Software Systems Architecture (11491 &8746)

Semester 2 2023

UNIVERSITY OF CANBERRA

SOFTWARE SYSTEMS ARCHITECTURE

Lecture 2

Dr. Richa Awasthy

DISTINCTIVE BY DESIGN

WEEK 2: AGENDA

UNIVERSITY OF CANBERRA

1. Agile – Part II

2. Software Systems Architecture

ANNOUNCEMENT/REMINDERS

UNIVERSITY OF CANBERRA

- Quiz 2 opens today, closing on Sunday 13th Aug;
- There are no extensions for quizzes.
- Computer labs are only for Week 6 for the mid-term assessment. Make sure you are enrolled in one of these.
- Tutorial Classes
  - Tutorials begin this week (Week 2). Most of the tutorials are face-to-face on campus. Please find the venue and time details on Allocate.
  - Two tutorials are online as indicated in the timetable. These will be accessible via the Virtual Room link through the Canvas site for the unit.
  - Students must attend the tutorial classes they are enrolled in, unless authorized by the Unit Convener.
  - If you still haven't enrolled into a tutorial and lab, please do it at the earliest.
    - If you have any issue related to tutorial allocation, send an email to richa.awasthy@canberra.edu.au so we can sort out your situation.

UNIVERSITY OF CANBERRA

1. Agile – Part II

THE SCRUM

UNIVERSITY OF CANBERRA

- Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.
- The Scrum framework consists of:
  - ✓ Scrum Teams and their associated roles,
  - ✓ events,
  - ✓ artifacts, and
  - ✓ rules.
  - Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.
- **Note** - Across the industry, there are misconceptions that Scrum means no documentation, scrum team consists of only developers, and so on. It is not entirely so.

THE SCRUM PROCESS

UNIVERSITY OF CANBERRA

- In Scrum, the prescribed events are used to create regularity. All events/tasks are time-boxed events, such that every event has a maximum duration.
- **Sprint**: it is at the heart of Scrum, is a time-box of two weeks or one month during which a potentially releasable product increment is created.
- In **Sprint planning**, the work to be performed in the Sprint is planned collaboratively by the **Scrum Team**.
- The **Daily Scrum Meeting** is a 5 to 15-minute for the Scrum Team to synchronize the activities and create a plan for that day.
- A **Sprint Review** is held at the end of the Sprint to inspect the Increment and make changes to the Product Backlog, if needed.
- The **Sprint Retrospective** occurs after the Sprint Review and prior to the next Sprint Planning. In this meeting, the Scrum Team is to inspect itself and create a plan for improvements to be enacted during the subsequent Sprint.

SCRUM ROLES

UNIVERSITY OF CANBERRA

- The Scrum Team consists of three roles, namely a **ScrumMaster**, a **Product Owner**, and the **Team**.
- **ScrumMaster**: manages the scrum process.

  He/she is responsible for-
  - making the process run smoothly
  - removing obstacles that impact productivity
  - organizing and facilitating the critical meetings
- **Product Owner:** is the voice of the customer/ Stakeholder.
- **The Scrum Team**: creates the shippable/ deliverables of the project.

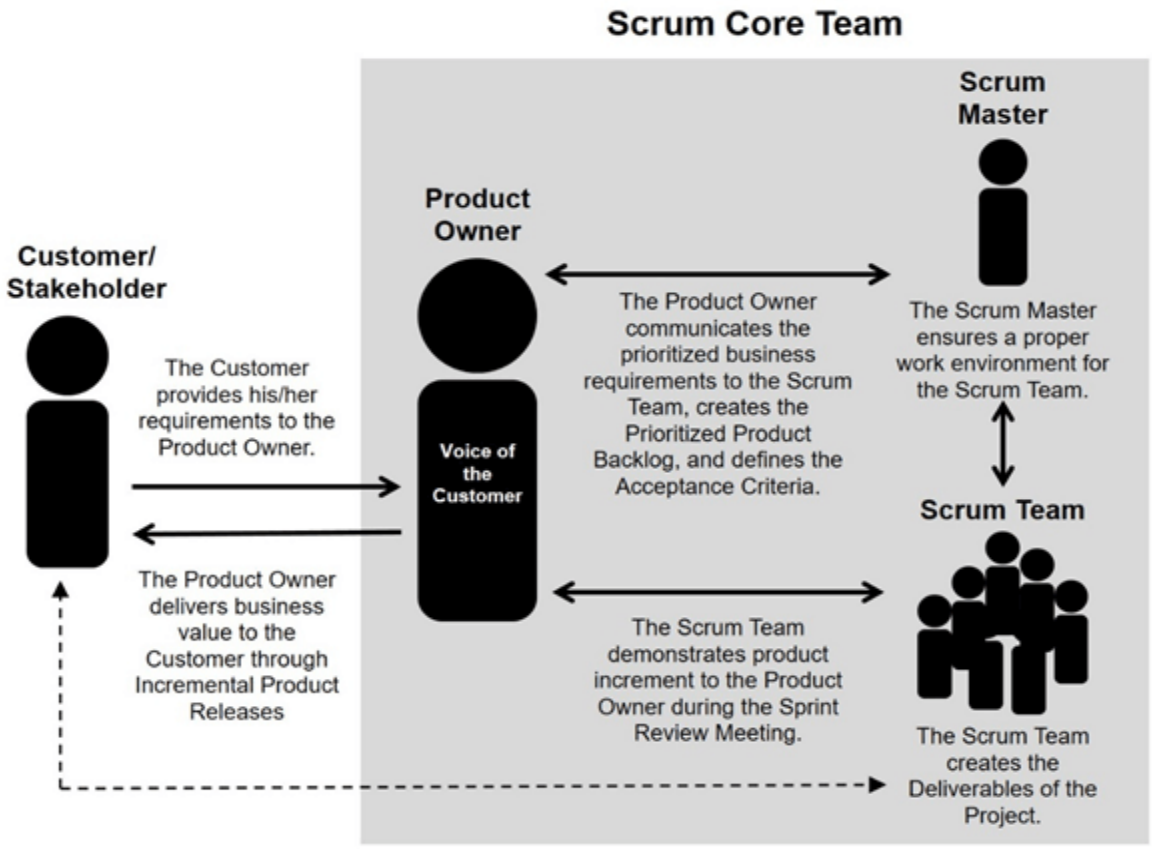


Image Source: SBOK Guide 3rd Edition

USER STORIES

UNIVERSITY OF CANBERRA

What is a user story?

- In Agile a user story is a short, informal, plain language description of what a user wants to do within a software product to gain something they find valuable.

- A user story represents a small piece of business value that a team can deliver in an Agile iteration;

- While traditional requirements (like use cases) try to be as detailed as possible, a user story must have only enough information to guide the development; then it can be incremented over time.

