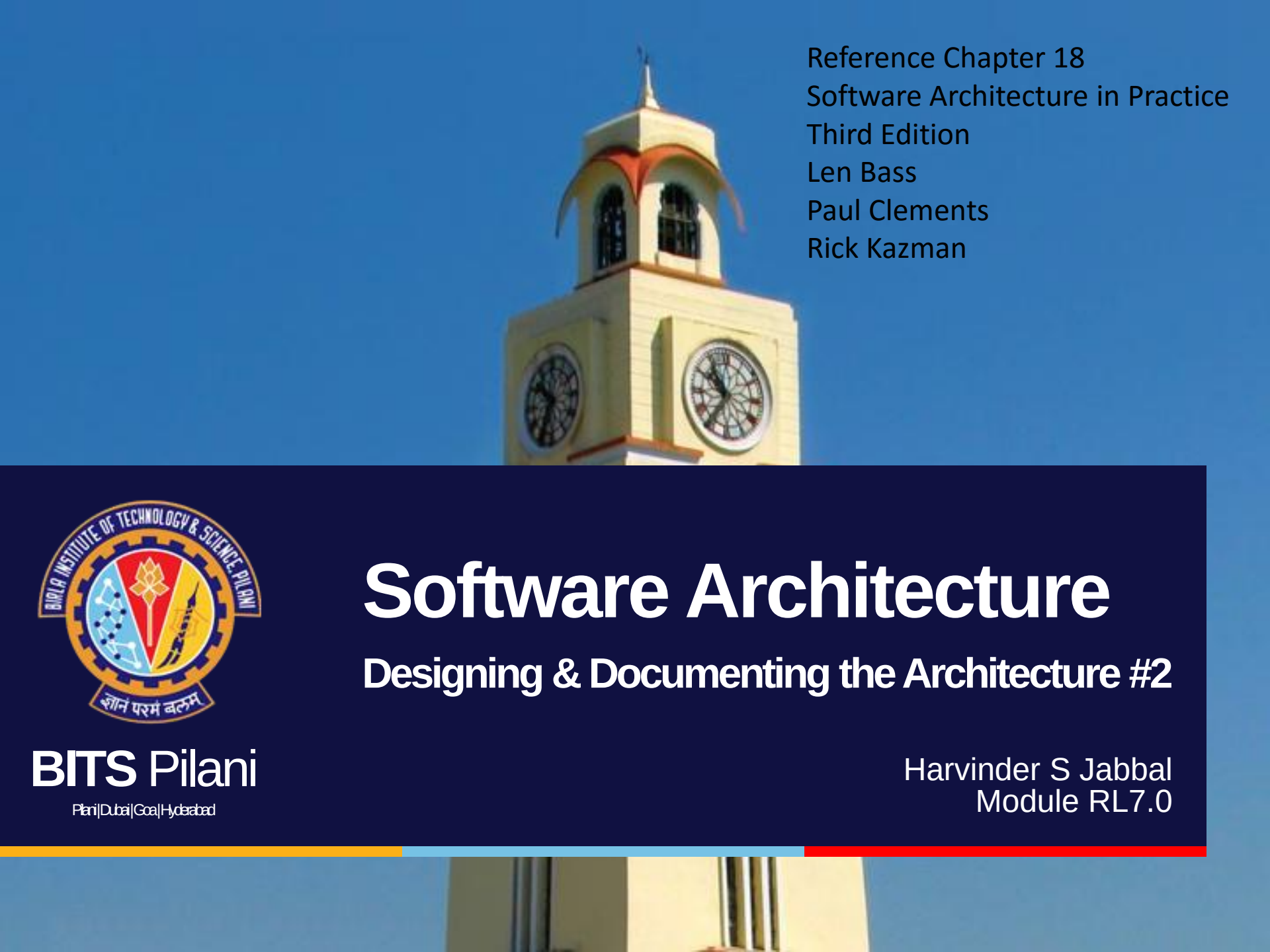


Reference Chapter 18
Software Architecture in Practice
Third Edition
Len Bass
Paul Clements
Rick Kazman

Software Architecture

Designing & Documenting the Architecture #2

Harvinder S Jabbal
Module RL7.0

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Designing & Documenting the Architecture #2

innovate achieve lead

Documenting the Architecture with relevant views

- Module,
- C&C,
- Allocation &
- Quality

Documenting beyond views,

- Documenting behaviours with sequence diagram

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Documenting the Architecture with relevant views – Module, C&C, Allocation & Quality

Architecture Documentation

Even the best architecture will be useless if the people who need it

- do not know what it is;
- cannot understand it well enough to use, build, or modify it;
- misunderstand it and apply it incorrectly.

All of the effort, analysis, hard work, and insightful design on the part of the architecture team will have been wasted.

Chapter Outline

innovate achieve lead

1. Uses and Audiences for Architecture Documentation
2. Notations for Architecture Documentation
3. Views
4. Choosing the Views
5. Combining Views
6. Building the Documentation Package
7. Documenting Behavior
8. Architecture Documentation and Quality Attributes
9. Documenting Architectures That Change Faster Than You Can Document Them
10. Documenting Architecture in an Agile Development Project
11. Summary

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

1. Uses and Audience for Architecture Documentation

1. Uses and Audience for Architecture Documentation

innovate achieve lead

Architecture documentation must	• be sufficiently transparent and accessible to be quickly understood by new employees • be sufficiently concrete to serve as a blueprint for construction • have enough information to serve as a basis for analysis.
Architecture documentation is both prescriptive and descriptive.	• For some audiences, it prescribes what *should* be true, placing constraints on decisions yet to be made. • For other audiences, it describes what *is* true, recounting decisions already made about a system's design.
Understanding stakeholder uses of architecture documentation is essential	• Those uses determine the information to capture.

Three Uses for Architecture Documentation

Education

- Introducing people to the system
  - New members of the team
  - External analysts or evaluators
  - New architect

Primary vehicle for communication among stakeholders

- Especially architect to developers
- Especially architect to future architect!

Basis for system analysis and construction

- documentation serves as the basis for architecture evaluation.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

2. Notations

2. Notations

Informal notations	• Views are depicted (often graphically) using general-purpose diagramming and editing tools • The semantics of the description are characterized in natural language • They cannot be formally analyzed
Semiformal notations	• Standardized notation that prescribes graphical elements and rules of construction • Lacks a complete semantic treatment of the meaning of those elements • Rudimentary analysis can be applied • UML is a semiformal notation in this sense.
Formal notations	• Views are described in a notation that has a precise (usually mathematically based) semantics. • Formal analysis of both syntax and semantics is possible. • Architecture description languages (ADLs) • Support automation through associated tools.

