Architectural Design & Drivers

"You never really understand a person until you consider things from his point of view”

Atticus Finch
In to kill a Mockingbird by Harper Lee

LECTURE 04
DR USMAN NASIR

Designing Software Architecture

Design

- Design is both a verb and a noun.
  - Design is a process, an activity, and hence a verb.

- Designing is about making decisions to achieve goals and satisfy requirements and constraints.
  - The outputs of the design process are a direct reflection of those goals, requirements, and constraints.

- Why traditional houses in Switzerland look different from those in Algeria?
  - They are designed according to the requirements

- Igloos (The most famous house in snow)

Both are igloos!!

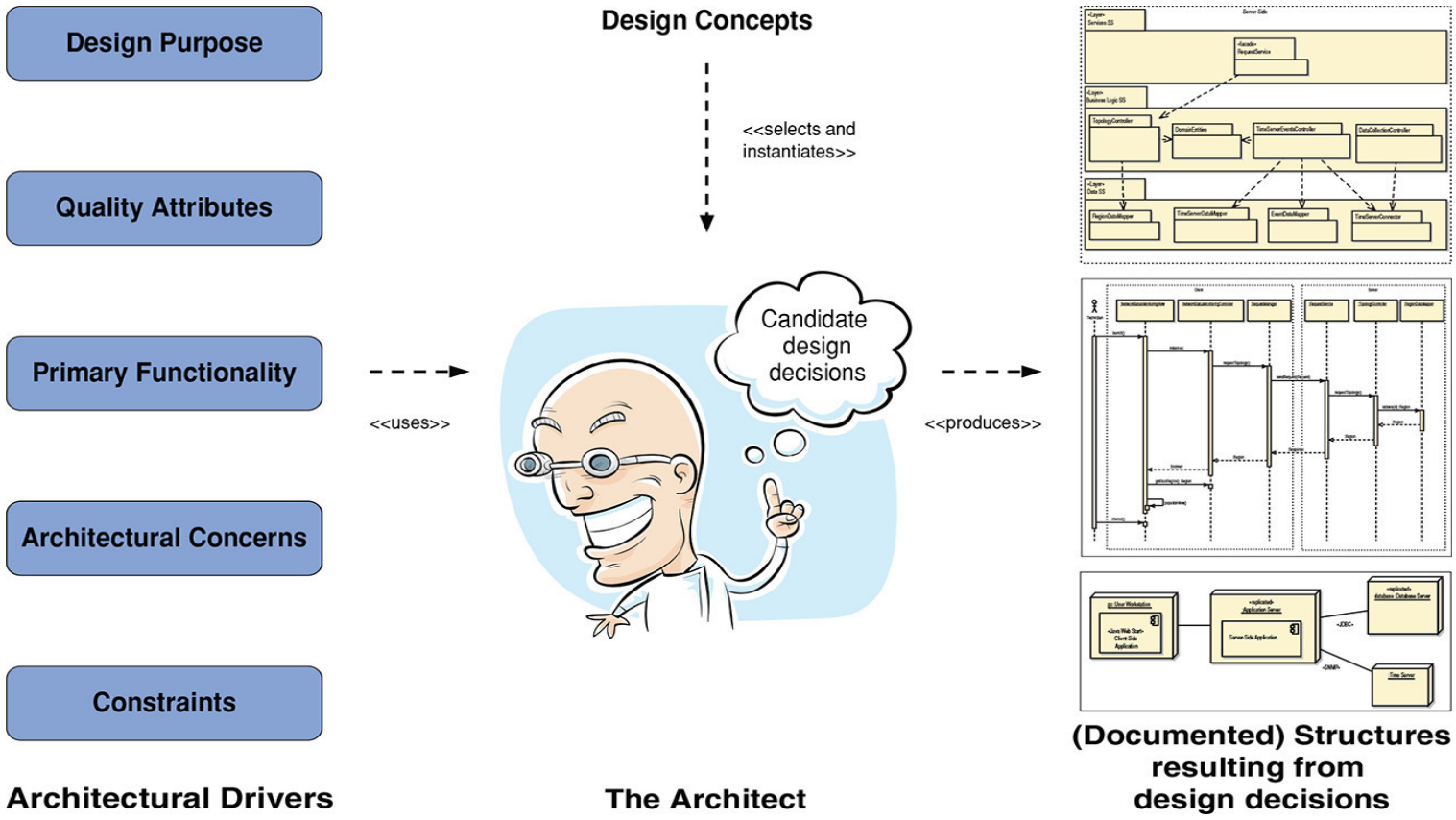
but different designs conforming to the needs, quality and constrains

