Selecting Architecture Products for a Systems Development Program

Michael S. Russell
Anteon Corporation

Determining what architecture products are needed to support software development within a larger systems engineering process is a challenge. Existing documents such as the Department of Defense Architecture Framework provide some guidance, but no defined product identification process. The method proposed in this article provides a repeatable process for selecting the architecture products required to support a larger systems engineering effort, defines the content of each product, and identifies each product's customer.

The purpose of architecture is to answer questions. In other words, the architecture provides the information needed by decision makers during the course of the systems engineering process to define concepts and processes, allocate functionality, define test metrics, design software, and make other development decisions. However, the output of the systems engineering (SE) process is not the architecture; rather, it is the system being produced [1]. Unfortunately, the architecture often becomes divested from the SE process it is meant to support, becoming an output or deliverable in and of itself [2].

In part, this is perpetuated by government regulations specifying that each acquisition program produce architecture with little thought as to how architecture will be used by the program [3, 4, 5]. So, the output of most architecture efforts tends to be a three-ring binder that weighs five pounds or so, which no one ever reads. This has given a bad name to the architecting process, and has left many decision makers asking why they spent their limited money and time producing architectures.

To correct this situation, both the software-centric and system-centric communities need to reexamine the architecting process, rediscover the intended uses for architecture, and ensure architecting is always done in support of the SE life cycle. For software designers, this will mean a break from the concept that four or five standard Unified Modeling Language (UML) diagrams will solve the needs of all stakeholders. This article presents one method for ensuring the architecture is producing the products needed to support the overall SE process.

Tying Architectures Into the SE Process

The SE process has been described as an elaborate engineering decision process that includes the following [6]:

- Begins with understanding the system requirements and specifications.
- Translates these specifications into a conceptual design in the form of a functional architecture.
- Translates this functional architecture into logical design or physical architecture.
- Translates the physical architecture into a detailed design or implementation architecture for the system ultimately to be produced or acquired.
- Completes development through the production of a product that conforms to this architecture, potentially through various strategic sourcing efforts and associated integration and test.

"Developing architecture as part of a program's SE [systems engineering] process can be a critical component of the program's success, but only if the right products are identified and produced at the right time for the right customer."

These steps show the SE process for what it really is – a series of decision points leading up to a delivered product. This means the architecture's products must be tied to the rest of the SE process and, more specifically, to decision points within that process. This allows the architect to determine what products need to be produced, when to produce them, and what level of detail needs to be present. To tie architecture products to decision points, the architect must understand what decisions need to be made. It is a process that begins with these questions:

1. What is the output of my SE process?
2. What decisions need to be made to produce the output?
3. What sort of information is needed to make those decisions?
4. Which architecture products provide the information in a format that is understandable to the decision maker?
5. When are the architecture products needed?
6. How should my project be staffed to produce those products?
7. Which software tools will help the architect build those products?

For example, if the output of the SE process is an electronic timecard system, then decisions include these: "What are the expectations of its users?" "How many employees will use it concurrently?" "What technology options exist?" and "Which technical option works best with the existing timekeeping business process?" Concerning the time card software itself, the decision maker would probably like to know the advantages and disadvantages of different ways of entering time card data, ease of updating and maintenance, and how the software will interoperate with the business' existing software.

Each of these decisions is made at a different point in the SE process, and many of the decisions are interdependent. It is the architect's job to recognize this and suggest the appropriate mix of architecture products that will provide the needed information – in essence, answer the questions. Figure 1 describes this process.

Please note that the last step in the process concerns what software tools are needed to build the architecture products. Too often, a project starts with the notion, "We're using XX company's tool," without considering how that software will be used or even if it can be used within the program's existing SE process. In fact, many of these tools assume an SE process, and the software is optimized to produce architecture products designed for that process (i.e., the Rational Unified Process) [7]. While this is not necessarily a bad thing, it does mean that the chosen tool is driving product selection and the overall process to produce the architecture products. Sometimes this situation is directed such as a program that is required to use a model-driven architecture [8] approach. Ideally, the software tool should be chosen after the products to be built have been identified.

Remember that decisions made early in the SE process require different information than those decisions made closer to the end. For instance, describing the business process that a new system will support is done early on, while defining software to support that process is done later. Understanding when information is useful to a decision maker is just as important as understanding what information he or she needs.

