

The Object Process Graph

Abstract

GraphLogic's core technology, the Object Process Graph (OPG), is a general purpose executable graph that incorporates every aspect of an application, including process, user interface, and database. No programming language, tool, or database is required to handle any part of an application, no matter how large or complex. A complete high-level visual programming environment is used to define an OPG. No code is generated; the graph is the code. Object Process Graphs interface with traditional programming languages and databases through industry standard protocols.

The fusion of formerly disparate application aspects compresses the development cycle and results in a substantial increase in developer productivity when compared to traditional development environments. The dynamic graph structure allows developers to easily construct and modify program logic, database functionality, and user interfaces even while a program is running. The real time development platform allows the design stage of the application to proceed concurrently alongside requirements gathering and testing. Sophisticated security and change control features are an intrinsic part of the platform, allowing lock down and change control of all production environments. Users with relatively little computer training can learn to create complex custom applications with the system.

This paper begins with an overview of the current state of software development and the need for a new paradigm. The OPG paradigm is introduced, with discussion of the technical aspects of how systems are designed using OPG methodology. The paper then discusses the practical application of the OPG and is followed by a brief conclusion.

Overview

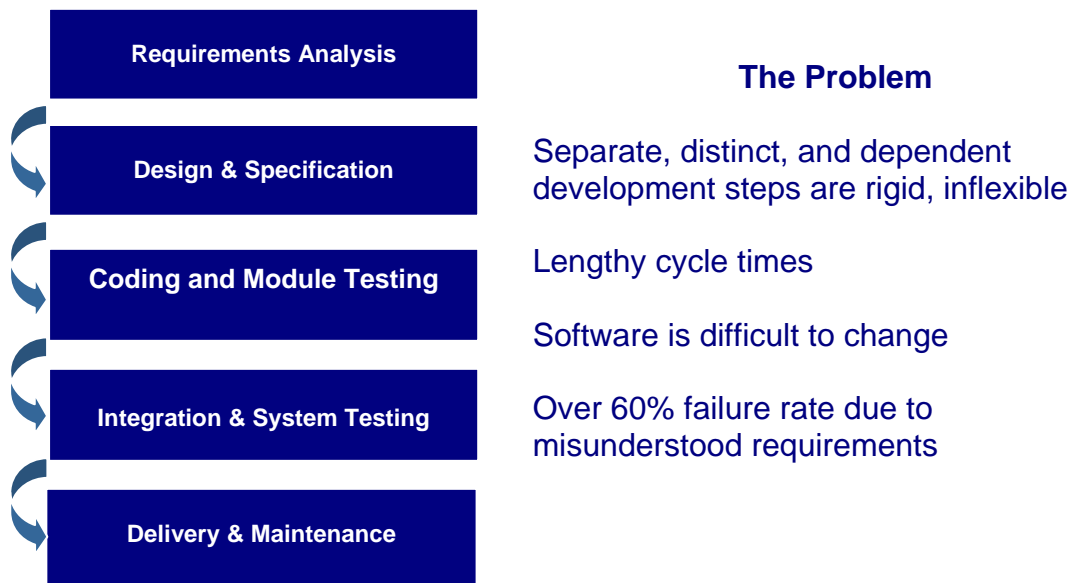
In the past 50 years the computing industry has witnessed a billion-fold increase in available computing power. Yet the fundamental paradigm for creating computer software has remained unchanged. The vast majority of today's software is still created using traditional programming languages that interact with a data repository. Tools have been developed that make this task easier, but software development is still a slow, labor intensive process, prone to error and replete with unpredictable results. Software engineers have searched for a better, more efficient way to develop complex systems; but the code dependency inherent in traditional system development methodologies has hampered meaningful progress.

The concept of building complex applications without programming has been the holy grail of software development. Although many have tried to create

development tools and environments that promise to reduce or eliminate hand coded application systems, Object Process Graph technology is a radically disruptive shift from all other attempts. The OPG is a revolutionary new way to create computer software systems that fundamentally alters the development process and introduces new ideas – thus shattering the dominant software paradigm. Existing technologies all rely on a sequentially based programming language interacting with a data repository. In contrast, the OPG completely synthesizes data and program information. Both program logic and data are stored within the OPG in a very similar form, as nodes and edges. No traditional programming languages or databases are used to describe any part of an application. The program is the data; the data is the program. No code is generated in the process.

The Problem

When using traditional development methods, software passes through five distinct phases when it is placed in a live production environment. These five steps can be understood as follows:



Traditional software languages such as FORTRAN, COBOL, C, Java, and others are used in the coding phase and require highly skilled programmers working with lines of software code and commercial databases. At some point during this process, development stops and the code is frozen. Testing begins, and then the software undergoes a lengthy debugging process. As developers uncover coding errors, they proceed to rewrite large chunks of adjacent code to make everything work together.