Choosing a Notation

innovate achieve lead

Tradeoffs

- Typically, more formal notations take more time and effort to create and understand, but offer reduced ambiguity and more opportunities for analysis.
- Conversely, more informal notations are easier to create, but they provide fewer guarantees.

Different notations are better (or worse) for expressing different kinds of information.

- UML class diagram will not help you reason about schedulability, nor will a sequence chart tell you very much about the system's likelihood of being delivered on time.
- Choose your notations and representation languages knowing the important issues you need to capture and reason about.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

3. Views

3. Views

Views let us divide a software architecture into a number of (we hope) interesting and manageable representations of the system.

Principle of architecture documentation:

- *Documenting an architecture is a matter of documenting the relevant views and then adding documentation that applies to more than one view.*

Which Views? The Ones You Need!

Different views support different goals and uses.

We do not advocate a particular view or collection of views.

The views you should document depend on the uses you expect to make of the documentation.

Each view has a cost and a benefit; you should ensure that the benefits of maintaining a view outweigh its costs.

Overview of Module Views

innovate achieve lead

Elements

- Modules, which are implementation units of software that provide a coherent set of responsibilities.

Relations

- *Is part of*, which defines a part/whole relationship between the submodule—the part—and the aggregate module—the whole.
- *Depends on*, which defines a dependency relationship between two modules. Specific module views elaborate what dependency is meant.
- *Is a*, which defines a generalization/specialization relationship between a more specific module—the child—and a more general module—the parent.

Overview of Module Views

innovate achieve lead

Constraints

- Different module views may impose specific topological constraints, such as limitations on the visibility between modules.

Usage

- Blueprint for construction of the code
- Change-impact analysis
- Planning incremental development
- Requirements traceability analysis
- Communicating the functionality of a system and the structure of its code base
- Supporting the definition of work assignments, implementation schedules, and budget information
- Showing the structure of information that the system needs to manage

Module Views

innovate achieve lead

- It is unlikely that the documentation of any software architecture can be complete without at least one module view.

Overview of C&C Views

innovate achieve lead

Elements

- *Components.* Principal processing units and data stores. A component has a set of *ports* through which it interacts with other components (via connectors).
- *Connectors*. Pathways of interaction between components. Connectors have a set of roles (interfaces) that indicate how components may use a connector in interactions.

Relations

- *Attachments*. Component ports are associated with connector roles to yield a graph of components and connectors.
- *Interface delegation.* In some situations component ports are associated with one or more ports in an "internal" subarchitecture. The case is similar for the roles of a connector

Overview of C&C Views

innovate achieve lead

Constraints

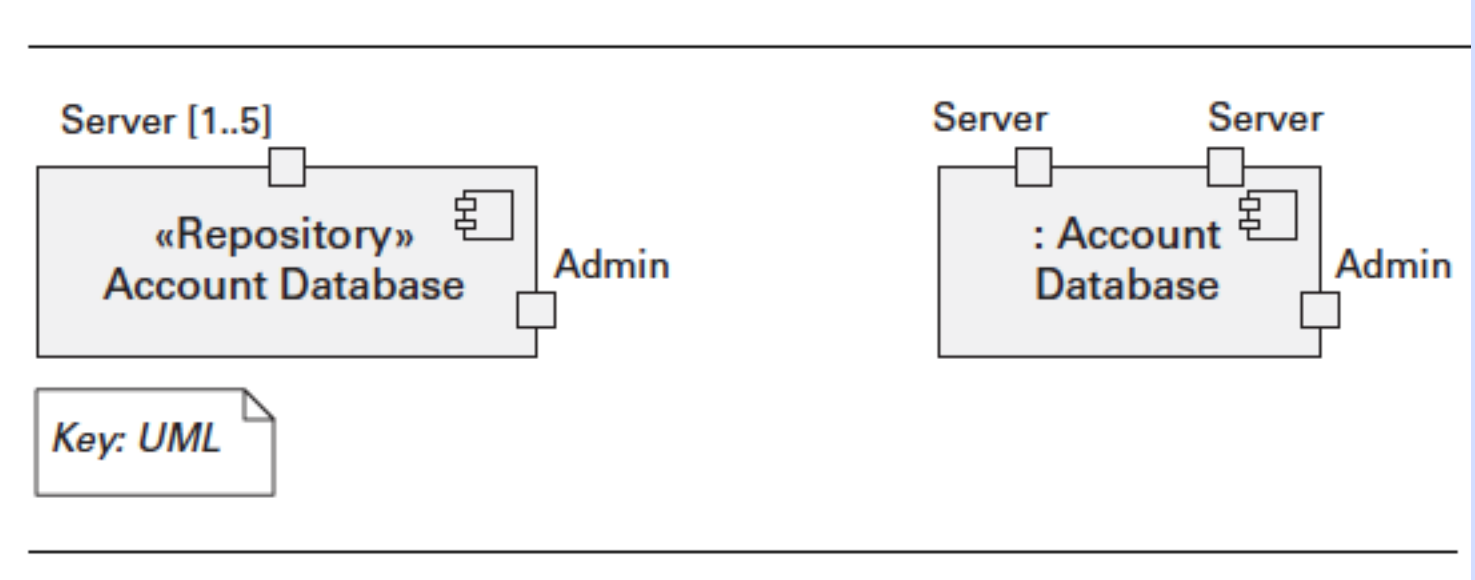
- Components can only be attached to connectors, not directly to other components.
- Connectors can only be attached to components, not directly to other connectors.
- Attachments can only be made between compatible ports and roles.
- Interface delegation can only be defined between two compatible ports (or two compatible roles).
- Connectors cannot appear in isolation; a connector must be attached to a component.

Usage

- Show how the system works.
- Guide development by specifying structure and behavior of runtime elements.
- Help reason about runtime system quality attributes, such as performance and availability.

Notations for C&C Views

UML components are good match for C&C components.



