselling specialized knowledge.

  For example, frameworks give systems integration companies with expertise in vertical markets a distribution mechanism for packaging, reselling, and deploying their expertise.

- **More reliable and robust code**. Code reused through frameworks has already been tested and integrated with the rest of the framework (and with the system as a whole). This allows an organization to build from a base that has been proven to work in the past, and minimizes the amount of testing required.
- **Improved consistency**. Because frameworks embody expertise, you solve problems once—when first creating the framework—and you use the business rules and designs you capture in a framework consistently across all problems in the framework's domain.

  Additionally, frameworks enforce the relationships among the objects and classes in the framework, providing a higher degree of consistency than is obtained with either procedural or class libraries.

Frameworks provide a mechanism for reliably extending functionality.

- **Improved integration and interoperability**. Because all programs based on a particular framework reuse not only common code but also common design, developers who are working on similar programs have a better chance of understanding and working with each other's code, and the programs they develop are more likely to work together consistently and reliably. Because standard behaviors are captured in the framework, there is consistent behavior from product to product.

  Frameworks support a high degree of integration among multiple customizations. Much as individual objects hide their internal complexity and present a simplified interface for use by other objects, many different programs can use frameworks simultaneously in a way that allows the programs to share common behavior without interfering with each other's specialized implementations. This is possible because the client API of the framework remains unchanged.
- **Reduced maintenance overhead**. Fixing a bug once in a framework fixes the same bug in all the software derived from that framework. Similarly, when you add new features or capabilities to a framework, the additions automatically appear and work in all applications based on the framework. And because you make the changes in only one place, you minimize the chance of introducing errors in the code.

  Maintenance is far easier, because you amortize maintenance of a framework over many implementations. Because the implementation adds or changes only the things that are unique to the particular implementation, there is less new code.

  Also, quality of the final product is higher. Because you constantly reuse the framework, it becomes very robust. Thus a new product contains significant amounts of mature code in the framework, plus a smaller amount of new code in the implementation, resulting in higher overall quality.
- **Orderly program evolution**. Frameworks provide a mechanism for reliably extending functionality. While objects and class libraries provide interfaces for extending functionality at a fine-grained level, frameworks provide this flexibility at a higher level. In this manner, you can develop applications by using the framework as a starting point and writing smaller amounts of code to modify or extend the framework's behavior. You can add these extensions without sacrificing compatibility and interoperability because the interfaces are well-defined.
- **Enculturation**. Developers who are used to dealing with frameworks tend to think in terms of generic solutions rather than special solutions, with the result that you can often design your own programs as frameworks that you or others can reuse.

Taligent, along with many other leading industry players, foresees object technology as a way to change both the development and use of software. Taligent's approach, hosting the CommonPoint application system on existing 32-bit operating systems, enables users and developers to make the change incrementally, at a pace that preserves the value of current information technology investments and fosters new innovative solutions.

Software developers will be able to focus on what they do best: build solutions that address real-world business problems and issues. End users will become more task-focused, with intuitive applications that enable them to use the computer as a powerful tool in accomplishing their work. And groups of people will be able to effectively collaborate on business-critical projects and problem solving.

Taligent's vision and technology will provide a common base for applications, a common platform for heterogeneous computing, and a common environment for people to work together—in essence, a common point for a new generation of computing.

For more detailed information on system features and capabilities, look for two new books coming early this summer from Addison-Wesley:

- *Inside Taligent Technology* describes what Taligent technology can do, why it is important, and where the people who created it believe it is going. It is written by Sean Cotter, a freelance writer, and Mike Potel, vice president of Technology Development at Taligent.
- *The Power of Frameworks* illustrates the value of frameworks in application development. Written by Taligent, it includes a demo CD.

You can also browse the Taligent web page: http://www.taligent.com.

Joyce Uggla, Taligent, Inc., 10201 North De Anza Boulevard, Cupertino, CA 95014-2233. 1-800-288-5545. Ms. Uggla is staff technical editor at Taligent. She has a Master's degree in Education from Stanford University, and has been writing and editing technical information in the Silicon Valley for 15 years.

TECHNICAL SUPPORT FOR SOLUTION DEVELOPERS

Solution Developer Operations (SDO)—a recently announced unit at IBM—wants the latest level of solution developer's products to run at optimum performance on the latest IBM platforms, exploiting the power and capabilities of those platforms. And that requires support.

SDO offers technical support to all worldwide solution developers working on OS/2 and AIX to help them exploit IBM technology while developing on or porting to IBM platforms. The OS/2 Developer Assistance Program offers technical support to OS/2 developers, while the POWER Team offers technical support for AIX developers.

Mass communications provide a vital part of the technical support. IBM and solution developers are leveraging the possibilities inherent in Internet and commercial online services through white papers and documents on migrating solutions, code fixes, plus assistance with questions during development via forums and bulletin boards. Information and support via Internet and commercial services are available to any solution developer—no defined relationship with IBM is required.

SDO wants to go beyond the generalized support delivered via Internet and commercial services; they want defined relationships with worldwide solutions developers that will promote success for everyone. For AIX developers, the POWER Team provides programs that will accelerate developing on and porting to AIX while exploiting the latest technology, such as Symmetric Multiprocessing (SMP)

As a member of the POWER Team, you can use these services:

- Ask directed questions of experts who specialize in supporting developers
- Request customized assistance in determining how to exploit new IBM technology
- Participate in beta programs
- Use Developer's Discount Program to purchase systems for use in development

To join the POWER Team, call us at 1-800-222-2363. If you are considering developing on or porting to OS/2, SDO can help. Call 407-982-6408 for more information. (See "IBM Marketing and Technical Services" in this issue for more information about support for solution developers.)

—John Falvey, IBM Corporation