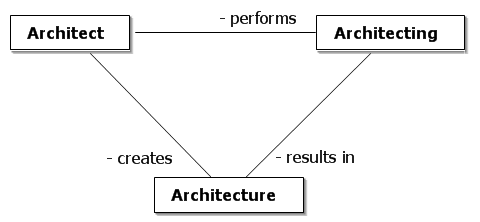
# Core Concepts



Architecture

* *Architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. [IEEE 1471]*
* *The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them. [Bass]*
* *[Architecture is] the organizational structure and associated behavior of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems. [UML 1.5]*

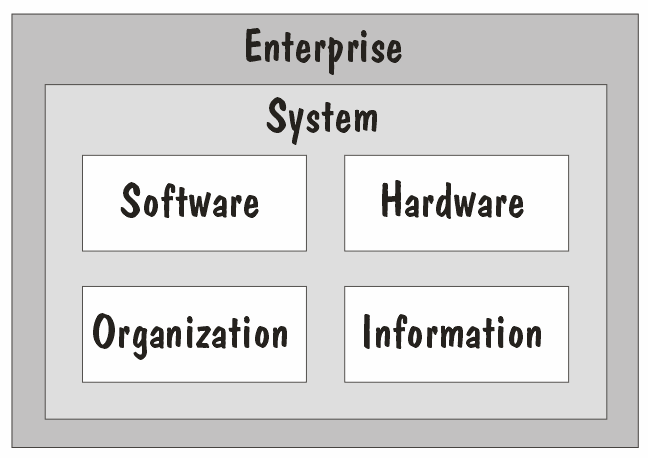
# Architecture versus Design



*All architecture is design but not all design is architecture. Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change.*

- Grady Booch

An architecture has a particular scope



# The benefits of architecting

* + Architecting helps manage complexity
  + Architecting ensures architectural integrity
  + Architecting provides a basis for reuse
  + Architecting addresses system qualities
  + Architecting drives consensus
  + Architecting reduces maintenance costs
  + Architecting supports impact analysis
  + Architecting supports the planning process

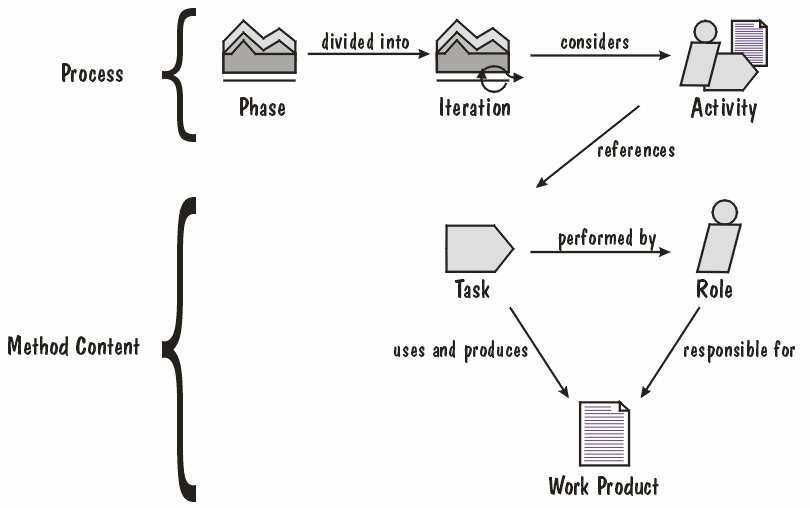
# Agenda

* + Introduction
  + Architecture, Architect, Architecting Method fundamentals

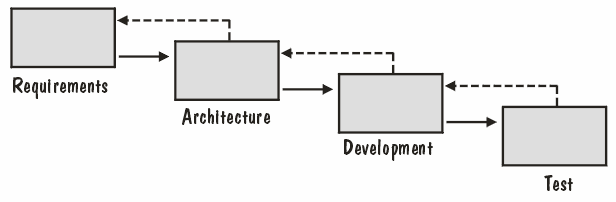
▪

* + Documenting a software architecture
  + Reusable architecture assets
  + A day in the life
  + Summary

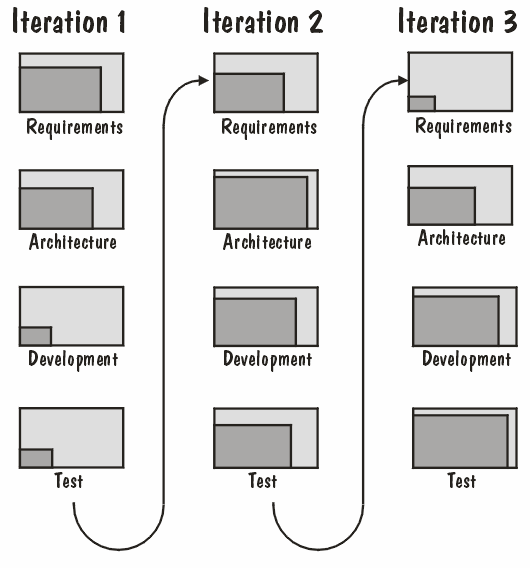
# Key Method Concepts



A Waterfall Process



An Iterative Process



Agile

* + Agile Manifesto

cIndividuals and interactions over processes and tools. cWorking software over comprehensive documentation. cCustomer collaboration over contract negotiation. cResponding to change over following a plan.