Finally, a system has many customers. The electronic timecard system described in the earlier example has an obvious customer: the employee. However, there are many other customers such as the system developers, the company's executives, time card system vendors, and the company's accounting department. Each of these customers has an interest in the new system, but each makes different decisions and needs different information to make those decisions. A well-built architecture takes all of these needs into consideration and is capable of producing the right architecture product, in the optimal format, to meet each customer's need.

The Product Identification Process

The product identification process outlined in this article is a matrix-based approach used successfully by several DoD organizations over the past five years, and is also taught in the Federal Enterprise Architecture Certification (FEAC) Institute's [9] certification program. The matrices can be produced with most any office automation or database software. However, the matrix is normally put together in a spreadsheet. To help examine how the process would be implemented, a completed set of matrices using the electronic timecard example are included in the online version of this article at <www.stsc.hill.af.mil/crosstalk/2005/11/0511Russell.html>. The matrices presented below contain a subset of the same information.

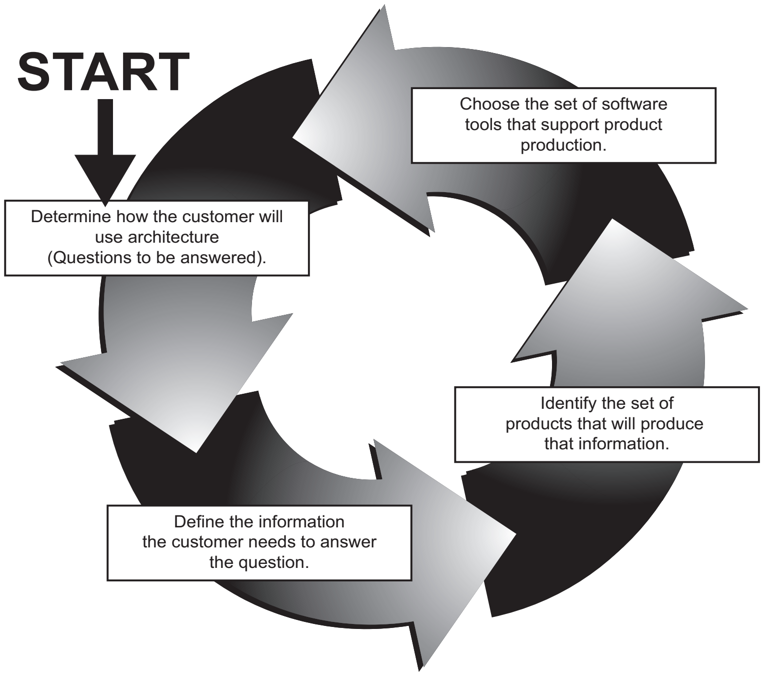
START
Choose the set of software tools that support product production.
Determine how the customer will use architecture (Questions to be answered).
Identify the set of products that will produce that information.
Define the information the customer needs to answer the question.

Figure 1: *Product Identification Process*

Step One

Step one focuses on identifying the architecture's customers and determining each customer's questions or decisions they need to make. In the Customer-Question Matrix (Figure 2), the customers are listed along the top axis and a list of questions along the left axis. Next, the information needed to answer each customer's question is added to the matrix. Developing this matrix can be greatly improved by letting each customer see the responses of other customers. This typically results in less question redundancy, especially when the information needs between customers are similar.

Step Two

In step two, the focus shifts from the customer's questions and decision points to which architecture products provide the information needed to answer the questions or support a decision. It is important to remember that even when several customers need similar information, the architecture product and the level of detail for that product might be different. For example, a business executive and a programmer would both need to understand the concept for the company's new timecard

Figure 2: *Customer-Question Matrix*

	Customers:			
Questions:	**Business Owner/ Managers**	**Finance Department**	**Information Technology (IT) Department**	**Users**
How will it be deployed?	Deployment milestones, overarching plan, IT integration plan.	Deployment milestones, overarching plan.	Deployment milestones, overarching plan, interaction with development team.	When the user must start using it.
What data has to be saved?	Audit data requirements, payroll data requirements, time card business rules.	Payroll data requirements, time card business rules.	Frequency of data backup and storage requirements.	Need to track available vacation time.
What is the testing plan?	Testing plan schedule, integration into overall development plan.	Finance department test input.	IT department test input and responsibilities.	User testing plan.

