

When You Have a Hammer, Everything Looks Like a Nail

Applying the right tools for the job can go a long way toward architecting for project success

Mike Vincent

Software architecture tools

Humongous Insurance is a large multinational company that sells insurance products for boat, automobile, and home owners. These products are sold to both consumers and corporate clients. Humongous Insurance has a large, mobile workforce of over 10,000 people that includes salespeople, claims adjusters, attorneys, IT staff, HR staff, and finance staff. Humongous Insurance is based in the United States, but has offices in Canada and several Latin American countries.

Through merger and acquisition, Humongous has a mixed application environment composed primarily of Microsoft .NET and IBM Websphere. These applications handle customer facing web sites, corporate and consumer policy sales and renewals, insurance claims processing, government permits and licensing, legal case management, customer and vendor relationship management, accounting, and communications with approved repair contractors and mobile agents. There is a great deal of repetition and overlap both geographically and functionally among operating divisions.

Today we are attending the first day of the Humongous Enterprise Architecture Strategy Conference. The objective is to create a unified view and strategic blueprints for streamlining operations of the various applications used within the enterprise. They have agreed on the need to move toward a greater implementation of agile methodologies.

Here I am dropping in on a round table discussion to help identify software architecture tools appropriate to supporting agile development. People involved in the discussion include architects from our four regional development centers in the United States, Canada, Mexico, and Brazil include Scott McDonald, Lionel Penuchot, Cesar Garcia, Kirk Nason, Lisa Miller, Qiong Wu, and Yolanda Sánchez.

“Software architecture is more than an overall guide to system design and construction for developers. As architects we have a responsibility to keep IT and the business stakeholders requirements in alignment.” said Scott kicking off the discussion. “Before jumping in and running with your favorite architectural design tool, you can get off to a better start by architecturally understanding the big picture and context of your project.”

Understanding Agile Development

“Let’s summarize the principles of agile development to keep in mind as we move through this discussion. Scott Ambler has a great concise definition (Agile Definition, 2007 Feb 15):”

Agile is an iterative and incremental (evolutionary) approach to software development

which is performed in a highly collaborative manner

with "just enough" ceremony

that produces high quality software

which meets the changing needs of its stakeholders.

“There really is not set of standards of required artifacts for agile development. You just do what is really needed, just barely good enough. In iteration 0, the goal is to identify an architectural strategy, not write mounds of documentation (Agile System Development Lifecycle, 2007 Feb 15). Design details will be worked through in later development iteration model storming sessions and via test driven development. One of the real benefits of test first design is keeping code and design in sync. Initially, you need to get some code working to prove that the architecture works. Try to identify problem areas up front.”

One Size Does Not Fit All

“We also need to consider the size and scope of projects such as large line of business, corporate wide, large or small departmental, needs for internationalization and localization, and integration requirements including purchased products,” added Lisa. “It wouldn’t make sense or be the best use of resources to take the same approach for enterprise wide customer relationship management as it would for a small departmental specific application. Different situations (Choose the Right Method, 2007 Feb 15) call for different approaches”

“Not only the size and scope of projects, but we need to consider what we are trying to do with software architecture tools.” Kirk contributed. “We may be interested in focusing on requirements, class modeling or data modeling. Perhaps **CASE** tools to do some base code generation. It is really important for us to keep in perspective the fundamental architecture we build our applications upon. The use of development **frameworks** and **software factories** can help us keep consistency and maintainability among our broad range of applications. And, tools for management of our entire software development life cycle (SDLC, 2007 Feb 15) are of very high importance”

“There are lots of software architecture tools available. While we want to consider the best of breed products for our use we also have to look at how well they will work with the development environments we will be using going forward. Certainly the direct costs as well as the indirect costs such as how much education and training we will need and how much effort is required to best leverage these tools must be considered.” Yolanda added. “Also, since many tools are vendor specific, we will need to be comfortable being ‘married’ to that vendor’s technology.”

“I think we’ve got a great overview so let’s drill into some more specific areas to identify important attributes we should be considering.” said Scott. “To put some structure around our discussion, let’s organize around the categories that Lionel has put on the whiteboard – SDLC, Development Frameworks, Diagramming and Modeling, and Other Tools.”

SDLC Tools

“Microsoft Solutions Framework for Agile Software Development (Agile MSF, 2007 Feb 15) was introduced with Visual Studio Team System. We have been using it for about a year.” said Kirk. “Agile MSF is a context based process for building .NET applications that provides innovative techniques to extend agile software development to all the traditional IT roles including business analysts, architects, and testers as well as business stakeholders. It is composed of a set of proven practices that complement each other with the sum being greater than each one used in isolation.”

