developerWorks®

The benefits of software architecting

Peter Eeles

May 15, 2006

from The Rational Edge: This fourth and final article in a series, Peter Eeles covers the benefits that a business and an IT organization can derive from a sound software architecture.

My first article in this series described what a software architecture is, the second article defined the characteristics of the role of the software architect, and the third article considered the themes, or characteristics, that underly the process of software architecting. This fourth and final article in the series covers the benefits that a business and an IT organization can derive from a sound software architecture.

In general terms, software architecting is a key factor in reducing cost, improving quality, timely delivery against schedule, and delivery against requirements. In this article I will focus on specific benefits that contribute to meeting these objectives. It is also worth noting that, as an architect, we sometimes have to justify our existence. This section provides some useful ammunition for treating architecting as a critical part of the software development process.

Architecting addresses system qualities

The functionality of the system is supported by the architecture, through the interactions that occur between the various elements that comprise the architecture. However, one of the key characteristics of architecting is that it is the vehicle by which system qualities are achieved. Qualities such as performance, security, and maintainability cannot be achieved in the absence of a unifying architectural vision, since these qualities are not confined to a single architectural element but rather permeate the entire architecture. For example, in order to address performance requirements, it may be necessary to consider the time required for each component in the architecture to execute, and also the time spent in inter-component communication. Similarly, in order to address security requirements, it may be necessary to consider the nature of the communication between components, and introduce specific "security-aware" components where necessary. All of these concerns are architectural and, in these examples, concern themselves with the individual components and the connections between them.

A related benefit of architecting is that it is possible to assess such qualities early on in the project lifecycle. Architectural prototypes are often created in order to specifically ensure that such qualities are addressed. This is important since, no matter how good an architecture looks on paper, executable software is the only true measure of whether the architecture has achieved such qualities.

Architecting drives consensus

The process of architecting drives consensus between the various stakeholders, since it provides a vehicle for enabling debate about the system solution. In order to support such debate, the process of architecting needs to ensure that the architecture is clearly communicated and understood. An architecture that is effectively communicated allows decisions and tradeoffs to be debated, reviews facilitated, and agreement reached. Conversely, an architecture that is poorly communicated will not allow such debate to occur. Without such input, the resulting architecture is likely to be of lower quality.

On a related note, the architecture can drive consensus between the architect (and his vision) and new or existing members, as part of training. Again, the architecture must be effectively communicated for this benefit to be achieved. Development teams with a clear vision of what they are implementing have a better chance of implementing the product as desired.

To this end, it's important to document the architecture appropriately, and this is a primary responsibility of the architect. Similarly, the creation of an architectural proof-of-concept is an excellent way of demonstrating that the architecture either does, or does not, meet the stated requirements.

Architecting supports the planning process

The process of architecting supports a number of disciplines. Clearly, it supports the design and implementation activities, since the architecture is a direct input to these activities. However, in terms of the benefits that the process of architecting brings, arguably the major benefits are those related to the support provided to project planning, and project management activities in general: specifically, scheduling, work allocation, cost analysis, risk management, and skills development. The process of architecting can support all of these concerns, which is one of the main reasons why the architect and the project manager should have such a close relationship. Much of this support is derived from the fact that the architecture identifies the significant components in the system and the relationships between them. Consider the UML component diagram in Figure 1, which has been kept deliberately simple for the purposes of this discussion. This figure shows four components with dependencies between them.

