CodeMorphis

Prototyping Windows Applications Visually

White Paper
Revision 1.3
June 5, 2004
www.codemorphis.com

The Challenges of Software Development

Software development is very labor-intensive and often requires large investments of human resources and capital. In addition to the tangible costs there also exist hidden costs that are directly derived from typical implementations of software projects. These hidden costs often arise from inefficient development planning and execution that result in late to market scenarios. In such cases, software producers not only lose potential revenues in software sales but also risk losing substantial market shares to competitive products. In the ever-increasing competitive software industry, it is imperative to have a streamlined production pipeline in order to deliver cutting-edge software products to market in a timely fashion.

An organization's ability to produce competitive software applications is to a very large extent, determined by the quality of project planning and the communication that takes place between the involved development groups. Some of the most costly problems in software development, in terms of capital, time and resources, include:

- *Technological communication gaps between an organization's groups:* Erroneous designs and difficult communications between marketing and R&D, for example, lead to incorrectly designed and/or improperly targeted products.

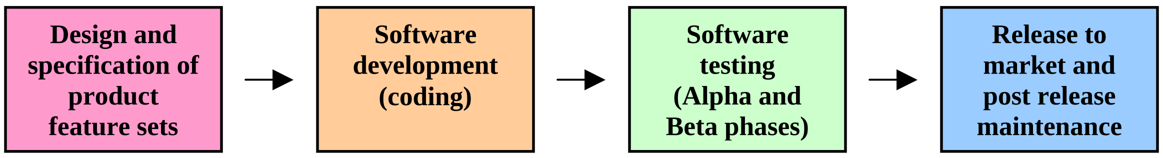
- *Excessively long implementation cycles:* Long development and beta phases, slow times to market, the inability to adapt to increasingly fast changing technology market conditions lead to lost revenues and can jeopardize the organization's overall market position.

- *Inefficient development planning:* Unnecessary waste of human resources and task redundancies can result in the poor allocation of an organization's most valuable resources and a low performance product.

- *Unnecessary post release costs:* Development errors, software patches/updates, customer support & training and customer on-site maintenance visits often account for even greater proportions of total project costs. Design errors not detected during design phases typically cost 100 times more to fix once products have shipped ("*What We Have Learned About Fighting Defects*", Shull et al., Proceedings of the Eight IEEE Symposium on Software Metrics, 2002).

The Waterfall Approach: An Unwieldy Software Development Scheme

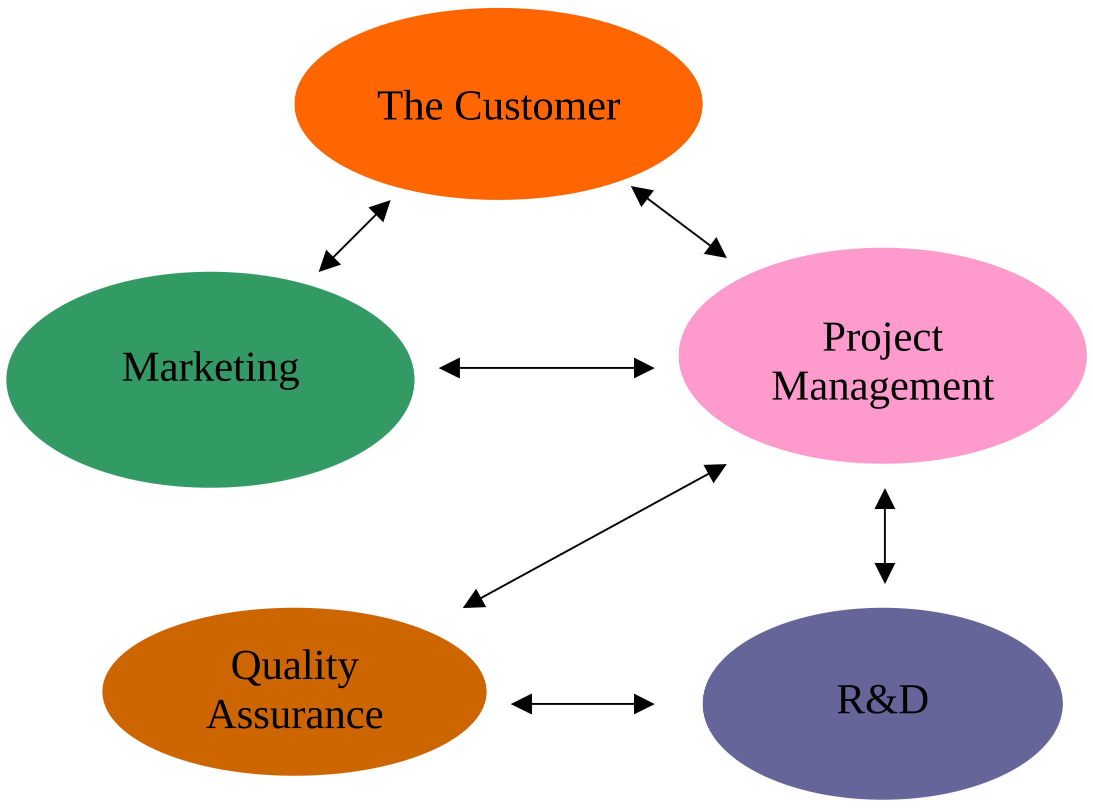
Many software companies, even large entities, often lack clear development planning processes and tools. Most start with some initial documentation and plans for proceeding but then these methods and communication tools are quickly abandoned as the project moves through its many milestones. Often, once development has begun, major re-evaluations of feature sets must be performed due to hasty planning. Such unexpected revisions not only slow down the development process but also affect marketing decisions taken prior to the project's debut, thereby potentially affecting the organization's business model. Why does this phenomenon occur over and over in most software companies? The reason is that the traditional approach to software development follows a waterfall cycle, in which we move from a distinct marketing stage to a distinct planning stage, to a development stage and then finally a quality assurance stage. The problem is that software development is highly iterative in nature, requiring many refinements and redesigns that go beyond even the current development project and into future releases. Thus, the waterfall approach runs counter to the actual requirements of a software development project.


