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

innovate achieve lead

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-

Ideas key to architectural design methods

Idea 1

- Decomposition

Idea 2

- Designing to Architecturally Significant Requirements

Idea 3

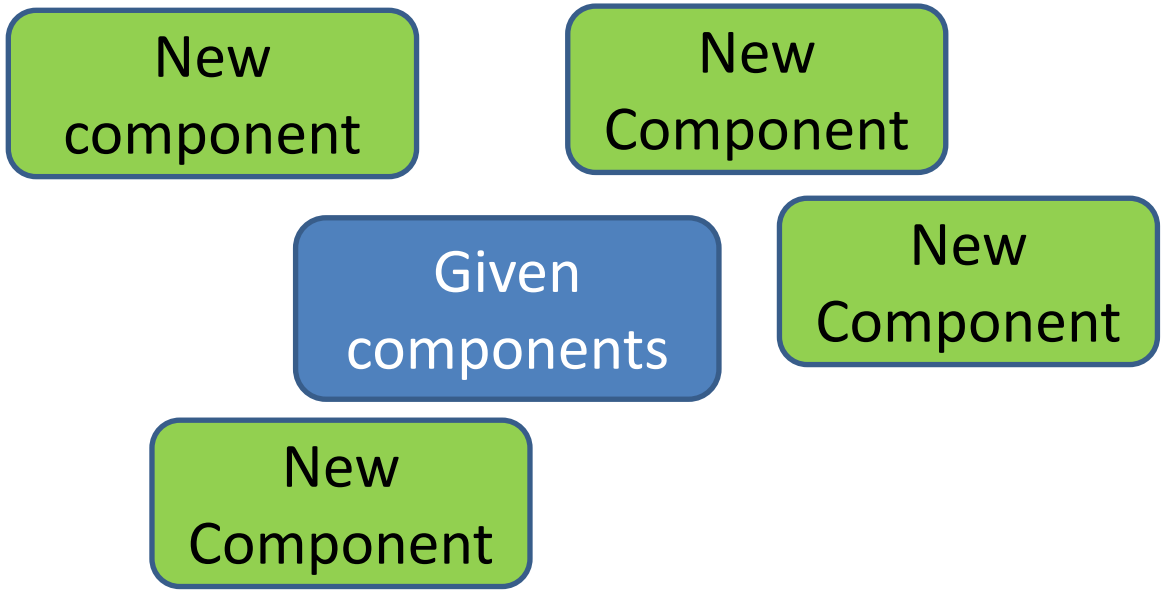
- Generate and Test

Idea 1: Decomposition

- 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

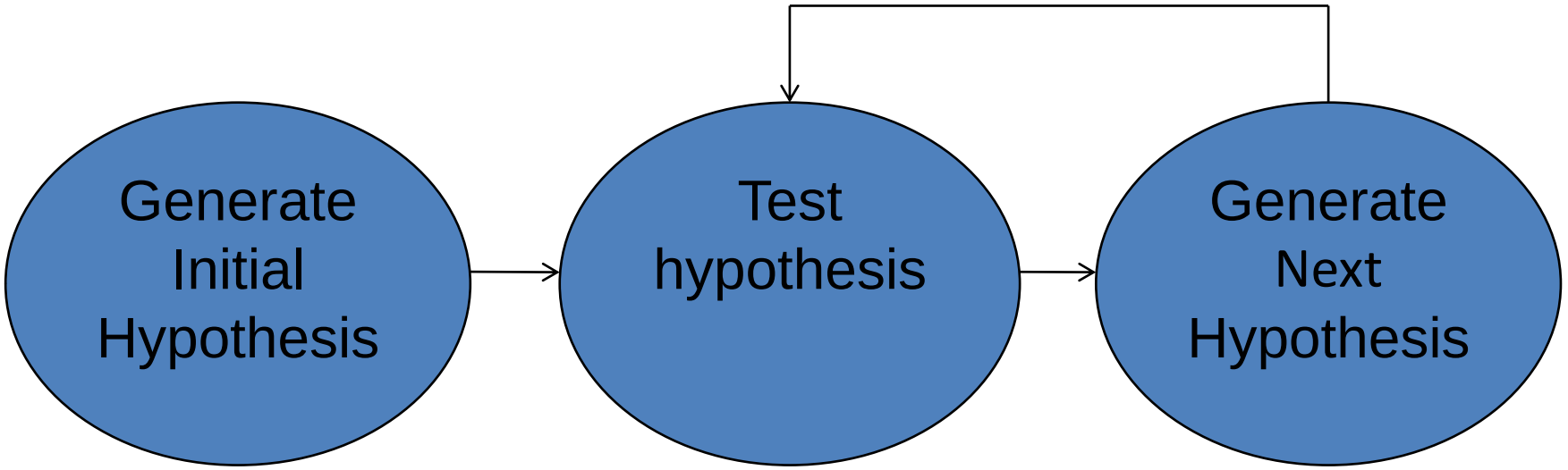
View the current design as a hypothesis.

