Documenting software architecture, Part 2: Develop the system context

Skill Level: Introductory

Tilak Mitra (tmitra@us.ibm.com)
Senior Executive IT Architect
IBM

13 May 2008

In this series, learn why and how you should document software architecture. This second article provides guidance for documenting your system context information. The system context is the first architecture artifact you should capture. Learn how to use a system context diagram and information flows to develop and document the system context for your system or application's software architecture.

Introduction

Part 1 of this series explained the importance of a disciplined approach to documenting software architecture. It also introduced commonly used mechanisms to capture the architectural artifacts used in a typical software development process. This article continues the discussion and focuses on the first important architecture artifact: the *system context*. Learn how to document the system context information with diagrams and information flows.

At a very high level, you should consider two views of a system:

- In the most common view, the software or application is represented by a set of architecture building blocks (also called the overview architecture).
- The second view is more outward-facing. Think of the system to be designed as a black box. Surrounding the system are the users and roles that access the system, and in which contexts they access it. The diagram should also depict each of the external systems that the system would need to interact with in order to perform specific functions.

The black box representation of the system context shows you how your software is dependent upon external systems, and how the dependencies need to be architected into the overall solution.

The role of the system context in software architecture

The system context is a fundamental artifact in the software architecture of a system. Developing the system context view is important, because this view is used as a mechanism to trace back to the business context, and downstream to the functional and operational architecture. We shall provide a brief overview of the business context to understand why traceability to it is important.

The business context

Provides an organizational view of how the system needs to interact with other enterprises to depict the business ecosystem where the software will reside. This view is particularly important in systems with a high dependency on external organizations. The high-level view does not differentiate between the various users and roles. Rather, it depicts them as a user community that interacts with the business.
For example, if you are building software for a university, the business context might depict the university as a central entity and represent dependencies on:

- The government, to request funding and to obtain and perform regulatory conformance checks.
- The IT industry, to request research projects and educational services.
- The user community, to which the university will provide hardware and software support.
- Other universities in the consortium, to obtain student history and records.

The system context

Uses the business context to identify the external organizations. After the external organizations are identified, the system context will identify the specific IT systems and applications that the system will need to interact with in order to receive or send information. It does the same exercise for each of the external organizations, and it collectively creates a system level view that shows which external systems need to be brought into the scope of the overall solution.
The system context provides a decomposition of the business context and provides traceability to the business context information.

The system context helps identify some key architectural artifacts that will be

required in order to build the complete solution. The information flow between the system-to-be-built and each external system provides key inputs to the information model. The characteristics of the external system determine the need for adapters that will facilitate the technology integration. The information flows also represent architecturally significant activities that can be traced back to the business process models, which represent a key portion of the system's requirements.

The importance of the system context cannot be underestimated. It plays a very important role in developing the software architecture of an application system. I strongly recommend that you document the system context.

Documenting the system context

The first task in documenting the system context is to create a system context diagram. As shown in Figure 1, the system context diagram:

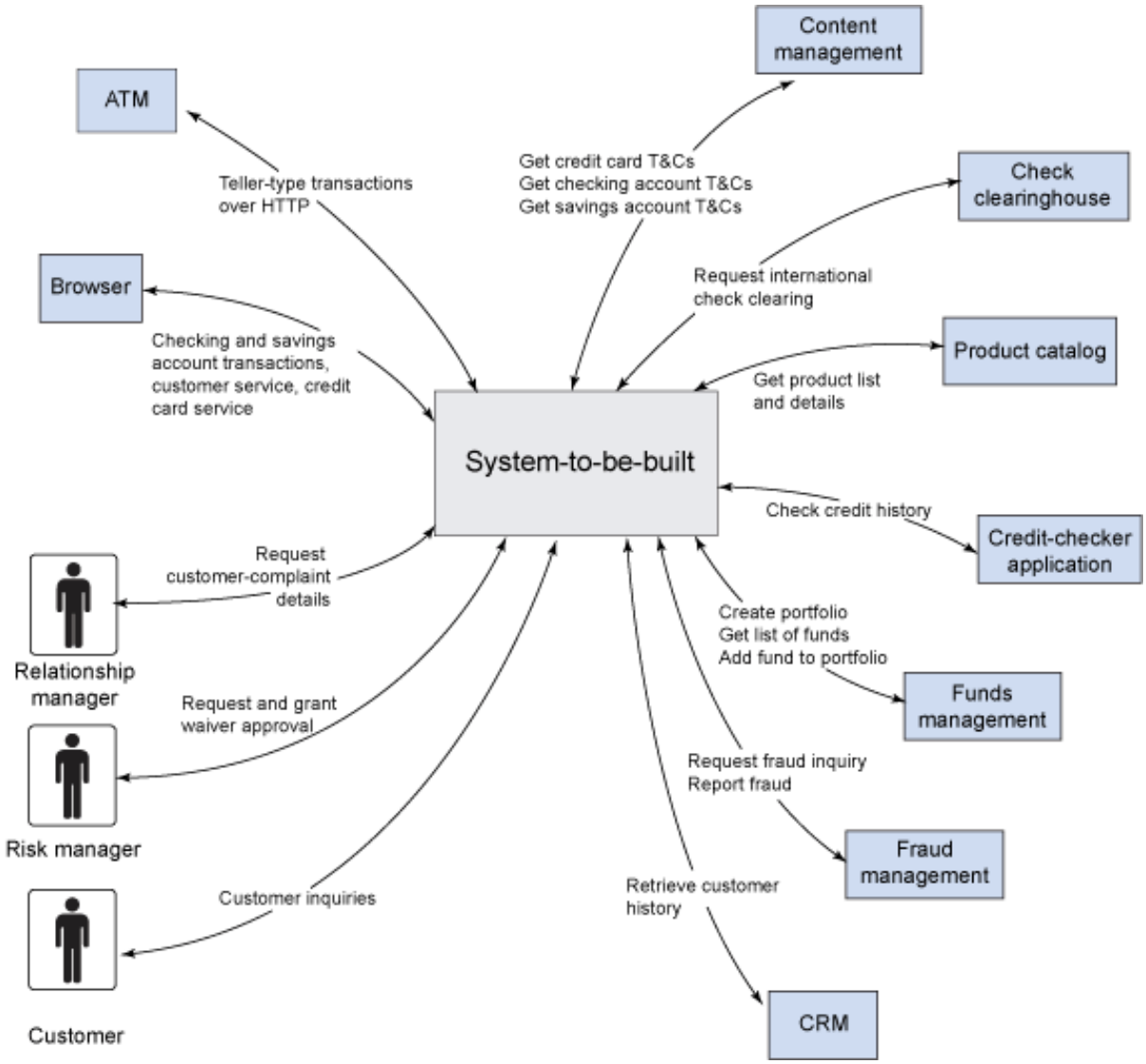
- Represents the system-to-be-built as a black box
- Depicts its interaction with external entities (systems and end users)
- Identifies the information and control flows between the system and external entities

External entities are not necessarily systems that are outside the enterprise perimeter. An existing enterprise application or a database can be represented as an external entity in the system context. I recommend that a section be dedicated to the system context diagram and given a self-explanatory name, such as "System context diagram."

System context diagram

Figure 1 shows a system context diagram for a banking application.

Figure 1. System context diagram



It's a good idea to supplement the diagram with detailed examples of each artifact.

The system in the center box in Figure 1 has a generic name. Of course, you would substitute a name to represent the application that is to be built.

Artifacts fall into the following categories:

Users and roles

These artifacts show the users and roles that interact with the system. Although there's no hard and fast rule, it's common practice in IT to show users and roles to the left of the system. I recommend that you create a subsection under the main section in which the users and roles will be documented. Users are usually categorized by roles.

For example, in Figure 1 there are two types of roles: the relationship manager and the risk manager. The documentation for each role should have:

- A description of the role and the context in which it accesses the system.
- A description of the information for which the role accesses the system.
- The volume of transactions that a typical user in a given role would be performing in a given unit of time.

Channels

Users will use different channels to access the system. I recommend that you create a separate subsection to document this channel information. Documentation for each of the channels should minimally capture:

- A description of the channel and the type of roles or users that typically use it to interact with the system. For example interactive voice response (IVR), browser, smart phone, and so on.
- The network and bandwidth supported by the channels, such as T1 line, 8.02 11g, a partial T3, and so on.
- The access protocol to send and receive data to and from the system, such as HTTP, sockets, IVR, and so forth.

External systems

You must document the external systems that your system interacts with when it's carrying out the required functions. A lot of analysis takes place behind the scenes to identify the external systems that need to be brought into the scope of the solution. The business analyst and the domain experts usually participate in this analysis. You should also sufficiently document the results of this analysis.
It is recommended that you dedicate a subsection to the documentation of external systems. This documentation should minimally capture:

- A descriptive overview of the external system, including background information about the placement of the system in relation to the system to be built.
  For example, the external system may be placed inside the enterprise intranet, in the extranet as defined by the business, or on the Internet.
- The access protocol required to interface with the external system, such as secure HTTP, sockets, a proprietary access mechanism, and so on.
- The data formats supported or expected by the external system to facilitate integration.
- Any specific compliance requirements that need to be followed in order to interact with the external system.

- Nonfunctional specifications of the system, such as security, availability, information throughput, and so on.

Not all nonfunctional requirements for the external systems need to be documented. Document only those that may influence the architecture and design of the system that needs to be built.

When documented adequately, the previous information should provide a good description of the system context diagram. However, the information captured so far provides only a static view of the system context, represented through the users, roles, channels, and external systems. Identifying and capturing the information that is exchanged between the system and each of the external systems provides the dynamic view of the system context. The next section addresses this information flow.