Design and specification of product feature sets
Software development (coding)
Software testing (Alpha and Beta phases)
Release to market and post release maintenance

The waterfall approach to software development.

Breaking Away from the Waterfall Approach

Traditionally, the software development cycle is deemed to start with project management and R&D groups. In today's highly competitive market, however, software producers are finding that it is necessary to rethink this process and start with the customer. Software planning actually starts with the customer and is initiated by marketing to determine the needs of the target customer and the ensuing feature sets that must be delivered. Once these product requirements are determined, constant communications must be maintained between various groups involved with the project.



Software development requires constant interaction and communication between groups.

Harnessing the Power of Rapid Application Technology

To support an iterative communication and design approach, new tools must be used. Instead of exclusively using traditional design documents and specification sheets, tools that foster iterative work strategies must be integrated. A Rapid Application Development (RAD) tool can serve to:

- *Help reduce the lag time between design concepts and actual implementation by more accurately describing how the software product must behave.* By having a visual prototype that groups such as project management, R&D and quality assurance can experiment with simultaneously, increased communication on a design scheme that is representative of the final product is achieved.

- *Allow all involved development groups to better contribute to the final software product.* Design documents and specification sheets are insufficient for describing and appreciating actual software workflows and behaviors, especially when most software products must constantly evolve to satisfy changing market needs. As well, when working with customers to identify product requirements, a visual prototype can be used to validate the relationships between customer needs and feature sets, delivering huge financial and temporal savings.

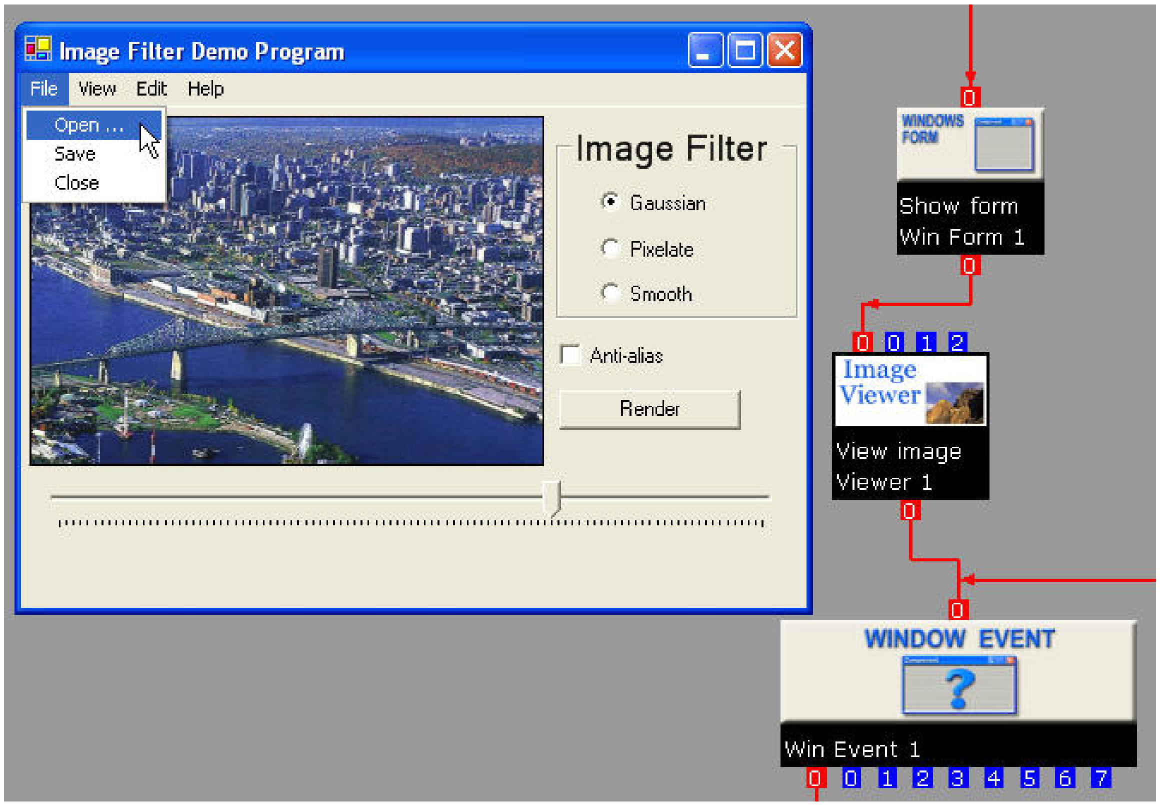
- *Help to test and validate designs early in the development cycle.* Traditionally, groups such as quality assurance only see the near-final product. By allowing such groups to evaluate designs in early stages, design flaws can be avoided.