- 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

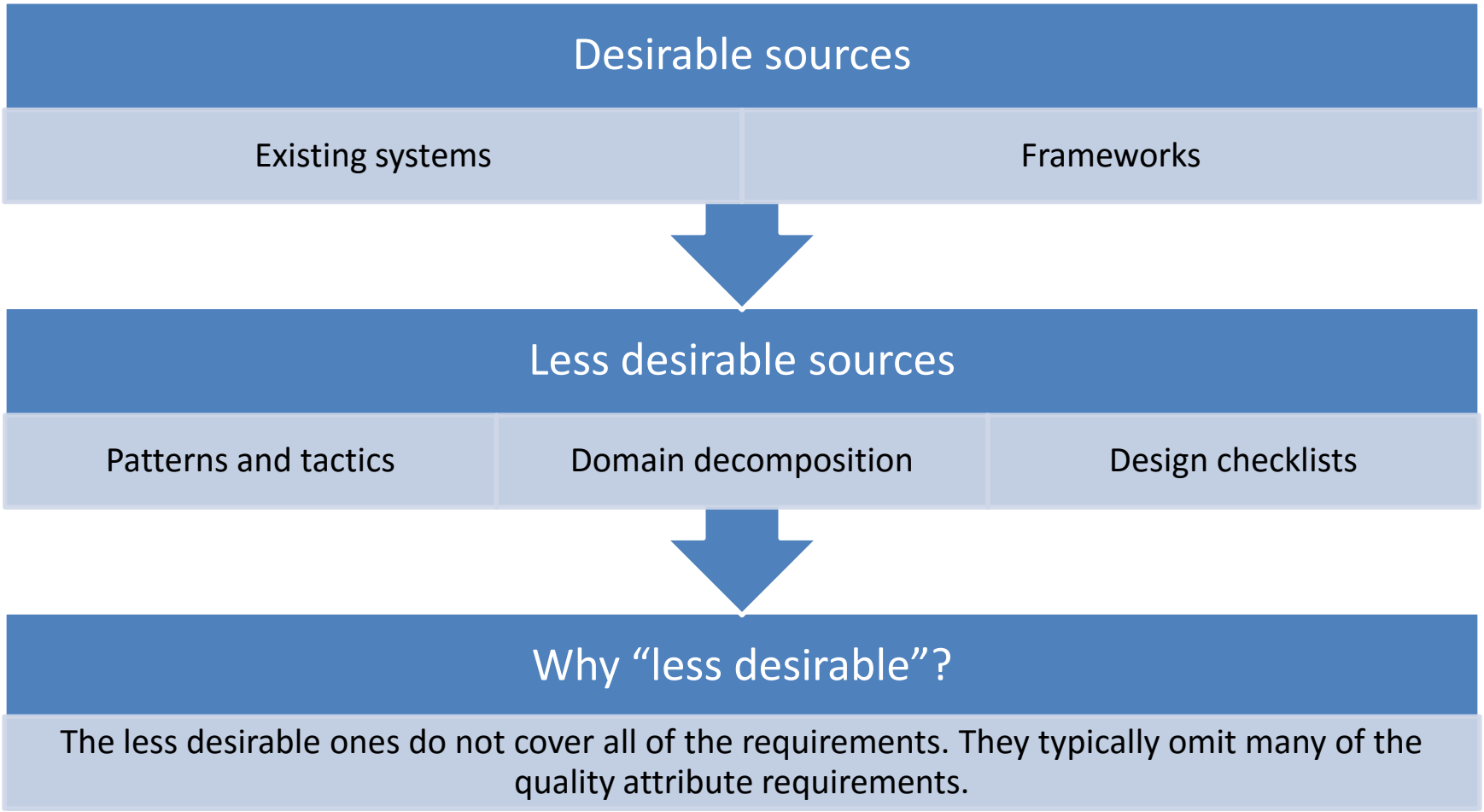


Raises the Following Questions

