Reference Chapter 16
Software Architecture in Practice
Third Edition
Len Bass
Paul Clements
Rick Kazman

Software Architecture

Designing & Documenting the Architecture #1

Architecture and Requirements

Harvinder S Jabbal
Module 19 (RL6.1)

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

innovate achieve lead

Architecture and Requirements

- Define Architecturally Significant Requirements (ASR)
- Gathering (ASR) from Requirement document, Business goals & Stakeholders
- Capturing ASR in an Utility Tree for further refinement
- Architecture Design strategy and Attribute –Driven Design Method

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Define Architecturally Significant Requirements (ASR)

Requirements

- Architectures exist to build systems that satisfy requirements.
- But, to an architect, not all requirements are created equal.

- An *architecturally significant requirement* (ASR) is a requirement that will have a profound effect on the architecture.

- How do we find those?

ASRs and Requirements Documents

innovate achieve lead

- An obvious location to look for candidate ASRs is in the requirements documents or in user stories.
- Requirements should be in requirements documents!
- Unfortunately, this is not usually the case.
- Why?

Don't Get Your Hopes Up

- Many projects don't create or maintain the detailed, high-quality requirements documents.
- Standard requirements pay more attention to functionality than quality attributes.
- Most of what is in a requirements specification does not affect the architecture.
- No architect just sits and waits until the requirements are "finished" before starting work. The architect *must* begin while the requirements are still in flux.

When does the Architect Start?

Don’t Get Your Hopes Up

innovate achieve lead

- Quality attributes, when captured at all, are often captured poorly.
  - “The system shall be modular”
  - “The system shall exhibit high usability”
  - “The system shall meet users’ performance expectations”
- Much of what is useful to an architect is not in even the best requirements document.
  - ASRs often derive from business goals in the development organization itself
  - Developmental qualities (such as teaming) are also out of scope

Most ASRs are not obvious.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

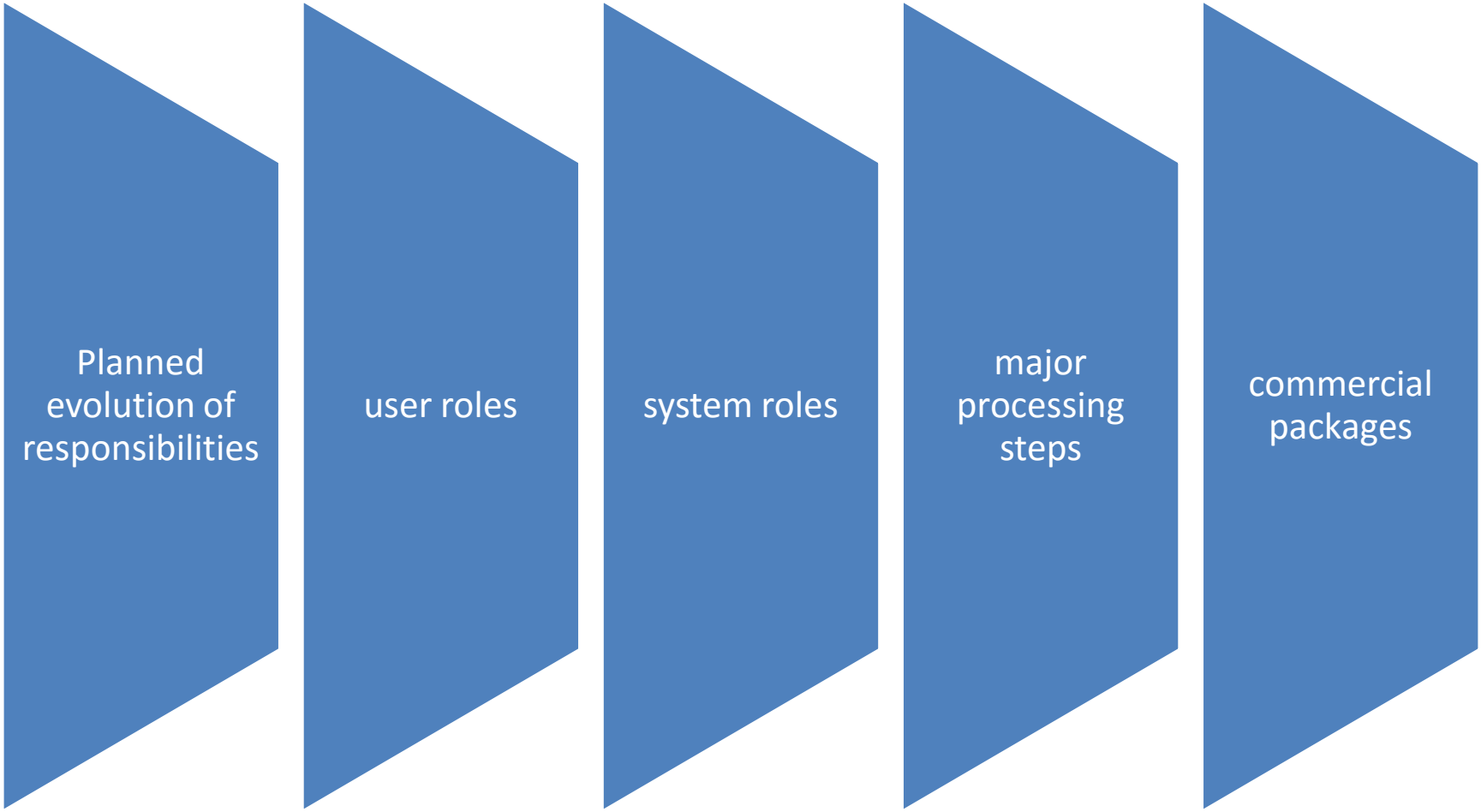
Gathering (ASR) from Requirement document, Business goals & Stakeholders

Sniffing Out ASRs

Design Decision Category	Look for Requirements Addressing . . .
Allocation of Responsibilities	Planned evolution of responsibilities, user roles, system modes, major processing steps, commercial packages
Coordination Model	Properties of the coordination (timeliness, currency, completeness, correctness, and consistency) Names of external elements, protocols, sensors or actuators (devices), middleware, network configurations (including their security properties) Evolution requirements on the list above
Data Model	Processing steps, information flows, major domain entities, access rights, persistence, evolution requirements
Management of Resources	Time, concurrency, memory footprint, scheduling, multiple users, multiple activities, devices, energy usage, soft resources (buffers, queues, etc.) Scalability requirements on the list above
Mapping among Architectural Elements	Plans for teaming, processors, families of processors, evolution of processors, network configurations
Binding Time Decisions	Extension of or flexibility of functionality, regional distinctions, language distinctions, portability, calibrations, configurations
Choice of Technology	Named technologies, changes to technologies (planned and unplanned)

Requirements that can affect:

ALLOCATION OF RESPONSIBILITY



Requirements that can affect: COORDINATION MODEL

innovate achieve lead

Properties of coordination

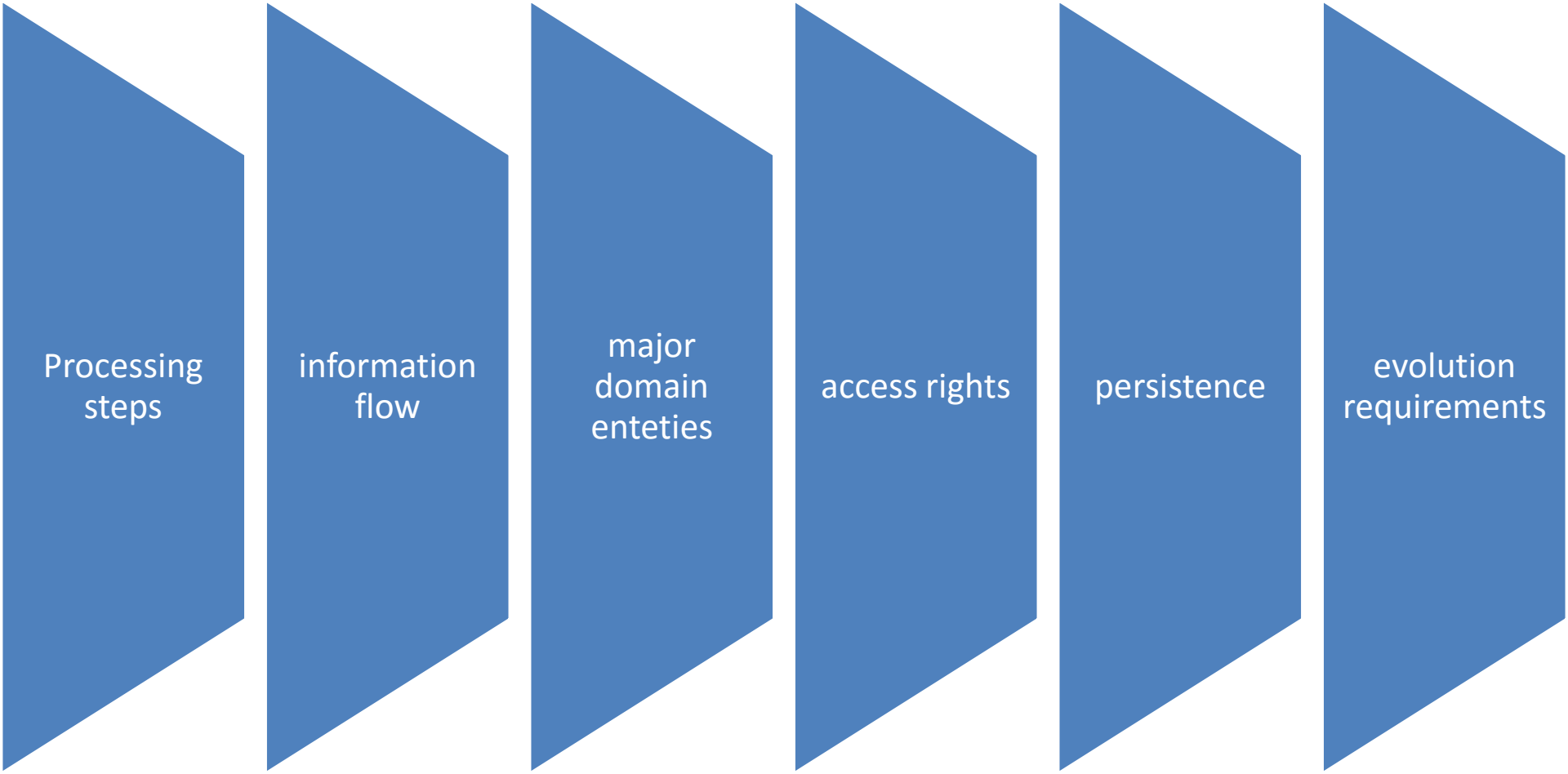
- timeliness
- currency
- completeness
- correctness
- Consistence