“Agile MSF builds upon the value statements of the Agile Alliance by introducing the techniques of **personas**, **shadow applications**, and **test thresholds** based on Microsoft’s own development experience. Agile MSF facilitates architects, business analysts, project managers, developers and testers working in parallel by staggering the work, setting up coordination points, and providing only what is needed in a just-in-time fashion. For example, we only write the scenarios for the upcoming iteration, we plan one iteration at a time, architect only the necessary pieces, develop the functionality in the iteration plan only for this iteration, and write test cases for functionality planned in the current iteration.”

“And, Agile MSF is customizable and extensible. It is designed to be mapped to a wide variety of existing methodologies which makes for substantially easier adoption.”

“Within our division, we have been using the Rational Unified Process (RUP, 2007 Feb 15) for the last three years for our J2EE based projects.” said Qiong. “The RUP is not a single concrete prescriptive process, but like rather an adaptable process framework intended to be tailored by the development organizations and software project teams that will select the elements of the process that are appropriate for their needs.”

“RUP is based on a set of six key principles for business driven development:

1. Adapt the process - A project or organization must right-size the process to their needs.
2. Balance stakeholder priorities - Widens the discussion from a pure software requirements point of view, to a higher discussion including business goals and stakeholder needs.
3. Collaborate across teams - With a broad variety of stakeholders, all voices need to be heard, especially with globally distributed development
4. Demonstrate value iteratively - Projects are delivered, even though often only internally, in increments in an iterative fashion. The increment, which includes the value of the past iteration, is used to measure the progress of the project.
5. Elevate the level of abstraction – This principle motivates the use of reusable assets. A higher level of abstraction also allows discussions on different architectural levels.
6. Focus continuously on quality - Quality checks are not only at the end of each iteration but are a continuous ongoing activity in the software engineering project, often performed in a daily rhythm supported by the entire team.”

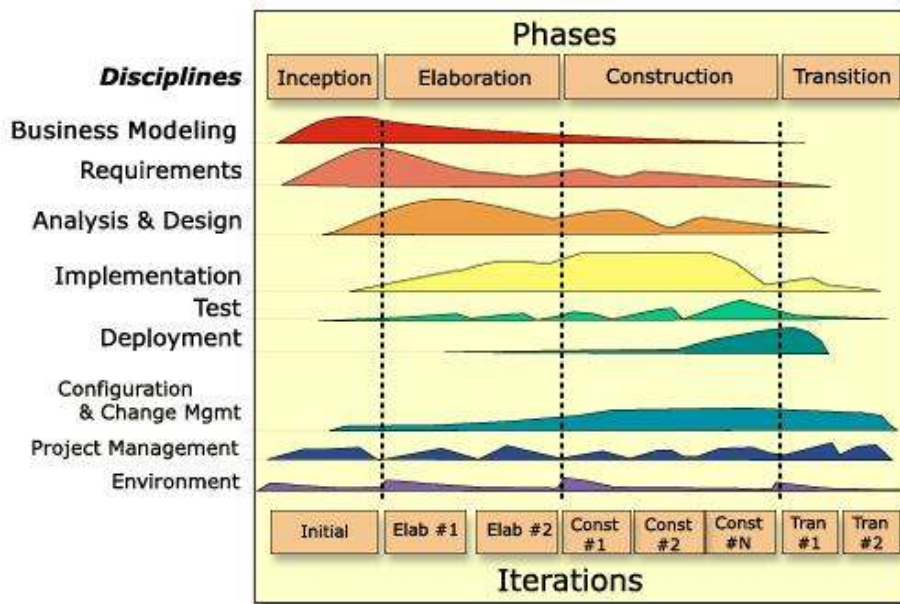


Figure 1, RUP Phases and Iterations

“RUP is based on a set of building blocks, or content elements, describing what is to be produced, the necessary skills required and the step-by-step explanation describing how specific development goals are achieved.”

Development Frameworks

“Development Frameworks (Framework, 2007 Feb 15) provide us with another important category of software architecture tools.” contributed Lionel. “A Framework facilitates software development with a degree of architectural consistency by allowing developers to focus on solving business problems rather than spending time and effort on low level ‘plumbing’ issues. There are a wide variety of commercial and open source frameworks available. With almost all frameworks, one needs to recognize that there must to

be a commitment to learn it. Many frameworks may include some code generation capabilities which can be a time saver but look for the real value in the class and object structure while not placing undo restrictions on developers. Also, for most organizations, a framework should be customizable and extensible.”

Diagramming and Modeling Tools

“Moving to the fundamental architectural functions for conceptualization and communication there are a multitude of products for diagramming and modeling.” said Cesar. “Visualization is highly important to communication. Diagrams based on standards provide a means of common understanding. Most are based on the UML (Unified Modeling Language, 2007 Feb 15). UML diagrams include Activity, Class diagrams, Collaboration, Component, Deployment, Interface, Frame, Note, Package, Sequence, State machine, Stereotype, and Use case.”