Suggested Reading:

http://agilemodeling.com/artifacts/componentDiagram.htm

Notations for C&C Views

UML connectors are not rich enough to represent many C&C connectors.	• UML connectors cannot have substructure, attributes, or behavioral descriptions.
Represent a “simple” C&C connector using a UML connector—a line.	• Many commonly used C&C connectors have well-known, application-independent semantics and implementations, such as function calls or data read operations. • You can use a stereotype to denote the type of connector.
Connector roles cannot be explicitly represented with a UML connector.	• The UML connector element does not allow the inclusion of interfaces. • Label the connector ends and use these labels to identify role descriptions that must be documented elsewhere.
Represent a “rich” C&C connector	• using a UML component, or by annotating a line UML connector with a tag that explains the meaning of the complex connector.

Overview of Allocation Views

innovate achieve lead

Elements

- *Software element* and *environmental element.*
- A software element has properties that are *required* of the environment.
- An environmental element has properties that are *provided* to the software.

Relations

- *Allocated to*. A software element is mapped (allocated to) an environmental element. Properties are dependent on the particular view.

Overview of Allocation Views

innovate achieve lead

Constraints

- Varies by view

Usage

- Reasoning about performance, availability, security, and safety.
- Reasoning about distributed development and allocation of work to teams.
- Reasoning about concurrent access to software versions.
- Reasoning about the form and mechanisms of system installation.

Quality Views

innovate achieve lead

A *quality view* can be tailored

- for specific stakeholders or to address specific concerns.

A quality views is formed

- by extracting the relevant pieces of structural views and packaging them together.

Quality Views: Examples

- Show the components that have some security role or responsibility, how those components communicate, any data repositories for security information, and repositories that are of security interest.
- The view's context information would show other security measures (such as physical security) in the system's environment.
- The behavior part of a security view
  - Show how the operation of security protocols and where and how humans interact with the security elements.
  - Capture how the system would respond to

Security view

- Especially helpful for systems that are globally dispersed and heterogeneous.
- Show all of the component-to-component channels, the various network channels, quality-of-service parameter values, and areas of concurrency.
- Used to analyze certain kinds of performance and reliability (such as deadlock or race condition detection).
- The behavior part of this view could show (for example) how network bandwidth is dynamically allocated.

Communications view

Quality Views: Examples

innovate achieve lead

- Could help illuminate and draw attention to error reporting and resolution mechanisms.
- Show how components detect, report, and resolve faults or errors.
- It would help identify the sources of errors

Exception or *error-handling view*

- Models mechanisms such as replication and switchover.
- Depicts timing issues and transaction integrity.

Reliability view

- Shows those aspects of the architecture useful for inferring the system's performance.
- Show network traffic models, maximum latencies for operations, and so forth.

Performance view

Sample separator slide for the presentation

4. Choosing the Views

You can determine which views are required, when to create them, and how much detail to include if you know the following:

What people, and with what skills, are available	Which standards you have to comply with	What budget is on hand	What the schedule is	What the information needs of the important stakeholders are	What the driving quality attribute requirements are	What the approximate size of the system is

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

5. Combining Views

5. Combining Views

innovate achieve lead

At a minimum, expect to have		
at least one module view,	at least one C&C view,	and for larger systems, at least one allocation view in your architecture document.

Method for Choosing the Views

innovate achieve lead

Step 1. Build a stakeholder/view table.

- Rows: List the stakeholders for your project's software architecture documentation
- Columns: Enumerate the views that apply to your system.
  - Use the structures discussed in Chapter 1, the views discussed in this chapter, and the views that your design work in ADD has suggested as a starting list of candidates.
  - Include the views or view sketches you have as a result of your design work so far.
- Some views (such as decomposition, uses, and work assignment) apply to every system, while others (various C&C views, the layered view) only apply to some systems.
- Fill in each cell to describe how much information the stakeholder requires from the view: none, overview only, moderate detail, or high detail.

Method for Choosing the Views

innovate achieve lead

Step 2. Combine views to reduce their number

- Look for marginal views in the table; those that require only an overview, or that serve very few stakeholders.
- Combine each marginal view with another view that has a stronger constituency.
- These views often combine naturally:
  - *Various C&C views.* Because C&C views all show runtime relations among components and connectors of various types, they tend to combine well.
  - *Deployment view with either SOA or communicating-processes views.* An SOA view shows services, and a communicating-processes view shows processes. In both cases, these are components that are deployed onto processors.
  - *Decomposition view and any of work assignment, implementation, uses, or layered views.* The decomposed modules form the units of work, development, and uses. In addition, these modules populate layers.

Method for Choosing the Views

innovate achieve lead

Step 3. Prioritize and stage.

- The decomposition view (one of the module views) is a particularly helpful view to release early.
  - High-level decompositions are often easy to design
  - The project manager can start to staff development teams, put training in place, determine which parts to outsource, and start producing budgets and schedules.
- You don't have to satisfy all the information needs of all the stakeholders to the fullest extent.
  - Providing 80 percent of the information goes a long way, and this might be "good enough" so that the stakeholders can do their job.
  - Check with the stakeholder if a subset of information would be sufficient.
- You don't have to complete one view before starting another.
  - People can make progress with overview-level information
  - A breadth-first approach is often the best.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

6. Building the Documentation Package

6. Building the Documentation Package

Documentation package consists of

Views	Documentation beyond views

Documenting a View

innovate achieve lead

Section 1: The Primary Presentation.

- The *primary presentation* shows the elements and relations of the view.
- The primary presentation should contain the information you wish to convey about the system—in the vocabulary of that view.
- The primary presentation is most often graphical.
  - It might be a diagram you've drawn in an informal notation using a simple drawing tool, or it might be a diagram in a semiformal or formal notation imported from a design or modeling tool that you're using.
  - If your primary presentation is graphical, make sure to include a key that explains the notation.
  - Lack of a key is the most common mistake that we see in documentation in practice.
- Occasionally the primary presentation will be textual, such as a table or a list.
  - If that text is presented according to certain stylistic rules, these rules should be stated or incorporated by reference, as the analog to the graphical notation key.

Documenting a View

innovate achieve lead

Section 2: The Element Catalog.

- The *element catalog* details at least those elements depicted in the primary presentation.
  - For instance, if a diagram shows elements A, B, and C, then the element catalog needs to explain what A, B, and C are.
  - If elements or relations relevant to this view were omitted from the primary presentation, they should be introduced and explained in the catalog.
- Parts of the catalog:
  - *Elements and their properties*. This section names each element in the view and lists the properties of that element. Each view introduced in Chapter 1 listed a set of suggested properties associated with that view.
  - *Relations and their properties*. Each view has specific relation types that it depicts among the elements in that view.
  - *Element interfaces*. This section documents element interfaces.
  - *Element behavior*. This section documents element behavior that is not obvious from the primary presentation.

