Enterprise Development

with the Microsoft Solutions Framework™ and Elsinore's Visual Intercept™

Prepared by Kenny M. Felder: Co-founder of One Tree Software, and former Group Program Manager of Visual SourceSafe for Microsoft Corporation

Visual Intercept in the Microsoft Solutions Framework

“The success of most development projects depends more upon effective project management and development processes than on writing clever code. The Microsoft Solutions Framework is based on best practices in development projects that have made Microsoft so successful. The MSF provides a flexible, interrelated set of concepts, models, and processes to help plan, build, and manage the development of custom business solutions. Visual Intercept is an excellent tool for MSF-based development projects because it facilitates documentation and prioritization of development issues. In addition, its scalability and integration with Microsoft development tools make it useful for a wide range of development projects.”

- Jim Wilson, MSF Product Manager, Microsoft Corporation

“What makes Visual Intercept such a vital tool is the fact that it integrates tightly into all of the Visual Studio tools. This isn't just about convenience—by integrating seamlessly into the development environment, Visual Intercept becomes a very powerful way of keeping a development team and a development process on track."

- Greg Leake, Visual Studio Group Product Manager, Microsoft Corporation

"Visual Intercept provides a structured development environment for Microsoft Developer Tools users. Visual Intercept has become the glue that binds the domain knowledge of the software development cycle from specs and requirements, rapid prototyping, development, QA, through release. Visual Intercept allows the development team to understand where they have been and what they are expected to do to succeed."

- Mark Uland, Development Manager, Elsinore Technologies

“In recent years, it has become increasingly obvious that product development of any kind (software, infrastructure or even widgets) is a team effort. Further, risk management is now mandated as the driving force for all decisions within the product development cycle. Especially in large teams, there needs to be a tool that can act as the basis for reliable project and risk assessment throughout the entire cycle. Until now, this has been the domain of cumbersome custom workflow and web applications, or worse—the domain of the ignored. Visual Intercept provides all the features necessary to simply and effectively manage issues from bugs to shipping delays to typos on the marketing materials. And you don’t need to be a code cowboy to use it!”

- Marc Aniballi, Enterprise Program Manager, Microsoft Corporation

MSF, the *Microsoft Solutions Framework*, is a set of best practices, processes and strategies for producing a software application or deploying an infrastructure. Based on the techniques developed over the last 20 years by Microsoft Corporation, its consulting arm, and its partners, MSF provides a systematic approach to ensure that high quality and on-time delivery are not left to chance.

Note: In this paper, following common usage, the term "MSF" will be used to refer specifically to Solutions Development Discipline (SDD), the most popular course in the MSF series.

Companies who take an MSF course from Microsoft are left with a very detailed knowledge of what processes and systems to put in place, but much less understanding of what tools they can use to help them regulate and automate these processes. Marc Aniballi, a Microsoft enterprise program manager and MSF instructor, argues "It is impossible to apply and benefit from MSF without a good issue tracking system. To be truly effective, the issue tracking system has to scale to the size of your entire enterprise, and integrate tightly into the other development and management tools you use. Of course, it also has to track a lot more than bugs."

This white paper explains some of the specific ways that one product—*Visual Intercept by Elsinore Technologies*—can be used to help implement an MSF development strategy.

A brief introduction to the Microsoft Solutions Framework

The *Microsoft Solutions Framework* (MSF) is based on a theory that has proven successful in over twenty years at Microsoft Corporation: shipping a successful software product is no accident. It requires an organized team, structured code, and systematic processes that minimize risk and maximize the ability to make intelligent decisions. As a development project becomes larger and more complex, the need for a systematic approach becomes even more important.

At the heart of MSF are three models.

The *team model* divides a product team into six roles. Each role (one or many people) is represented by a manager (or team lead), and these managers form a group of peers who are ultimately responsible for all major decisions, and for the success or failure of the product. The six roles are: development, testing, logistics, user education, product management, and program management. One of the critical elements of MSF is good, clear communication *within* and *between* these six roles.