Names of

- external elements
- protocols
- sensors
- actuators (devices),
- middleware.
- network configurations (including their security properties)

Evolution requirements on the list at the left

Requirements that can affect: DATA MODEL



Requirements that can affect:

innovate achieve lead

MANAGEMENT OF RESOURCES

- Time
- concurrency
- memory footprint
- scheduling
- multiple users
- multiple activities
- devices
- energy usage
- soft resources (buffers, queues etc)
- Scalability requirements on the list above

Requirements that can affect:

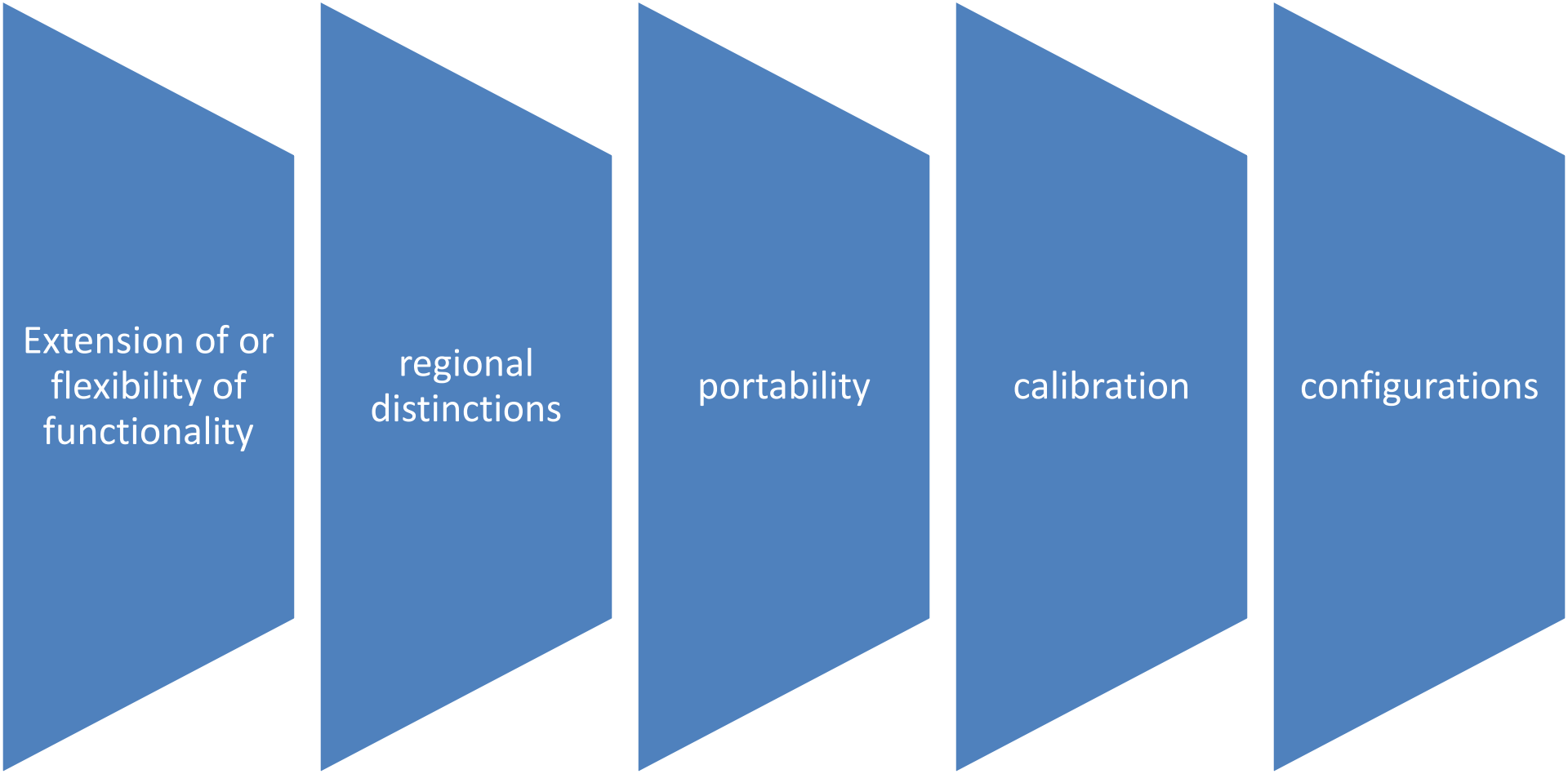
MAPPING AMONG ARCHITECTURAL ELEMENTS

Plans for

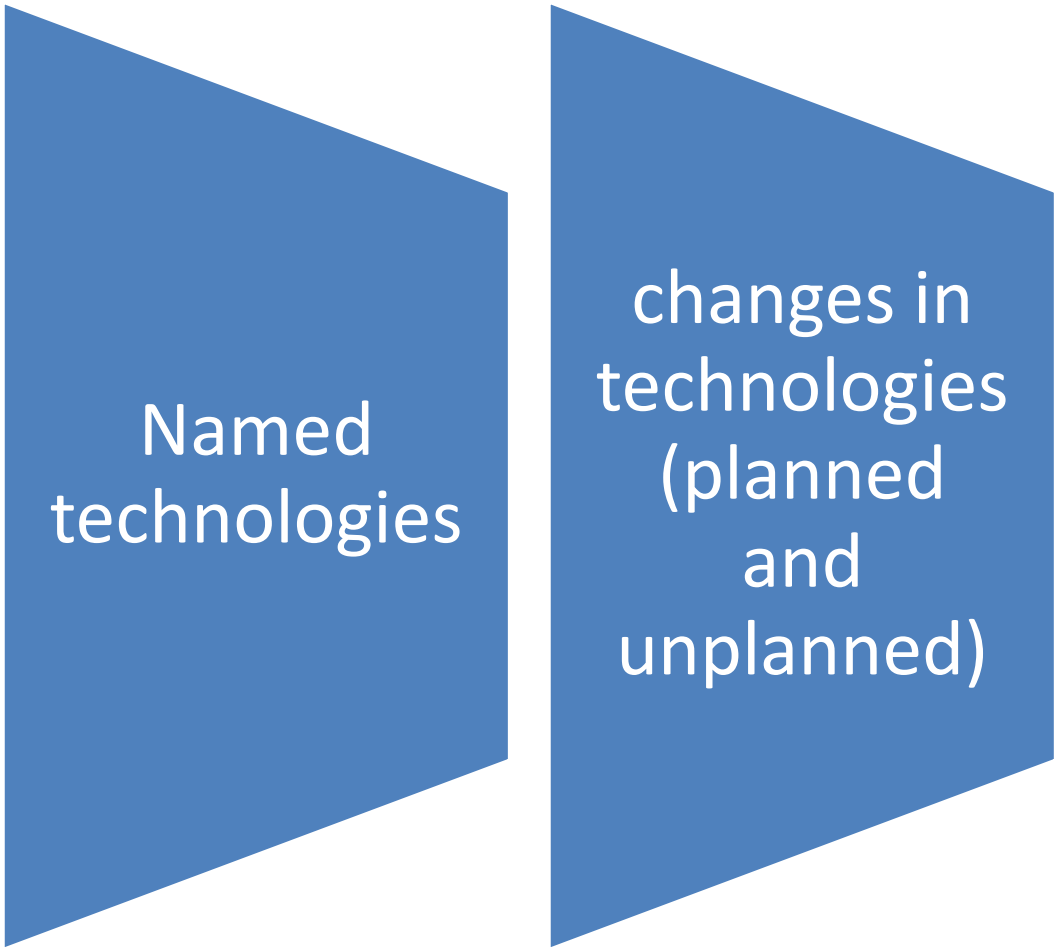
- teaming
- processors
- families of processors
- evolution of processors
- network configuration

Requirements that can affect:

BINDING TIME DECISIONS



Requirements that can affect:
CHOICE OF TECHNOLOGY



Gathering ASRs from Stakeholders

innovate achieve lead

- Say your project won't have the QAs nailed down by the time you need to start your design work.
- What do you do?
- Stakeholders often have *no idea* what QAs they want in a system
  - if you insist on quantitative QA requirements, you're likely to get numbers that are arbitrary.
  - at least some of those requirements will be very difficult to satisfy.
- Architects often have very good ideas about what QAs are reasonable to provide.
- Interviewing the relevant stakeholders is the surest way to learn what they know and need.

Gathering ASRs from Stakeholders

The results of stakeholder interviews should include

- a list of architectural drivers
- a set of QA scenarios that the stakeholders (as a group) prioritized.

This information can be used to:

- **refine** system and software **requirements**
- understand and clarify the system's **architectural drivers**
- provide rationale for why the architect subsequently made certain **design decisions**
- guide the development of **prototypes and simulations**
- influence the **order in which the architecture** is developed.

Quality Attribute Workshop

- The QAW is a facilitated, stakeholder-focused method to generate, prioritize, and refine **quality attribute scenarios** before the software architecture is completed.
- The QAW is focused on **system-level concerns** and specifically the role that software will play in the system.

QAW Steps

Step 1: QAW Presentation and Introductions.

- QAW facilitators describe the motivation for the QAW and explain each step of the method.

Step 2: Business/Mission Presentation.

- The stakeholder representing the business concerns behind the system presents the system's business context, broad functional requirements, constraints, and known quality attribute requirements.
- The quality attributes that will be refined in later steps will be derived largely from the business/mission needs presented in this step.

Step 3: Architectural Plan Presentation.

- The architect will present the system architectural plans as they stand.
- This lets stakeholders know the current architectural thinking, to the extent that it exists.

