Checkpoint #2

## **Checkpoint #2**

*Worth 10% of the final grade*

Your group will create a list of architectural drivers for the assigned system that include:

* + Design concerns
  + Quality attribute requirements
  + Primary functionality
  + Architectural concerns
  + Constraints

Submit Group Project Checkpoint #2.

**From the module:**

**Design Purpose:**

You may undertake architecture design for various purposes and the effort expended in designing the architecture will be commensurate with that purpose. Architecture design may be undertaken as:

* a part of a ***request for proposal*** and an initial architecture is created to explore its feasibility and provide a schedule and budget estimate. Such an undertaking requires fleshing out the architecture just enough to figure out the major units of work so that effort estimates can be determined.
* a part of a ***proof-of-concept*** study to explore a problem domain, a technology, a quality attribute, or to place something before a customer to elicit rapid feedback. This requires designing enough architecture for creating an exploratory prototype.
* a part of the ***development of a new system*** (greenfield project) or *evolution of an existing system* (brownfield project). In this case, substantive work may need to be undertaken to design an architecture that satisfies the given requirements, helps create and allocate units of work to project teams, and assists in release planning.

These purposes are subject to interpretation and realized differently depending on the context. In a greenfield project for a mature domain, the effort expended for responding to request for proposal may be straightforward; the architect can examine organization’s existing systems and confidently make estimates through analogy. In novel domains, however, one may need to flesh out the architecture enough to make reasonable estimates. Brownfield projects may entail understanding a complex system that exists before planning its evolution. Additionally, an organization may impose its own interests on top such as extending the shelf life of a product through design for reuse, future extension, or continuous delivery, leveraging the existing project team capabilities and skillset, and so forth.

These considerations are important because they affect both the process of design and the design output. Architectures exist to help achieve business goals. The architect should be clear about these goals and should communicate them (and negotiate them!) and establish a clear design purpose before beginning the design process.

**Quality Attributes:**

As discussed in the previous lesson, quality attributes are measurable and testable properties of a system used to indicate how well the system satisfies the needs of its stakeholders. Among the drivers, these are the most significant forces that shape the architecture of a system.

The best way to discuss, document and prioritize the quality attribute requirements is as a set of scenarios that describe how a system is required to respond to a stimulus. Figure 4 shows the different parts of a scenario that were discussed in the previous lesson.

**Figure 4.** Different parts of a quality attribute scenario (Cervantes and Kazman, 2016)

Because they are testable with explicit stimuli and responses, we can evaluate a design in terms of how likely it is to support the scenario.

As with any other requirements, scenarios should be prioritized. This can be achieved by considering two dimensions. First is the importance of the scenario with respect to the success of the system or its business criticality. This is ranked by the customer. The second corresponds to the degree of technical risk associated with the scenario. This is ranked by the architect.

A low/medium/high (L/M/H) scale is used to rank both dimensions. Once the dimensions have been ranked, scenarios are prioritized by selecting those that have a combination of (H, H), (H, M), or (M, H) rankings.

While it is possible to modify traditional requirements elicitation techniques, such as Joint Requirements Planning (JRP), Joint Application Design (JAD), discovery prototyping, and accelerated systems analysis, to focus on quality attribute requirements we will discuss two additional techniques, Quality Attribute Workshop (QAW) and Utility Tree, that can be helpful.

**Primary Functionality:**

While functionality is orthogonal to architecture design, there is a subset that is critical to achieving the business goals that motivated the development of the system. As a rule of thumb, this subset is typically 10 percent of the total use cases or user stories for a system and is referred to as primary functionality.

An architect must consider primary functionality when performing architecture design for two reasons:

* It helps establish a precedent on how responsibilities inherent in use cases or user stories are allocated to different elements (usually modules) of a system to promote modifiability or reuse, and also allocate work to the project teams. This precedent helps guide downstream element interaction design for non-primary use cases (something that the architect may not be involved with).
* Some quality attribute scenarios are directly connected to the primary functionality of a system. For example, in a movie streaming application, one of the primary use cases is, of course, to watch a movie. This use case is associated with a performance quality attribute scenario such as “Once the user presses play, the movie should begin streaming in no more than 5 seconds”. In this case, the quality attribute scenario is directly associated with the primary use case, so making decisions to support this scenario also requires making decisions about how its associated functionality will be supported. This is not the case for all quality attributes. For example, an availability scenario can involve recovery from a system failure, and this failure may occur when any of the system’s use cases are being executed.

**Architectural Concerns:**

There are some concerns that not spelled out explicitly as requirements but are part of architecture design for most systems and must be addressed. These include:

* ***General concerns*** such as establishing an overall system structure, the allocation of functionality to modules, the allocation of modules to teams, organization of the code base, startup and shutdown, and supporting delivery, deployment, and updates to the system.
* ***Specific concerns*** internal to a system such as exception management, dependency management, configuration, logging, authentication, authorization, caching, and so forth. Sometimes these internal concerns may arise from a previous design decision; for instance, if you establish an architecture for a web-enabled system you may need to address session management.
* ***Issues*** arising from design reviews that require changes to the existing architecture.