Source: © https://help.rallydev.com/writing-great-user-story

HOW DO I WRITE USER STORIES?

UNIVERSITY OF CANBERRA

- When getting started with user stories, a template can help:
  - *As a* <user type>, *I want to* <function> *so that* <benefit>.
- Examples:
  - *As a* consumer, *I want to* have a shopping cart functionality *so that* I can easily purchase items online.
  - *As an* executive, *I want to* generate a report *so that* I can understand which departments need to improve their productivity.

Source: © https://help.rallydev.com/writing-great-user-story

CREATING USER STORIES

UNIVERSITY OF CANBERRA

- The user story needs to have, at a minimum, the following parts:

  - **Title:** *<a name for the user story>*
  - **As a** *<user or persona>*
  - **I want to** *<take this action>*
  - **So that** *<I get this benefit>*

- The story should also include validation steps - steps to take to know that the working requirement for the user story is correct. That step is worded as follows:

  - **When I** *<take this action>*, **this happens** *<description of action>*

- User stories may also include the following:

  - **An ID:** A number to differentiate this user story from other user stories.
  - **The value and effort estimate:** *Value* is how beneficial a user story is to the organization creating that product. *Effort* is the ease or difficulty in creating that user story.
  - **The person who created the user story:** Anyone on the project team can create a user story.

Source: © Agile for Dummies by Ambler & Holitza

EXAMPLE OF A USER STORY

UNIVERSITY OF CANBERRA

- Card-based user story example.

Title	Transfer money between accounts
As	Carol,
I want to	review fund levels in my accounts and transfer funds between accounts
so that	I can complete the transfer and see the new balances in the relevant accounts.

Value | Jennifer (Author) | Estimate

Title	
As	<personal/user>
I want to	<action>
so that	<benefit>

Value | Author | Estimate

Source: © Agile for Dummies by Ambler & Holitza

ACCEPTANCE CRITERIA & DEFINITION OF DONE

UNIVERSITY OF CANBERRA

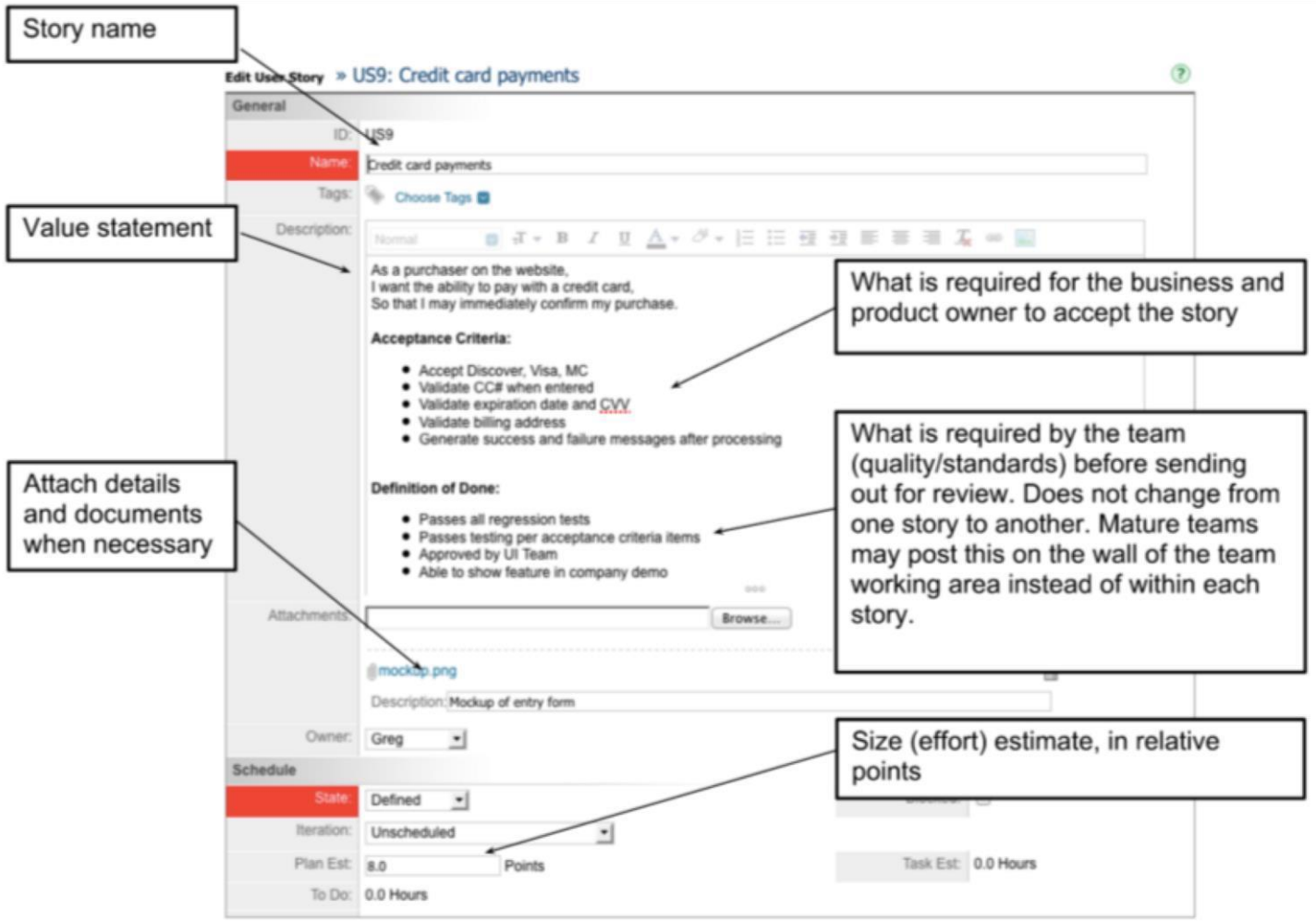
- Acceptance criteria:
  - What is required for the business and product owner to accept the user story;
  - Related to "functionalrequirements."

- Definition ofdone:
  - What is required by the team (quality/standards) before sending something for review;
  - It usually does not change from a user story to the other.