Information flow

Information that flows between the system and the external systems, users, and channels is an essential part of your system. Information can flow in batches or in real time. Documenting the information and its characteristics as a part of the system context is paramount when defining the overall software architecture.

The information flow is usually represented using a short phrase that can either be a noun or a verb. Whether you choose a noun or a verb is a matter of preference, but it is recommended that you stick to one form instead of using both. I use a verb form in the example. For each of the information flows, I recommend that, at a minimum, you document the following set of artifacts:

- A description of the information that is flowing between the system and the users, channels, and external systems.
- Categorize the information as batch, real time, or semi-real time.
- The number of transactions that must be supported per unit of time.
- The type of data that constitutes a typical transaction.
- The volume of data that will typically flow per transaction.
- The frequency with which transactions are executed.

These artifacts, as stated earlier, do not address the sequence of the interactions between the system and the external entities. When there is information flowing between two systems, there can be a sequence of information exchange between the systems that completes a transaction. In such situations, the sequence of information exchange should also be documented.

The importance of documenting the information flow is manifold:

- The information flow identifies important information that will influence the final information model for the software to be built.

- It helps you to understand the data formats supported by the external system by analyzing the information elements.
  For systems outside the enterprise perimeter, the data format is usually different from the format that's prescribed to be supported by the system, which means the two interacting systems will also require data translation.

- The access protocol (network and data) supported by the external system may be different from the protocol that will be supported by the system-to-be-built. This protocol disparity raises technology requirements for application and systems integration. The requirements are usually addressed through the choice of technology adapters.
  A technology adapter normalizes the data format and access protocol differences between external systems and the system-to-be-built. The choice of technology adapters is an important facet of an integration architecture that supports the system to be rolled out.

- Business process modeling is a top-down approach for requirements gathering and for analyzing and understanding the business processes that are in the scope of the business or IT transformation initiative. Process decomposition identifies a set of subprocesses, activities, or tasks that constitute a larger business process. Some activities or tasks require interaction or integration with external systems, such as data dependencies between one system and the other. Such activities can be traced back to one or more information flow definitions from the portfolio of information flows. This provides key traceability between the requirements and their implementation dependency on external systems. Traceability is a fundamental tenet of an efficient and well-organized software development life cycle.

- The data, protocol, and network adapters are essential recipes that go into the definition of the architecture overview of the system or application. In effect, the heterogeneity of the external systems influences multiple layers of the architecture (to be covered in the next article of this series).

With the system context diagram and the information flows, you can define a rigorous system context document that will prove to be very important in shaping the software architecture of the system.

Conclusion

In this article, you learned about the system context view of software architecture. The system context should be one of the first artifacts that you develop as a part of the system architecture. You learned about effective ways to develop thorough documentation of the system context. This article also highlighted the importance of documenting the system context as it pertains to the traceability of developed artifacts in a typical software development life cycle.

Stay tuned for subsequent articles in this series, which will elaborate on a set of guidelines for documenting the rest of the architecture artifacts.

Resources

Learn

- Read Part 1 of this series, "What software architecture is, and why it's important to document it" (developerWorks, Apr 2008).
- Read a compendium of published software architecture definitions.
- Read the two-part series "From business modeling to Web services implementation" (developerWorks, Feb 2005).
- Learn about "The information perspective of SOA design, Part 1: Introduction to the information perspective of a Service Oriented Architecture" (developerWorks, Jan 2008).
- "*Architecture in practice* is a developerWorks series by Tilak that delves into SOA.
- Get an RSS feed for the series *Documenting software architecture*.
- In the Architecture area on developerWorks, get the resources you need to advance your skills in the architecture arena.
- Learn more about SOA in *Executing SOA: A Practical Guide for the Service-Oriented Architect*, a recently published developerWorks series book that Tilak coauthored. Use coupon code IBM3748 for a 35 percent discount.
- Browse the technology bookstore for books on these and other technical topics.

Get products and technologies

- Download IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.

About the author

Tilak Mitra
Tilak Mitra is a Senior Certified Executive IT Architect in IBM. He specializes in SOAs, helping IBM in its business strategy and direction in SOA. He also works as an SOA subject matter expert, helping clients in their SOA-based business transformation, with a focus on complex and large-scale enterprise architectures. His current focus is on building reusable assets around Composite Business Services

(CBS) that has the ability to run on multiple platforms like the SOA stacks for IBM, SAP and so on. Tilak recently coauthored a developerWorks series book: *Executing SOA: A Practical Guide for the Service-Oriented Architect*, available in Resources. He lives in sunny South Florida and, while not at work, is engrossed in the games of cricket and table tennis. Tilak did his Bachelors in Physics from Presidency College, Calcutta, India, and has an Integrated Bachelors and Masters in EE from Indian Institute of Science, Bangalore, India. Find out more about SOA at Tilak's blog. View Tilak Mitra's profile on LinkedIn or e-mail him with your suggestions at tmitra@us.ibm.com.

Trademarks