SADE: SA Design as Ethnographic for Software Architecture

M. Sunil Kumar, Dr. A. Rama Mohan Reddy
1SVUCE, S.V. University, Tirupati, AP, India
2S.V.U.College of Engineering, S.V.University, Tirupati, AP, India

Abstract
“Software Architecture” or SA Development is not a new term in software development life cycle. Starting from a small web site development to application or software development, Software Architecture plays a very important role. Many people believe, SA development is just about coding HTML, CSS and JavaScript but SA is way beyond these technical terms. SA Development is not limited to Website development but extended to any kind of application development including client server. Even in wireframes and Prototyping we make use of SA technologies. A user is not going to experience a software or website based on the backend technology used, but based on the Software Architecture and its experience.

The intention of this topic is to understand
•   The role and importance of Software Architecture development in the SDLC
•   How SA Development can be catered in each phase of SDLC starting from the Requirement gathering phase.

Keywords
SA, UML, SDLC, Ethnographic

I. Introduction
Product development is the process of designing, creating, and marketing an idea or product. The product can either be one that is new to the marketplace or one that is new to a particular company, or an existing product that has been improved. In many instances a product will be labeled new and improved when substantial changes have been made.

Today Product development is becoming a need for every business, from a simple website to ecommerce websites, banking domain to mobile technology every business needs software or a product to communicate with the users.

We can categorize the product into this broad bucket viz., Consumer Products: Public Safety, Rugged Handheld Computers Utilities, Mobile & Fleet Management, Mapping & GIS, Embedded Systems: Portable Devices In-Vehicle Navigation & Telematics Tracking, Instrumentation, Security, Engineering & Construction: Construction, Construction Asset Management, Marine Construction, Mining, Surveying, Infrastructure, Agriculture Solutions: Precision Agriculture, Guidance, Flow and Application Control, Water Management, Information Management, Advance Devices: Defense, Precise Timing. Each of these products need a Software Architecture through which a user can accomplish their tasks. Having the best Software Architecture provides the best customer experience and today’s competitive market-place demands that companies maximize the ease of use for their customers. All product development goes through a similar development lifecycle, starting from requirement gathering to testing phase. Unfortunately, in the development cycle of most of the applications, the Software Architecture design does not begin until most of the development has been completed, and is usually starved for time and resource. The results are hardly positive. Either the SA design is severely affected by assumptions made during the application development or additional resources are added to rework or completely re-program all or part of the application to accommodate new design. This leads to problems like design compromises, delay, budget overruns etc. Given this the most effective strategy is to integrate and design the SA parallel to each phase of development lifecycle. In short, integrating SA design in the development process has several businesses benefits as follows:
•   Create a better product
•   Increase customer satisfaction and retention
•   Allow greater flexibility in responding to customer feedback
•   Lower development costs
•   Shorten the development lifecycle

II. Integrating SA Design in the SDLC Process

Software Architecture development process can be categorized into 4 phases as below:
•   Research and Analysis
•   Design and Branding
•   Prototype Development
•   Production

This structured process can maximize communication between the company and its clients, lower development costs, and most importantly deliver the best Software Architecture design. This process is flexible and modular, thus allowing client to determine which phase are appropriate for their needs, and can be applied to both the creation of Software Architecture for new applications as well as enhancements of existing applications.

A. Research and Analysis

1. Process
In traditional Research & Analysis phase, two categories of information are gathered and analyzed by the user experience team.

Fig. 1: SDLC Process
This establishes the context for Software Architecture design and this context informs the entire design process. Additionally, during this phase as a part of SA development, SA developers can start researching about current market trends, SA technologies, and competitive analysis. This research can help the UX team during the consecutive phase of SDLC and both SA and UX team will have clarity of things possible and impossible and LOE (level of effort) required during development.

For existing applications the Research & Analysis phase also determines the strengths and weakness of current SA.

2. **How?**
The user data is gathered answering the following questions

1. Who are the users?
2. What are their skill levels?
3. What are their qualitative expectations for the application?
4. What task do they need to perform with this application?
5. Under what environmental conditions will they use this application?
6. Under what time conditions and constraints will they use this application?

The application information is gathered answering the following questions

1. What problem is the application intended to solve?
2. What are the functional purposes of the application?
3. What are the operational purposes of the application?
4. What are the marketing purposes of the application?
5. What are the Software Architecture specifications of the application?
6. What are the software, hardware and graphical requirements to create and deliver the application?