Source: © https://help.rallydev.com/writing-great-user-story

ACCEPTANCE CRITERIA & DEFINITION OF DONE

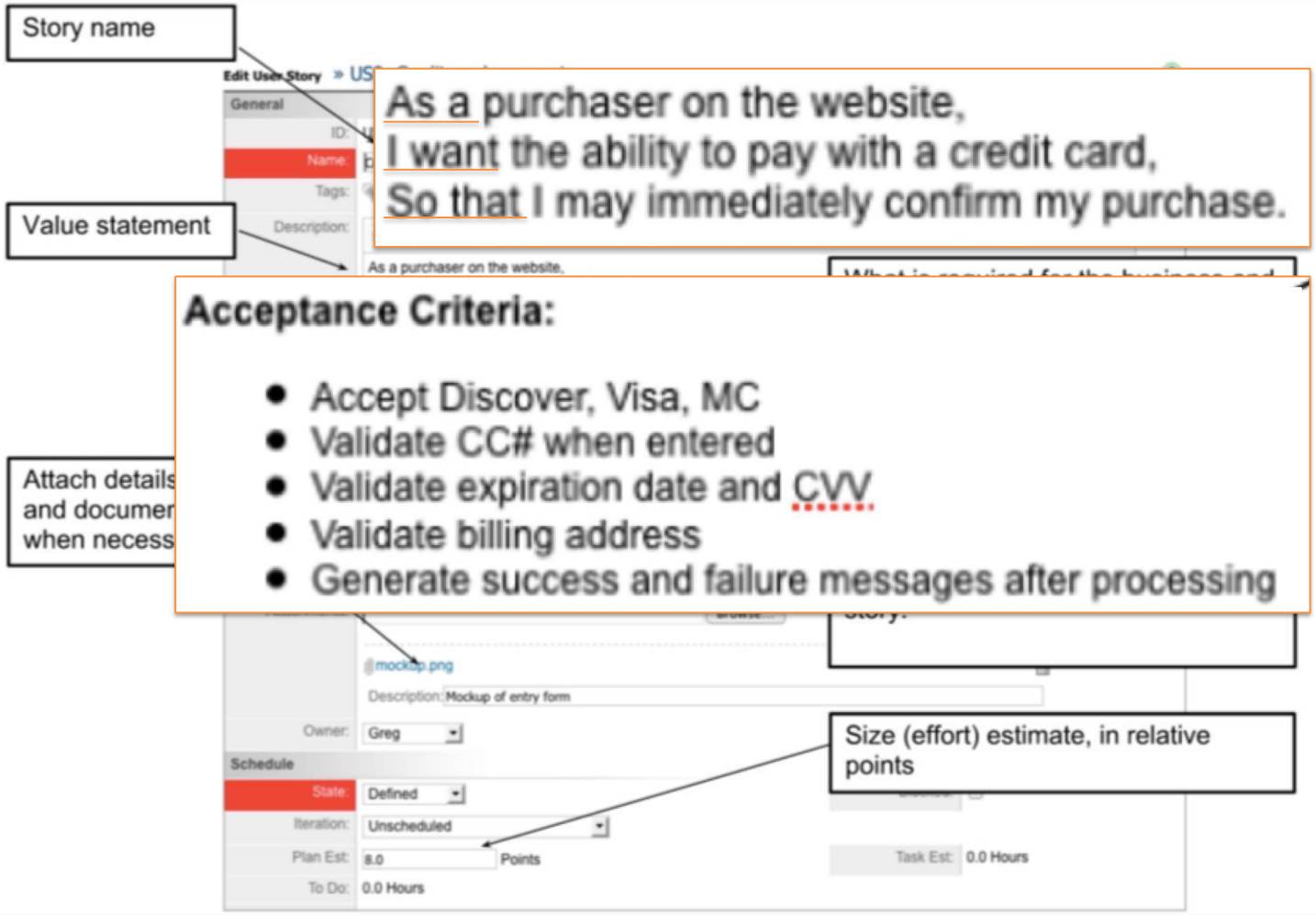
UNIVERSITY OF CANBERRA



Source: © https://help.rallydev.com/writing-great-user-story

ACCEPTANCE CRITERIA & DEFINITION OF DONE

UNIVERSITY OF CANBERRA



Source: © https://help.rallydev.com/writing-great-user-story

WELL-FORMED USER STORIES

UNIVERSITY OF CANBERRA

Well-formed stories will meet the criteria of Bill Wake's INVEST acronym:

Independent	We want to be able to develop in any sequence.
Negotiable	Avoid too much detail; keep them flexible so the team can adjust how much of the story to implement.
Valuable	Users or customers get some value from the story.
Estimatable	The team must be able to use them for planning.
Small	Large stories are harder to estimate and plan. By the time of iteration planning, the story should be able to be designed, coded, and tested within the iteration.
Testable	Document acceptance criteria, or the definition of done for the story, which lead to test cases.

Source: © https://help.rallydev.com/writing-great-user-story

WRITING USER STORIES FOR THE DRMS

UNIVERSITY OF CANBERRA

- In order to create a user story, you have to imagine yourself as a stakeholder/client of the system;

Source: © https://help.rallydev.com/writing-great-user-story

YOU NEED TO UNDERSTAND THE SYSTEM

UNIVERSITY OF CANBERRA

- You *need to know what you want from the system to be able to write a user story*!
- For our tutorials, if you have questions about the DRMS, you should read the article about DRMS made available on week 1, named as "*RMSComparison*", or take a look to similar systems:
  - SLURM: http://slurm.schedmd.com/
  - PBS Professional: http://www.pbsworks.com/
  - MOAB HPC SUITE: http://www.adaptivecomputing.com/
- They should give you adequate information to complete the tutorial assessment activities.

WRITING USER STORIES FOR THE DRMS

UNIVERSITY OF CANBERRA

For example:

- "As a user of DRMS system, I want to be able to submit jobs so that I can use the distributed resources available on the cloud environment/supercomputer."

Source: © https://help.rallydev.com/writing-great-user-story

TUTORIAL ASSIGNMENT: THE SYSTEM

UNIVERSITY OF CANBERRA

- The system that will be used during the tutorial classes is:

Distributed Resource Management System (DRMS)

2. Software Systems Architecture

LEARNING OUTCOMES

UNIVERSITY OF CANBERRA

Software Systems Architecture:

1. Demonstrate a firm understanding of the principles of software architecture, architectural best-practices, and how architecture is used in modern software engineering;

2. Understand the role of a software architect in software engineering practice;

5. Communicate the architecture to stakeholders and demonstrate that it has met their requirements.

WHAT IS SOFTWARE ARCHITECTURE

UNIVERSITY OF CANBERRA

- The **software architecture** of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both."
- A **structure** is a *set of elements* held together by *a relation*.
- **Software systems** are composed of *many structures*, and no single structure holds claim to being the architecture.
- There are **three important categories of architectural structures**.
  - Module
  - Component and Connector
  - Allocation

MODULE STRUCTURES

UNIVERSITY OF CANBERRA

- Some structures partition systems into implementation units (related to codes) with specific objectives, which we call modules.
- Modules are assigned specific computational responsibilities and are the basis of work assignments for developers/programming teams.
- In large projects, these elements (modules) are subdivided for assignment to sub-teams.