Design in Software Architecture

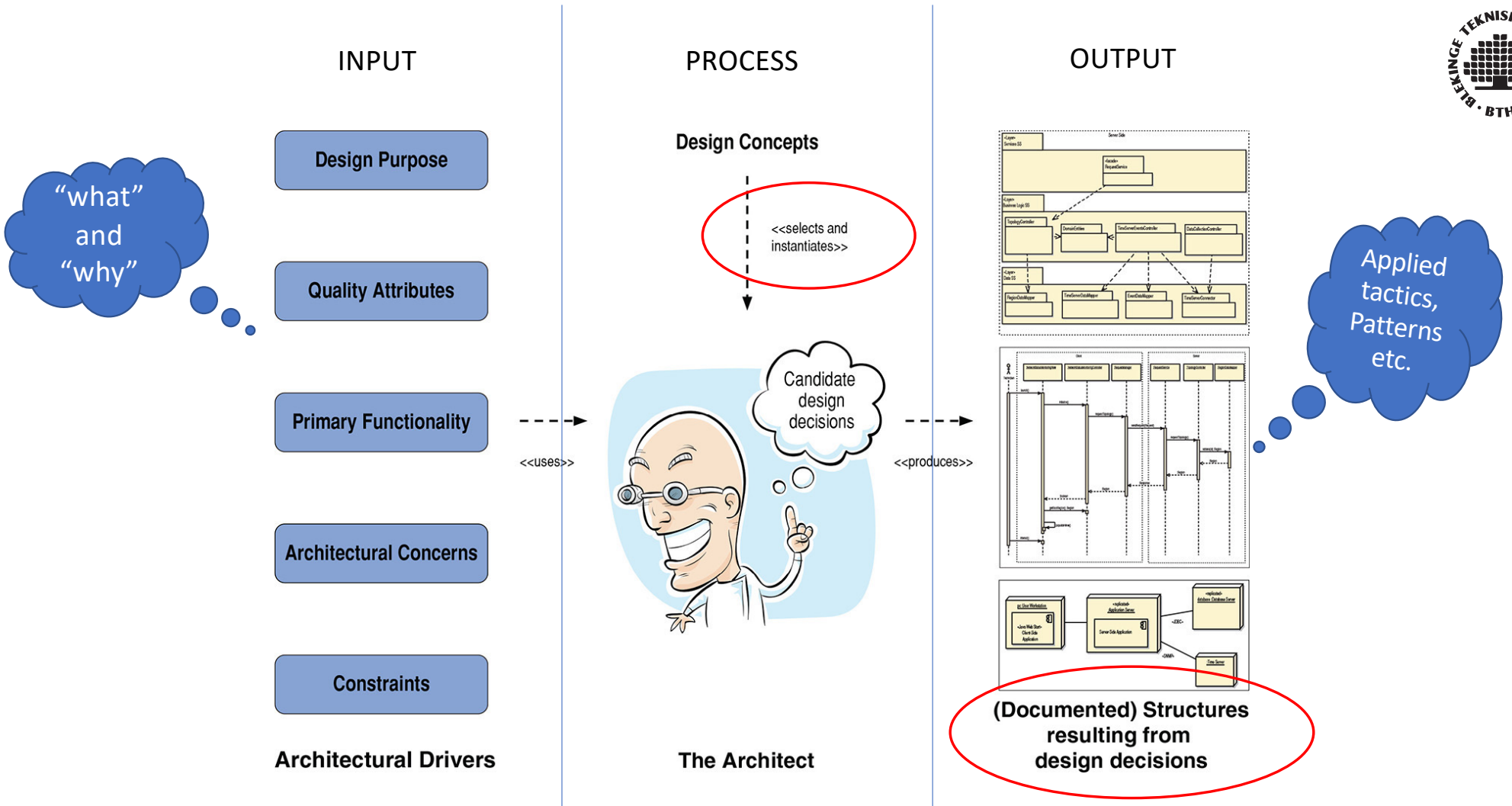
- Architectural design of software systems involves making decisions, working with available resources (developers etc..) skills and materials (platform etc..), to satisfy requirements and constraints.

- ***Architectural design activity*** is a key step to achieve product or project goals.
- **The activity eventually guides the project.**
  - Analysis, Cost, schedule estimations, team formation, risk analysis and mitigation, and implementation.

Architectural design activity



Overview of the architecture design activity (Architect image © Brett Lamb | Dreamstime.com)



Overview of the architecture design activity (Architect image © Brett Lamb | Dreamstime.com)

Architectural drivers

- **“What” are we doing and “why” are we doing.**
- These drivers include:
  - design purpose
  - quality attributes
  - primary functionality
  - architectural concerns
  - and constraints.
- These drivers are critical for to the success of the system they drive and shape the architecture.