Step 4: Identification of Architectural Drivers.

- The facilitators will share their list of key architectural drivers that they assembled during Steps 2 and 3, and ask the stakeholders for clarifications, additions, deletions, and corrections.
- The idea is to reach a consensus on a distilled list of architectural drivers that includes overall requirements, business drivers, constraints, and quality attributes.

QAW Steps

Step 5: Scenario Brainstorming.

- Each stakeholder expresses a scenario representing his or her concerns with respect to the system.
- Facilitators ensure that each scenario has an explicit stimulus and response.
- The facilitators ensure that at least one representative scenario exists for each architectural driver listed in Step 4.

Step 6: Scenario Consolidation.

- Similar scenarios are consolidated where reasonable.
- Consolidation helps to prevent votes from being spread across several scenarios that are expressing the same concern.

Step 7: Scenario Prioritization.

- Prioritization of the scenarios is accomplished by allocating each stakeholder a number of votes equal to 30 percent of the total number of scenarios

Step 8: Scenario Refinement.

- The top scenarios are refined and elaborated.
- Facilitators help the stakeholders put the scenarios in the six-part scenario form of source-stimulus-artifact-environment-response-response measure.

ASRs from Business Goals

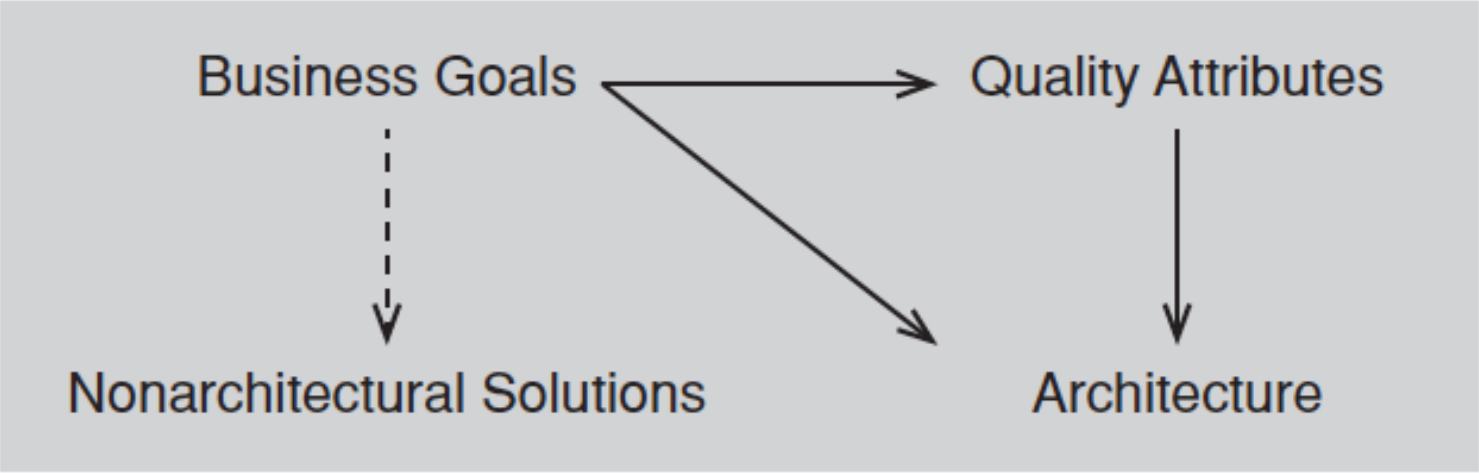


FIGURE 3.2 Some business goals may lead to quality attribute requirements (which lead to architectures), or lead directly to architectural decisions, or lead to nonarchitectural solutions.

A General Scenario for Business Goals

A Goal-subject (example stakeholder)

..... has the goal that ....

a goal-object (example Individual/System/Organisation)

..... Achieves ....

a goal (example improve business processes)

.... In the context of .....

the environment (example competitive)

... And will be considered to be satisfied if ...

Goal-measure (based on goal (categories)

... Is which is measured based on .

Value

... of output vis-a-vis resources you are willing to expend.

Examples of Business Goals For the following goal-objects

- Individual
- System
- Portfolio
- Organisation's Employees
- Organisation's Shareholders
- Organisation
- Nation
- Society

Goal-Object: Individual Business Goals

- Personalwealth
- power
- honor/face/reputation
- game and gambling spirit
- maintain or improve
- reputation(personal)
- family interests

Goal-Object: System

Business Goals->

Manage flexibility	distributed development	portability	Opensystems/standards	testability
product lines	integrability	interoperability	ease of installation and ease of repair	flexibility/configurabiiity
performance	reliability/availability	ease of use	security	safety
scalability/extendibility	functionality	system constraints	internationalization	reduce time to market

Goal-Object: Portfolio

Business Goals->

Reduce cost of development	costleadership,	differentiation,	reducecostof	retirement
smooth transition to follow-on systems	replace legacy systems	replace labor with automation	diversify operational sequence	eliminate intermediate stages
automate tracking of business events	collect/communicate/retrieve operational knowledge	improve decision making	coordinate across distance	align task and process
manage on basis of process measurements	operate effectively within the competitive environment, the technological environment, or the customerenvironment	Create something new	provide the best quality products and services possible	be the leading innovator in the industry

Goal-Object: Organisation's Employees Business Goals->

innovate achieve lead

Provide high rewards and benefits to employees,

create a pleasant and friendly work place,

have satisfied employees

fulfill responsibility toward employees

maintain jobs of work force on legacy systems

Goal-Object: Organisation's Shareholders

Business Goals->

Maximize dividends for the shareholders

Goal-Object: Organisation Business Goals->

Growth of the business	continuity of the business	maximize profits over the short run	maximize profits over the long run
survival of the organization	maximize the companys net assets and reserves	be a market leader	maximize the market share
expand or retain market share	enter new markets	maximize the companys rate of growth	keep tax payments to a minimum
increase sales growth	maintain or improve reputation	achieve business goals through financial objectives	run a stable organization

Goal-Object: Nation
Business Goals->

Patriotism

national pride

national security

national welfare

Goal-Object: Society Business Goals->

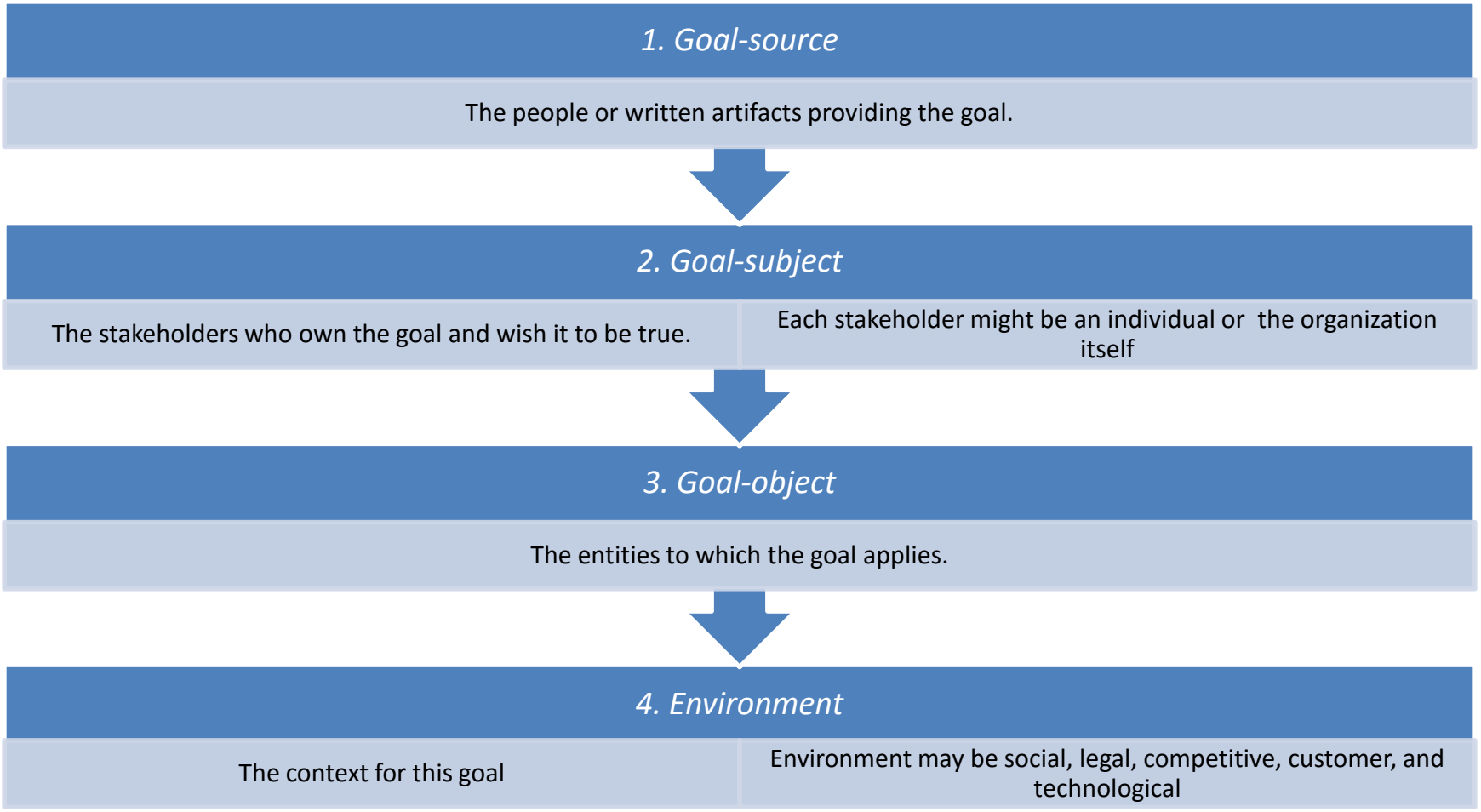
innovate achieve lead

Run an ethical organization	responsibility toward society	be a socially responsible company,
be of service to the community	operate effectively within social environment	operate effectively within legal environment

Categories of Business Goals, to Aid in Elicitation

BUSINESS GOAL	• GOAL-MEASURE
Contributing to the growth and continuity of the organisation	• Time that business remains viable
Meeting Financial objectives	• Financial Performance vs. objectives
Meeting personal objectives	• Promotion or raise achieved in period
Meeting responsibilities to employees	• Employee satisfaction; turnover rate
Meeting responsibilities to society	• Contribution to trade deficit
Meeting responsibilities to state	• Stock price, dividends
Meeting responsibilities to shareholders	• Market Share
Managing market position	• Time to carry out business
Improving business processes	• Quality measures of products
Managing the quality and reputation of the products	• Technology-related problems
Managing change in environmental factors	• Time window for achievement

Expressing Business Goals
Business goal scenario, 7 parts



Expressing Business Goals
Business goal scenario, 7 parts

innovate achieve lead

5. Goal

Any business goal articulated by the goal-source.