At some point, testing is deemed complete and once more the code is frozen. As the code is readied for hardware installation the user training process begins. In

complex or dynamic environments, users often notice inconsistencies with mission critical processes that are dictated or constrained by the code. Occasionally the software fails because change shifted the ground of the environment during the creation process. The best case scenario for the hardened code is that it will remain operational up to a point where the changing environment ultimately necessitates the need for an entirely new system. At this point, the cycle begins again. Object Process Graph powered systems avoid these problems, because they go from design to testing in one seamless step.

The structural rigidity inherent in traditional software development methodologies has made the process of developing complex software systems difficult, slow, and fraught with error. End users must specify their requirements early on, often during the planning stage of a project. In many cases, the users responsible for generating the project requirements specifications document lack a clear understanding of how a software system could help solve their business needs. A second problem exists because each of the individual development steps behaves like a single link in a link-dependent chain. Each subsequent link builds on the prior link; when a prior link fails the entire system is at risk. Since the most critical of the development steps, requirements gathering, occurs at the beginning of the dependent chain, subsequent changes in system requirements threaten to compromise, or render useless, the entire project. The Standish Group has estimated that over 60% of system development projects fail – either because customers do not get what they need or thought they were getting or the requirements changed in the meantime.

The OPG Paradigm Solution

The OPG shatters the old paradigm through the fusion of program, data, and model. No longer must software development occur within the confines of distinct, separate, time consuming link-dependant phases because OPG based systems perform all the steps necessary for system design in parallel.

GraphLogic has solved the persistent problem of rigid, link-dependent development environments by removing some steps entirely, by compressing other steps, and by allowing the remaining steps to proceed concurrently. The result is a substantially depressed development cycle that also allows for real time, on-the-fly changes to the emerging software system. Application development can now keep pace with the changing needs of end users. The following chart illustrates the OPG systems life cycle.



Development steps completed in parallel
Compressed development cycle
Real-time, on-the-fly changes

Nodes and Classes as Fundamental Components

Over the last decade, object oriented technology has emerged as the dominant software paradigm. With object oriented technology, the basic component of most development systems is a class or object. Graphical views of the application within these systems revolve around class or object diagrams of the application.

The fundamental components of an OPG, however, are nodes and edges, not classes and objects. All applications are built from these fundamental components, not from classes. The use of nodes and edges results in an internal structure that closely resembles the high level design view provided by the graphical editor. Little translation is needed to convert the internal representation of the system into a high level external representation; one that is easily comprehensible by an end user.

At its core, an Object Process Graph system is complete with the following functionality: it interprets, accepts, and processes data; produces information output; allows real time modifications and saving; and deploys graphs and an interpreter.

The OPG architecture fuses the elements of an application into a novel, dynamic, graph structure. This dynamic graph structure makes it possible to easily construct and modify program logic, database functionality, and user interfaces even while a program is running. This new technology allows for extremely short development cycles and end users can create complete and immediately usable custom applications dynamically – on the fly – with little or no assistance from professional programmers.

Synthesis of Program and Data

To date, persistent data and program code have been treated separately and very differently in all general purpose computing architectures. Object oriented methodologies synthesize transient data and program code; however, program

code and persistent data still remain very distinct entities – the Java code of an object method is separate from the attribute values of a persistent object instance. These differences are maintained throughout all general purpose software architectures; persistent data is stored in a relational or object oriented database and the program, such as Java or C++ code, is typically stored in an object or binary file. Although some database systems allow program code to be stored in a database, these systems store the data and program code in separate structures and in very different forms.

Some very specialized architectures have been developed that do achieve a complete synthesis of data and program; neural networks are one example. However, these have very specialized uses and no one would consider building a complete web based on line ordering system exclusively using neural networks.

The OPG is the first general purpose computing architecture that completely synthesizes data and program. An OPG is a type of graph object oriented database that stores all the process and computational logic, display properties, metadata, and instance data of an application. Both program logic and data are stored in the OPG as nodes and edges in a very similar form. No binary or object files are required to run an OPG system.

The Absence of Code

Existing technologies rely on automated code generation to translate a visual, graphical high level design view of an application into an executable application and rely on the developer's ability to modify the generated code to address application requirements that could not be defined in the high level design; for which code could not be generated. Object Process Graph development platforms, however, never generate any code at any point in the development process. A high level graphical interface is used to directly edit the OPG, and this interface can be used at any point in the development process, including during post-production maintenance. The graphical view of the OPG provided by this interface always provides an accurate representation of the entire current state of the application. As a result, developers are able to implement changes to an application at any time in the application life cycle – including post-production changes – using the same OPG graphical interface.