The *process model* divides the development process into four phases: envisioning, planning, developing, and stabilizing. The process model is discussed in more detail below.

Finally, the *application model* divides the application into three tiers: the user interface, the underlying data, and a "middle tier" which defines the *business rules,* or constraints on the data.

Students in an MSF course learn to organize their teams according to the team model—often, this means formalizing certain functions (such as testing) which may have previously been left to be done "on the side." They learn to organize their development cycle according to the process model—this means moving certain tasks, such as collecting requirements lists from customers, to center stage. They learn to organize their products along the application model, which consists of pulling the business rules out of the user interface code or the underlying data. They learn to identify and manage risk.

Most importantly, they learn that a systematic approach to the development process does *not* mean adding bureaucratic overhead that will get in the way. On the contrary, it means streamlining systems in a way that can reduce overall development times, increase quality, and provide a crucial competitive advantage.

A brief introduction to Visual Intercept

Visual Intercept from Elsinore Technologies is an incident management system. Similar to a bug tracker, but more comprehensive, Visual Intercept is best thought of as an electronic "to-do list" designed specifically for the needs of a Microsoft developer. Used properly, Visual Intercept provides the following key benefits to a development team.

- It facilitates communication about key issues, by providing a centralized forum where those issues are discussed and tracked.
- It provides individual contributors with specific task lists.
- It provides managers with a clean overview of all the tasks being done, and an easy way to add new items, remove or postpone less important items, and prioritize items.
- It allows managers to maintain efficient load balancing by viewing a list of each team member's assigned incidents and adjusting assignments as necessary.
- It ties together the variety of Microsoft tools—including Visual SourceSafe, Visual Studio, Microsoft Word, and Microsoft Excel—which are used throughout the project.
- It provides reports for management and documentation of a project.
- It provides a tracking mechanism and a permanent record of the project life cycle, even allowing certain issues from one cycle to be postponed to the next.

At the heart of the product is the *list of incidents.* An "incident" can be a bug that has to be fixed, a feature that has to be implemented, a customer question that has to be answered, or any other important item that has to be tracked systematically until it is dealt with. Associated with each incident is a general description of the issue, a priority, a person who is currently assigned to work on the incident, and a status. In addition, an incident may be associated with one or more contacts (such as customers), one or more files (such as documentation or related source code)—and of course, many other properties not listed here. Incidents are organized into a hierarchical system of "projects."

Built on top of this list are the processes, or rules for manipulating an incident. For instance, the typical life cycle of a bug is "report, assign, resolve, close." A tester *reports* the bug, the development lead *assigns* the bug to a developer, the developer fixes and *resolves* the bug, and finally the tester checks the fix and *closes* the incident. Visual Intercept guides the bug through this process, making sure that at each step the bug is assigned to the right person and is never lost.

And finally, on top of the processes, there are the user interfaces to the system: an easy-to-use Windows GUI, a Web-based interface that lives inside the browser, and a floating toolbar inside Microsoft Visual Studio. MSF veterans will recognize immediately that Visual Intercept is organized in the application model ("three-tier" or "n-tier" architecture) recommended by the MSF application model.

Of course, Visual Intercept has many more features, some of which are discussed in the appropriate sections of this paper. However, there is one overarching feature, which is at the center of the Visual Intercept philosophy: integration with Microsoft tools. At every stage, Visual Intercept works so closely with the other Microsoft products you already own that the boundaries are almost seamless. This is why the product is so often described as the “glue” that binds together all the different Microsoft applications.

Integration Note

A note like this one will be made whenever there is particular mention of a way that Visual Intercept integrates with the other tools you use.

The MSF Process Model

MSF divides the process of development into four distinct phases:

- Envisioning
- Planning
- Developing
- Stabilizing

MSF instructors suggest that Visual Intercept can be useful in all four phases. However, different goals, different milestones, and overall control by different team members characterize the four phases: therefore, the specific uses of Visual Intercept vary considerably from phase to phase.