Design concept

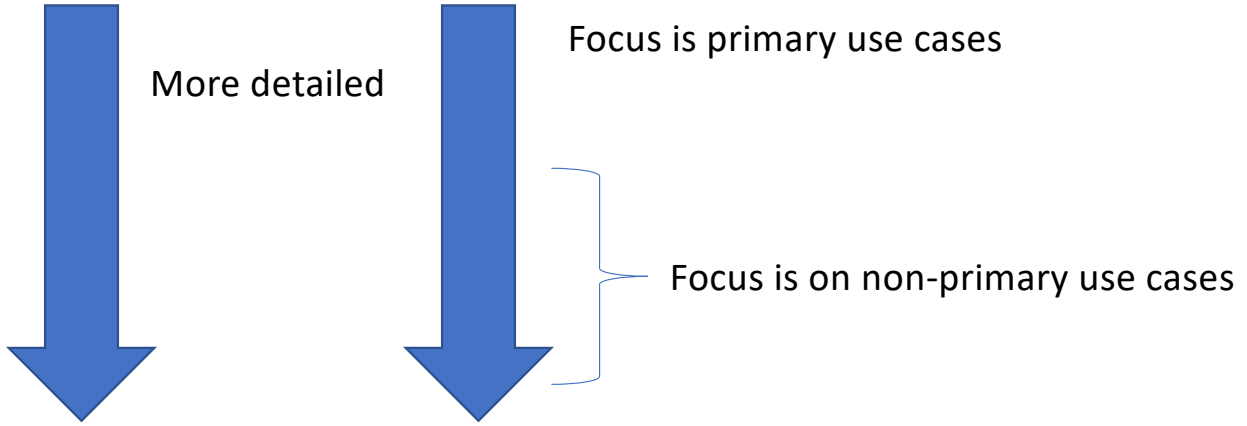
- The building blocks from which the structures that make up the architecture are created.
  - Architectural drivers are eventually transformed into structures using design concepts
- Different types of design concepts are:
  - **Tactics**
  - **Architectural patterns**
  - Reference architectures
  - Deployment patterns
  - Technology families
  - Frameworks
- These are proven designs and design fragments that have been tried and tested by community.

Architectural decision

- Design decision is a **decision** that is made during the **design process**, including the selection of a **design concept** and the instantiation of the selected design concept.

- Architectural decision
  - A design decision becomes architectural if it has non-local consequences and those consequences matter to the **achievement of an architectural driver**.

- Architectural design mostly results in the **identification of only a subset of the elements** that are part of the system's structure.
- There are three levels of design activities
  - Architectural design
  - Element interaction design
  - Element internals design

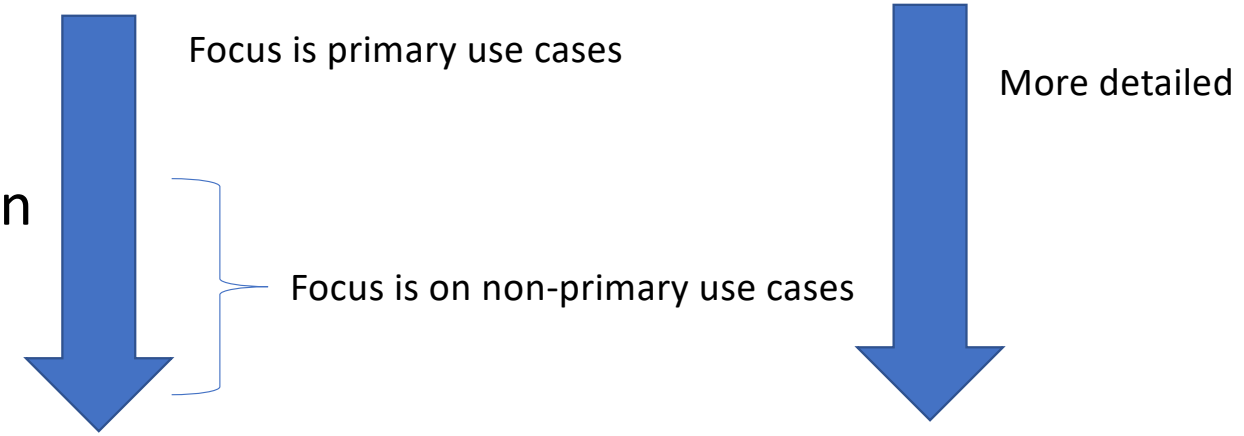


- Element are part of the system that compose the structures of the architecture.
  - Modules

- Element interaction design
  - The identification of the modules and their associated interfaces to support the non-primary use cases.
    - This is typically performed using sequence diagrams according to the decisions made during architectural design.