- 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

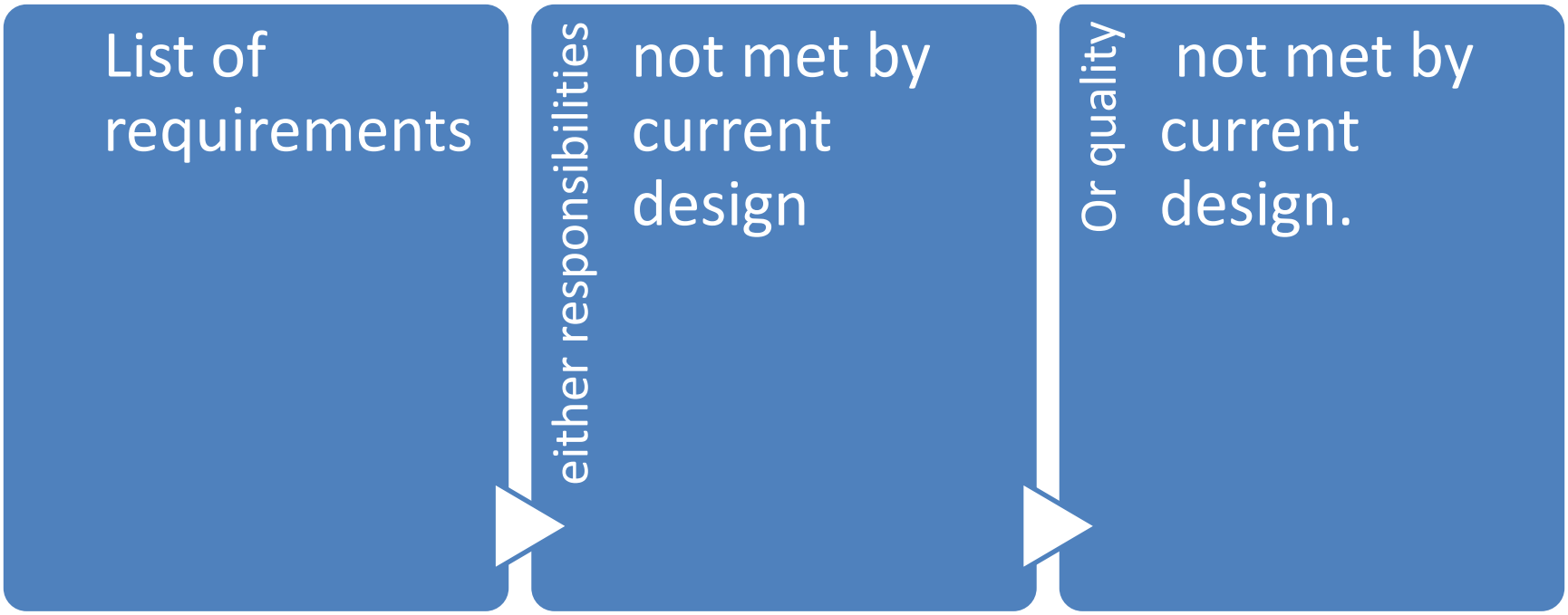
Design checklists from quality attribute discussion.