SEE LIST IN PREVIOUS SLIDE

6. Goal-measure

A testable measurement to determine how one would know if the goal has been achieved. The goal-measure should usually include a time component, stating the time by which the goal should be achieved.

7. Pedigree and value

The degree of confidence the person who stated the goal has in it	The goal's volatility and value.

PALM: A Method for Eliciting Business Goals

PALM is a seven-step method.

Nominally carried out over a day and a half in a workshop.

Attended by architect(s) and stakeholders who can speak to the relevant business goals.

PALM Steps

PALM overview presentation

Overview of PALM, the problem it solves, its steps, and its expected outcomes.

Business drivers presentation.

Briefing of business drivers by project management	What are the goals of the customer organization for this system?	What are the goals of the development organization?

Architecture drivers presentation

Briefing by the architect on the driving business and quality attribute requirements: the ASRs.

Business goals elicitation

Business goals are elaborated and expressed as scenarios.	Consolidate almost-alike business goals to eliminate duplication.	Participants prioritize the resulting set to identify the most important goals.

PALM Steps

Identification of potential quality attributes from business goals.

For each important business goal scenario, participants describe a quality attribute that (if architected into the system) would help achieve it. If the QA is not already a requirement, this is recorded as a finding.

Assignment of pedigree to existing quality attribute drivers.

For each architectural driver identify which business goals it is there to support.	If none, that's recorded as a finding.	Otherwise, we establish its pedigree by asking for the source of the quantitative part.

Exercise conclusion

Review of results, next steps, and participant feedback.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Capturing ASR in an Utility Tree for further refinement

Capturing ASRs in a Utility Tree

An ASR must have the following characteristics:

A profound impact on the architecture

- Including this requirement will very likely result in a different architecture than if it were not included.

A high business or mission value

- If the architecture is going to satisfy this requirement it must be of high value to important stakeholders.

Utility Tree

A way to record ASRs all in one place.

Establishes priority of each ASR in terms of

- Impact on architecture
- Business or mission value

ASRs are captured as scenarios.

Root of tree is placeholder node called “Utility”.

Second level of tree contains broad QA categories.

Third level of tree refines those categories.

Utility Tree Example (excerpt)

utility

Quality Attribute	Attribute Refinement	ASR
Performance	Transaction response time	A user updates a patient's account in response to a change-of-address notification while the system is under peak load, and the transaction completes in less than 0.75 second. (H,M)
		A user updates a patient's account in response to a change-of-address notification while the system is under double the peak load, and the transaction completes in less than 4 seconds. (L,M)
	Throughput	At peak load, the system is able to complete 150 normalized transactions per second. (M,M)
Usability	Proficiency training	A new hire with two or more years' experience in the business becomes proficient in Nightingale's core functions in less than 1 week. (M,L)
		A user in a particular context asks for help, and the system provides help for that context, within 3 seconds. (H,M)
	Normal operations	A hospital payment officer initiates a payment plan for a patient while interacting with that patient and completes the process without the system introducing delays. (M,M)
Configurability	User-defined changes	A hospital increases the fee for a particular service. The configuration team makes the change in 1 working day; no source code needs to change. (H,L)
Maintainability	Routine changes	A maintainer encounters search- and response-time deficiencies, fixes the bug, and distributes the bug fix with no more than 3 person-days of effort. (H,M)
		A reporting requirement requires a change to the report-generating metadata. Change is made in 4 person-hours of effort. (M,L)
	Upgrades to commercial components	The database vendor releases a new version that must be

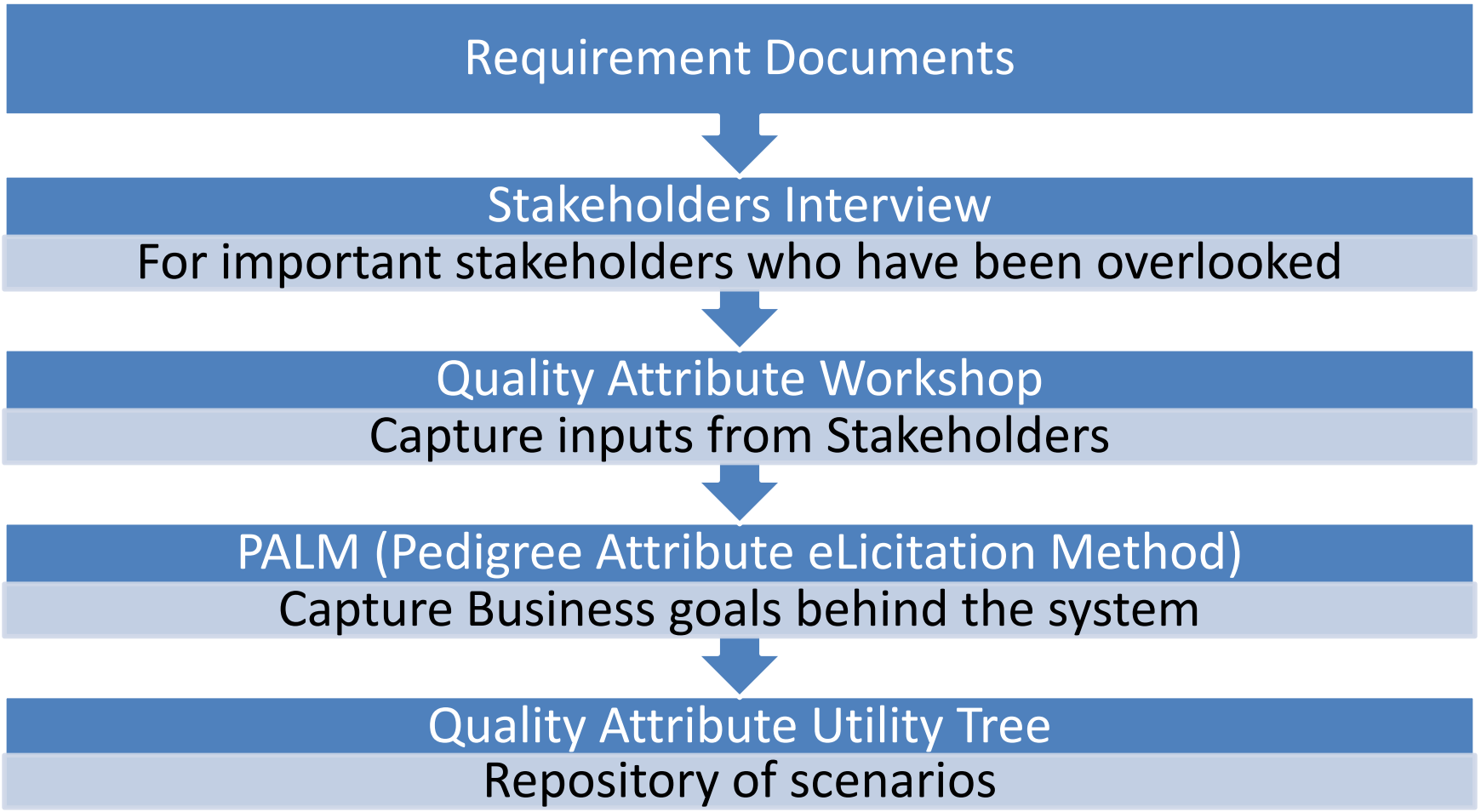
…

Key: Utility
H=high
M=medium
L=low

Utility Tree: Next Steps

- A QA or QA refinement without any ASR is not necessarily an error or omission
  - Attention should be paid to searching for unrecorded ASRs in that area.
- ASRs that rate a (H,H) rating are the ones that deserve the most attention
  - A very large number of these might be a cause for concern: Is the system is achievable?
- Stakeholders can review the utility tree to make sure their concerns are addressed.

Tying the Methods Together



Summary

- Architectures are driven by architecturally significant requirements: requirements that will have profound effects on the architecture.
  - Architecturally significant requirements may be captured from requirements documents, by interviewing stakeholders, or by conducting a Quality Attribute Workshop.
- Be mindful of the business goals of the organization.
  - Business goals can be expressed in a common, structured form and represented as scenarios.
  - Business goals may be elicited and documented using a structured facilitation method called PALM.
- A useful representation of quality attribute requirements is in a utility tree.
  - The utility tree helps to capture these requirements in a structured form.
  - Scenarios are prioritized.
  - This prioritized set defines your “marching orders” as an architect.

Thank you……..