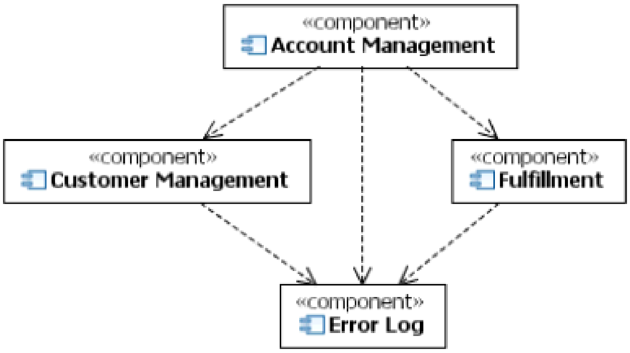

Figure 1: UML component diagram showing architecturally significant elements

In terms of scheduling, the dependencies imply an order in which each of these elements can be considered. From an implementation perspective, for example, the dependencies tell us that the Error Log component must be implemented before anything else, since all of the other components use this component. Next, the Customer Management and Fulfillment components can be implemented in parallel since they do not depend upon one another. Finally, once these two components have been implemented, then the Account Management component can be implemented. From this information, we can derive the Gantt chart (one of the key planning tools of a project manager) shown in Figure 2. The duration of each of the tasks shown does require some thought, but can be partially derived from the complexity of each of the architectural elements.

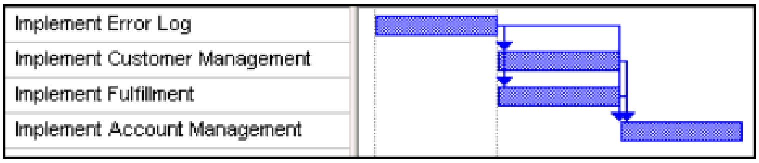

Figure 2: Gantt chart based on dependencies between architecturally significant elements

Clearly, this is a gross simplification for a number of reasons. For example, all of these elements are unlikely to be implemented as a single component, since they're much too complicated for this to be the case. Also, it is possible to create "stubbed" or partial implementations so that implementation of each of the elements can occur in parallel. I use this example to simply demonstrate the principle.

In terms of work allocation, the architecture can again help us identify areas that require particular skills and therefore particular resources (people) to which work can be allocated.

The architect can also assist in the cost estimation for the project. The costs associated with a project come from many areas. Clearly, the duration of the tasks and the resources allocated to each will allow the cost of labor to be determined. The architecture can also help determine costs related to the use of third-party components that will be used in the delivered system, and also the cost of any tools that are required to support the development effort, since one of the activities that the architect is involved in is the selection of an appropriate development environment that will allow the designers, implementers and other team members to work together in an effective manner.

Another focus of the architect is to identify and manage technical risks associated with the project. The management of technical risk involves the prioritization of each risk, and the identification of an appropriate risk mitigation strategy. The priorities and risk mitigation strategies are provided as input to the project manager.

Finally, the architecture identifies discrete components of the solution that can provide input in terms of the skills required on the project. If sufficient resources are not available within the project or within the organization, then this clearly helps identify areas where skills acquisition is required. This may be achieved through developing existing personnel, through outsourcing, or through new hires.

Architecting drives architectural integrity

One of the primary objectives of the process of architecting is to make sure that the architecture provides a solid framework for the work undertaken by the designers and implementers. Clearly, this is more than simply conveying an architectural vision. In order to ensure the integrity of the resulting architecture, the architect must clearly define the architecture itself, which identifies the architecturally significant elements, such as the components of the system, their interfaces, and their interactions.

The architect must also define the appropriate practices, standards, and guidelines that will guide the designers and implementers in their work. One of the objectives of architecting is to eliminate unnecessary creativity on the part of the designers and implementers, which is achieved by imposing the necessary constraints on what they can do and making it clear that deviation from the constraints can break the architecture. For this reason, the adoption of appropriate review and assessment activities helps ensure architectural integrity. A focus of these activities is to consider the work of the designers and implementers, and determine the degree of adherence to the architectural standards and guidelines that have been put in place.

Architecting helps manage complexity

Systems today are more complex than ever, and this complexity needs to be managed. Since an architecture focuses on only those elements that are significant, it provides an abstraction of the system and therefore provides a means of managing complexity. Also, the process of architecting considers the recursive decomposition of components. This is clearly a good way of taking a large problem and breaking it down into a series of smaller ones.