CodeMorphis Rapid Application Development (RAD) Tool

To address the software development needs of today's applications developers, CodeMorphis has introduced ***Synopsis***, a rapid application development tool built on visual programming technology. By utilizing the power of visual programming, Synopsis allows for all development groups to share a common design prototype. Therefore, design can actually begin at the marketing stage and continue throughout the implementation phases until the product is released to market.

Synopsis uses simple and easy to understand graphic icons to represent modular software components that can be combined like building blocks to quickly create a program. The result is that it is possible to create complex and functional software applications in minutes.

Synopsis gives a clear separation between the user interface development of a Windows application and the logic that drives the behavior of the program. The entire user interface of the application can be created visually with simple drag and drop operations from component trays. As the user interface is put together, Synopsis allows the user to add components to control the logic flows of the program. The component logic is assembled in a flow chart manner with color-codings to represent data and logic transfers.



Using Synopsis to build user interfaces and control programming logic visually.

Synopsis allows the user to think in terms of high-level tasks rather than get bogged down in low-level programming details. This frees the user and allows for maximum creativity. The effects of changes to the program prototype are seen immediately without the need to leave the visual environment. Even the most minute changes can be made on the fly and verified instantaneously. Stand-alone executables can even be generated from within the Synopsis product to serve either as prototypical versions or even full-fledged functional applications or tools. Synopsis also allows for integration with existing softwares (e.g. Dlls and databases) and features a proprietary plug-in technology to import new and evolving software components for application assembly.

Synopsis can help the overall development process by accelerating the software product design phase. Non-development human resources can become actively involved in the design requirements and specifications phases of product development. With Synopsis it is also easy to validate marketing and business objectives early on, before actual resources (monetary, temporal, human) are committed to the project.

Benefits of Synopsis

By using Synopsis the following benefits are enjoyed:

- **Time and cost savings for development.**
  Synopsis provides the user with modular software components that can literally be assembled within seconds to create a functional and very complex Windows application. The rapid prototyping features of Synopsis give development accelerations of several orders of magnitude. As an example, some multimedia Windows applications that normally would take 2 or 3 man-weeks to complete have been developed in Synopsis in under 1 hour.

- **Gains in software stability and robustness.**
  Synopsis is built on proprietary software toolsets that have been tested and verified for application development. Thus, using Synopsis adds further time gains by avoiding common software development pitfalls and bugs. This helps to accelerate product to market times even more.

- **Gains in flexibility and scalability.**
  You can use Synopsis to access existing software, for example Windows Dlls (Dynamic Load Libraries). This allows for the reuse of existing technologies. Synopsis also features a proprietary plug-in technology to allow for the integration of new software components that extend the programming power of the user. These features allow Synopsis to be used according to the current and/or preferred methods of the user. It also makes it much easier to move from the prototype to the final software development phase.

- **Improved software design processes.**
  Synopsis features *real-time programming*. This means that it is possible to change the design of your program while it is running! You can make programming changes during the run-time session of your program so that you can get immediate feedback. This technique breaks the traditional waterfall approach and results in much faster and tighter design, prototyping and implementation cycles.

Using Synopsis Across the Development Schedule

Using Synopsis, you can improve software application development across the entire development schedule, from start to finish. Here are just a few of many possible case scenarios:

- *Synopsis as a powerful marketing tool:* Develop business cases with customers from the start and benefit from their valuable input early on. Use Synopsis to conduct software focus groups or interactive marketing sessions with customers. In a few minutes you can capture the essential product requirements of your customers. Prepare variations of user interfaces easily and quickly and allow customers to provide critical feedback before incurring any software development costs.

- *Synopsis as an easy to use and flexible software design prototyping tool:* Break away from the traditional, inefficient and very costly waterfall approach to software development. Integrate your marketing, quality assurance, project management and R&D teams by allowing various human resources to incrementally refine the application designs before actual development is started.

- *Synopsis as a quality assurance and software refinement tool:* Use Synopsis for project management of implementation phases to verify development concepts and experiment with application solutions.

Summary

Synopsis provides a powerful prototyping and design communications tool to facilitate the production of any Windows application. The completely visual environment of Synopsis means that applications and prototype designs can be created with tremendous time and money savings. Furthermore, the ease of use means that Synopsis can be used by even those who have little programming knowledge. This key characteristic allows for Synopsis to be used throughout the entire software development project by various involved groups, from marketing to project management to R&D. With Synopsis, organizations can gain market advantages by delivering more robust and better targeted software applications than their competitors.

To learn more about Synopsis, visit http://www.codemorphis.com.