Information Needed by Each Customer:	Architecture Products:					
	System Concept and Program Plan	**Activity and Process Diagrams**	**Business Rules Description**	**Web-Portal Screen Mock-up**	**Regulation Compliance Traceability Matrix**	**Data Model**
System Concept Information	White paper			Graphics		
Federal and State Timekeeping Regulation Compliance			Text document		Spreadsheet	
Data Needed for Auditing, Payroll, Vacation Tracking, and Other Requirements		UML activity diagram	Data flow diagram	Graphics	Spreadsheet	UML class diagram
Time Card Usage, Review, and Auditing Business Rules		UML activity diagram	State chart		Spreadsheet	UML class diagram

Figure 3: *Product-Information Matrix*

system, but the level of detail needed by each would be much different.

For this Product-Information Matrix (Figure 3), the information needed by each customer is listed on the left axis, and architecture products that could provide that information are listed along the top axis. Quite often, the information needed will span several architecture products. For instance, determining who can update a company's human resources database and under what conditions an update can occur may require activity diagrams, a logical data model, a business rules model, and other architecture products.

When possible, information needs as described by the customers should be consolidated so the matrix does not become unwieldy, but only when the information needed is at the same level of detail. Although two customers may need the same type of information to answer their question, the information they need may reflect a different level of detail. For example, if a business owner wants to understand database access permissions, then a business rules model might be the best architecture product to provide the information. However, the database developer would need a data model based on the business rules to build the database. Few business owners will look at or understand a data model, but it is the right product for the developer.

Step Three

The next step in the process matches each customer with the set of architecture products they need to answer questions and make decisions in a timely manner. This step gives the architecture developers the information they need to plan the architecture development process and integrate it into the overall SE process. The Customer-Product Matrix (Figure 4) supports this mapping, identifies when each product is needed, and highlights dependencies between products. In this matrix, the customers are again listed along the top axis, while the architecture products are listed on the left axis. Delivery dates, product formats, i.e., activity diagram versus data flow diagram, metrics, and other information should be used to indicate the mapping between customers and products.

It is likely that a specific product such as the *activity model* will be identified as being used by multiple customers. Keep in mind that each customer will probably use his or her activity model in a different manner, which should be captured in the *information* section of the first matrix. This means that there probably will not be one activity model for the architecture. Rather, there will be several views of the activity model that are relevant to a specific customer's information needs, derived from the same pool of architectural data.

Step Four

Now that architecture products needed to support the SE process have been identified, the last step is to choose the software tools that the architects will use to produce products in a format useful for each customer's decision-making process. Typically, an executive-level decision maker will not want to look at a product as displayed in a Computer Aided Systems Engineering (CASE) tool, while a software developer will not get the information they need from a Power Point presentation. So the architecture should rely on a suite of tools to produce, store, and display the architecture's products. Each tool serves a defined purpose within the architecture, and together they support the creation and integration of the architecture.

Generally, there are four types of software tools used to support architecture development: CASE tools, databases, executable modeling tools, and Web sites. Normally, one tool of each type is needed because each customer has different needs for viewing and using his or her architecture products. One word of caution, even though many vendors will try to sell a *one-size-fits-all* software solution, very few tools support all aspects of your development process. So choose the tool that provides the information needed by that customer to support his or her decision-making process. Do not expect a customer to modify his or her SE process or decision-making criteria just because your software cannot deliver the architecture in a form he or she can use.

Other Uses for the Matrices

Understanding what products to produce is only half the challenge. The next chal-

Figure 4: *Customer-Product Matrix*

Architecture Products:	Customer:			
	Business Owner/ Managers	**Finance Department**	**IT Department**	**Users**
System Concept	DEC	DEC	DEC	
Activity and Process Diagrams	JAN – High Level APR – Final	JAN – High Level MAR – Detailed	JAN – High Level MAR – Detailed	MAR – Detailed
Business Rules Description	JAN – Initial	MAR – Detailed	JAN – Initial MAR – Detailed APR – Final	
Regulation Compliance Traceability Matrix	APR			
Data Model		MAR – Business Rule Text	FEB – UML Class Diagram	

lenge is determining how to staff the development project, including how many people to hire and what skills those people should have. The information contained in the matrices can also help answer these questions. For example, data models are generally produced later in the SE life cycle than system use cases are produced, meaning a project may not need to hire a data modeler at the very beginning of the project. Likewise, a completed set of matrices will normally show a need for several systems analysts toward the beginning of the life cycle, one or two during the middle, and a large group at the end during the test and evaluation stage.