“Products range from basic diagramming tools where objects are dropped on a design surface and linked up with connectors to products able to forward and reverse engineer code. Several products are supported for both Visual Studio .NET and Eclipse. Consider capabilities such as being able to import models from other products, support for audits and metrics, template customization, import and export of business process execution language and Web Services definitions, design patterns, and of course, the languages you develop in. Be sure support is included for recent language enhancements such as partial classes, anonymous classes, generics, and other new language features available in .NET Framework 2.0. Also consider the ability to keep software artifacts synchronized.”

“A word of caution, though.” added Lisa. “With any reasonably complex vendor product, you’ll need to decide if you are comfortable being ‘married’ to the vendor’s offerings and methodologies.”

“Object Role Modeling (ORM) is another diagramming modeling technology to consider.” said Scott. “ORM diagrams are an incredibly effective way to explore domain concepts with your stakeholders. An ORM diagram depicts objects (entity types), the relationships (fact types) between them, the roles that the objects play in those relationships, constraints within the problem domain, and optionally examples (called fact type tables). ORM has a solid mathematical foundation and can produce plain English output that business non technical stakeholders can understand and respond to.”

Other Tools

The other ORM aka O/RM (Object Relational Mapping) should also have a place in the architect’s toolkit even though is technically a technique.” said Yolanda. “Most development today is object oriented and data manipulation in code is best handled as objects. Additionally, proper relational data structures do not map directly to objects and need to be structured. Commercial and open source O/RM products handle the mapping and persistence management.

“Don’t forget the simple tools – the whiteboard.” said Lisa. “This may be the most effective means of communications with developers. Keeping architecture and development progress synchronized should be as simple as whiteboard drawings and as equally expressive.”

“We used to try to apply the same architectural tools and methodology on projects of all types and complexities.” remarked Lionel. “It was like if I had a hammer, everything looked like a nail! I think we can all see a great value in matching the right tools from a vast array of architectural tools available to the job.”

Review

We have covered a broad range of software architecture tools from SDLC, Development Frameworks, Diagramming and Modeling, and Other Tools including the whiteboard. While our focus has been through the lens of agile methodologies, most of the concepts covered here apply to other methodologies as well. Many tools are vendor specific, so decide if you want to be ‘married’ to that vendor’s offerings. Most importantly, choose the right tool for the job.

Critical Thinking Questions

- Are you currently using the right tool for the job?
- How much education, training and experience is required for the tools you are using or considering?
- The tools you use should help you achieve project success
 - Are you using them to get things done better, faster, and with higher quality?
 - Are you producing just enough documentation for your project and regulatory needs?
- How well do you use whiteboards?

References and Further Reading

- Manifesto for Agile Software Development, <<http://agilemanifesto.org/>>. Accessed 2007 Feb 15.
- [Agile Definition, 2007 Feb 15] Agile Software Development: Definition, Ambler, Scott W., <<http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm>>. Accessed 2007 Feb 15.
- [Choose the Right Method, 2007 Feb 15] Choose the Right Software Method for the Job, Ambler, Scott W., <<http://www.agiledata.org/essays/differentStrategies.html>>. Accessed 2007 Feb 15.
- [Agile System Development Lifecycle, 2007 Feb 15] The Agile System Development Lifecycle (SDLC), Ambler, Scott W., <<http://www.ambysoft.com/essays/agileLifecycle.html>>. Accessed 2007 Feb 15.
- [Artifacts, 2007 Feb 15] Artifacts for Agile Modeling: The UML and Beyond, Ambler, Scott W., <<http://www.agilemodeling.com/essays/modelingTechniques.htm>>. Accessed 2007 Feb 15.
- Agile Modeling and the Rational Unified Process (RUP), Ambler, Scott W., <<http://www.agilemodeling.com/essays/agileModelingRUP.htm>>. Accessed 2007 Feb 15.
- [Object Role Modeling, 2007 Feb 15] Object Role Model (ORM) Diagrams, Ambler, Scott W., <<http://www.agilemodeling.com/artifacts/ormDiagram.htm>>. Accessed 2007 Feb 15.
- MSF for Agile Software Development, <<http://msdn2.microsoft.com/en-us/teamssystem/aa718801.aspx>>. Accessed 2007 Feb 15.
- [Agile MSF, 2007 Feb 15] Agile Software Development for the Entire Project, Miller, Granville, <<http://www.stsc.hill.af.mil/crosstalk/2005/12/0512miller.html>>. Accessed 2007 February 15.
- Visual Studio 2005 Team System Modeling Strategy and FAQ, <[http://msdn2.microsoft.com/en-us/library/ms379623\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/ms379623(VS.80).aspx)>. Accessed 2007 Feb 15.
- [Object Role Modeling, 2007 Feb 15] Object-Role Modeling (ORM) standard version 2, <<http://sourceforge.net/projects/orm>>. Accessed 2007 Feb 15.
- EclipseUML Free Edition, <<http://www.omondo.com/>>. Accessed 2007 Feb 15.
- Rational Software, <www.rational.com>. Accessed 2007 Feb 15.
- IBM Rational Unified Process, <http://en.wikipedia.org/wiki/Rational_Unified_Process>. Accessed 2007 Feb 15.
- [Framework, 2007 Feb 15] Framework, <<http://en.wikipedia.org/wiki/Framework>>
- Scrum (management), <[http://en.wikipedia.org/wiki/Scrum_\(management\)](http://en.wikipedia.org/wiki/Scrum_(management))>. Accessed 2007 Feb 15.
- Computer-aided software engineering, <http://en.wikipedia.org/wiki/Computer-aided_software_engineering>. Accessed 2007 Feb 15.
- Object Relational Mapping, <<http://sharptoolbox.com/categories/object-relational-mappers>>. Accessed 2007 Feb 15.
- Hibernate Relational Persistence for Java and .NET, <<http://www.hibernate.org/>>. Accessed 2007 Feb 15.
- Mere Mortals .NET Framework, <http://www.oakleafsd.com/pgProducts_mmnet.htm>. Accessed 2007 Feb 15.
- Rockford Lhotka's CSLA .NET Framework, <<http://www.lhotka.net/Area.aspx?id=4>>. Accessed 2007 Feb 15.
- McGovern, James; Ambler, Scott W.; Stevens, Michael E.; Linn, James; Essential ASP.NET 2.0, Addison-Wesley Professional. ISBN: 0-321-23770-6. 2006.