The Software Architecture trends can be gathered answering the following questions

1. What are the current SA trends?
2. What are the current SA design patterns?
3. Which are the latest tools and technologies being used?
4. What tools and technologies are our competitors using?
5. Pros and Cons of using any latest technology for the current product?

3. **Deliverables**

1. Personas
2. Competitive landscape
3. Usability goals
4. Functional and non functional specifications

**B. Design and Branding**

1. **Process**

During the design and branding phase, Software Architecture design is created that addresses the specific needs identified in the research & analysis phase and creates, revise or leverage the applications brand.

2. **Involving SA design into Design phase**

During design phase SA developers can work closely with the UX team to define the Software Architecture (Wireframes, Visual design). A User Experience team may think out of the box while creating wireframes and visual design, but may not be aware of challenges, possibilities and limitations. Involving SA developers in this phase may ease the process, as SA developers understand the technologies and possibilities. This will reduce the last minute efforts from the SA developer’s side and additionally both the SA and UX team, as well as client will have a clear expectations set.

3. **Deliverables**

- Discussion of the design philosophy and strategy used to create the SA design, including an explanation of how research brief informed the SA design
- A comprehensive set of screen layouts illustrating every part of the SA design
- Key screen that present the visual design and branding for the application.

**C. Prototype Development**

1. **Process**

Using the approved design document as blueprint, prototypes of the Software Architecture designs are created. Based on clients needs the prototypes are created using HTML or flash. Prototypes can be low fidelity or high fidelity based on user needs. The scope of the prototype created during this phase is tailored to the specific application and the user testing requirements. Some applications require a comprehensive click through or working model of the entire interface, while others only require a prototype of core functionality.

The prototype serves as:

- A communication and review tool for the client
- An implementation guide for the software developer
- A user testing tool
- A working prototype enables clients to see how the Software Architecture will perform without having a commit programming resource

Another benefit to a working prototype is that it dramatically improves the communication among SA design, marketing and application development teams, as well as any other key stakeholders since marketers, designers and developers often think and use language differently, having a functional prototype leaves fewer issues open to interpretation or mis-understandings. Prototypes are in fact an opportunity to identify any remaining Software Architecture issues. Should there be design document and prototype are modified to incorporate changes prior to final approval.

Once the prototype is approved, development team can work to implement new Software Architecture design quickly and efficiently without interrupting the development process to correct design issues.

The utilization of prototype also has significant marketing benefits. Since the prototype closely stimulates the actual application, extensive user testing can be one before the programming starts, which also produces significant cost savings. Additionally positive user feedback is a validation of design, confirming that the application delivers a user experience that meets or exceeds customer expectations. This positive feedback can be used for both internal and external marketing purpose, by building confidence in the rollout and promoting user adoption.

2. **Deliverable**

Interactive prototype of the new Software Architecture
D. Production
A proper SA and UX team collaboration and Integration of SA design from the starting of SDLC can reduce lot of efforts and confusion. Also it can help in successful and timely delivery of the products in any company without any slippages and can increase customer satisfaction. Additionally it can help building ongoing relations with our clients.

III. UML as a Design Tool for SA
The Unified Modeling Language allows developers to create a model of a software system similar to blueprints of a house. To keep an overview of the whole model, the UML provides several diagrams. Each diagram shows only a few aspects of the system and omits all that are irrelevant to this specific perspective. Together, these diagrams form a complete picture of the model with all its aspects. In UML 2.0, 13 diagrams are defined. They reach from a high level of abstraction – how users perceive and use the system – to a low level of abstraction – detailed timings of critical processes and fine-grained object structure.

<table>
<thead>
<tr>
<th>Diagram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>class diagram</td>
<td>Fine-grained internal object structure of the system, both real-world and abstract or implementation aspects</td>
</tr>
<tr>
<td>object diagram</td>
<td>Runtime structure of class instances</td>
</tr>
<tr>
<td>composite structure diagram</td>
<td>Internal structure and collaborations</td>
</tr>
<tr>
<td>component diagram</td>
<td>Dependencies among system components</td>
</tr>
<tr>
<td>deployment diagram</td>
<td>Physical arrangement of computer systems and the components being executed on them</td>
</tr>
<tr>
<td>package diagram</td>
<td>Logical groupings of classes and dependencies between groups</td>
</tr>
<tr>
<td>activity diagram</td>
<td>Computations, workflows, object flow and control flow</td>
</tr>
<tr>
<td>use case diagram</td>
<td>Behavior of the system to an outside user, describe what can be done with the system</td>
</tr>
<tr>
<td>state machine diagram</td>
<td>Various possible states of an object and their transitions</td>
</tr>
<tr>
<td>sequence diagram</td>
<td>Timing of messages between objects during an interaction</td>
</tr>
<tr>
<td>interaction overview diagram</td>
<td>Overview of the flow of control</td>
</tr>
<tr>
<td>timing diagram</td>
<td>Time-dynamic behavior and state changes</td>
</tr>
<tr>
<td>collaboration diagram</td>
<td>Objects and links within an interaction</td>
</tr>
</tbody>
</table>