So, by matching the expected delivery dates for each architecture product captured in the matrix, a good initial staff skill set and loading matrix can be produced. Staffing choices are not always intuitive, and the completed matrices can significantly help the program manager justify his project funding requirements by having hard numbers to base his staffing plan upon.

Conclusion

The purpose of architecture is to provide the information needed by decision makers to make decisions. Developing an architecture as part of a program's SE process can be a critical component of the program's success, but only if the right products are identified and produced at the right time for the right customer.

The matrix-based approach outlined here has proven to be a repeatable and successful process for its users. When used early in the SE life cycle, it helps focus developers on how the architecture should be used by zeroing in on the questions to be answered and the information needed to answer those questions. During production, it helps set customer expectations concerning the types of products to be produced, the level of effort and skills required to produce the products, and when each product will be delivered. At any time during the life cycle, the information can be used to show what decisions need to be made and what architectural information these decisions should be based on.

Lastly, the matrices form a body of knowledge that can be reused when planning future projects.◆

References

1. Rechtin, Eberhardt, and Mark Maier. The Art of Systems Architecting. Boca Raton, FL: CRC Press, 1997.
2. Maier, Mark, David Emery, and Rich Hilliard. "ANSI/IEEE 1471 and Systems Engineering." Systems Engineering 2.3 (1990): 168-176.
3. Office of the Secretary of Defense: Acquisition Technology and Logistics. DoD 5000.1 The Defense Acquisition System, Washington: Department of Defense, 2003.
4. The Joint Staff. CJCSI 3170-01E, The Joint Capabilities, Integration and Development System, Washington: Department of Defense, 2005.
5. Office of the Secretary of Defense National Information Infrastructure. The DoD Architecture Framework Vers. 1, Washington: Department of Defense, 2004.
6. Sage, Andrew, and Charles Lynch. "Systems Integration and Architecting: An Overview of Principles, Practices, and Perspectives." Systems Engineering 1.3 (1998): 176-227.
7. Kruchten, Philippe. The Rational Unified Process. Reading, MA: Addison Wesley Longman Inc., 1998.
8. Miller, Joaquin, and Jishnu Mukerji. MDA Guide Vers. 1.0.1. Needham, MA: Object Management Group, 12 June 2003.
9. The Federal Enterprise Architecture Certification Institute. FEAC Institute. 25 Feb. 2005 <www.feacinstitute.org>.

About the Author

Michael S. Russell is the director of Enterprise Architectures for the Anteon Corporation's Systems Engineering Group. He has served as lead architect on numerous federal, Department of Defense, and industry enterprise architecture efforts. He is a lecturer with the Federal Enterprise Architecture Certification Institute and is a member of the International Council on Systems Engineering. He has taught courses in systems engineering for the past seven years, and has published several articles on systems engineering topics. He has a Master of Science in systems engineering from George Mason University.

Anteon Corporation
2231 Crystal DR STE 600
Arlington, VA 22202
Phone: (703) 769-6160
E-mail: mrussell@anteon.com

Coming Events

December 4-7
WSC '05
Winter Simulation Conference
Orlando, FL
www.wintersim.org

December 5-8
The 26th Institute of Electrical and Electronics Engineers Real-Time Systems Symposium
Miami, FL
www.rtss.org/rtss2005.html

December 5-9
Annual Computer Security Applications Conference ACSAC 2005
Tucson, AZ
www.acsac.org

December 7-9
I-SPAN 2005
International Symposium on Parallel Architectures, Algorithms, and Networks
Las Vegas, NV
http://sigact.acm.org/ispan05

December 11-14
ICIS 2005
International Conference on Information Systems
Las Vegas, NV
http://icis2005.unlv.edu/

December 19-21
Collaborate COM 2005
San Jose, CA
www.collaboratecom.org/index.php

May 1-4, 2006
2006 Systems and Software Technology Conference

STC Systems & Software Technology Conference

Salt Lake City, UT
www.stc-online.org