- 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.

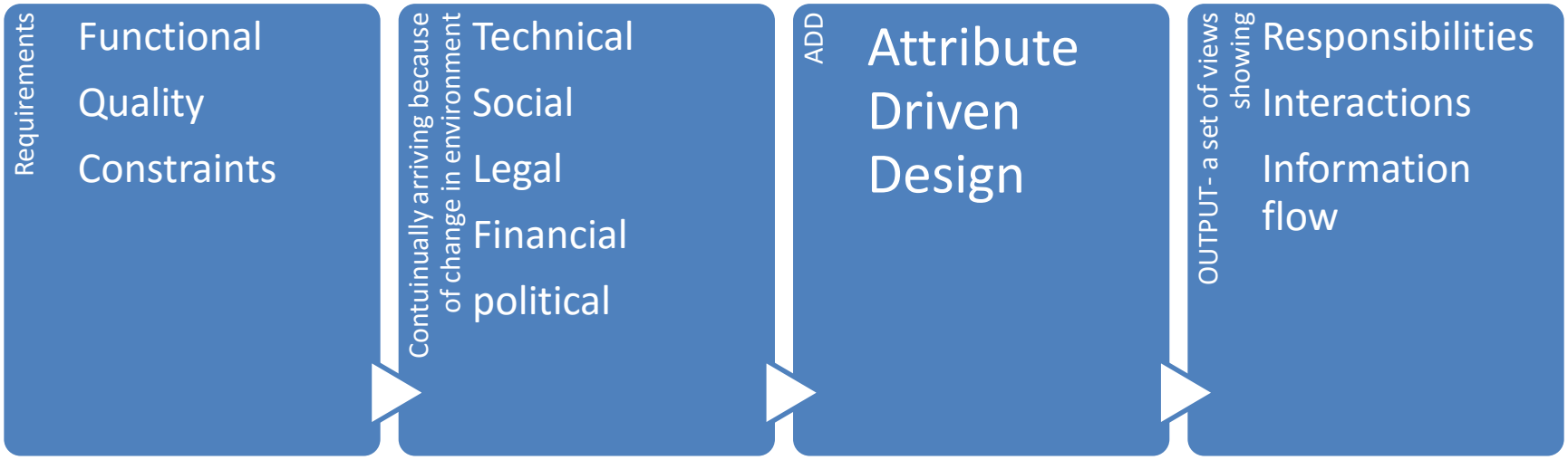
ADD does not result in a complete designbut the main design approach

- 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



innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

The Steps of ADD

The Steps of ADD

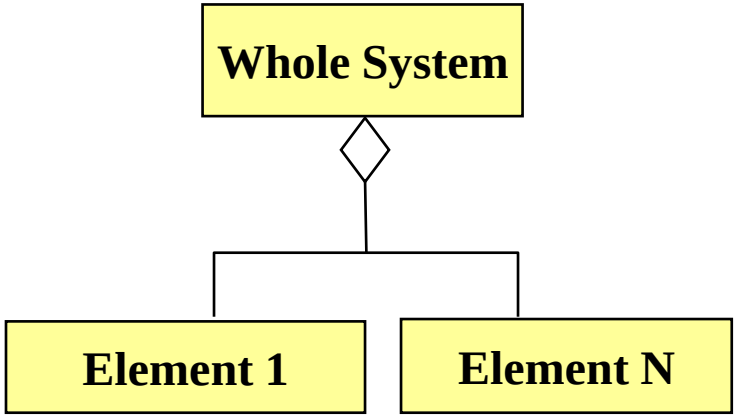
1. Choose an element of the system to design.
2. Identify the ASRs for the chosen element.
3. Generate a design solution for the chosen element.
4. Inventory remaining requirements and select the input for the next iteration.
5. Repeat steps 1–4 until all the ASRs have been satisfied.