UML is defined as a model itself, using a metamodeling language called Meta Object Facility (MOF). The prefix ‘meta’ is of Greek descent and means ‘higher, beyond’. A metamodel is a model on a higher level of abstraction, i.e. a model of a model; it is used to specify models. This implies that every model is an instance of a metamodel. Consequently, a meta-metamodel is used to specify metamodels.

A. System Analyst
The system analyst is responsible for the domain of requirements capture. He accomplishes this by talking to the customer and then identifying actors and use cases. He only creates some (often the most important) use cases and is assisted by several use case specifiers for the rest. Often this role is filled by the account manager or project manager.

B. Architect
The architect is responsible for describing the architecture of the system during requirements capture and for prioritizing the use cases specified by the use case specifiers. Use cases that are required on multiple occasions or ones that are crucial for the success of the project are assigned a higher priority than unimportant, rarely used ones. The output the architect creates is used later on to determine which elements of the architecture are implemented first.

C. Use Case Specifier
A use case specifier is responsible for detailing one or more use cases into their flow of events, including how it starts and ends, and how actors can interact. The use case specifiers create detailed use cases from the outlined use cases specified by the system analyst and include other information like the supplementary requirements. Use case specifiers have to work with the real users of the use cases frequently in order to tune the detailed use case to their situation.

D. Software Architecture Designer
The Software Architecture designer creates the layout of the Software Architecture, but he does not implement it. However, he may create prototypes for the Software Architecture of some use cases.

E. Artifacts
While working on a software project, a multitude of artifacts is created. RUP contains description of some common artifacts used during requirements capture and Software Architecture design, which we will reproduce here.
“Some of the earliest prototypes are simple hand-drawn pictures. As they progress, they may take the form of wireframes, typically created with a drawing tool, such as Visio or Adobe Illustrator. Combined with use case specifications, these prototypes are good
for storyboards describing specific scenarios of flow through the system’s screens. The prototypes are also good for understanding the structure of compartmentalized screens and can give stakeholders an early and tangible view of the system”. - Jim Conallen

**F. Storyboards**

Storyboards show the individual steps of an interaction sequence. The interaction is some task a user wishes to accomplish and involves multiple steps – in terms of UML: storyboards show the individual steps of a use case. A Storyboard consists of multiple simplified screens, which contain only the basic elements and links needed for the interaction. The links used are connected to the next screen in the interaction. Figure 6 shows how a user of a tutorial would interact with the system in order to find information on a specific topic. Storyboards are used by information designers as well as the whole design team, to obtain an understanding of workflows and navigation.

**G. Prototypes**

Software Architecture prototypes differ slightly from the prototypes used in pure software development. They do not necessarily implement any or all the functions of the product; they do show the complete Software Architecture of the system, though with varying levels of detail. The Software Architecture prototype is iteratively evaluated by future users of the system and improved by the designers.

**IV. Conclusion**

“Software Architecture” or SA Development is just not an artifice; there is immense understanding of ethnographic art. The work process from the start to finish of type understanding of an end user is observed in several Software Architectures that have been carried out in our survey, where is the strengths and limitations are documented and identified opportunities to adapt to the process. The survey had been carried out with a sufficient number of case studies, that belong to varied platforms, users and applications.

**References**


Mr. M Sunil Kumar has completed B.Tech in Computer Science & Information Technology from JNT University and M.Tech in Computer Science from JNT University. Presently he is pursuing Ph.D in Computer Science and Engineering, S.V.University, TIRUPATI. He is currently working as Assistant Professor in the Department of CSE, Sree Vidyanikethan Engineering College, A. Rangampet, Tirupati, A.P.

His main research interest includes Software Engineering, Software Architecture, Information Retrieval and Database Management Systems.

Dr. A. Rama Mohan Reddy, obtained his Bachelor Degree in Mechanical Engineering and Master’s degree in Computer Science Engineering from NIT Waranagai and Ph.D degree from Sri Venkateswara University and at present working as an Professor in Department of Computer Science and Engineering, Sri Venkateswara University College of Engineering. His areas of interest include Software Architecture and data mining. He has more than 27 years of experience in teaching and research.