- Element internals design
  - The internal design of the elements identified as part of element interaction design

- There are three levels of design activities
  - Architectural design
  - Element interaction design
  - Element internals design



- There are three levels of design activities
  - Architectural design } high level design
  - Element interaction design } detailed design
  - Element internals design

Humberto and Rick both do not like terms "*high-level design*" or "*detailed design*"

Architectural drivers

Architectural Drivers

- Architectural drivers are critical to the success of the system.
- Architectural drivers need to be baselined and managed throughout the development life cycle.

  - Design purpose
  - Primary functionality
  - Quality attributes
  - Architectural concerns
  - Constraints

1) Design purpose

- Architect should be clear about the purpose of the design
  - When and why are you doing this architecture design?
  - Which business goals is the organization most concerned about at this time?
- Architecture design can be created for many reasons
  - as part of a project proposal
    - Could be for pre-sales process, or for internal project selection and prioritization in a company
  - as part of the process of creating an exploratory prototype.
    - to explore the domain or new technology new setting or to explore some quality attribute
  - architecture during development
    - For an entire new system, or major portion of a new system, or to add a portion of an existing system

- Purpose depends on other factors too
  - greenfield systems vs brownfield systems
  - **novel** domains vs **mature** domains vs **immature** domains

- Architect wants to add a new module to an existing banking system.
  - Designing software architecture of a brownfield system in a mature domain might be relatively straightforward.
  - Or the requirements of new module are simple but the existing system itself is complex and difficult to understand

- Greenfield systems in novel domains would be far more complex and risky

- The development organization's goals during development or maintenance may affect the architecture design process too.
  - The organization might be interested in designing for future extension

  Or
  - the CIO might have a specific like or dislike and wants to impose it

- ***Listing the design purpose helps the architect to be clear about business goals.***

2) Primary functionality

- Functionality is the ability of the system to do the work for which it was intended.
  - As opposed to quality attributes, the way the system is structured does not normally influence functionality.
- In terms of architectural design, allocation of functionality to elements, matters most rather than the functionality of system.

- A good architecture is one in which the most common changes are localized in a single or a few elements, and hence easy to make.

- Primary functionality is usually defined as functionality that is critical to achieve the business goals that motivate the development of the system.

- Typically, a **small percentage of your use cases or user stories** are likely to be primary.
- Some quality attribute scenarios are directly connected to the primary functionality in the system.
  - Example: Watch movie use case in a movie streaming application

