Your Paper

You

May 9, 2017

Contents

Chapter 1

Quality Attriburtes

1.1 Some Background

1.1.1 Stakeholders

Stakeholders represent different(typically conflicting) perspectives on the system and attemp to influence the architect. The architiect needs to tradeoff the different influences and resolve the conflict. e.g. - Marketing might want to have a big market but maintence would prefer system to be simple.

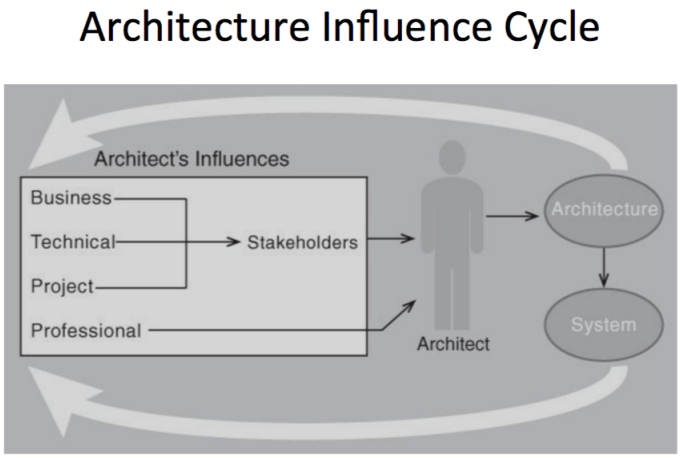

Figure 1.1: Architect Influence Cycle

1.1.2 Functional Requirements

these specify what the system does, architecture is important to them as they structure the containers that hold functionality !!shit in lec! !. Functional requirements are things the system does.

1.1.3 Constraints

hese are decisions that have already been taken e.g. we will use the Java programming language (because we have a Java development team available) or the system will only support certain web browsers

1.2 Quality Attributes

Quality Attributes specify, usually quantitatively, the requirements on particular bits of functionality or on the whole system. Software Quality Attributes are the benchmarks that describe system's

intended behavior within the environment for which it was built. The quality attributes provide the means for measuring the fitness and suitability of a product. !!Is there a difference between QA's and Non-Fucntional requirements? !!

1.2.1 Problems with Quality Attributes

- Often QA's are not testable. e.g. what does it man to say something is modifiable or usable or dependable or resilient?
- It can be difficult to map from a concern about the system to a QA. For example, a high failure rate in some transaction could be a performance issue or it could be an availability issue
- Communities around a particular quality attribute have developed their own terminology (e.g. security has attacks, performance has events!!what?!!, etc).

One Solution: use quality attribute scenarios to provide sufficient specificity to avoid some of these issues.

1.2.2 QA Scenarios

QA scinario has following components:

1. Source of Stimulus: Person or another System
2. Stimulus: An action system responds to e.g. using the wrong configuration specification for the system.
3. Environment: Captures wider aspects of the system
4. Artifact: Part of the system that is stimulated
5. Response: Activity resulting from stimulus
6. Response measure: Measure of the response so that the scenario is testable e.g time taken to detect wrong config.

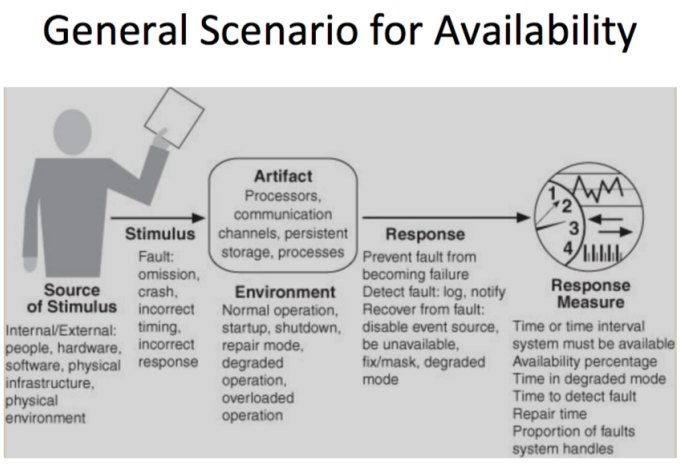

Figure 1.2: Example QA Scenario Components

Each QA has a general scenario that it tries to capture with its components. This acts like a guide for the architect.

1.2.3 Architectural Tactics

1. Artitectural tacticas are a way of documenting routes to achieve a QA requirement.
2. Architectural Tactic is a design Decision that influences the achievement of a QA response. They are more primitive than design patterns.
3. Tactics are based on single QA's and dont take trandeoffs into account.
4. Teactics are usually more generic and need to be specialised to a specific context.
5. They allow the architect to systematically enumerate possible design decision.