RUNTIME/COMPONENT-AND-CONNECTOR STRUCTURES

UNIVERSITY OF CANBERRA

- These structures focus on the way the elements interact with each other at **runtime** to carry out the **system's functions**.
- Help to determine quality attributes such as security, performance and availability.

ALLOCATION STRUCTURES

UNIVERSITY OF CANBERRA

- Allocation structures describe the mapping from software structures to the system's environments; where software is created and executed:
  - Organizational
  - Developmental
  - Installation
  - Execution
- For example:
  - Modules are assigned to teams to develop and assigned to places in a file structure for implementation, integration, and testing.
  - Components are deployed onto hardware in order to execute.

WHICH STRUCTURES ARE ARCHITECTURAL

UNIVERSITY OF CANBERRA

- Structures play such an important role in our perspective on software architecture because of the analytical and engineering power they hold.
- A structure is architectural if it supports reasoning about the system and the system's properties.
- The reasoning should be about an attribute of the system that is important to some stakeholder.

ARCHITECTURE IS AN ABSTRACTION

UNIVERSITY OF CANBERRA

- Abstraction must be applied in software architecture.
- An architecture comprises software elements and how the elements relate to each other.
  - An architecture specifically **omits certain information** about elements that is **not useful for reasoning** about the system.
  - It omits information that has no ramifications outside of a single element.
  - An architecture selects certain details and suppresses others.
  - Private details of elements—details having to do solely with internal implementation—are not architectural.
- The architectural abstraction lets us look at the system in terms of its elements, how they are arranged, how they interact, how they are composed, what their properties are that support our system reasoning, and so forth.

“GOOD” AND “BAD” ARCHITECTURES

UNIVERSITY OF CANBERRA

“

– There is no such thing as an inherently good or bad architectures

– Architectures are either more or less fit for some purpose

– Architectures can be evaluated but only in the context of specific stated goals

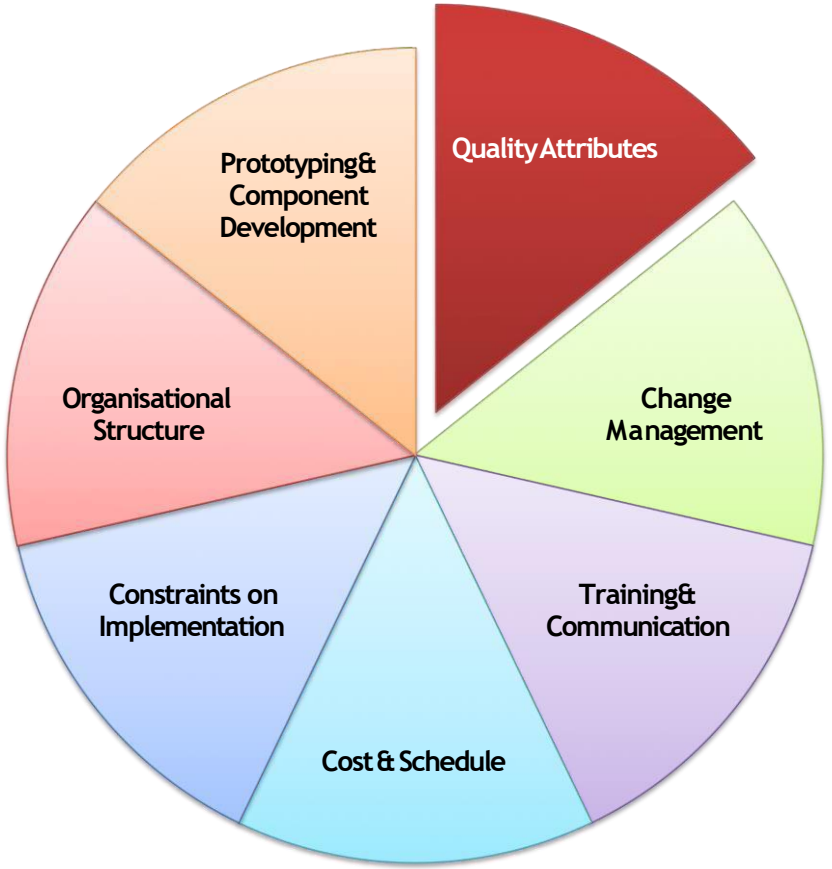
– There are, however, good ‘rules of thumb’

”

- The purpose is to learn how to design **adequate** architectures, and solve the problem satisfactorily within the architectural context

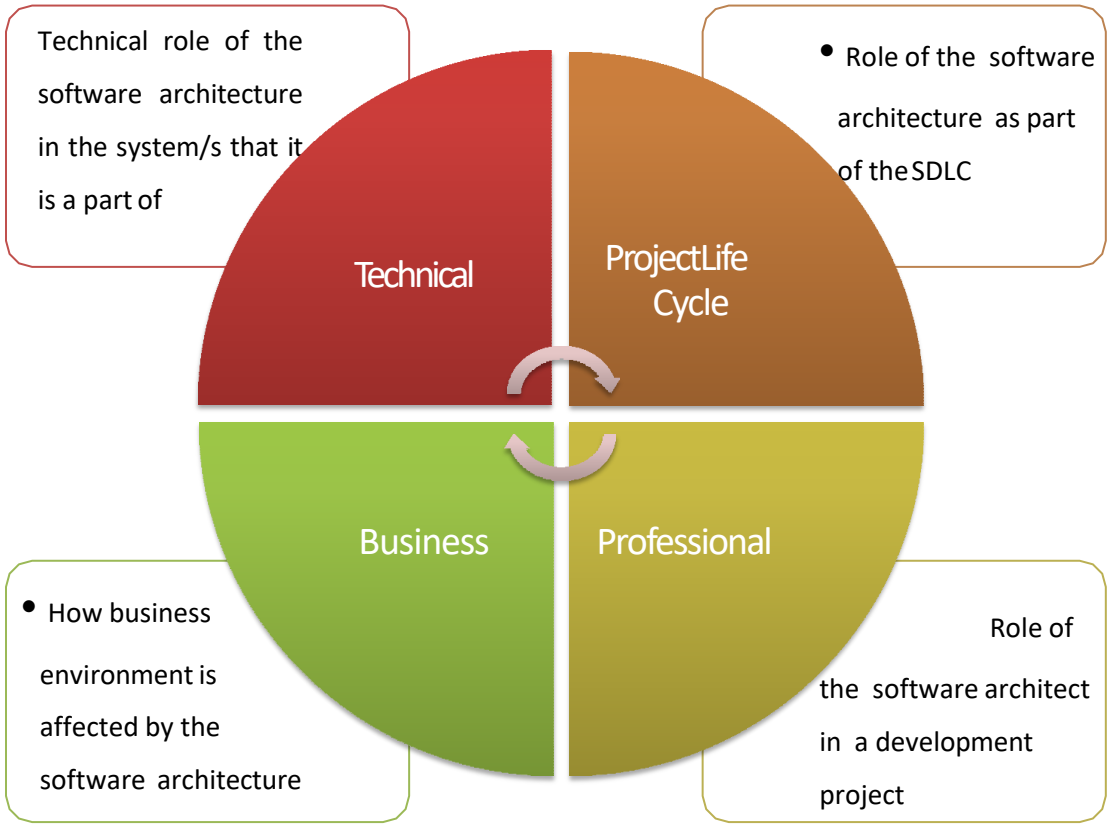
WHY IS SOFTWARE ARCHITECTURE IMPORTANT?