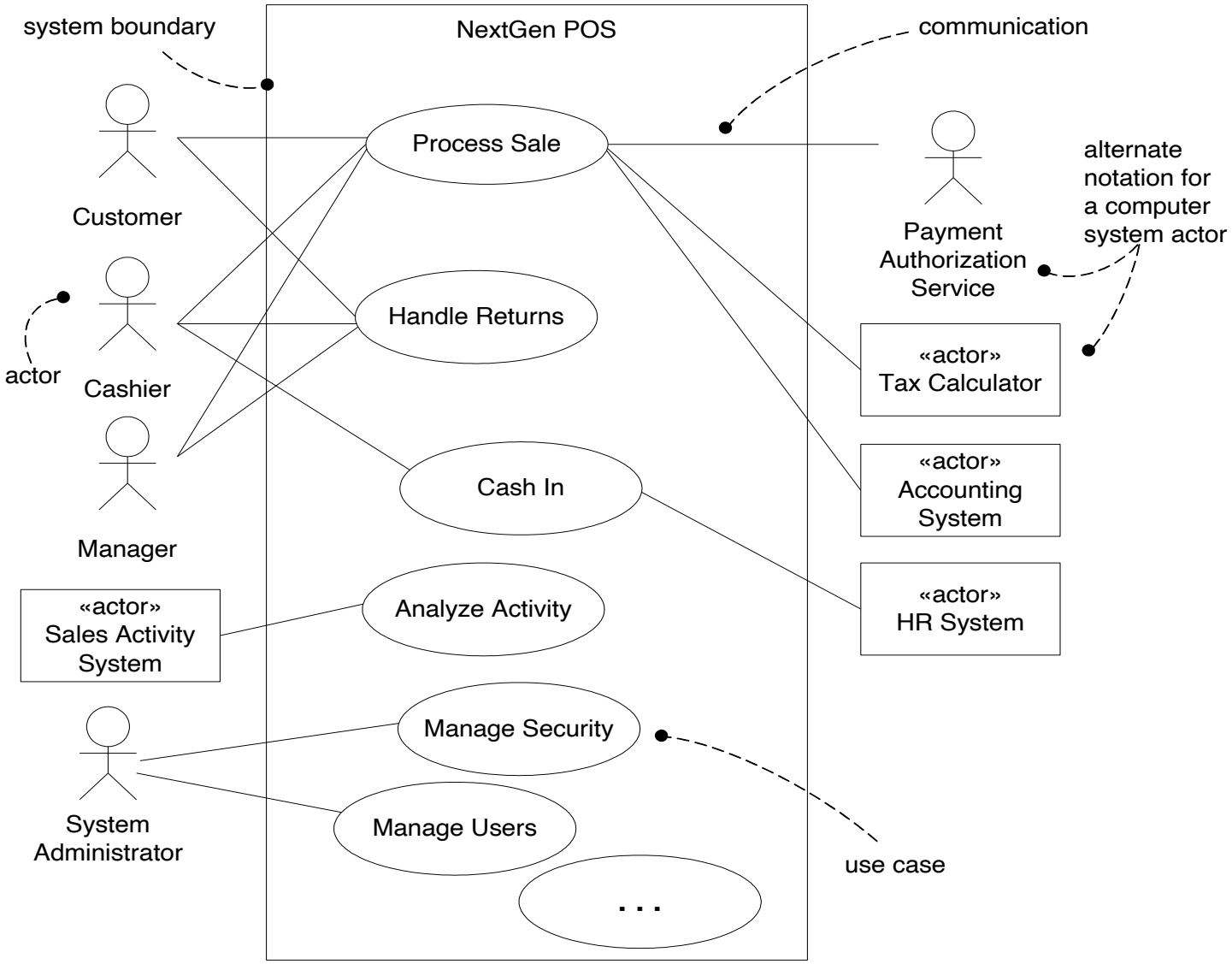
Use cases

- Use cases are narrative descriptions of domain processes as a specific sequence of actions and interactions between actors (users) and the system.

- Use Case Style
  - Black Box Use cases
    - Focus on what not how
- Use Case Formats
  - Brief/Summary or Fully dressed (detailed) use case

Use case model

- One UC Model for entire system representing all UCs



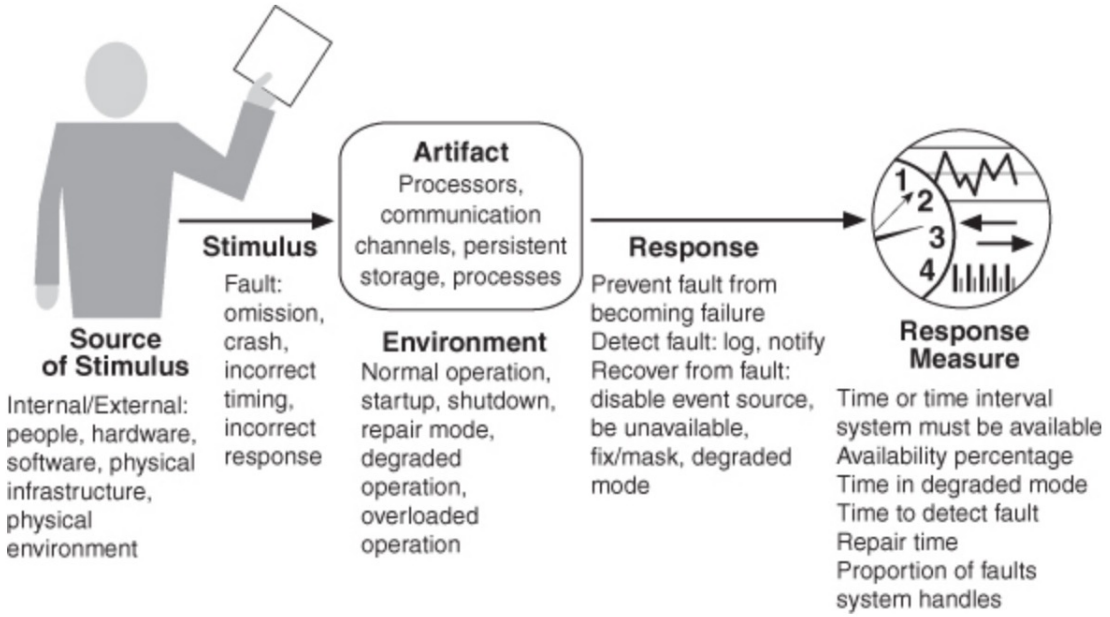
- Fully dressed format

Use case Section	Comment
Use case name	Start with a verb
Scope	The system under design
Level	“user goal” or “sub function”
Primary Actor	Calls on system to deliver its services
Stakeholders and interests	who cares about the system and what do they want
Preconditions	what must be true on start
Success Guarantee/Post conditions	What must be true on successful completion
Main Success Scenario	Unconditional happy path scenario of success
Extensions	Alternate scenario of success or failure
Special Requirements	Related NFRs
Technology and Data variation list	Varying I/O methods
Frequency of occurrence	Influences investigation, testing
Miscellaneous	Any open issues

