



## TenStep Supplemental Paper

---

5 February 2004

### Development Architecture

Companies, especially large ones, can have a difficult time making the most efficient use of company resources. Of course, most people don't start their day trying to figure out how to waste the company's money. The problem, however, is that people make decisions based on what they know. If you always had all of the relevant information you needed, you could make the best decisions possible. However, when you don't have all of the best information, it's possible that what seems like the best decision from your perspective may not be the best one from a company perspective.

#### Enter development architecture

In general, IT architecture is a framework. It is a structure that provides guidance to help make decisions in a way that makes best use of resources on an organization-wide or company-wide basis. There are a number of these frameworks, or architectures, in the IT organization. For instance, you can define a Technical Architecture that identifies your technical products and tools and when it is appropriate to use each. You can define a Data Architecture that describes important data fields and characteristics about each one.

Here, development architecture is defined in a broad sense to include three major areas:

- The development life-cycle and processes used to build business applications.
- The application models that show the appropriate technical design that will best fit the business requirements.
- The inventory and categorization of the business applications that exist within the organization today. In some companies, this area is called an "Application Architecture," but here it is considered part of the broader development architecture.

Let's look at each of these areas in more detail to see how they can positively affect your development environment.

#### Application development process

The first part of Development Architecture is to define the processes associated with your development life-cycle. This is a good exercise for companies of all sizes. If your company produces software for a living, chances are you already have this in place, although some organizations still do not.

There is a fair amount of the development process that requires the creativity and skills of the individual analysts, designers and programmers. However, there are also many aspects of the life-cycle that can be standardized. For instance, there are many ways to collect business requirements. However, there are certainly some ways that are better than others. There are also many techniques available to gather requirements, from personal interviews to surveys to group meetings. There are also certain proven techniques for testing. Your development architecture might provide an overall testing



## TenStep Supplemental Paper

---

process that is applicable to general projects, but allow some customization of the processes based on the specific solution being developed.

There are also many ways that applications can be developed. You could use traditional waterfall methods (analyze, design, code, test, etc.), or you may use a Rapid Application Development (RAD) approach of building the solution in successive smaller increments.

As was mentioned earlier, one of the strengths of architectures is that they provide guidance to assist with decision making. In this case, the initial guidance would come in terms of the type of development life-cycle you should choose. For example, there are some projects where it is better to use a waterfall approach than RAD. Before the project gets too far along, the project manager should evaluate the business requirements against a predefined set of criteria. These criteria lead to guidance on the type of life-cycle to utilize. For instance, if the solution is heavily on-line, and the requirements are not well known, then it may be that a RAD life-cycle would be better. If the solution is heavily batch oriented and requires a lot of integration into other current applications, a traditional waterfall approach might be a better choice. If the solution is actually a major enhancement to an existing application, then an enhancement lifecycle may be more appropriate.

### **Setting up the development lifecycle**

Anyone who has created methodologies knows that they can be very difficult and time-consuming. They can be, but they do not have to be. You can purchase development lifecycle methodologies to use as your starting point, or you can get a group together and brainstorm the processes. One of the critical decisions to make is the level of detail that you provide. A good rule of thumb is to provide enough detail that you give guidance to the project managers, but not so much detail that the methodology gets cumbersome. You can spend a year defining the life-cycle at a very detailed level, but there is a much earlier point where you have covered 80% of what the project manager needs to know. That is the point where the process should end.

In terms of guidance, you also need to determine if there are portions of the life-cycle that are mandatory. If so, these are considered company standards that everyone must follow. For instance, you may have standard templates that must be used at certain points. Your life-cycle can also issue guidelines that are recommendations, but not absolutely mandatory.

### **Defining the technical models**

Remember that one of the purposes of architecture is to provide guidance for decision-making. When an application is being built, there are many decisions to be made. One of the most important decisions is the overall technical design. The technical design is created after business requirements are generated and before the detailed design and coding work begins.