Documenting a View

Section 3: Context Diagram.

- A *context diagram* shows how the system or portion of the system depicted in this view relates to its environment.
- The purpose of a context diagram is to depict the scope of a view.
- Entities in the environment may be humans, other computer systems, or physical objects, such as sensors or controlled devices.

Section 4: Variability Guide.

- A *variability guide* shows how to exercise any variation points that are a part of the architecture shown in this view.

Section 5: Rationale.

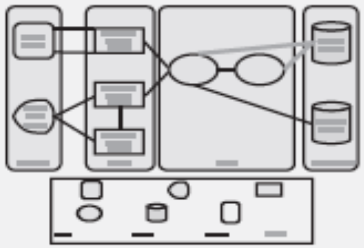
- *Rationale* explains why the design reflected in the view came to be.
- The goal of this section is to explain why the design is as it is and to provide a convincing argument that it is sound.
- The choice of a pattern in this view should be justified here by describing the architectural problem that the chosen pattern solves and the rationale for choosing it over another.

View Template

innovate achieve lead

Template for a View

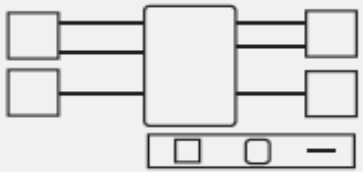
Section 1. Primary Presentation



Section 2. Element Catalog

- Section 2.A. Elements and Their Properties
- Section 2.B. Relations and Their Properties
- Section 2.C. Element Interfaces
- Section 2.D. Element Behavior

Section 3. Context Diagram



Section 4. Variability Guide

Section 5. Rationale

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Documenting beyond views,

Documenting Information Beyond Views

innovate achieve lead

Document control information.

List the • issuing organization, • the current version number, • date of issue and • status, • a change history, and • the procedure for submitting change requests to the document.	Usually captured in the front matter

Documenting Information Beyond Views

Section 1: Documentation Roadmap. The documentation map tells the reader what information is in the documentation and where to find it.

- *Scope and summary*. Explain the purpose of the document and briefly summarize what is covered.
- *How the documentation is organized*. For each section in the documentation, give a short synopsis of the information that can be found there.
- *View overview*. Describes the views that the architect has included in the package. For each view::
  - The name of the view and what pattern it instantiates, if any.
  - A description of the view's element types, relation types, and property types.
  - A description of language, modeling techniques, or analytical methods used in constructing the view.
- *How stakeholders can use the documentation*.
  - This section shows how various stakeholders might use the documentation to help address their concerns.
  - Include short scenarios, such as "A maintainer wishes to know the units of software that are likely to be changed by a proposed modification."
  - To be compliant with ISO/IEC 42010-2007, you must consider the concerns of at least users, acquirers, developers, and maintainers.

Documenting Information Beyond Views

innovate achieve lead

Section 2: How a View Is Documented.

- Explain the standard organization you're using to document views—either the one described in this chapter or one of your own.

Section 3: System Overview.

- Short prose description of the system's function, its users, and any important background or constraints.
- Provides your readers with a consistent mental model of the system and its purpose.
- This might be a pointer to your project's concept-of-operations document for the system.

Documenting Information Beyond Views

innovate achieve lead

Section 4: Mapping Between Views.

- Helping a reader understand the associations between views will help that reader gain a powerful insight into how the architecture works as a unified conceptual whole.
- The associations between elements across views in an architecture are, in general, many-to-many.
- View-to-view associations can be captured as tables.
  - The table should name the correspondence between the elements across the two views.
  - Examples
    - "is implemented by" for mapping from a component-and-connector view to a module view
    - "implements" for mapping from a module view to a component-and-connector view
    - "included in" for mapping from a decomposition view to a layered view

Documenting Information Beyond Views

innovate achieve lead

Section 5: Rationale.

- Documents the architectural decisions that apply to more than one view.
  - Documentation of background or organizational constraints or major requirements that led to decisions of system-wide import.
  - Decisions about which fundamental architecture patterns are used.

Section 6: Directory.

- Set of reference material that helps readers find more information quickly.
  - Index of terms
  - Glossary
  - Acronym list.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

7 Documenting behaviours

7. Documenting Behavior

Behavior documentation complements each views by describing how architecture elements in that view interact with each other.

Behavior documentation enables reasoning about

- a system's potential to deadlock
- a system's ability to complete a task in the desired amount of time
- maximum memory consumption
- and more

Behavior has its own section in our view template's element catalog.