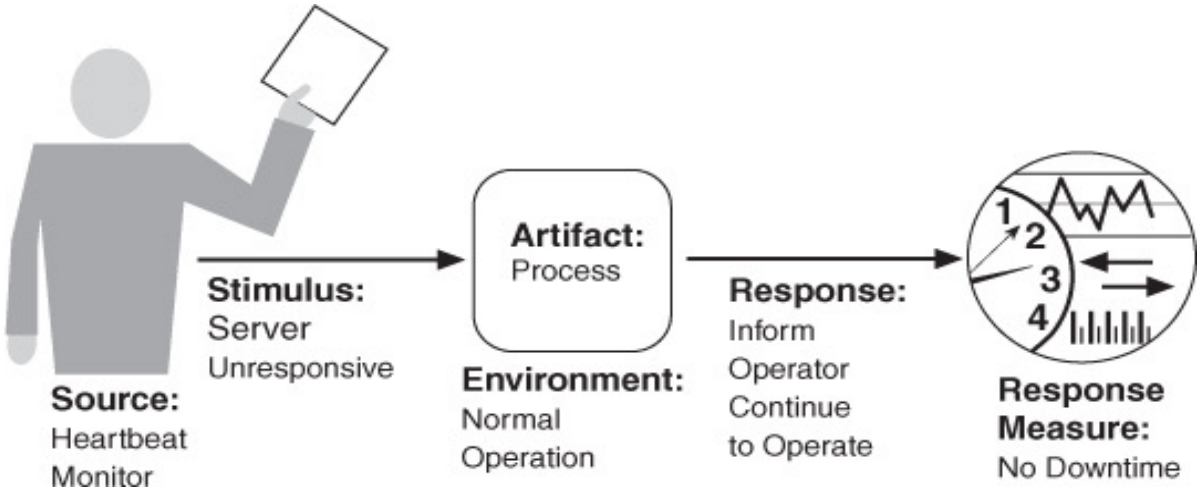
3) Quality attribute

- A measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders. Quality attributes are orthogonal to functionality.
  - unambiguous and testable.
- Quality attribute scenario parts:
  1. *Source of stimulus*
  2. *Stimulus*
  3. Environment
  4. *Artifact*
  5. *Response*
  6. *Response measure*

- An example of a general scenario



- A concrete availability scenario example
  - The heartbeat monitor determines that the server is nonresponsive during normal operations. The system informs the operator and continues to operate with no downtime.



Identifying and prioritizing quality attributes

- Analyzing requirements document

- Quality Attribute Workshop (QAW)
  - QAW is a facilitated, stakeholder-focused method to identify, prioritize and refine quality attribute scenarios
    - A modified version of QAW can be used involving all group members.

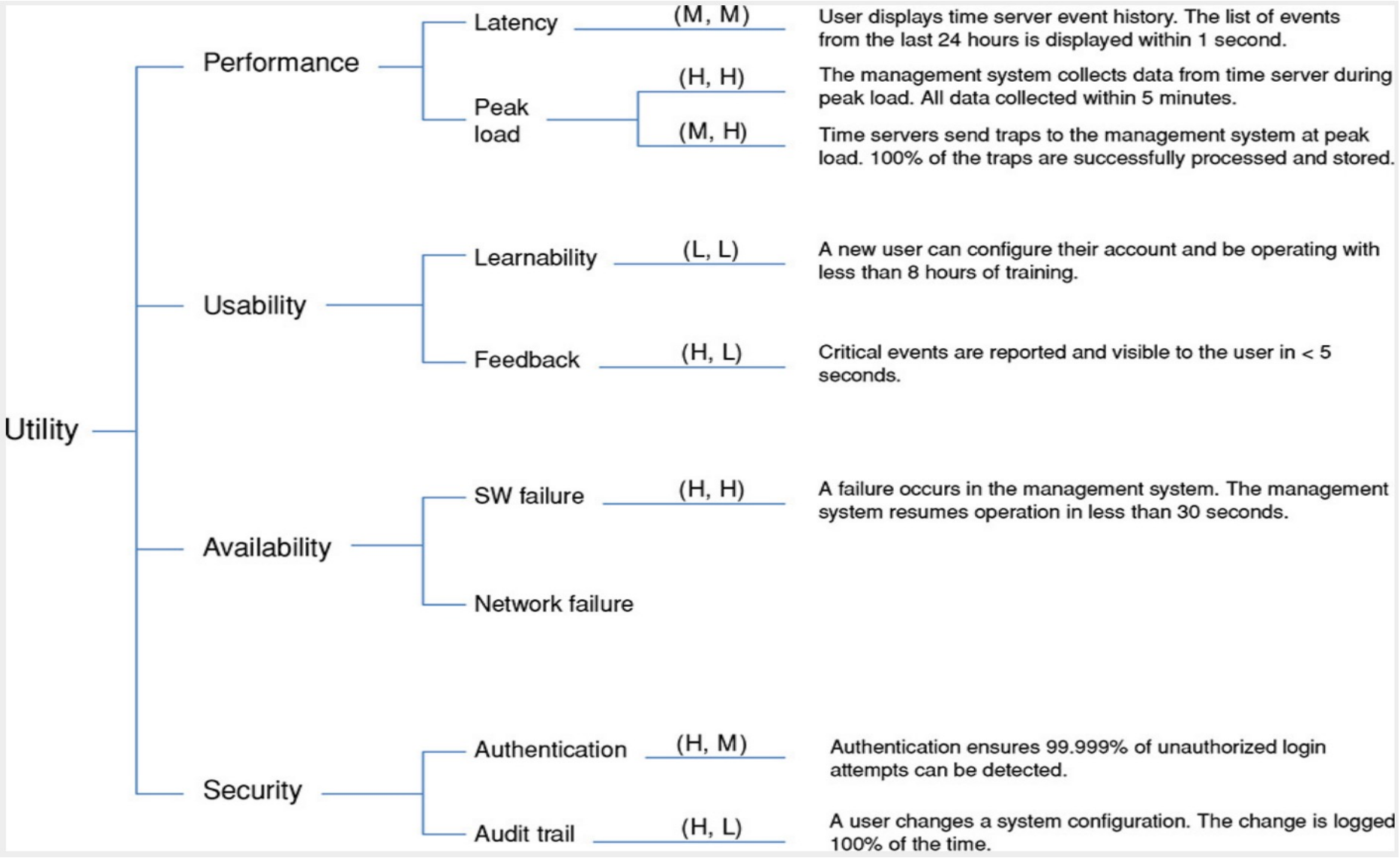
- Quality attribute utility tree attributes can be used to prioritize quality attribute