An architect must be aware of these concerns and can characterize them as quality attribute scenarios elicited from the customer which in this case would include development and operations team.

**Constraints:**

Constraints are decisions over which an architect has little or no control, and architectures that satisfy the business goals need to be designed around them. These decisions come in form mandated technologies, integrating legacy components, interoperating with existing systems, complying with existing laws and regulations, leveraging the capabilities and skillsets of your project team, and so on. An architect must design the best system despite these constraints.

|  |  |
| --- | --- |
| Checkpoint 2 Rubric 22 |  |
| **Criteria** | **Ratings** |
| This criterion is linked to a Learning Outcome Design concerns You may undertake architecture design for various purposes and the effort expended in designing the architecture will be commensurate with that purpose. | 2 pts Full Marks  Design purpose IS clear     1.6 pts Partial Marks  Design purpose is MOSTLY clear.     1 pts Minimum Expectations  Design purpose is NOT clear.     0 pts No Marks  Design purpose is MISSING from the submission. |
| This criterion is linked to a Learning Outcome Quality attribute requirements Quality attributes are measurable and testable properties of a system used to indicate how well the system satisfies the needs of its stakeholders. Among the drivers, these are the most significant forces that shape the architecture of a system and are expressed using quality attribute scenarios | 2 pts Full Marks  Quality attribute scenarios are well-formed and properly parsed.     1.6 pts Partial Marks  Quality attribute scenarios are MOSTLY well-formed and correctly parsed.     1 pts Minimum Expectations  Quality attribute scenarios are NOT well-formed and/or correctly parsed.     0 pts No Marks  Quality attribute scenarios are missing from the submission. |
| This criterion is linked to a Learning Outcome Primary functionality While functionality is orthogonal to architecture design, there is a subset that is critical to achieving the business goals that motivated the development of the system. As a rule of thumb, this subset is typically 10 percent of the total use cases or user stories for a system and is referred to as primary functionality. | 2 pts Full Marks  Primary functionality IS correctly identified.     1.6 pts Partial Marks  Primary functionality is MOSTLY correctly identified.     1 pts Minimum Expectations  Primary functionality is NOT correctly identified.     0 pts No Marks  Primary functional use cases / user stories are missing from the submission. |
| This criterion is linked to a Learning Outcome Architectural concerns Some concerns that not spelled out explicitly as requirements but are part of architecture design for most systems and must be addressed | 2 pts Full Marks  Architectural concerns ARE correctly identified.     1.6 pts Partial Marks  Architectural concerns are MOSTLY correctly identified.     1 pts Minimum Expectations  Architectural concerns are NOT correctly identified.     0 pts No Marks  Architectural concerns are missing from the submission. |
| This criterion is linked to a Learning Outcome Constraints Constraints are decisions over which an architect has little or no control, and architectures that satisfy the business goals need to be designed around them. | 2 pts Full Marks  Constraints ARE correctly identified.     1.6 pts Partial Marks  Constraints are MOSTLY correctly identified.     1 pts Minimum Expectations  Constraints are NOT correctly identified.     0 pts No Marks  Constraints are missing from the submission. |

Brian: Here is what I thought of as far as some drivers/constraints:

Nick: I made some change in blue

Manish: my changes are in purple color

Design Concerns:

* Development of a new system
* Supporting Intergration with existing tools used by the organization
* Data management and storage

Technical constraints:

* Programming language
* Support of a specific platform
* Use of a specific library/framework
* Security and privacy
* Data syncing across multiple devices

Business Constraints:

* Timing – regarding trad-off decisions or alternative scenarios
* Budget – This could limit drivers that can be invested in
* Team Composition - skills available to the development process
* Regulation – must follow industry regulation and data protection laws (HIPPA, GDPR)
* User adoptions – comprehensive documentation and user guide

Functional Requirements

* Meeting scheduling – users should be allowed to schedule and send invitations. (record meetings??)
* Notifications – users should get notification for new messages, call and etc
* File Transfer – use should be able upload/download file (share with access control??)
* Communcation Methods – Call, Message, Video Call, whiteboard for real time collaborations

Quality Attributes:

* “Our system should be fast”
* “User friendly”

Not sure how Kassab wants us to format this.... There wasn’t really a table to reference. I'm thinking we will be utilizing our previous table and building off of the previous? Maybe set this one up in a table format as well?