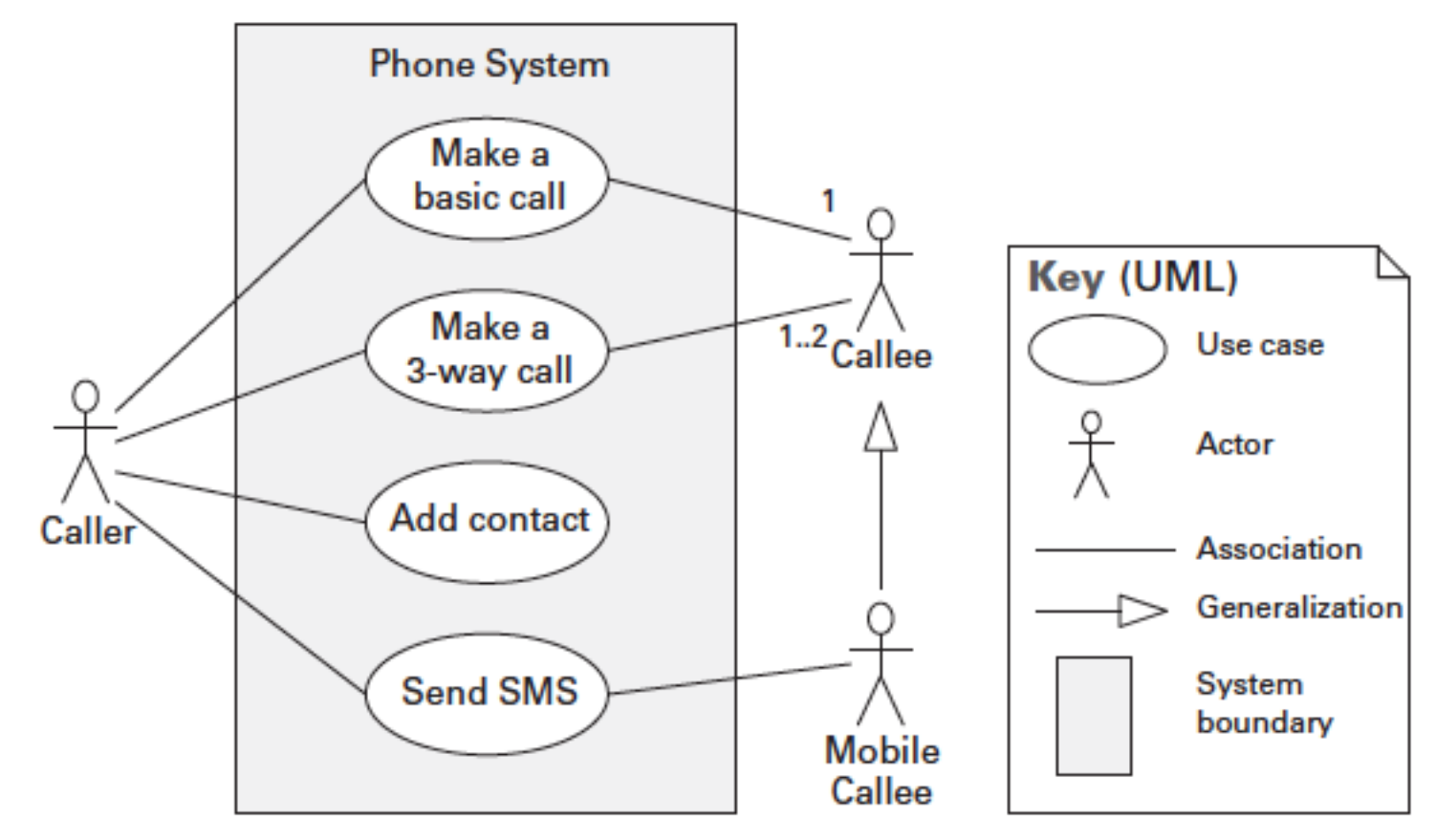
Notations for Documenting Behavior

innovate achieve lead

Trace-oriented languages

Traces are sequences of activities or interactions that describe the system's response to a specific stimulus when the system is in a specific state.	A trace describes a particular sequence of activities or interactions between structural elements of the system.	Examples • use cases • sequence diagrams • communication diagrams • activity diagrams • message sequence charts • timing diagrams • Business Process Execution Language

Use Case Diagram



From *Documenting Software Architectures: Views and Beyond*, 2nd edition, Addison Wesley, 2010

Use Case Description

Name: Make a basic call

Description: Making a point-to-point connection between two phones.

Primary actors: Caller

Secondary actors: Callee

Flow of events:

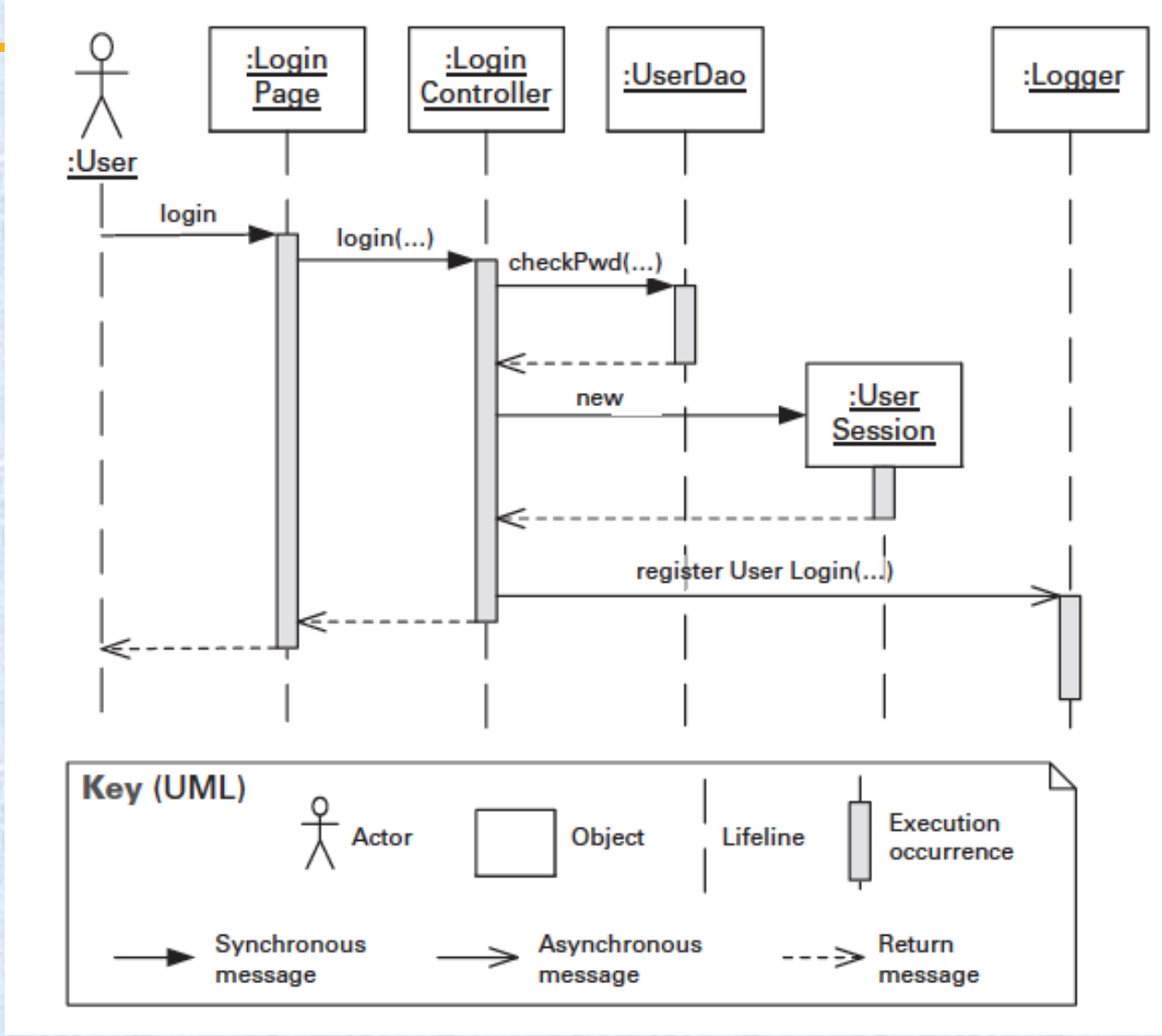
The use case starts when a caller places a call via a terminal, such as a cell phone. All terminals to which the call should be routed then begin ringing. When one of the terminals is answered, all others stop ringing and a connection is made between the caller's terminal and the terminal that was answered. When either terminal is disconnected—someone hangs up—the other terminal is also disconnected. The call is now terminated, and the use case is ended.