- Utility Tree Matrix

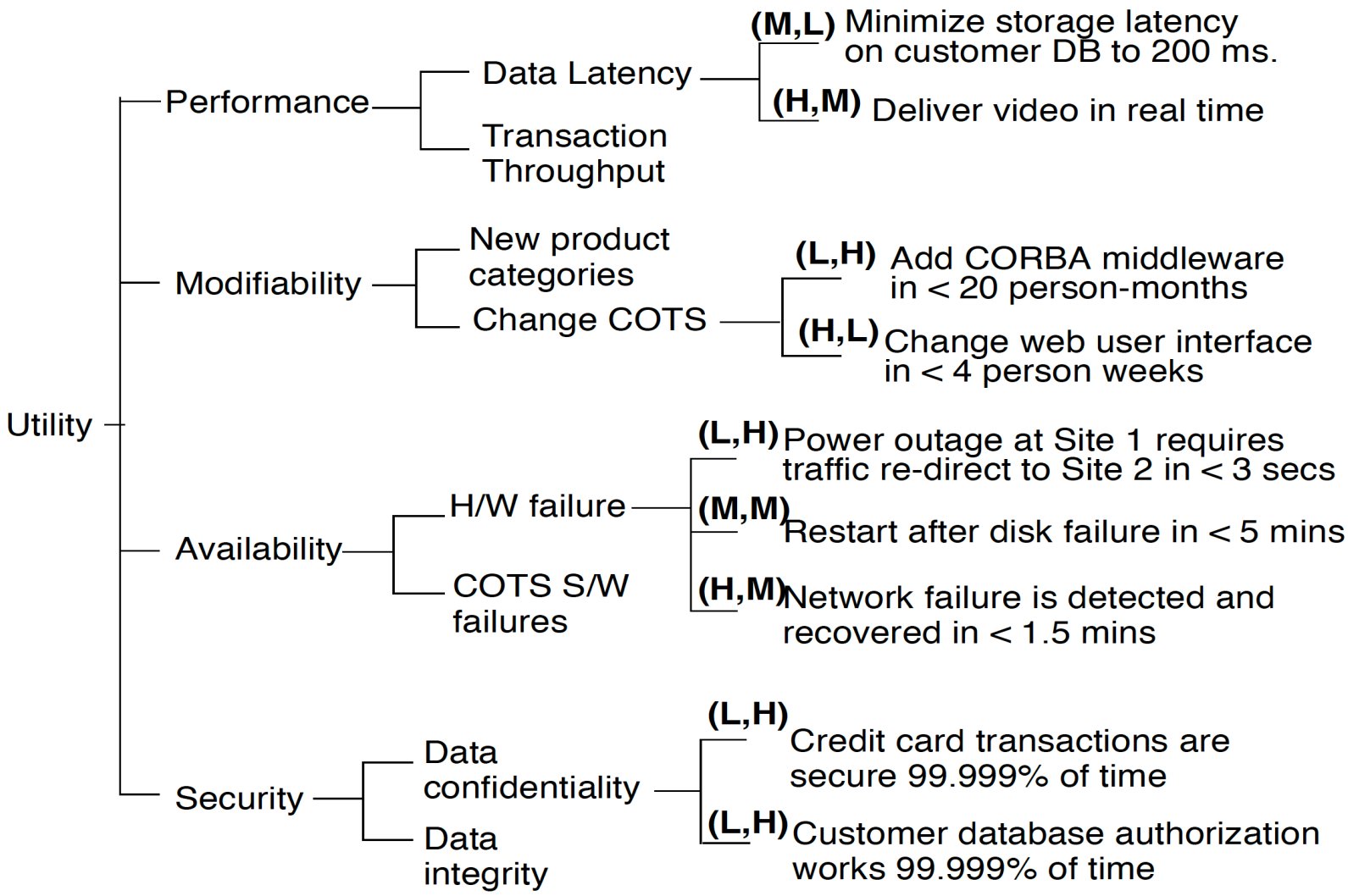
- Quality Attribute Utility tree uses **two dimensions of prioritization to rank scenario** using (X,Y):
  - Business Importance (X) Importance of each node to the success of the system
  - Technical Risk (Y) Degree of perceived risk posed by importance of each node