The Envisioning Phase

The “envisioning” phase of a project, driven by the “product manager” in the MSF Team Model, involves working with customers to set out the overall vision of the product. At the end of this phase (the “Vision Scope Approved” milestone), the entire team agrees to the list of product requirements.

In practice, this often means maintaining a long list of high-level feature requests. It is very important that no feature request is lost forever: at the same time, everyone on the team knows that some of the feature requests will not “make the cut” for this version of the product, so prioritizing them is critical. Often, a feature is associated with one or more customers who requested it.

Integration Note: Microsoft Word

Often, a written document will be used to describe all the customer requests and requirements. Such a document can be attached directly to one or more incidents, and then launched and modified from directly inside Visual Intercept.

Clearly, during the envisioning phase, one of the primary uses of Visual Intercept is to maintain that feature list. Visual Intercept allows you to prioritize all the feature requests, and to say “show me only the priority-1 or priority-2” items as you begin to settle on a final list. Some features will be postponed to the next version, and can be filtered out of your query; however, they are not lost, and you will bring them back the next time to reconsider them. By setting up all your key customers in the Visual Intercept contact list, you can associate individual feature requests with the people who need to be notified of their status.

Integration Note: Contact List

Of course, the Visual Intercept contact list should be the same as your Microsoft Outlook contact list! A future version of Visual Intercept will provide automatic Outlook integration, so that one common contact list is used in both programs.

By the end of this phase, all of the feature requests have been entered into Visual Intercept; some are still active, while others have been postponed. By printing out a list of all the active ones, the product manager automatically creates a large part of the Vision Scope document that must then be approved by the team.

The Planning Phase

The "planning" phase of a project, driven by the "program manager" in the MSF Team Model, involves creating specific designs and schedules for the product. At the end of this phase (the "Project Plan Approved" milestone), the list of requirements has been turned into a list of specific features with plans and schedules for implementation. For instance, a customer requirement may be "I want to keep track of where my salesmen are in the field." The features in the planning phase are farther away from the customer request, but closer to the actual work that needs to be implemented: create a new "location" field in the database, restrict it to the following values, expose it through the user interface in these ways, and so on.

Integration Note: Microsoft Project

The list of to-do items used in this phase is often the same list of items that define the project in Microsoft Project. Later, Visual Intercept incidents will be associated with particular Microsoft Project milestones.

Once again, Visual Intercept serves as the final storehouse of all features. However, it is important to keep the requirements from the envisioning phase distinct from the features in the planning phase. Therefore, it is often useful to create different *projects* in Visual Intercept: one for requirements, and one for features. A feature in the planning phase can be associated with a particular requirement, since Visual Intercept allows different incidents to be related to each other: this allows the user to move quickly between a requirement, and the features which implement that requirement.

The MSF courseware suggests that the bug database can be used to record and track change requests made against the functional specification. The feature list is not truly frozen after the planning phase: during the development phase, as the product is used in-house and then by Beta testers, new important feature requests will be added to the list. This problem is often referred to as *change management*—how do you cope with the fact that, while you are developing the product, the requirements are continually changing under your feet? Clearly, it is vital to be able to add new features to the list, and at the same time, to view and prioritize them in the context of the existing features. The more complex this process becomes, the more important it is to have a very simple mechanism—such as Visual Intercept—for listing and reporting on all the features in the product.

The Developing Phase

The “developing” phase of a project, focused on the developers in the MSF Team Model, is often the longest phase of the project, since this is when the actual application is written. There are interim milestones (often referred to as “M1,” “M2” and so on) during this phase, but the developing phase is not over until the “Scope Complete” or “First Use” milestone is reached.

During the developing phase, the feature list developed in the planning phase is one of the key drivers of the process. Each developer can run a simple Visual Intercept query to bring up a list of all the features he is supposed to implement, along with the priority, deadline, and other information about the feature. When the feature is complete, the developer can resolve the incident, thus indicating that the work has been done.