Exceptional flow of events:

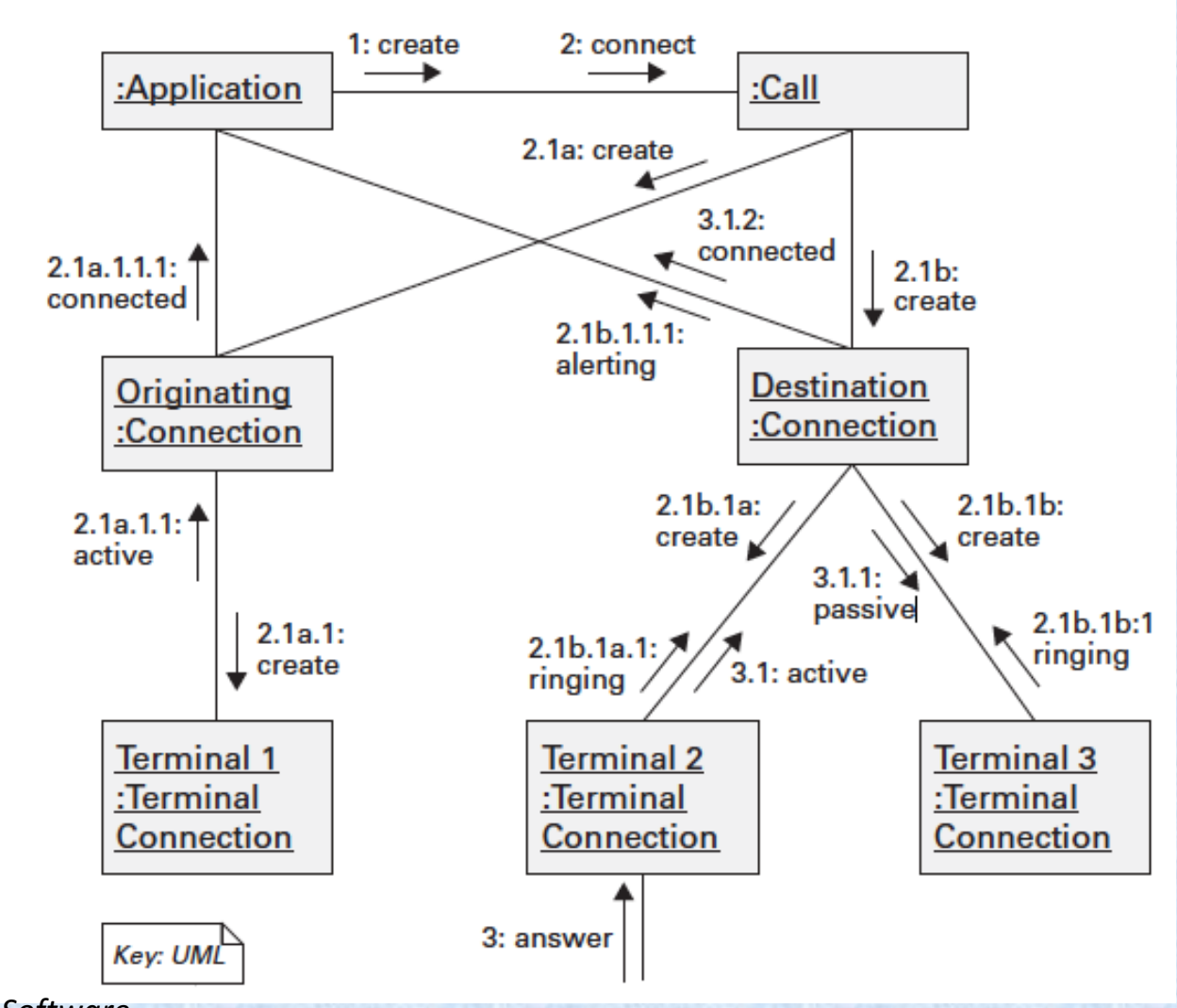
The caller can disconnect, or hang up, before any of the ringing terminals has been answered. If this happens, all ringing terminals stop ringing and are disconnected, ending the use case.

From *Documenting Software Architectures: Views and Beyond*, 2nd edition, Addison Wesley, 2010