UNIVERSITY OF CANBERRA



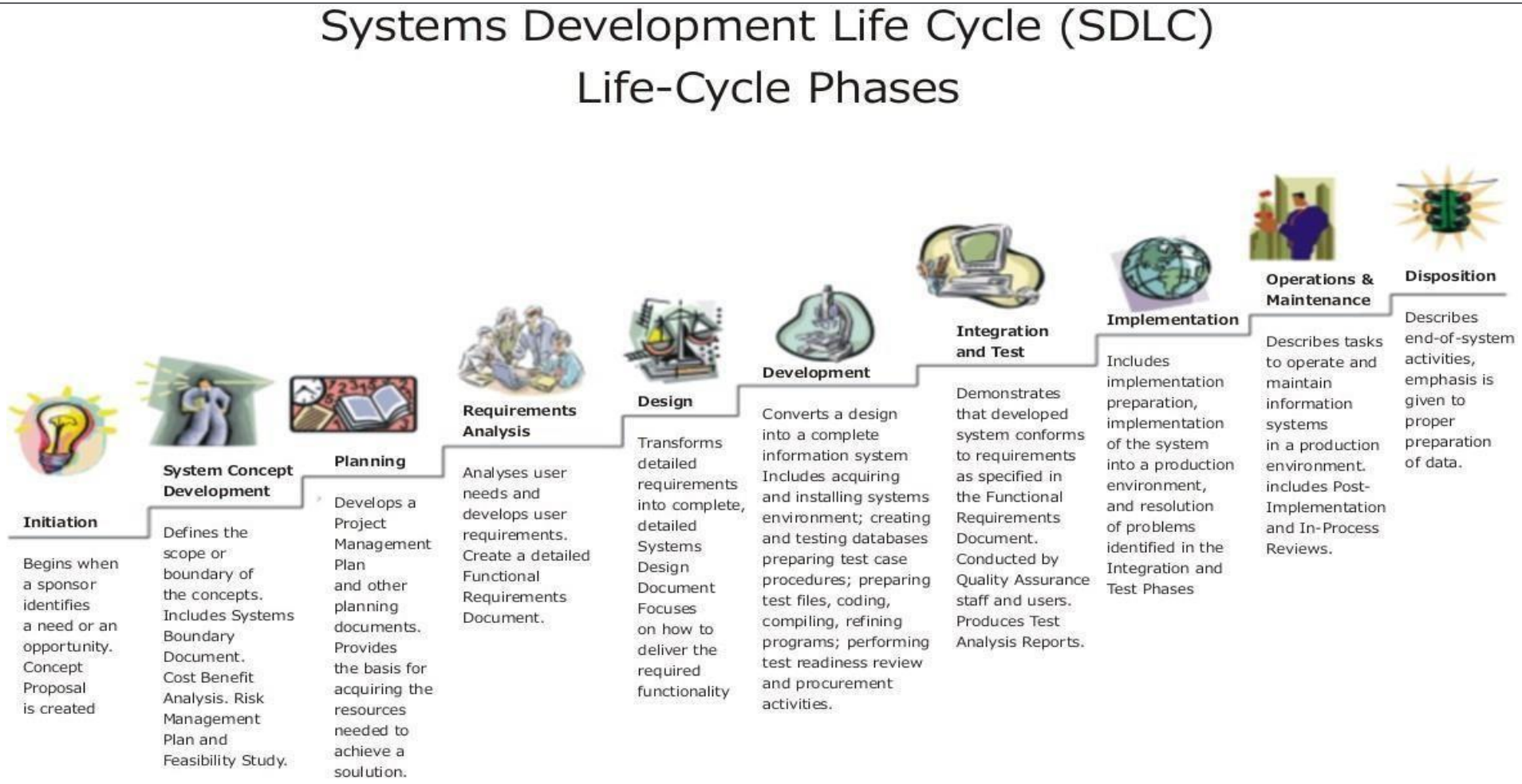
CONTEXTS OF SOFTWARE ARCHITECTURE

UNIVERSITY OF CANBERRA



SDLC: PHASES

UNIVERSITY OF CANBERRA



Source: US Department of Justice (redrawn by Eugene Vincent Tantog) - Information Resources Management, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5530145

BUSINESS CONTEXT

UNIVERSITY OF CANBERRA

- **Architectures and systems must be built for a reason**
  - The investment should be justified
  - Reasons may change as the business evolve
  - The quality attributes of a system should be justified in terms of added value
- **Architects must understand the business context!**

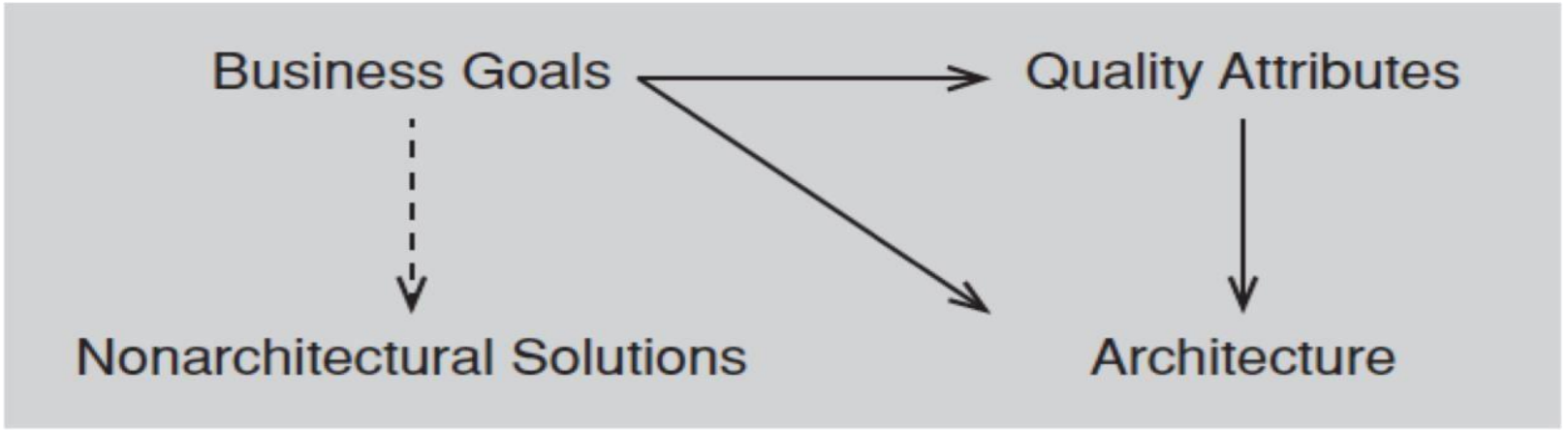


FIGURE 3.2 Some business goals may lead to quality attribute requirements (which lead to architectures), or lead directly to architectural decisions, or lead to nonarchitectural solutions.