Example Tactics

Example of Availaibilty Tactics

DIfferent tactics identify different forts of reposnsibilities and how to achieve them. For exam for availaibility:

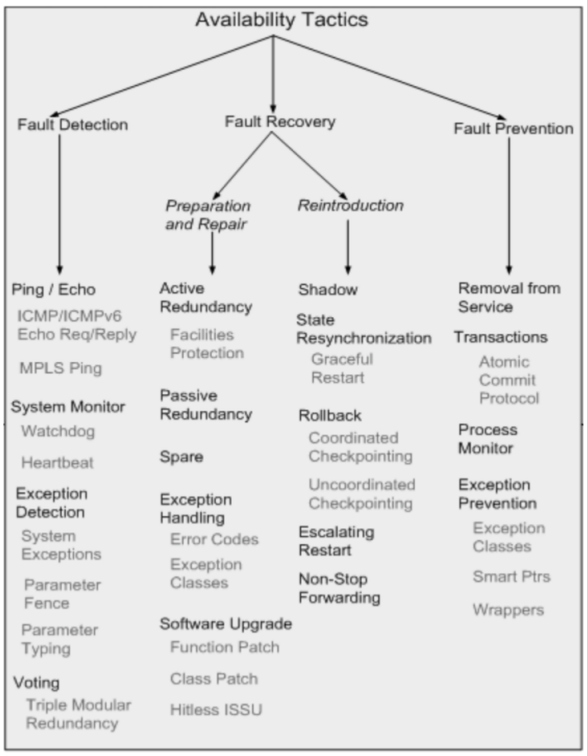

Figure 1.3: Availaibilty Tactics

- Example- Watchdog times ensure the availability of systems by initialing a restart in the event of apparent inactivity.
- Another example- Heartbeat is used in high availibility clusters to provide availaibility of a shared resource. One system owns the resource and regularly sends out heartbeat to indicate system is alive. If heartbeat is missed the backup system takes over responsibility of the shared resource.

1.2.4 Categories of Architectural Design Decisions

There are seven broad categories of design decisions:

1. **Allocation of responsilbilties**- identify the most important responsibilities and determine how to allocate these to runtime and static elements. These are often specific to a QA and so the specifics depend on the QA, for example in the case of availability fault detection is an important responsibility that will be further decomposed and distributed in the architecture.

2. **Coordination Model** - Components in the architecture interact with one another via a collectioon of mechanisms. This is called the coordination model. Things it takes into account are:

   - What elemsts in the system need to coordinate with one another.
   - What properties(eg timing, security) does the coordination need to have
   - The mechanisms and their properties(eg statefullness, synchrony, dilevery guaranties).

3. **Data Model** - How data is created, destroyed. Data acess methods, operations on the data, the properties of the data are all included in the Data Model. We also need to decide on how the data would be organised, stored, backed up and recovered in case of data loss. It also includes Maintaining Metadata that cotrols the interpretation of the data.

4. **Management of resources** We can have both Hardware(eg CPU, memory, battery) and Software(Buffers, Processes) resources which need to be Managed. Management includes:

   - Identifying which resources need to be managed.
   - What system element should manage a resource.
   - Work out sharing strategies and how to arbitrate in contention situations
   - Consider the consequences of running out of a resource (e.g. Memory).

5. **Mapping Among Architectural Elements** We have 2 types of mapping:

   - Mapping b/w differnt types of elemts in the architecture e.g. b/w static development structures and threads or processes
   - Mapping b/w software elements and environment elements e.g. from process to specific processors.

   Some important Mappings: code to runtime structures, runtime elements to environment, data model elements to data stores.

6. **Binding Time Decisions** !!LOOK UP AGAIN- SHIT IN LECTURES!!

7. **Choice of technology**- Important so that we can realise other decisions in a concrete system. Things to consider:

   - What technologies are avalaible?
   - what tools are availaible to support the technologies?
   - How much training would be needed for each technology?
   - What are the consequesces and restrictions of the the choice of a technology(eg. incompatible with other tech)?
   - How does the tech fit in other technologies used in the organisation?

Chapter 2

Security

We need to consider atacks on Confidentiality,Integrity and Availability.AIacks need to be monitored andd etected, resisted where possible, otherwise we may need some other form of reaction, eventually we should be able to fully recover.

2.1 Important QA's

1. **Confidentiality** - Only those who should have accessare given access
2. **Integrity** - Data or services are not subject to unauthorised manipulation.
3. **Availibility** - The system is availble for legitimate use.
4. **Security Mechanisms** Authentication, Authorisation, non-repudation