About the Author

Mike Vincent is a solutions architect with speakTECH, a Microsoft Gold Certified Partner in Orange County, California providing clients with enterprise and software architecture and development services, primarily focusing on Microsoft .NET technology. He founded and leads both the IASA SoCal Chapter and the Orange County C# Developers group. Mike is a frequent presenter at local and regional user groups and is a driving force for the SoCal .Net Technical Summit conferences.

Glossary

The technologies behind software architecture tools include:

- **CASE.** Computer-aided software engineering (CASE) is the use of [software](#) tools to assist in the [development](#) and maintenance of software. Tools used to assist in this way are known as CASE Tools. Some typical CASE tools are:
 - [Code generation](#) tools
 - [Data modeling](#) tools
 - UML
 - [Refactoring](#) tools
 - [QVT](#) or [Model transformation](#) Tools
 - [Configuration management](#) tools including [revision control](#)

It is the tools that are concerned with analysis and design, and with using design information to create parts (or all) of the software product, that are most frequently thought of as CASE tools.

- **Agile Modeling.** Agile Modeling (AM) is a practice-based methodology for effective modeling and documentation of software-based systems.
- **UML.** Unified Modeling Language (UML) is a general-purpose modeling language that includes a standardized graphical notation used to create an abstract model of a system, referred to as a UML model.
- **ORM.** Object Role Modeling (ORM) is a powerful method conceptual class design and for designing and querying database models at the conceptual level, where the application is described in terms easily understood by non-technical users. In practice, ORM data models often capture more business rules, and are easier to validate and evolve than data models in other approaches.
- **O/RM.** Object-Relational mapping (aka O/RM, ORM, and O/R mapping), is a programming technique for converting data between incompatible type systems in databases and Object-oriented programming languages.
- **Framework.** A framework is a defined support structure in which another software project can be organized and developed. A framework may include support programs, code libraries, a scripting language, or other software help develop and glue together the different components of a software project.
- **Personas.** Personas are respectful, fictitious people that represent groups of users. The personas describe usage patterns, knowledge, goals, motives, and concerns of a group of users. The key to good personas is that they are memorable and represent a set of typical customers. Personas can be compared to actors in use-case models with more details about the user community. Personas allow all members of the development team to obtain a deeper understanding of the user community.
- **Shadow Applications.** A shadow is architecture for the functionality to be completed in the iteration. The shadow leads the working code at the beginning of iteration as the architects get out in front of the development for the iteration. During this time, the architecture and the working code are not in sync. As the pieces of the leading shadow are implemented, the architecture begins to reflect the working code base. The original parts of the system that were architected but

not implemented now become implemented. When the architecture represents the working code, we call the shadow a trailing shadow.

- **Software Factories.** A Software Factory is a development environment configured to support the rapid development of a specific type of application. While Software Factories are really just the logical next step in the continuing evolution of software development methods and practices, they promise to change the character of the software industry by introducing patterns of industrialization.
- **Test Thresholds.** Test metric thresholds are objective definitions of when product quality is high enough to ship. Test thresholds are context based. Testing that is acceptable on one project may not be on another.