PROFESSIONAL CONTEXT

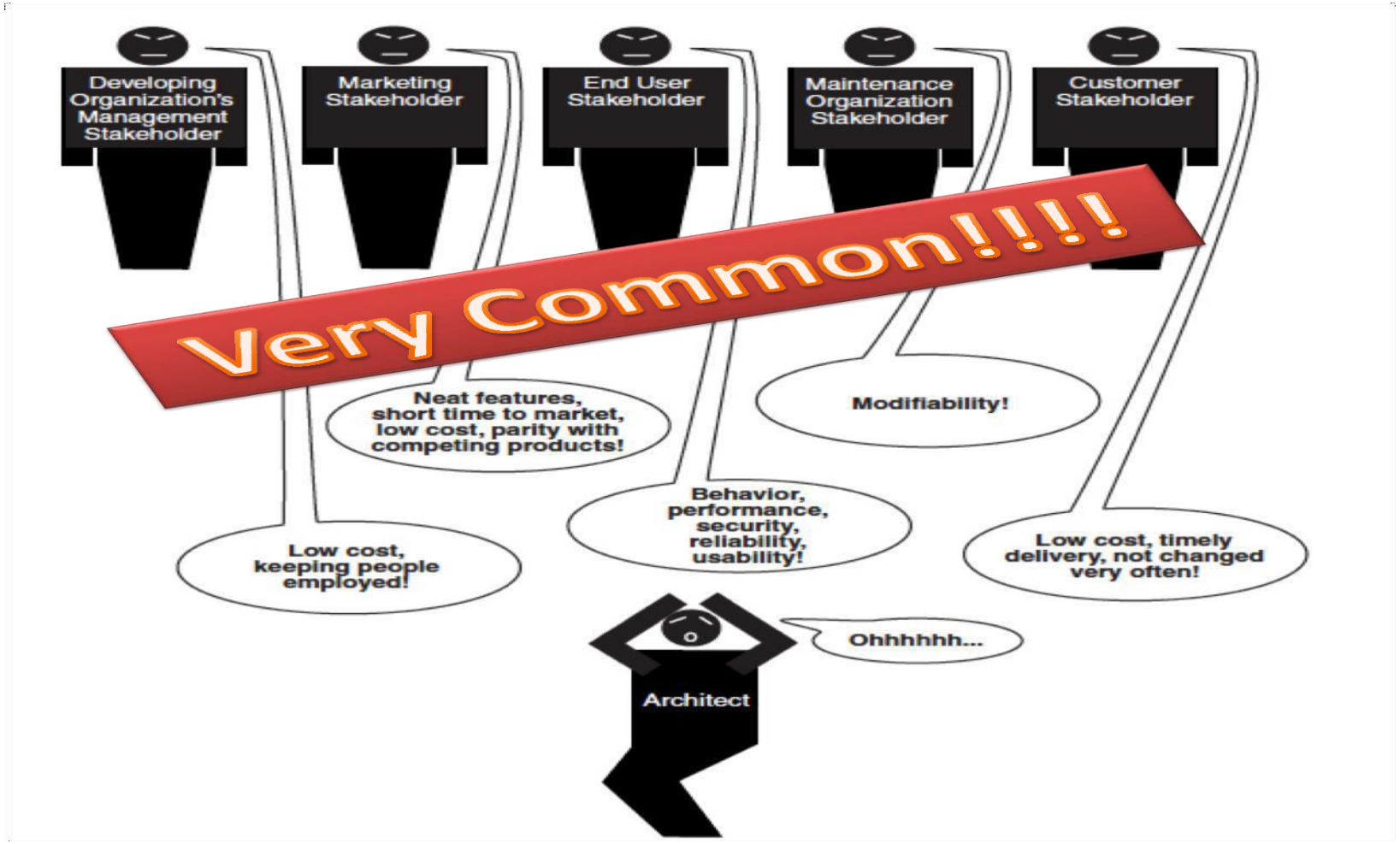
UNIVERSITY OF CANBERRA

- Architects need knowledge
- Architects need non-technical skills and abilities
- Architects must work and think outside the “architectural bubble”
- Stakeholders
    - Stake: “Something that is staked for gain or loss; e.g., prize in a contest; interest
    - Stakeholder: has a stake in the system, i.e., something to gain if system is successful

Source: Merriam-Webster onlinedictionary

PROFESSIONAL CONTEXT (STAKEHOLDER OF A SYSTEM)

UNIVERSITY OF CANBERRA



Source: © Len Bass, Paul Clements, Rick Kazman, distributed under Creative Commons Attribution License.