Credits

- **Chapter Reference from Text T1:** 16, 17, 18
- Slides have been adapted from Authors Slides Software Architecture in Practice – Third Ed.
  - Len Bass
  - Paul Clements
  - Rick Kazman

Reference Chapter 17
Software Architecture in Practice
Third Edition
Len Bass
Paul Clements
Rick Kazman

Software Architecture

Designing & Documenting the Architecture #1

Designing an Architecture

Harvinder S Jabbal
Module RL6.2

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Designing & Documenting the Architecture #1

- Define Architecturally Significant Requirements (ASR)
- Gathering (ASR) from Requirement document, Business goals & Stakeholders
- Capturing ASR in an Utility Tree for further refinement
- Architecture Design strategy and Attribute –Driven Design Method

Architecture Design strategy and Attribute –Driven Design Method

Chapter Outline

Design Strategy

The Attribute-Driven Design Method

The Steps of ADD

Summary

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Design Strategy

Design Strategy-

innovate achieve lead

Ideas key to architectural design methods

Idea 1

- Decomposition

Idea 2

- Designing to Architecturally Significant Requirements

Idea 3

- Generate and Test

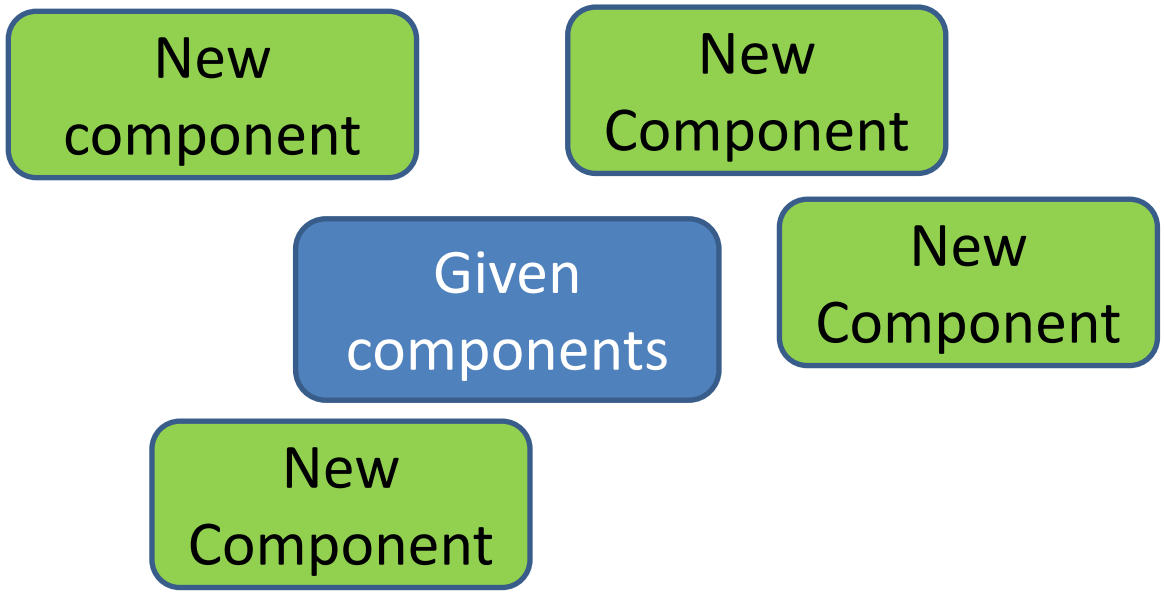
Idea 1: Decomposition

innovate achieve lead

- Architecture determines quality attributes
- Important quality attributes are characteristics of the *whole* system.
- Design therefore begins with the whole system
  - The whole system is decomposed into parts
  - Each part may inherit all of part of the quality attribute requirements from the whole

Design Doesn't Mean Green Field

- If you are given components to be used in the final design, then the decomposition must accommodate those components.



Idea 2: Designing to Architecturally Significant Requirements

- Remember architecturally significant requirements (ASRs)?

- These are the requirements that you must satisfy with the design

  - There are a small number of these
  - They are the most important (by definition)

How Many ASRs Simultaneously?

- If you are inexperienced in design then design for the ASRs one at a time beginning with the most important.

- As you gain experience, you will be able to design for multiple ASRs simultaneously.

What About Other Quality Requirements?

If your design does not satisfy a particular non ASR quality requirement then either

- **IMPROVE THE DESIGN**
  - **Adjust your design** so that the ASRs are still satisfied and so is this additional requirement or
- **WEAKEN THE REQUIREMENT**
  - **Weaken the additional requirement** so that it can be satisfied either by the current design or by a modification of the current design or
- **CHANGE PRIORITIES**
  - **Change the priorities** so that the one not satisfied becomes one of the ASRs or
- **DECLARE NON-SATISFIABLE**
  - **Declare the additional requirement non-satisfiable** in conjunction with the ASRs.

Idea 3: Generate and Test

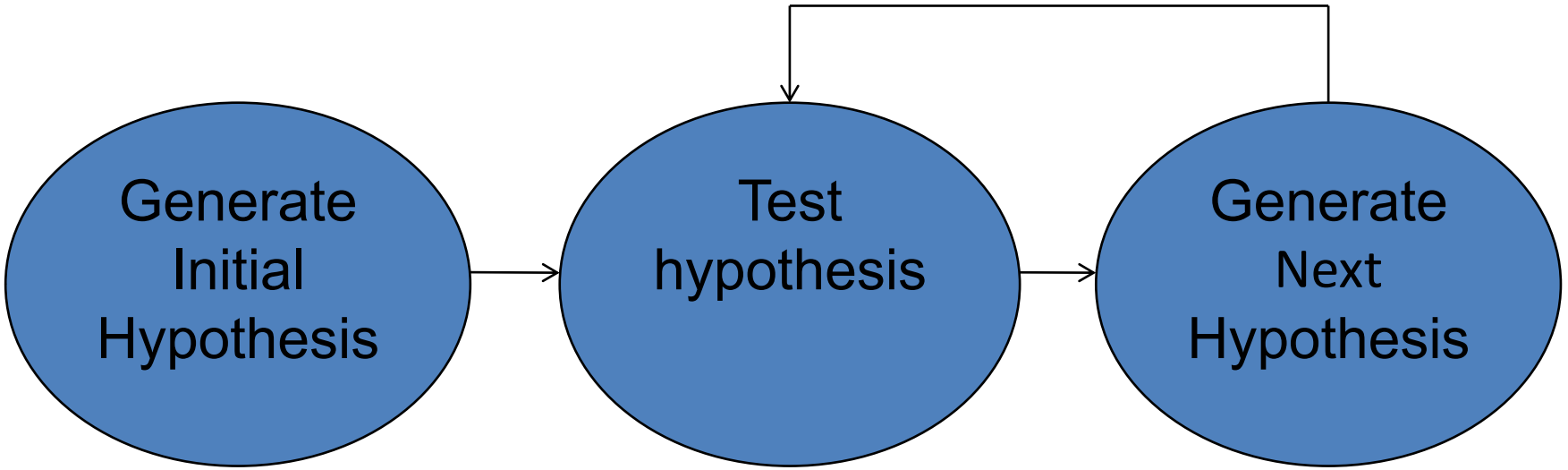
View the current design as a hypothesis.

- Assume requirement will be satisfied

Ask whether the current design satisfies the requirements

- Test if requirement is satisfied.

If not, then generate a new hypothesis

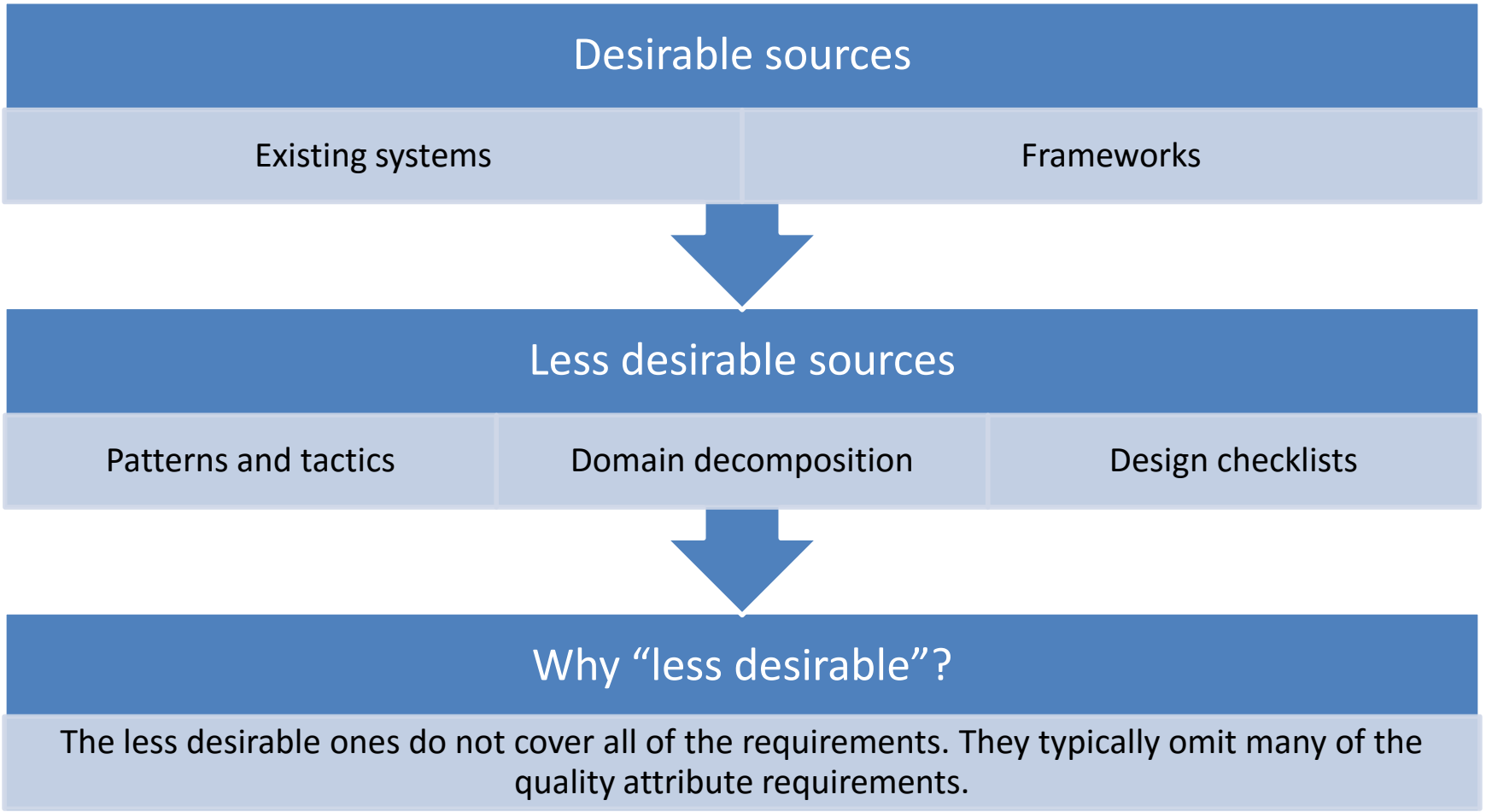


Raises the Following Questions

- Where does initial hypothesis come from?
- How do I test a hypothesis?
- When am I done?
- How do I generate the next hypothesis?