Sequence Diagram

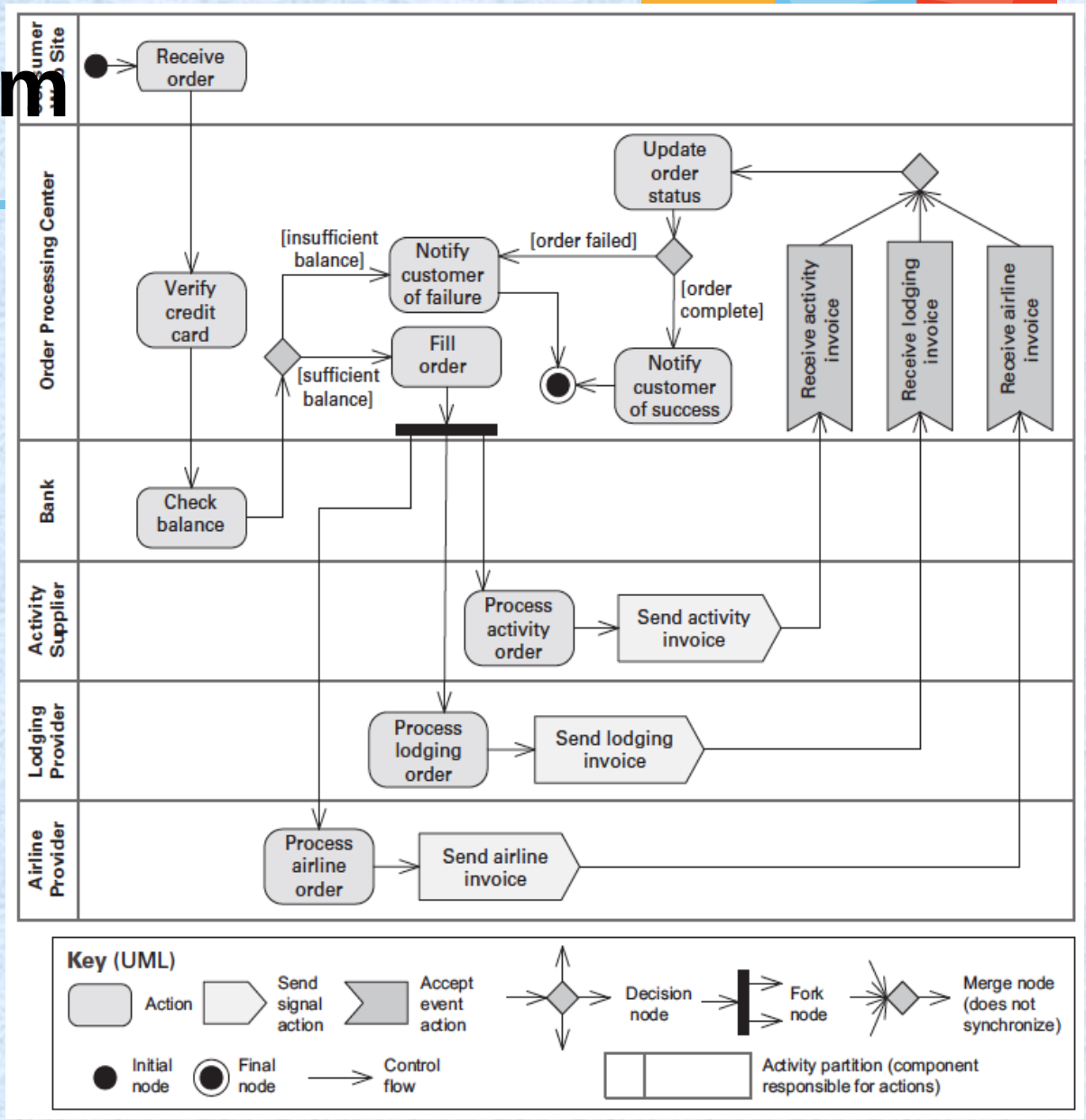


Communication Diagram



From *Documenting Software Architectures: Views and Beyond*, 2nd edition, Addison Wesley, 2010

Activity Diagram



From *Documenting Software Architectures: Views and Beyond*, 2nd edition, Addison Wesley, 2010