Nick: Here’s a start for what I was picturing. Like Brian said, there aren’t any examples to base off, but this what I was imagining. (Add more for all types of drivers)

Manish: I think we should be adding these to our existing table from checkpoint1

Brian: Like extend the table further to the right? [@Shrestha, Manish](mailto:mxs6992@psu.edu)

* This would also add formatting problems (not fitting the page)
* Or do we keep Checkpoint 1 as is, then use this table below? Then submit both copies and tell Kassab “Hey, due to formatting issues, we are submitting both as reference. But Checkpoint 2 is an extension of Checkpoint 1.”
* Also, should we add Kassab’s “Business goals” suggestions to our 1st Checkpoint? THEN do
* On the grading rubric he put “One notable strategic goal extracted from the problem description is the need to reach a wide market by incorporating a variety of input/output devices. So reaching to a wider market is the business goal while supporting diverse input/output devices becomes a feature (Engineering objective) that supports the business goal. Additionally, another business goal would be to "differentiate the product in a competitive market", and this can be achieved by integrating five essential communications tools in a single one.”
* Kassab wrote for Quality Attribute Scenario section: “This section holds significant importance within this checkpoint, and you have done an excellent job specifying each quality scenario with their corresponding stimulus, source of stimulus, artifact, environment, response, and response measure. To improve readability, I suggest incorporating the scenario as a small paragraph. Please do so in your final report.”
* Should I start a new doc that is strictly “Final draft?”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID #** | **Architectural Driver** | **Type** | **Ranking (Success of system/Technical Risk)** | **Explanation** |
| 1 | The system design shall support increased efficiency of collaboration between geographically dispersed employees | Design Concern/Purpose | (H, H) | A business goal of the system is to “make it easier for its geographically dispersed teams to collaborate with each other.” This driver is representative of how the software shall be built to support that goal. |
| 2 | The system design shall allow for the product to differentiate itself in the product in a competitive market | Design Concern/Purpose | (H, H) | Another business goal would be to be able to differentiate our product from existing products of the market. This is a new product that will combine 5 different features, so the architecture of the software shall support that |
| 3 | The 5 services are available to use 99% of the time | Quality attribute Requirement (Usability)  A user attempts to access any of the 5 services. They are able to connect and use any of the services at any time. | (H, M) | The success of the product relies on users being able to actually use the product whenever it is best suited for them. If one service is down, users should be able to rely on any of the other 4 to still communicate |
| 4 | The system shall resolve or notify user of network connection issues within 10 microseconds | Quality Attribute Requirement (Performance)  The system detects a change in connection strength or availability while a user is using a service. It will try to reconfigure the settings for an optimal connection within 10 microseconds or it will alert the user of possible network disruption. | (M, M) | If the system detects a network issue, it will attempt to resolve or at least notify the user of what the issue is. There may be instances where neither is possible, but providing user with as much information as possible will increase user satisfaction |
| 5 | The system shall compress all data without loss of any bytes to improve data transfer. | Quality Attribute Requirement (Performance)  When a users’ data in any of the 5 services is sent to the server, it will be compressed and then decompressed once the client receives it. No packets will be dropped. | (L, M) | The architectural drivers of data compression can optimize data transfer efficiency, reduce network bandwidth usage, and provide a reliable and user-friendly experience for collaborating teams. |
| 6 | The system shall have authentication to verify users' identity and access privileges based on their role. | Quality Attribute Requirement  (Security)  A user will sign into the product and the system will verify the credentials. Once authenticated, only features that are within the users’ role permission will be available. | (H, H) | This driver ensures that only authorized users can access the system's features and data, thereby maintaining security and protecting sensitive information. |
| 7 | The system shall support 5 different services including voice communication, video conferencing, instant chat, file sharing, and collaborative whiteboarding. | Primary Functionality | (H, H) | This is the base functionality that the system much achieve. |
| 8 | Real-time features such as voice, video, and whiteboarding shall have a response time of no more than 100ms. | Primary Functionality | (M, L) | The primary functionality driver comes from the request of the customer that it’s “critical operations” be “highly responsive”. |
| 9 | The system must be profitable | Primary Functionality | (H, M) | The product must be built in a way that will make it profitable in the long run. If it doesn’t make money, then the system won’t continue to be supported. |
| 10 | The 5 system components shall be developed and run independently within the system | Architectural Concern | (M, M) | This architectural driver will allow for each component of the system to be developed in parallel and each service will be able to function independently of the other services. |
| 11 | The system shall be continuously and quickly deployable | Architectural Concern | (L, L) | Deployments of the system should be as quick and easy as possible so that changes can be made available to the users as soon as possible |
| 12 | Metadata for the system shall be stored in a database | Architectural Concern | (L, L) | The system design will need to include a way to store system metadata so that it can be used for debugging, analytics, etc. |
| 13 | The system shall be able to support a variety of hardware and operating system inputs and outputs | Constraint | (H, H) | The system must be able to work on any device, accept input from multiple devices, and output to multiple devices. We have no control over end-user client devices. |
| 14 | Any registered user must be over the age of 13 | Constraint | (M, H) | This legal constraint comes from laws that prohibit the collection of data from users under the age of 13. |