- You already know most of the answers; it is just a matter of organizing your knowledge.

Initial Hypothesis come from "collateral" that are available to the project



How Do I Test a Hypothesis?

Use the analysis techniques already covered

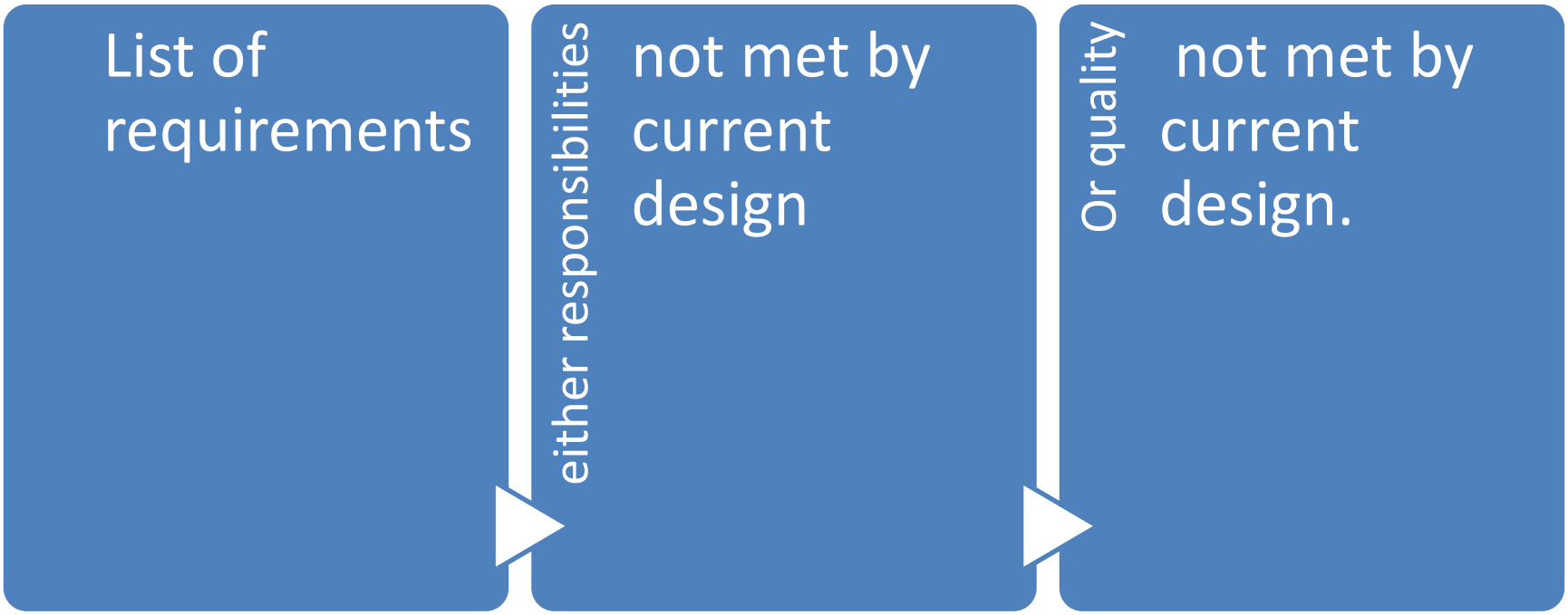
Design checklists from quality attribute discussion.

- Example- coordination model to support capturing activity to support testabilitycollect

Architecturally significant requirements

- Does the hypothesis provide a solution for the ASR.

What is the output of the tests?



How Do I Generate the Next Hypothesis?

Add missing responsibilities.

Be mindful of the side effects of a tactic.

Use tactics to adjust quality attribute behavior of hypothesis.

The choice of tactics will depend on which quality attribute requirements are not met.

When Am I Done?

All ASRs are satisfied and/or…

You run out of budget for design activity

- In this case, use the best hypothesis so far.
- Begin implementation
- Continue with the design effort although it will now be constrained by implementation choices.

The Attribute-Driven Design Method

The Attribute-Driven Design Method

Packaging of many of the techniques already discussed.

An iterative method. At each iteration you

- Choose a part of the system to design.
- Marshal all the architecturally significant requirements for that part.
- Generate and test a design for that part.

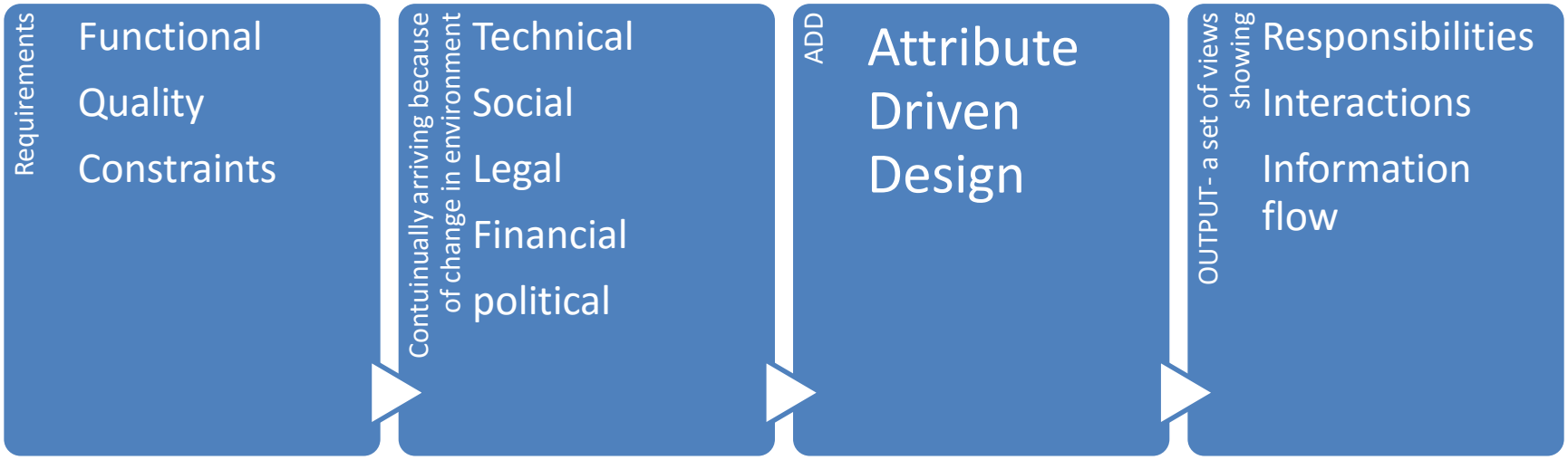
ADD does not result in a complete designbut the main design approach

- Set of containers with responsibilities
- Interactions and information flow among containers

Does not produce an API or signature for containers.

- Gives a “workable” architecture early and quickly

ADD Inputs and Outputs



The Steps of ADD

The Steps of ADD

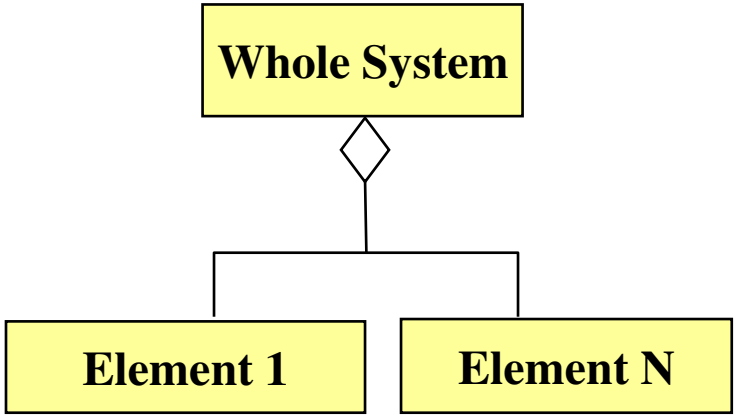
1. Choose an element of the system to design.
2. Identify the ASRs for the chosen element.
3. Generate a design solution for the chosen element.
4. Inventory remaining requirements and select the input for the next iteration.
5. Repeat steps 1–4 until all the ASRs have been satisfied.

Step 1:

- Choose an Element of the System to Design

- For green field designs, the element chosen is usually the whole system.
- For legacy designs, the element is the portion to be added.
- After the first iteration:

- *Initial iteration will be broad with less depth.*
- *Gradually get fine-grained.*



Which Element Comes Next?

Two basic refinement strategies:	Which one to choose?	If using new technology =>	If a team needs work =>	Otherwise =>
Breadth first	It depends ☺	depth first: explore the implications of using that technology.	depth first: generate requirements for that team.	breadth first.
Depth first				