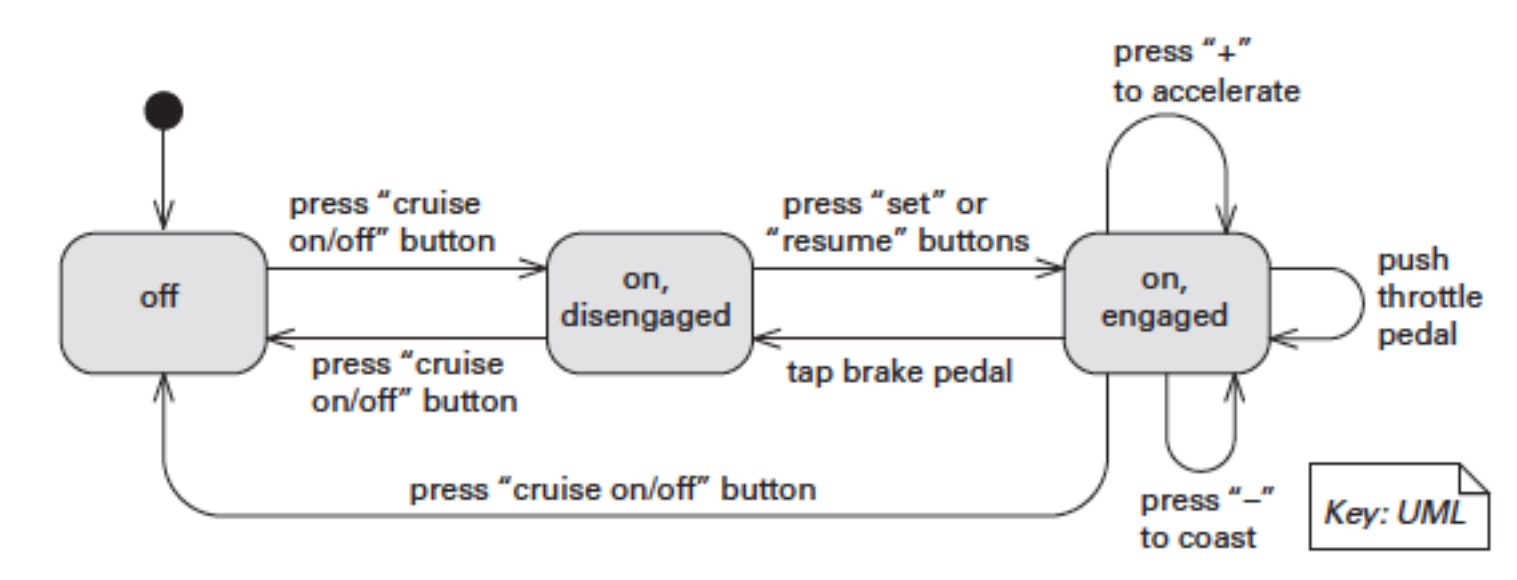
Notations for Documenting Behavior

innovate achieve lead

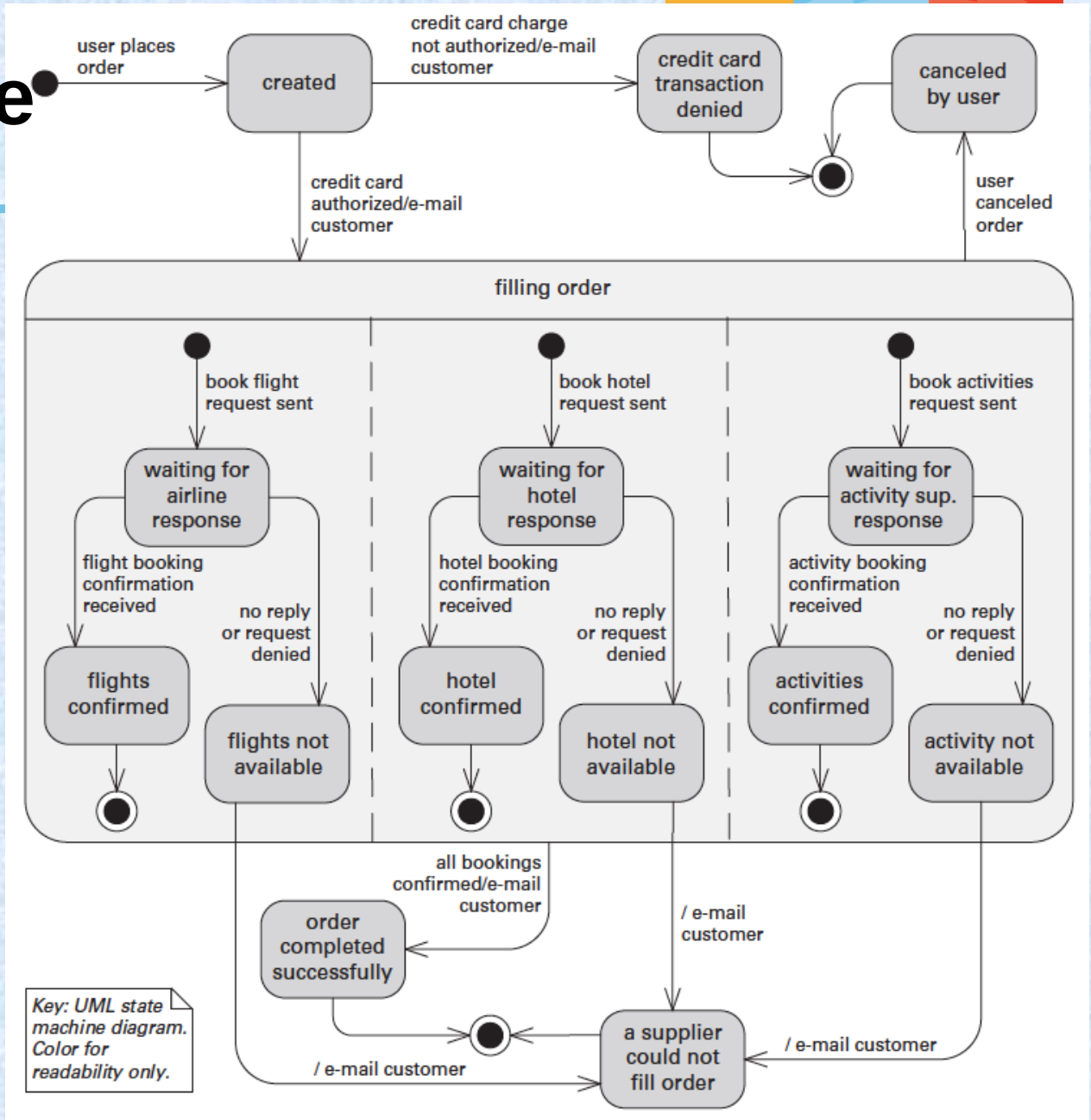
Comprehensive languages

- *Comprehensive models* show the complete behavior of structural elements.
- Given this type of documentation, it is possible to infer all possible paths from initial state to final state.
- The state machine formalism represents the behavior of architecture elements because each state is an abstraction of all possible histories that could lead to that state.
- State machine languages allow you to complement a structural description of the elements of the system with constraints on interactions and timed reactions to both internal and environmental stimuli.

State Machine



State Machine



From *Documenting Software Architectures: Views and Beyond*, 2nd edition, Addison Wesley, 2010

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

8. Documenting Quality Attributes

8. Documenting Quality Attributes

Where do quality attributes show up in the documentation? There are five major ways:

Rationale that explains the choice of design approach should include a discussion about the quality attribute requirements and tradeoffs.	Architectural elements providing a service often have quality attribute bounds assigned to them, defined in the interface documentation for the elements, or recorded as *properties* that the elements exhibit.	Quality attributes often impart a "language" of things that you would look for. Someone fluent in the "language" of a quality attribute can search for the kinds of architectural elements) put in place to satisfy that quality attribute requirement.	Architecture documentation often contains a *mapping to requirements* that shows how requirements (including quality attribute requirements) are satisfied.	Every quality attribute requirement will have a constituency of stakeholders who want to know that it is going to be satisfied. For these stakeholders, the roadmap tells the stakeholder where in the document to find it.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

9. Documenting Architectures That Change Faster Than You Can Document Them

9. Documenting Architectures That Change Faster Than You Can Document Them

An architecture that changes at runtime, or as a result of a high-frequency release-and-deploy cycle, change much faster than the documentation cycle.

Nobody will wait until a new architecture document is produced, reviewed, and released.

In this case:

- *Document what is true about all versions of your system.* Record those invariants as you would for any architecture. This may make your documented architecture more a description of constraints or guidelines that any compliant version of the system must follow.
- *Document the ways the architecture is allowed to change.* This will usually mean adding new components and replacing components with new implementations. The place to do this is called the variability guide.

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

10. Documenting Architecture in an Agile Development Project

10. Documenting Architecture in an Agile Development Project

Adopt a template or standard organization to capture your design decisions.

Plan to document a view if (but only if) it has a strongly identified stakeholder constituency.

Fill in the sections of the template for a view, and for information beyond views, when (and in whatever order) the information becomes available. But only do this if writing down this information will make it easier (or cheaper or make success more likely) for someone downstream doing their job.

Don't worry about creating an architectural design document and then a finer-grained design document. Produce just enough design information to allow you to move on to code.

Don't feel obliged to fill up all sections of the template, and certainly not all at once. Write "N/A" for the sections for which you don't need to record the information (perhaps because you will convey it orally).

Agile teams sometimes make models in brief discussions by the whiteboard. Take a picture and use it as the primary presentation.

Summary

- You must understand the uses to which the writing is to be put and the audience for the writing.
- Architectural documentation serves as a means for communication among various stakeholders, not only up the management chain and down to the developers but also across to peers.
- An architecture is a complicated artifact, best expressed by focusing on views.
- You must choose the views to document, must choose the notation to document these views, and must choose a set of views that is both minimal and adequate.
- You must document not only the structure of the architecture but also the behavior.

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