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 Depth first	It depends ☺	depth first: explore the implications of using that technology.	depth first: generate requirements for that team.	breadth first.

Step 2

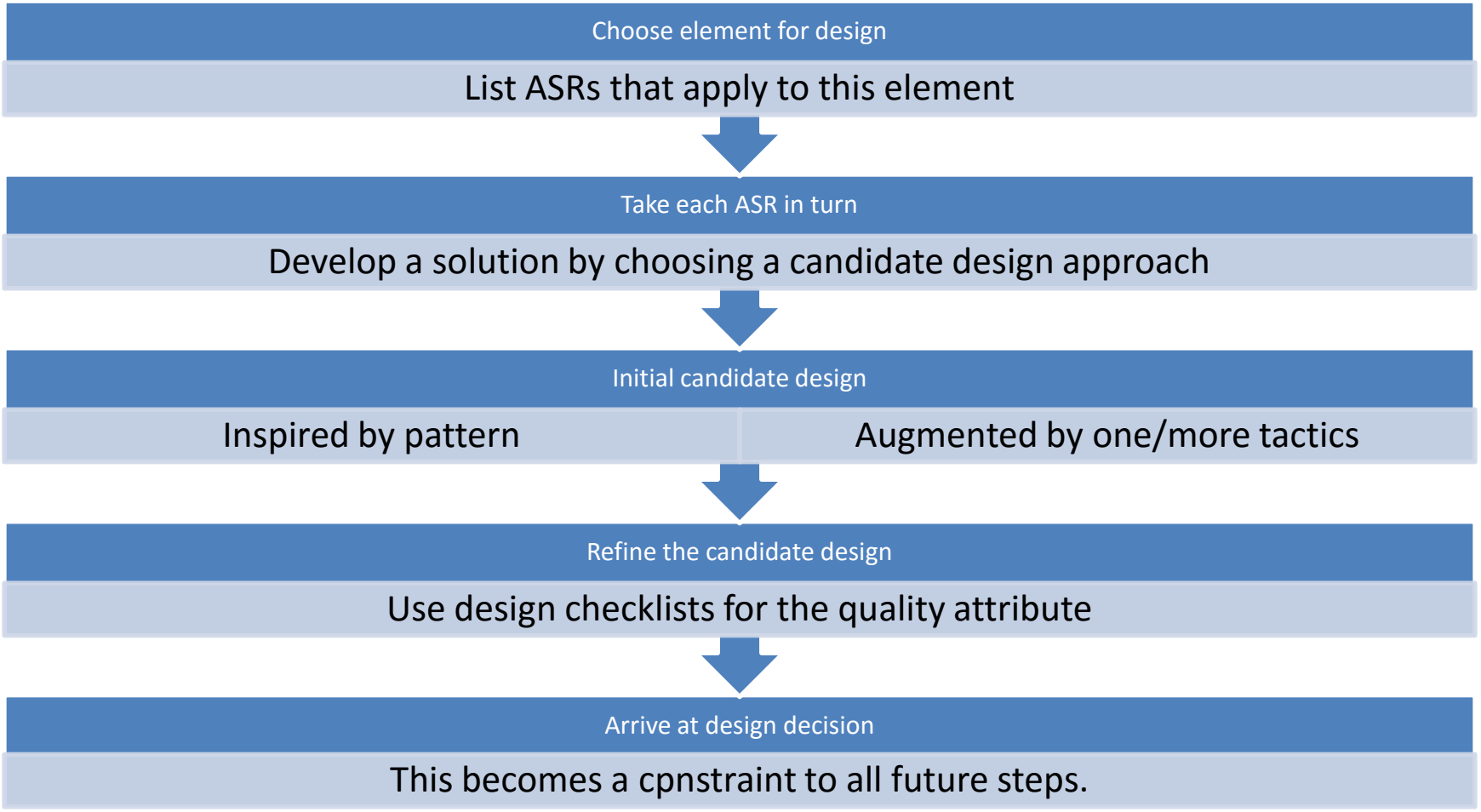
• Identify the ASRs for the Chosen Element