Step 2

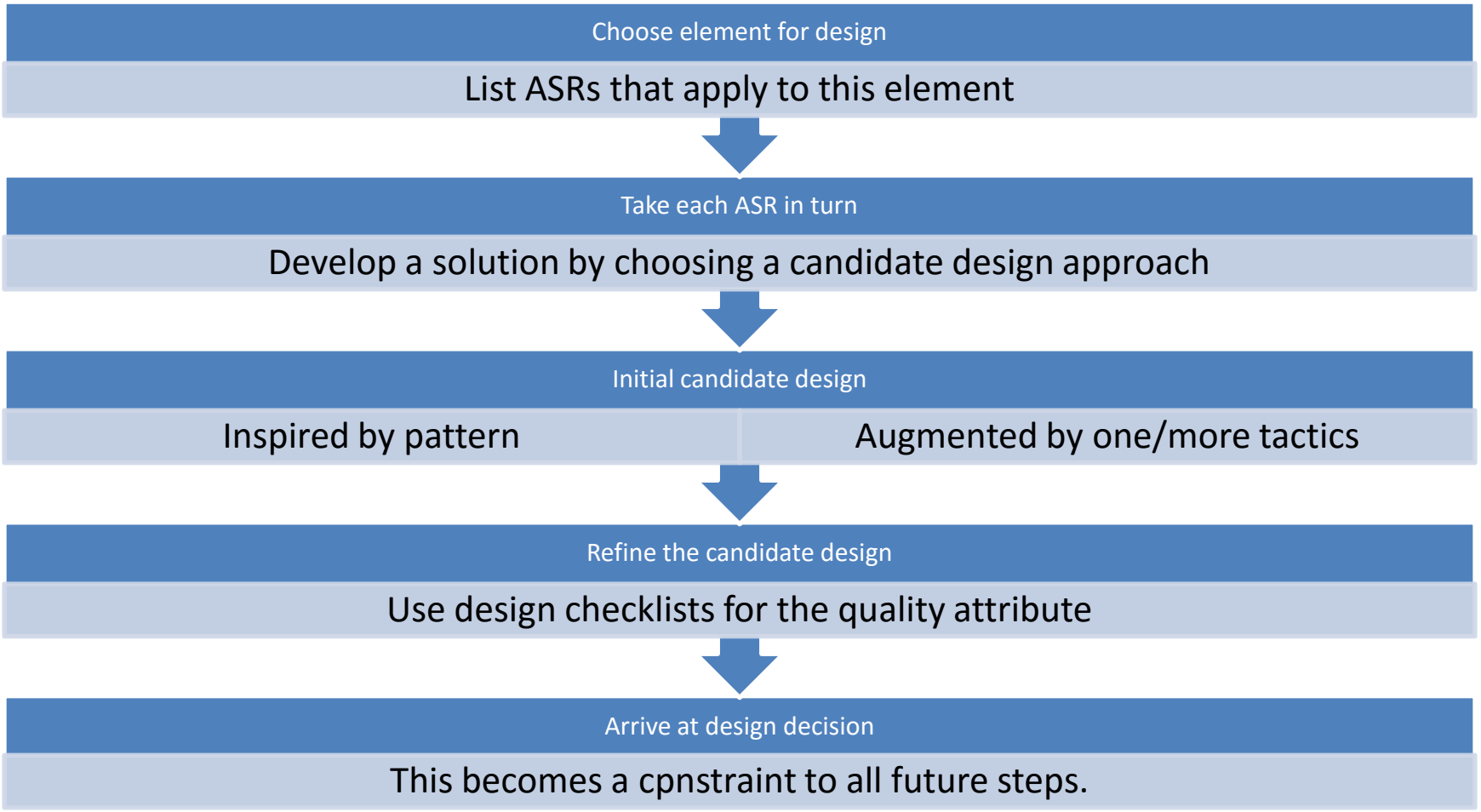
• Identify the ASRs for the Chosen Element

innovate achieve lead

- If the chosen element is the whole system, then use a utility tree (as described earlier).
- If the chosen element is further down the decomposition tree, then generate a utility tree from the requirements for that element.

Step 3

- Generate a Design Solution for the Chosen Element



Step 4 • Select the Input for the Next Iteration

innovate achieve lead

Ensure that requirement has been satisfied,

- if not:
- BACKTRACK.

ASR not yet satisfied should relate to

- Quality Attribute Requirement/
- Functional responsibility /
- constraint of the parent element

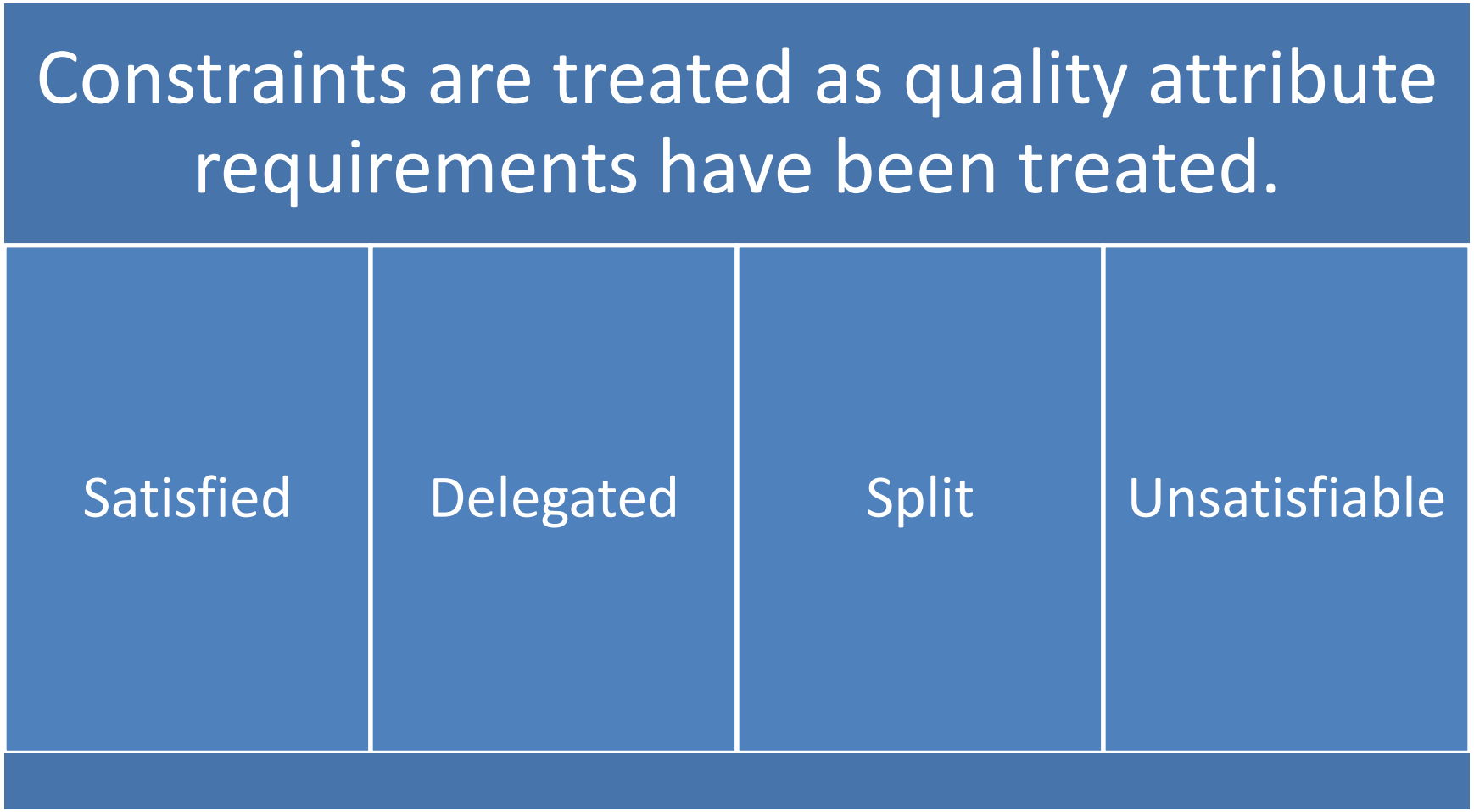
then add responsibilities to satisfy the requirement.

- Add them to container with similar requirements
- If no such container may need to create new one or add to container with dissimilar responsibilities (coherence)
- If container has too many requirements for a team, split it into two portions. Try to achieve loose coupling when splitting.

For each Quality Attribute Requirements, responsibility and constraint.

If the quality attribute requirement has been satisfied,	• it does not need to be further considered.
If the quality attribute requirement has not been satisfied then either	• Delegate it to one of the child elements • Split it among the child elements
If the quality attribute cannot be satisfied,	• see if it can be weakened. • If it cannot be satisfied or weakened then it cannot be met.

Constraints



Step 5

- Repeat Steps 1–4 Until All ASRs are Satisfied

innovate achieve lead

At end of step 3, each child element will have associated with it a set of:

functional requirements (responsibilities),	quality attribute requirements, and	constraints.

This sets up the child element for the next iteration of the method.

ADD PROCESS CAN BE TERMINATED IF

All requirements satisfied	High degree of trust between architect and implementation team.	Contractual arrangement satisfied.	Project's design budget exhausted.


innovate
achieve
lead
BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Summary