Integration Note: Visual SourceSafe files

Visual Intercept allows you to associate an incident with the specific files in Visual SourceSafe that implement that feature. You can easily check in or out all the files associated with a particular incident, from within Visual Intercept. Visual Intercept incident information is automatically inserted in the “Comment” field in Visual SourceSafe’s file history. And later, you can refer back to the incident to see exactly which files were touched, for what reason.

MSF teaches that every project is constrained by the “iron triangle” or “trade-off triangle”: *resources*, *features*, and *ship date*. If the project is falling behind, there are three choices: add more resources, cut features, or delay shipping.

But here is a subtle and very important point. Very often, the team is not actually aware that the project is falling behind schedule! If the team is not aware of the problem well in advance, they will not make a conscious choice about how to handle it: they will not add resources, they will not cut features, and therefore the only possible result is a delayed release. On the other hand, if every feature is being carefully tracked in Visual Intercept along with its current status, the project manager can see the “early warning signs” of a schedule slip, and make a conscious decision of how to adjust.

In addition to the feature list, another sort of incident (perhaps tracked in a different project inside Visual Intercept) also becomes prominent during this phase: bugs. According to MSF, bug tracking is not simply a matter of fixing the occasional mistake. In order to run a successful project, you must know at all times if your project is on track; and a bug tracking database is the crucial element in that determination. A tester who finds a bug must have confidence that it will be addressed. A developer must be able to see, at all times, a list of the bugs he is expected to fix. These are, of course, exactly the issues that Visual Intercept addresses by maintaining the list of bugs and providing easy searching and reporting on that list.

MSF also stresses “clear accountability”: at any given time, each bug must have one person who is *responsible for* that bug, so nothing falls between the cracks. However, this responsibility may move over time: for instance, shifting between the development and test teams. For this purpose, the most important Visual Intercept feature is its ability to ensure that each bug is assigned to one person at any given time, and to control the flow of that assignment from one person to another.

As a final note: MSF instructors recommend regular code reviews during the developing phase. Some note, however, that the issues raised during a code review are often lost soon thereafter: if there is no good follow-up, the review was a waste of time! There should therefore be one developer whose job during the code review is to act as the “recorder,” taking down notes on all the major issues that are raised—and then entering them immediately into Visual Intercept, so that they will be tracked to completion.

The Stabilizing Phase

During the stabilization phase, bugs are king. There is little or no new development going on, and the test manager runs the show. This is the part of the project where it is most common to run the project from a bug tracking tool. Each developer checks Visual Intercept in the morning for all the bugs that are assigned to him: this forms his “to-do list” for the day. The project managers hold regular “triage meetings” in which they look at all the bugs in Visual Intercept together, and decide which ones to postpone for a future version of the product.

The team managers will be monitoring the bug tracking system closely—at all phases, but especially this one—to gauge the status of the project. The “bug curve” (the total count of open bugs over time) should spike up at each Beta release, and then slide back down. Using Visual Intercept, the test manager can compare the state of the project at any time against the agreed upon criteria for release (*eg* “No priority 1-2 or severity 1-2 bugs open”).

One MSF instructor teaches all his students that they will also want to be able to get the following reports, describing them as “common metrics used within Microsoft.”