Technology Differentiators

All application software is composed of three distinct aspects: process, user interface, and database. Process includes all application logic, including control of flow and computational logic that one finds in traditional programming languages like Java or C++. The user interface includes everything needed to display information on, and receive information from, a computer screen or other interface. Web based interfaces are often developed with languages or tools like

HTML, JavaScript, or AJAX. The database component is typically handled with a relational database like Oracle, SQL-Server, or DB2.

Distinct sets of tools have been developed to increase application productivity within a specific aspect. For example, tools based on executable graphs, from Teranode, Scitegic, Inforsense, and JBoss jBPM handle the process aspect of an application. The chart below summarizes the difference between OPG technology and traditional development methods.

Application Aspects Of Software Systems	Traditional Technologies	Some newer technologies: Small steps in the right direction using graphs... Domain Specific Applications	OBJECT PROCESS GRAPH Full realization of graph power Domain Agnostic Development & Execution Platform
PROCESS	Programming Languages e.g. java, C++	Process Only Executable graphs; e.g. Teranode, Inforsense, Scitegic, Expert-24, Labview, Jboss jBPM Graph Models - code generators & design tools; e.g. Rational Rose, TogetherJ	General purpose executable graph Fusion of all application aspects
USER INTERFACE	UI Languages & Tools, e.g. javascript, AJAX	User Interface Only, Graphs very rarely used	
DATABASE	e.g. Oracle, SQLServer, DB2	Database Only Graphs rarely used Executable Graphs; e.g. Cogito	

In contrast to traditional technologies, program information, traditionally described with computer program code, is stored as nodes and edges within the Object Process Graph. This provides a fine-grained modular structure that can grow and be updated incrementally. Consequently, this structure shares the characteristics of data that traditionally reside in a database. The OPG can be changed at any time, including while an application is running. Changes are a natural part of an OPG system lifecycle. Changes are implemented by adding, updating, or deleting OPG nodes and edges or by changing their properties. Although this is similar to the lifecycle of a traditional database, traditional databases suffer from the trauma of separate program code maintenance. The OPG system is inherently flexible, adaptable, and designed for change.

An OPG is a high level model of a computer application system constructed and realized as an executable graph. Unlike previous attempts to develop software systems via high level design models, with the OPG, the model is the program. Thus, no code generation is needed to translate a high level design model into

program code, because the OPG is both the program and the model. Applications are run by simply directly executing their Object Process Graphs. Everything needed to define and execute an application is within the OPG. The creation of process logic, user interfaces, and database structures is performed within a single environment, in a transparent manner. Complete applications, including process, user interface, and database, can be created without the need to integrate the different application aspects.

The OPG in Practice

Developers who use OPGs are able to create fully functional, production ready enterprise systems faster than most organizations can produce a requirements specifications document. This is due to the inherent nature of the executable graph; because the design is the executable program, the application comes to life rapidly. End users are able to work with a developing OPG system almost immediately; they can see, work with, and fully test system functionality. Because the development process occurs alongside and concurrently with the requirements gathering process, end users are better able to define their system requirements.

The OPG technology also allows developers or security enabled users to modify any aspect of an application – its user interface, database structure, or programming logic – while the application is running. Both user and developer generated modifications can be applied to the base application for all users, to a specific user's version of the application, or to a specific run or execution of the application. One result of this feature is that OPG users are able to create whole new types of computer applications.

The OPG was also created specifically to manage the complexity required of large modern computer applications in a much more efficient manner and without the need for programming. Object Process Graph technology combines implementation speed, flexibility, and reliability advantages with the ability to define the complex program and data structures required to implement modern applications. It is now possible to build systems that previously were too complex to develop with traditional technologies by using OPG technology instead.

The departure from traditional programming techniques allows OPG created systems to be substantially more reliable. Additionally, OPG created systems result in greater end user satisfaction, due to the merged relationship between requirements gathering and system development that is inherent in the OPG development process.

Although no traditional programming languages or databases are required to build applications, the technology can closely interface and integrate with legacy technologies using industry standard protocols. Applications developed on the

OPG platform can integrate with any database, such as Oracle, or any application written in a traditional language, such as C++ or Java. Additionally, since the platform was built in Java and designed for the web, applications will run on all industry standard operating systems, web browsers, and hardware.

The user interfaces definable within the OPG are rich, interactive interfaces with capabilities equal to consumer oriented web sites. Potential products that can be built with OPG technology include enterprise business application systems for process control, manufacturing, finance, human resources, database systems, and software development systems.

Conclusion

The evolution of the software industry has proceeded along a continuum that embraced object oriented technology as the dominant model for system design. In practice, the lengthy, multi-step process involved effectuated a built-in obsolescence, due to the fragmented development steps and rigid distinctions between process, user interface, and database. The fusion of these functions within the OPG allows for the rapid deployment of complex enterprise quality systems that are flexible and dynamic. With requirements gathering, design, and testing all completed in parallel, the development cycle is significantly compressed and on-the-fly changes are available to both developers and end users.