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

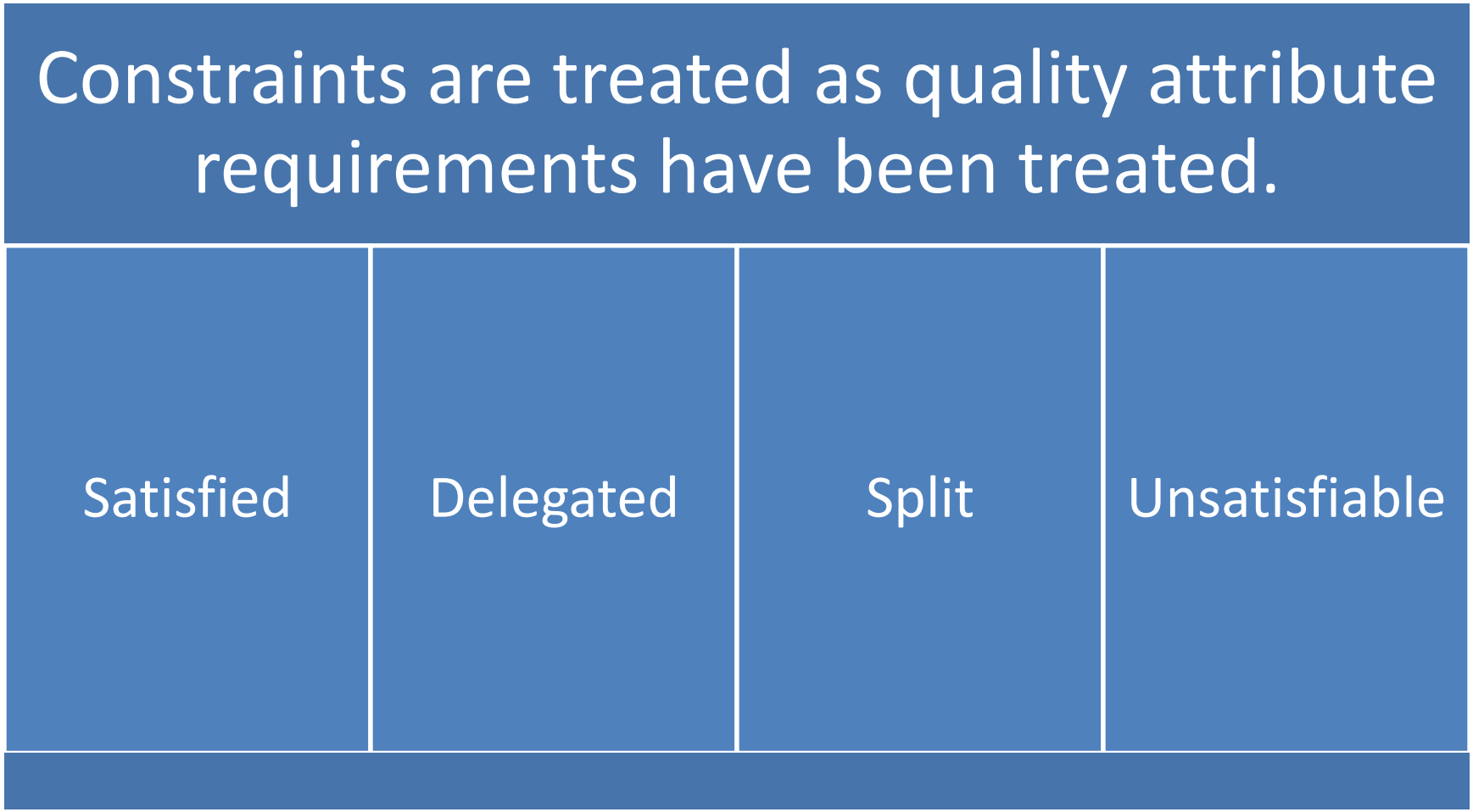
then add responsibilities to satisfy the requirement.

- 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

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

innovate achieve lead

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