ARCHITECTURE INFLUENCE CYCLE

UNIVERSITY OF CANBERRA

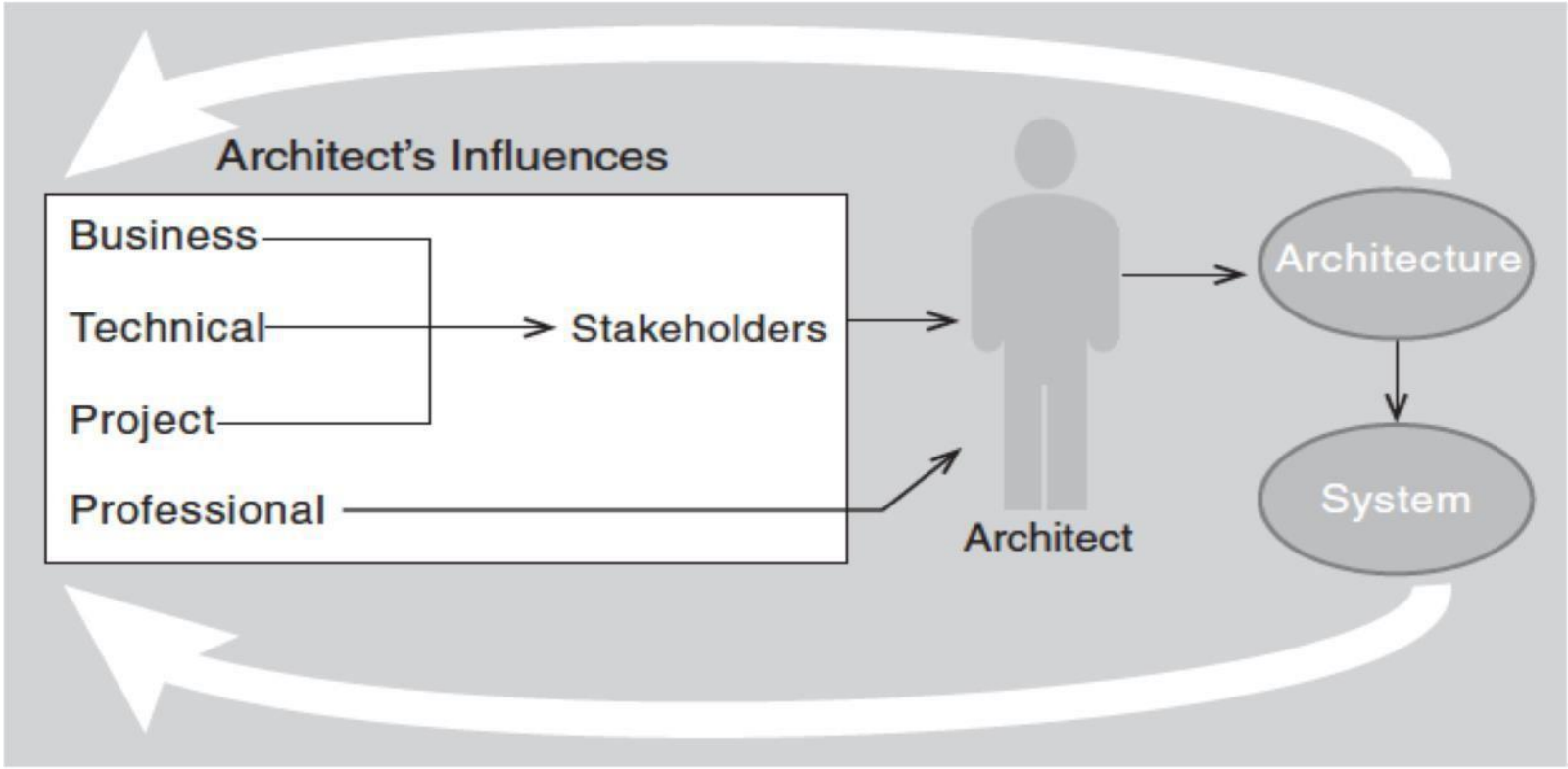


FIGURE 3.5 Architecture Influence Cycle

REQUIREMENTS

UNIVERSITY OF CANBERRA

- **Requirement**
  - "Something wanted or needed"
  - "Something essential to the existence or occurrence of something else"
    - Source: Merriam-Webster online dictionary
- **Requirement**

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

Source: IEEE Computer Society (1990). "IEEE Standard Glossary of Software Engineering

SYSTEM REQUIREMENTS

UNIVERSITY OF CANBERRA

- **System requirements can be categorized as:**

Functional Requirements	Quality Attribute Requirements	Constraints
• State what the system must do or how it must behave or react to run-time stimuli • Functionality is the ability of a system to do the work for which it was intended • Functionality does not determine architecture	• Annotate (qualify) functional requirements • Qualification might be how fast the function must be performed, how resilient it must be to erroneous input, how easy the function is to learn, etc.	• Design decisions that have already been made for you • They cannot be changed, so architecture has to be built on top of the constraints

ASR- ARCHITECTURALLY SIGNIFICANT REQUIREMENTS

UNIVERSITY OF CANBERRA

- Architectures are built to design systems; systems must satisfy the requirements
    - However, not all requirements are relevant to the architecture design
- Architecturally Significant Requirement (ASR):
    - ✓ Requirement which has an impact on the architecture

GATHERING ASR

UNIVERSITY OF CANBERRA

Requirements Documents

- Usually do not contain all ASRs
- Functional requirements usually do not affect the architecture

Business Goals

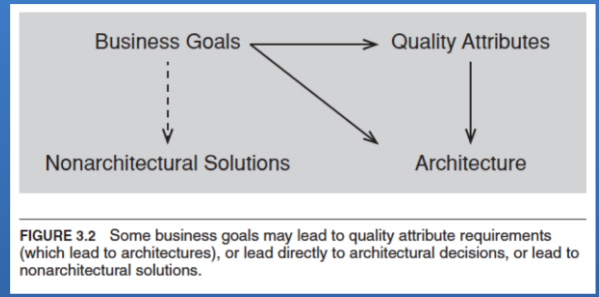


FIGURE 3.2 Some business goals may lead to quality attribute requirements (which lead to architectures), or lead directly to architectural decisions, or lead to nonarchitectural solutions.

Stakeholders

- Good practice to interview them
- But don't expect ASRs directly defined

Techniques

- QAW (Quality Assurance Workshop)
- PALM (Pedigreed Attribute eLicitation Method)

CAPTURING ASRS

UNIVERSITY OF CANBERRA

- ASRs should be captured in documents for further reference and approval

ARCHITECTURE DOCUMENTATION

UNIVERSITY OF CANBERRA

- Proper documentation is required to take full advantage of an architecture
- Architecture documentation has to be:
  - Aligned with the requirements
  - Descriptive and Prescriptive
    - Describes decisions that had already been made
    - Prescribes constraints on decisions that have yet to be made
  - Concrete and with enough information to support system development
  - Written in a way that can be understood by new employees and stakeholders

NOTATIONS

UNIVERSITY OF CANBERRA

"A system of characters, symbols, or abbreviated expressions used in an art or science or in mathematics or logic to express technical facts or quantities."