If you look at all of the various applications today, you can start to categorize them into application types, or models. Let's think about a large company that has over 200 separate applications. Regardless of the specific type of application and the type of data



## TenStep Supplemental Paper

---

that it processes, you should notice a handful of application types. This might include web applications, data warehouse applications, decision support applications, transaction processing applications, reporting applications, etc. You will also notice that certain types of models work better for certain categories of business requirements. For instance, you might see that a web application is better for external customers to order products from you. On the other hand, if you have an Accounts Receivable system processing 50,000 transactions a day, a web application might not be the right one. Likewise, business requirements that call for the storage and retrieval of millions of customer order records might point out the need for a data warehouse application rather than a traditional client-server application using normal database processing.

You don't want to be in a position where you have chosen the wrong platform and software. Many times you don't realize this until you start to do heavy systems testing, or worse, when the application goes live. That is way too late to have to re-work fundamental technical design decisions.

Development architecture can help by providing guidance as to the type of application that should be built based on the business requirements. Again, the architecture provides guidance

### **Rationalizing the existing application portfolio**

The third feature of development architecture is to understand and categorize what you already have. It might be surprising to know how many companies really don't have a high-level picture of every application in their company. It should not be a surprise, then, that companies end up redeveloping similar software multiple times.

The major step in this aspect of the development architecture is to take an inventory of all the business applications that exist today, as well as any that are in-progress. This sounds simple, but it can be a huge effort for a large company. You must first be very clear on what constitutes a business application. The ones supported by the IT development organization might be simpler to identify, but what about all of the applications that are created by business users, as well as the applications that are within IT, but managed and supported outside the development department? You must first decide the scope and definition of the inventorying process.

Next, you must determine what information you want to collect on each application. There are many obvious characteristics, such as the purpose, the client base, the types of data processed, and the technical environment. However, there are literally dozens and dozens (hundreds?) of pieces of information you could collect. You want to determine the information that will provide the most value, is easiest to capture, and will not require a huge process to keep up-to-date.

The application inventory is used for two main purposes. First, you can look for opportunities to rationalize wherever possible. One way to rationalize is to look for redundancies - that is, different applications being used for similar needs. For example, you may find two Customer Relationship Management (CRM) packages in use. Further follow-up may determine that you can standardize with one package. You may also



## TenStep Supplemental Paper

---

discover that you have development products only used by a small number of applications.

### **Rationalizing the application inventory is an ongoing process**

You don't make major decisions to retire duplicate applications and older technology in a short timeframe. In fact, in a large company, you may need to look at a five year plan to get you to a more rationalized application environment. You may even decide that you cannot make a business case to eliminate all the inefficiencies. However, this aspect of the development architecture at least gives you the information you need to make appropriate decisions.

The second purpose of the application inventory is to map requests for new development against the current business applications. This can be done as a part of the project approval process. When business clients are looking to build new solutions, they can refer to the development architecture to see if something similar might already exist. The people who are making funding decisions can also see what the new projects are, what business processes they relate to, and what applications exist in that space already. They can catch obvious duplications and ensure that redundant applications are not funded.

### **Summary**

Development architecture provides a framework to guide decision-making. Development architecture can cover three areas – the development lifecycle, the application technical design model, and an application inventory. Development life-cycles provide guidance into how applications should be developed and will save project managers the time and effort required to create development processes from scratch each time an application is built.

As companies get larger, the chances get much greater that development decisions will be made that lead to an inefficient use of company resources. Development architecture can help by providing guidance for the development process at three key areas. First, the development architecture shows you the best development life-cycle approach for the initial development project. Second, the development architecture shows you the best technical design model you should use based on your business requirements. Third, the development architecture gives you visibility into all of the applications in place and in progress so that you do not reinvent a business application that exists or substantially exists today. All of this guidance allows the development work that is approved to proceed more effectively and efficiently, with as little rework and redundancy as possible.