Managing complexity is further enabled by techniques that allow abstractions of the architecture to be communicated. The adoption of industry standards that allow expressing such abstractions, such as UML, is therefore commonplace in the industry today for documenting software systems.

Architecting provides a basis for reuse

The process of architecting can support both the use and creation of reusable assets. Reusable assets are beneficial to an organization, since they can reduce the overall cost of a system and also improve its quality, given that a reusable asset has already been proven (since it has already been used).

The creation of an architecture supports the identification of possible reuse opportunities. For example, the identification of the architecturally significant components and their associated interfaces and qualities supports the selection of off-the-shelf components, existing systems, packaged applications, and so on, that may be used to implement these components. The architecture itself may also prove to be reusable as a reference architecture for subsequent systems. Even components within the architecture may be deemed potentially reusable in other contexts.

Although the process of architecting can identify reuse opportunities within the current project, there is a much greater impact when reuse is considered across projects and across the enterprise.

Any discussion of reuse should be tempered with a word of caution. Very few reuse programs have been successful to date, for a variety of reasons, some technical and some not. From a technical perspective, a reuse program needs to ensure that appropriate standards, processes, and tools are in place. Fortunately, some foundation elements are starting to be addressed. A good example is the standardization of the Object Management Group's Reusable Asset Specification (RAS),[1] which defines a standard for describing reusable assets, packaging reusable assets, and for interfacing to a RAS repository service.

From a non-technical perspective, there are organizational considerations that must be kept in mind when implementing a reuse strategy. For example, who is responsible for creating a reusable asset given that it will be used in many contexts? Who maintains the asset once it has been created (and the team that created the asset disbanded)? What incentives are in place for the creation and use of reusable assets? How is the cost of creating a reusable asset, which is almost always more costly than creating one that is not, justified?

Architecting reduces maintenance costs

The process of architecting can help reduce maintenance costs in a number of ways. First and foremost, the process of architecting should always ensure that the maintainer of the system is a key stakeholder and that their needs are addressed as a primary concern, not as an afterthought. The result should not only be an architecture that is appropriately documented in order to ease the maintainability of the system; the architect will also ensure that appropriate mechanisms for maintaining the system are incorporated, and will consider the adaptability and extensibility of the system when creating the architecture.

The architect should also consider those areas of the system most likely to require change and work to isolate them. This can be fairly straightforward if the change impacts a single component, or a small number of components. However, it should be acknowledged that some changes, such as those relating to system qualities such as performance or reliability, cannot be isolated in this way. This is why the architect must ensure that, when architecting the current system, they consider any likely future requirements, since scaling up a system to support thousands of users rather than tens of users, for example, is not normally possible without changing the architecture in fundamental ways.

Architecting supports impact analysis

An important benefit of architecting is that it allows us to reason about the impact of making a change before it is undertaken. An architecture identifies the major components and their interactions, the dependencies between components, and traceability from these components to the requirements that they realize.

Given this information, a change to a requirement, for example, can be analyzed in terms of the impact on the components that collaborate to realize this requirement. Similarly, the impact of changing a component can be analyzed in terms of the other components that depend upon it.

Such analyses can greatly assist in determining the cost of a change, the impact that a change has on the system, and the risk associated with making the change. This information is then used when prioritizing changes and negotiating those changes that are absolutely necessary.

Acknowledgements

The contents of this article have been derived from a forthcoming book, provisionally entitled "The Process of Software Architecting." As a result, the content has been commented upon by many individuals that I would like to thank, who are Grady Booch, Dave Braines, Alan Brown, Mark Dickson, Luan Doan-Minh, Holger Heuss, Kelli Houston, Philippe Kruchten, Nick Rozanski, Dave Williams, and Eoin Woods.

Notes

[1] See "Reusable Asset Specification," Object Management Group Inc., Document number 04-06-06: June 2004.