Source: Merriam-Webster online dictionary

NOTATIONS

UNIVERSITY OF CANBERRA

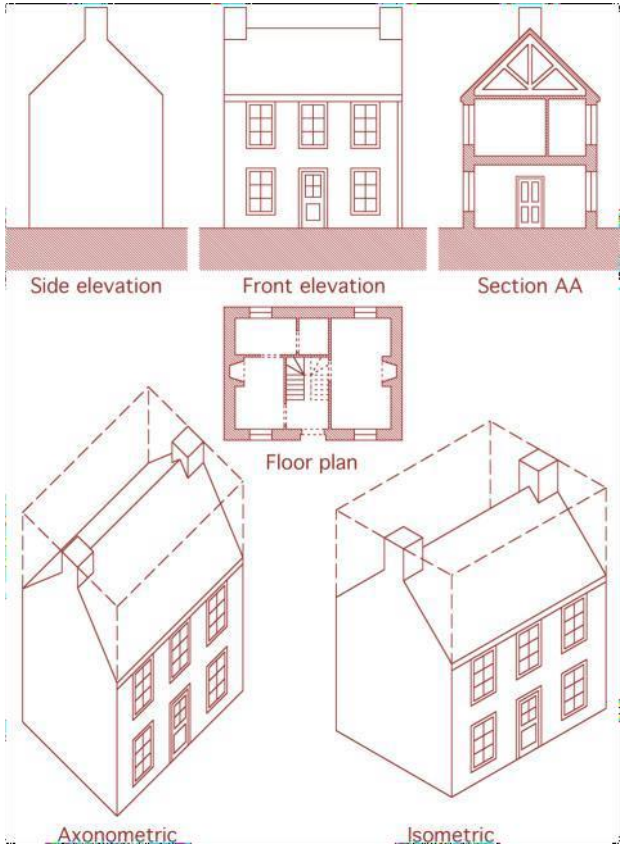
- A notation must be adopted to document an architecture

Informal	Semiformal	Formal
• Architecture is presented by using general-purpose diagrams and tools,and frequently natural language • Bespoke syntax and semantic • E.g.: Microsoft, PowerPoint	• Architecture is presented by using a standardised notation • Syntax is well organised • Semantics is rudimentary or non- existing • E.g.: UML	• Architecture is presented by using a precise, mathematically- based notation • Formal syntax and semantics • E.g.: ADLs – Architecture Description Languages

VIEWS

UNIVERSITY OF CANBERRA

- A notation must be adopted to document an architecture
- A view is a representation of a structure
- A view describes the interrelated architectural elements and the relations among them
- Views are used in architectural descriptions to document different aspects of a system



EXAMPLE: CLIENT-SERVER SYSTEM

UNIVERSITY OF CANBERRA

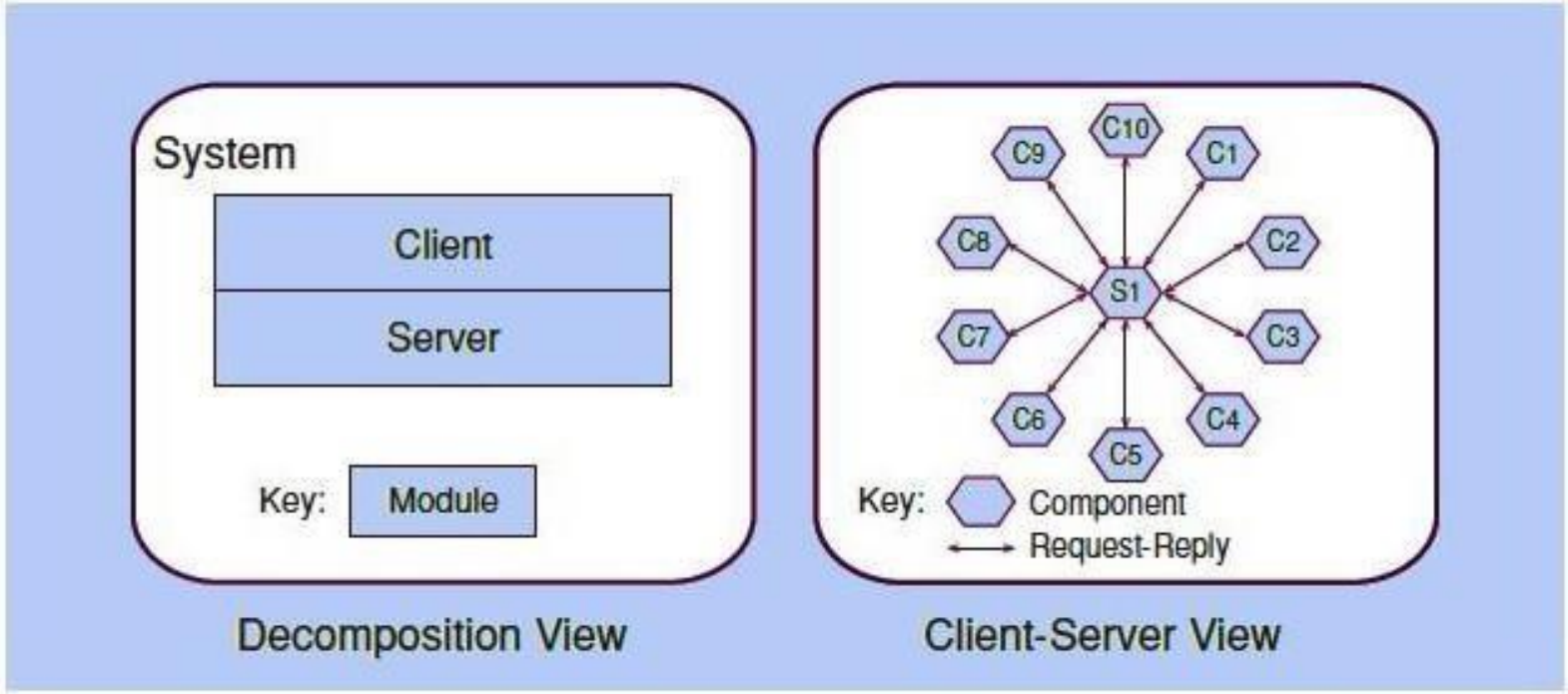


FIGURE 1.2 Two views of a client-server system

ADDITIONAL RECOMMENDATIONS

UNIVERSITY OF CANBERRA

Some architectures change faster than the architect's ability to document the changes

- E.g., systems with extraordinarily frequent release-and-deploy cycles
- For those cases:
  - Document the elements that are common to all versions of the system
  - Build and document an architecture that is allowed to change, and specify which changes are acceptable

ADDITIONAL RECOMMENDATIONS

UNIVERSITY OF CANBERRA

- For Agile development projects
  - ✓ Create a template or use a standard way to document the design decisions
  - ✓ Fill in the templates when the information becomes available, but only if this information will simplify someone's job in the future
  - ✓ Try to produce only the required information to allow the team progress
  - ✓ Be agile: make quick drafts or take pictures of manually designed models when working on whiteboards, and don't feel bad in using them in your presentations!

References

- Len, Bass, Clements Paul, and Kazman Rick. (2013) "Software architecture in practice." Boston, Massachusetts Addison. 3rd Edition. – Mainly Chapter 1 to 3.
- Santana, F.S., Stange, R. L., Saraiva, A. M., Pinaya, J.L., & Becerra, J.L. (2012, August). A complete RM-ODP case-study to integrate geospatial services and ecological niche modeling systems. In Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on (pp. 239-246). IEEE.
- Lecture by Dr. Karam Salam
- Microsoft Application Architecture Guide, 2nd Edition. Available online at https://msdn.microsoft.com/en-us/library/ff650706.aspx .