2.2 Security Scenario Components

2.2.1 Generic Example

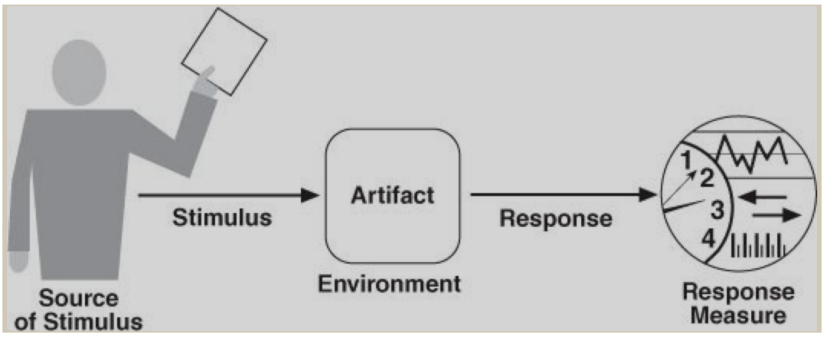

Figure 2.1: QA Scenario Components

- **Source**- Humans or systems that may or may not have been identified and can be either inside or outside the organisation
- **Stimulus**- Unauthorized attempt to access, Manipulate or disable the artifact.
- **Artifact**- System services, data, components, data produced or consumed by the system.
- **Environment**- online, offline, connected to network, disconnected to network, behind firewall, fully/partially operating, not operational
- **Response**- Transaction carried out s.t: Data or services protected from unauthorised access, no data manipulation without authorization, parties in a transaction are identified with assurance, Par'es cannot repudiate their participation, Data, resources etc are available for legititimate use, Appropriate people are notified when threat is identified.

- **Response Measure**- assessment of the degree of compromise, temporal and spatial data on the compromise, how many atacks were resisted, how much data is vulnerable

2.3 More Concrete example

Assume a more concrete example fo Denial of Service. The components could be:

- **Source**- A wide ranfe of systems with different IP.
- **Stimulus**- Access to the service provided
- **Artifact**- The service we are concerened with
- **Environment**- Normal operation
- **Response**- Detect Normal Load
- **Response Measure**- Mode of operation is changed to ensure normal service to trusted IP addresses.!!what?!!

2.4 Security Architecture tactics

Tactics are a high-level way of categorizing possible protection against aIack.

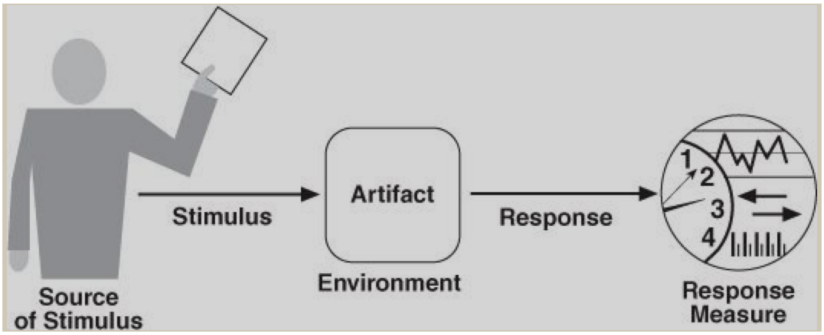

Figure 2.2: Security Architectural Tactics

2.5 Architectural Design Decisions

2.5.1 Allocation of resposibilities

For security-system responsibilities do the following:

- Ensure all actors have identities(like roles)
- Authenticate identities to apprppriate actors.
- Check and Ensure Authorization
- Ensure Data Encryption
- Log attempts, successes, failures and other senitive oprations.

2.5.2 Coordination Model

The following things in the coordination model need to be acoounted for:

- Ensure coordination mechanisms use authentication and
- Ensure the coordination model is not vulnerable to tampering, interception, impoersonation.
- The model data involved is encrypted.
- Monitor level of demand for communication to identify excessive demands

2.5.3 Data Model

- Ensure There is a vald data model that disallows invalid data flows.
- Ensure Logging of access, modification and attempted access or modification.
- Data is protected in flight at rest using appropriate encryption
- Ensure appropriate backup/recovery mechanisms are in place.

2.5.4 Mapping among Architectural Elements

- Explore and be wary be wary how different mappings change the way users can access resources.
- Ensure for all the mappings, the responsibilities(authorization, logging,encryption etc) are preserved.
- Ensure recovery from attack is possible.

2.5.5 Resource Management

- explore teh overheads resulting from responsibilities(logging, encryption, recovery etc).
- Analyse how a user can make demands on a critical resource.
- Make sure malicious use of resources is detected and managed.
- Identify and manage the potential fot corruption/contamination.
- explore the potential for resource use to be used as a covert channel to trnasmit data.
- limite resources used to manage sttemps at unauthorised use !!surely depends on use case?!!.

2.5.6 Binding Time

- Explore the consequesces of varying binding times on the ability to trust actor or components.
- place mechanisms to ensure trust given in bindin time.
- Explore impact on resource use, capacity/throughput, response time.
- Ensure the time bindings are ensured with all resposibilities(authorization, logging,encryption etc).
- Explore the potential of variation in binding time as a covert channel.

2.5.7 Choices of Technologies

- Ensure limitations of technologies are understood and the potential for future compromise is well identified.
- Ensure your chosen technologies support the tactics you want to deploy to protect the system.