- Attributes that rate (H,H) deserve the most attention, then move to (H,M) and (M, H) ones

The letters in the parenthesis (e.g., M,M denote the importance and risk)



Decipher this?



Utility Tree Matrix

- A utility matrix can be developed by listing all QA ids

Business Importance/ Technical Risk	L	M	H
L	5, 6, 17, 20, 22	1, 14	12, 19
M	9, 12, 16	8, 20	3, 13, 15
H	10, 18, 21	4, 7	2, 11

4) Architectural Concerns

- Architectural concerns encompass additional aspects that need to be considered as part of architectural design but that are not expressed as traditional requirements.
- There are several different types of concerns:
  - General concerns
  - Specific concerns
  - Internal requirements
  - Issues

5) Constraints

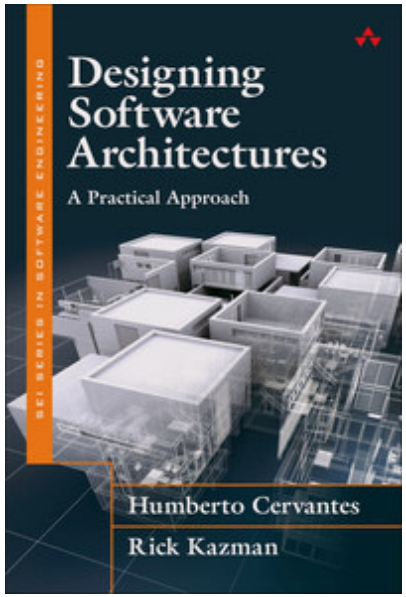
- A constraint is a decision over which an architect has little or no control
  - The constraints could be mandated technologies, needs to interoperate or integrate, laws and standards, Deadlines etc..
- Examples:
  - Technical constraint: Use of open source technologies
  - Non-technical constraint: System must adhere the Sarbanes-Oxley Act

Group Assignment

- Status??
  - What is happening now???
    - Any questions ☺
  - How much of today's lecture will help towards Assignment 1?
    - A lot

Reading (Must)

- Chapter 2 of T2 Designing Software Architectures, A Practical Approach, Authors: HumbertoCervantes and Rick Kazman, Publisher: Addison-Wesley, 2016



https://learning.oreilly.com/library/view/designing-software-architectures/9780134390857/

Chapters are really small, in fact the book is not thick at all thus don't be afraid. *The book has a very good* ***Glossary***

Do look up!

- System Use Case Diagrams
  - https://www.uml-diagrams.org/use-case-diagrams.html
- Plant UML use case diagramming tool
  - https://plantuml.com/use-case-diagram
    - An example (Just click and see): http://www.plantuml.com/plantuml/uml/LOzD2eCm44RtESMtj0jx01V5E_G4Gvngo2_912gbTsz4LBfylCV7p5Y4ibJlbEENG2AocHV1P39hCJ6eOar8bCaZaROqyrDMnzWqXTcn8YqnGzSYqNC-q76sweoW5zOsLi57uMpHz-WESslY0jmVw1AjdaE30IPeLoVUceLTslrL3-2tS9ZA_qZRtm_vgh7PxSqV

Acknowledgment

- Material (diagrams, text etc.) in this lecture is borrowed from the following sources:
  - Humberto Cervantes and Rick Kazman, “Designing Software Architectures, A Practical Approach”, Publisher: Addison-Wesley,2016

- Learning material and course ideation is provided by Dr. Javier Gonzalez Huerta, Dr. Muhammad Usman & Mr. Ehsan Zabardast