- <u>Number of bugs opened per day</u>, broken down by project, and by severity within a project. As the project approaches completion, this number should be going down: that is, it should be harder and harder for testers to find bugs, especially high-severity ones.
- <u>Number of bugs resolved per day,</u> once again broken down by project and severity. The purpose of this report is not just to see how quickly the developers are working. By monitoring this curve alongside the previous curve, you can project when the bug count will reach zero.
- <u>List of bugs in order of when they were entered into the bug database,</u> broken down by severity. The point of this report is to see how quickly bugs are generally being resolved. Related metrics are “average time between opened and resolved” and “average time between resolved and closed.”
- <u>Number of bugs resolved “fixed” and number of bugs resolved in other ways, over time.</u> This is primarily a measure of the quality of the bugs that the test team is finding. If a significant percentage of bugs is being resolved “unreproducible” or “duplicate,” the test team is not finding real new bugs. If a significant percentage of bugs is being postponed, the test team may be focusing on the wrong issues.
- <u>Bug reactivation rate</u>: if many of the bugs which are resolved “fixed” are later getting opened again, that indicates a problem that should be studied more closely.
- <u>Breakdown of how bugs were found</u>: for instance, automated testing *vs.* ad hoc testing *vs.* systematic manual testing. By seeing which method produces the most bugs, you can refine your methods, or (toward the end) focus on the ones that work.

Integration Note: ODBC and Excel

Visual Intercept is built on top of Microsoft's Open Database Connectivity (ODBC) specification. In a default installation, the underlying Visual Intercept database is Microsoft Access. However, Visual Intercept can store its information in any database that supports the ODBC standard, including Microsoft SQL Server. That means that, in addition to the reporting facilities provided natively within Visual Intercept, you can write your own reports using database-specific tools. And you can export those reports to Microsoft Excel for graphing and charting.

Risk Management

One underlying theme of MSF, at *all* phases of a project, is minimizing risk. In fact, one MSF instructor defines the overall MSF strategy as "giving the development and/or rollout team the necessary feedback to minimize risk on a milestone-driven process." He adds that "Most project teams don't seem to implement that very well. That's where most teams fall apart."

MSF instructors stress that a bug tracking database, such as Visual Intercept, is one of the necessary ways to give that essential feedback to the development team. In addition, they recommend that the program manager keep track at all times of a *risk management issues list.* Issues on this list might include "We have five developers for every tester—can the test team keep up?" or "Our product is built on Beta technology from another company: what if their timeline slips, or their product is unstable?" For each issue, there should be a consideration of the possibly dangerous scenarios, and backup plans.

In Visual Intercept, a risk management issue is simply another incident. It can be entered, tracked, prioritized, reported, and ultimately resolved or closed, just as any other incident.

The MSF model also stresses that the managers of each of the six roles in the team model should hold regular project tracking meetings. During these meetings, among other issues, the list of risks should be discussed. If the list is stored and tracked in Visual Intercept, any member of the team can review it or make comments at any time, and the meeting leader can easily print out a list to prepare for that part of the meeting.

In Summary

Microsoft and many other corporations have discovered that a structured, well managed development process leads to a shorter development cycle and higher quality. MSF provides the structure and processes, but tools are required to implement those processes. Visual Intercept is one of the key tools that can be used for this purpose.

- By facilitating communication and tracking individual responsibilities, Visual Intercept brings together the different people in the MSF Team Model.
- By tracking different kinds of incidents at different stages, and relating them to each other, Visual Intercept structures the tasks throughout the stages of the MSF Process Model.
- By integrating tightly with many different Microsoft tools, Visual Intercept creates a unified development environment that puts key information at the fingertips of both individual contributors and managers.

Used together, MSF and Visual Intercept can provide an important competitive advantage for any team which is responsible for delivering important software.

--This paper was compiled by Kenny M. Felder. As a co-founder of One Tree Software, Kenny Felder was one of the original creators of a version control system called SourceSafe. After selling the product to Microsoft, he went on to be the Group Program Manager for Microsoft Visual SourceSafe. He compiled this document with many hours of assistance from a number of MSF instructors from Microsoft Corporation, members of the Elsinore Technologies development team, and Visual Intercept sales consultants from Raleigh Group International.

For More Information

For more information on Visual Intercept, contact John Myers at Elsinore Technologies at (919)532-0022 ext 409 or kfreed@ciinc.com. For a free download of Visual Intercept visit http://www.elsitech.com/downloads/downloads.htm.

For more information on the Microsoft Solutions Framework, visit http://www.microsoft.com/msf.