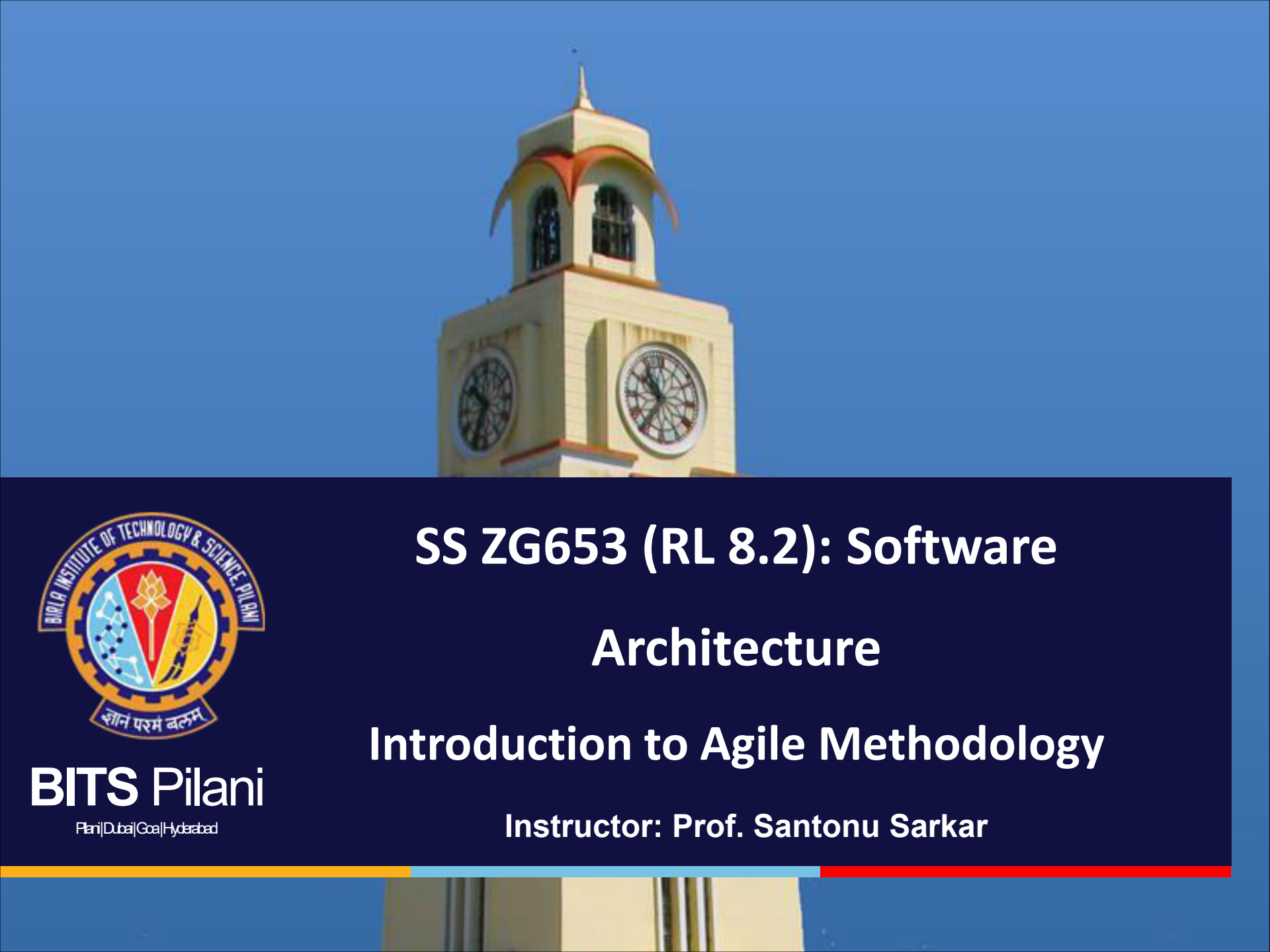
Summary

Designing the architecture is a matter of		
Determining the ASRs	Performing generate and test one an element to decompose it to satisfy the ASRs	Iterating until requirements are satisfied.

Thank you……..

Credits

- **Chapter Reference from Text T1:** 16, 17, 18
- Slides have been adapted from Authors Slides Software Architecture in Practice – Third Ed.
    - Len Bass
    - Paul Clements
    - Rick Kazman



SS ZG653 (RL 8.2): Software Architecture

Introduction to Agile Methodology

Instructor: Prof. Santonu Sarkar

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

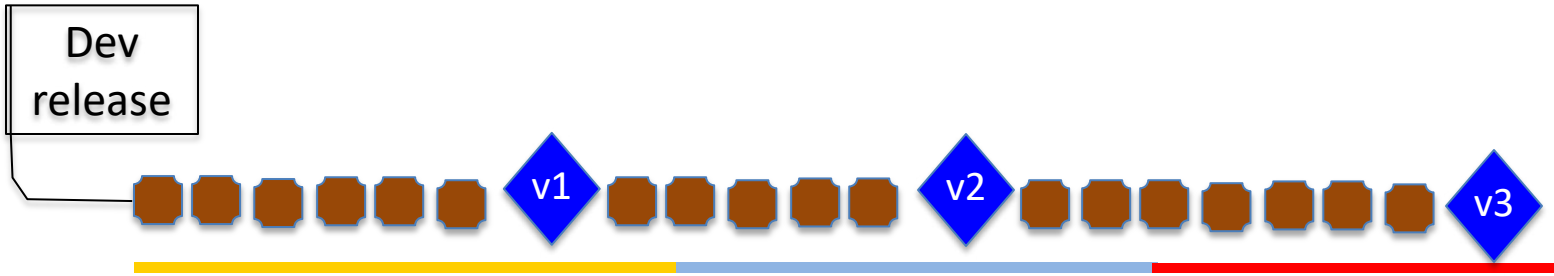
What is Agile Methodology

- Collaborative
  - Forms a pair for any development task to avoid error
  - Involves stakeholders from the beginning
- Interactive and feedback oriented
  - Teams interact frequently
  - Quick, and repeated integration of the product
  - Constant feedback from the stakeholder (customer)
- Iterative
  - Requirement, design, coding, testing goes through many iterations each having short duration
  - Refactoring is a part of the development process
- Test driven
  - Before building the component, define the test cases
  - Continuously test

– Scott Amber, Kent Beck

A Brief Overview

- There are 7 disciplines performed in an iterative manner
- At each iteration the software (or a part of the software) is built, tested
- Software architecture is more “agile” and it is never frozen
- UML based modeling is performed



Discipline Overview

- Model
  - Business Model
  - Analysis and Design (Architecture)
- Implementation
- Test
- Deployment
- Config Management
- Project Management
- Environment

Steps of Architecture Modeling in Agile

- Feature driven
  - Prioritize. Elaborate critical features more
- Model the architecture (UML)
- Suggested viewpoints for Agile
  - Usage scenarios
  - User interface and system interface
  - Network, deployment, hardware
  - Data storage, and transmission
  - Code distribution
- Suggested quality concerns
  - Reuse
  - Reliability, availability, serviceability, performance
  - Security
  - Internationalization, regulation, maintaince

Class Responsibilities and Collaborators (CRC) Card

What

- It is a physical (electronic) card
- One card for one class
  - Indicates the responsibilities of a class
- Collaboration
  - Sometimes a class can fulfill all its assigned responsibilities on its own
  - But sometimes, it needs to collaborate with other classes in order to fulfill its own responsibilities

Why?

- A good technique to identify a class and its responsibility during functional architecture design
- Highly collaborative and interactive process for a team of designers
- The team can do it fast

CRC Card

Class Name	Collaborators
Responsibilities assigned to this Class	If this class can not fulfill any of its assigned task on its own then which other classes it has to collaborate

CRC Card Example 1

```
class Box
{
private double length;
private double width;
private double height;
Box(double l, double w, double h)     {     length = l; width = w; height = h;     }
public double getLength()             {     return length;                         }
public double getWidth()              {     return width;                          }
public double getHeight()             {     return height;                         }
public double area()                  {     return 2*(length*width + width * height + height * length);     }
public double volume()                {     return length * width * height;                                  }
} // End of class BOX
```

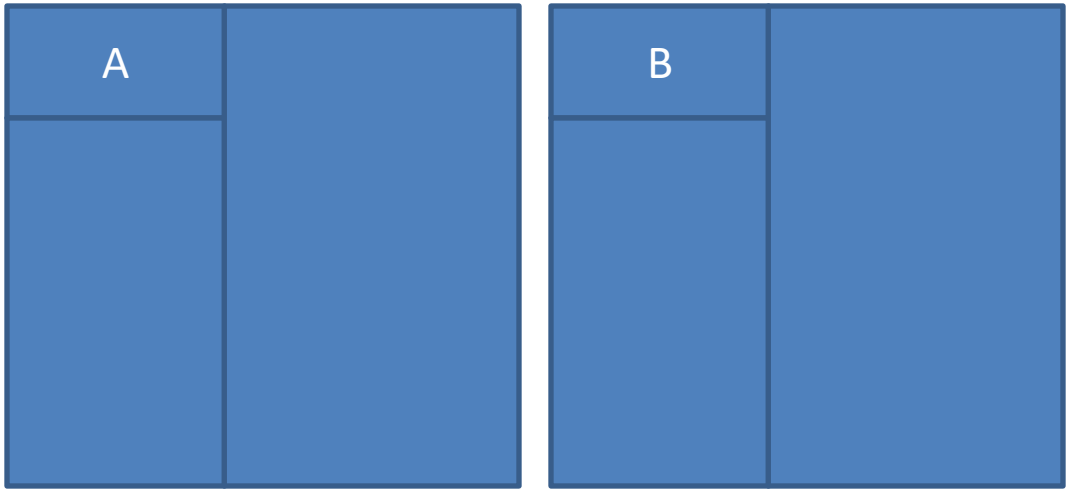
Write CRC Cards for Box Class

Box	Collaborators
Responsibilities 1. Getting length 2. Getting width 3. Getting height 4. Computing area and volume	<<None>>

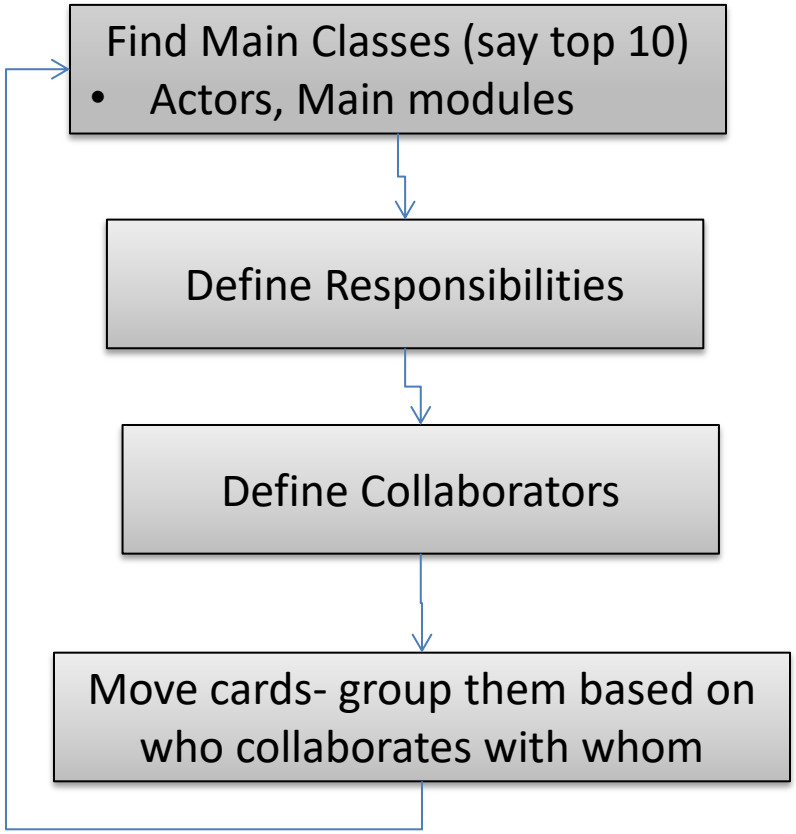
CRC Card Example 2

```
class B
{
  public void doB()
  {
    System.out.println("Hello");
  }
}
class A
{
  public void doS()
  {
      B b1 = new B();
      b1.doB();
  }
} //End of class Test
```

Write CRC Cards for Classes A & B



How do you create CRC Model?



Thank You