* + *Scrum is a management and control process that cuts through complexity to focus on building software to meet business needs. Scrum is superimposed on top of and wraps existing engineering practices, development methodologies and standards. [Schwaber]*

# Agenda

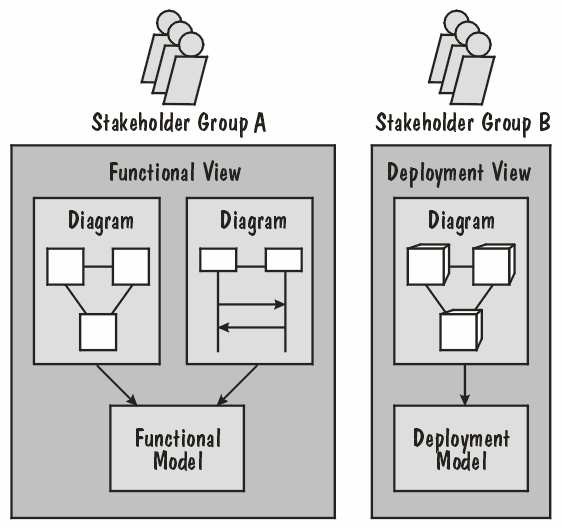
* + Introduction
  + Architecture, Architect, Architecting
  + Method fundamentals

Documenting a software architecture

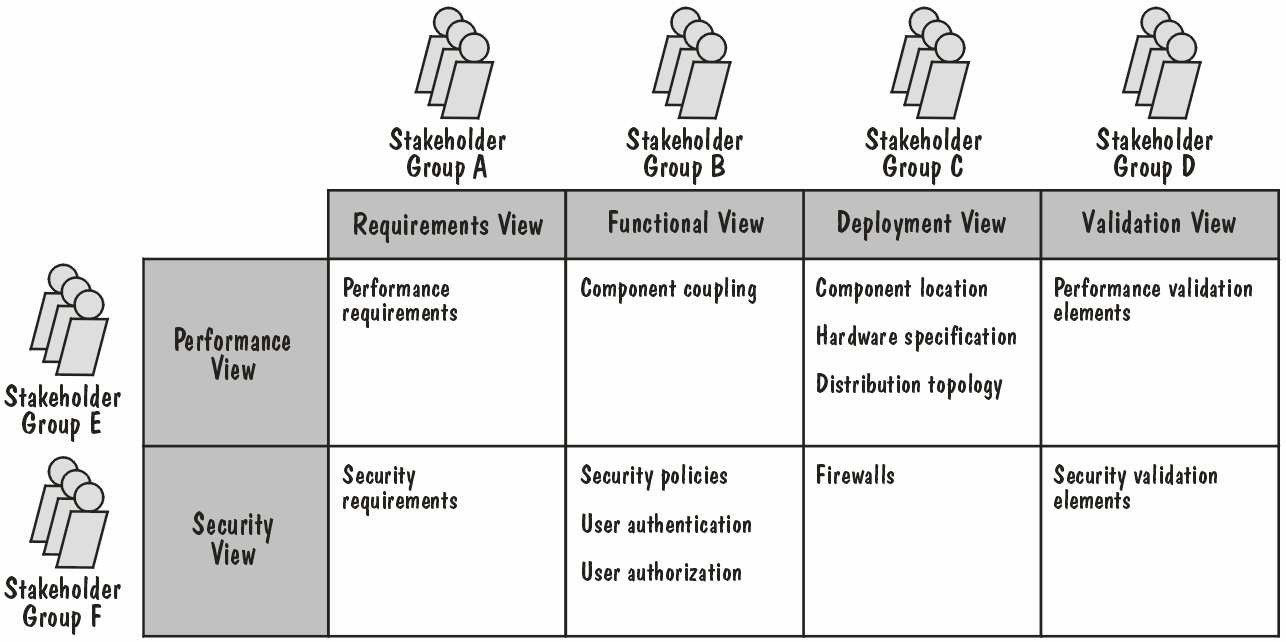
▪

* + Reusable architecture assets
  + A day in the life
  + Summary

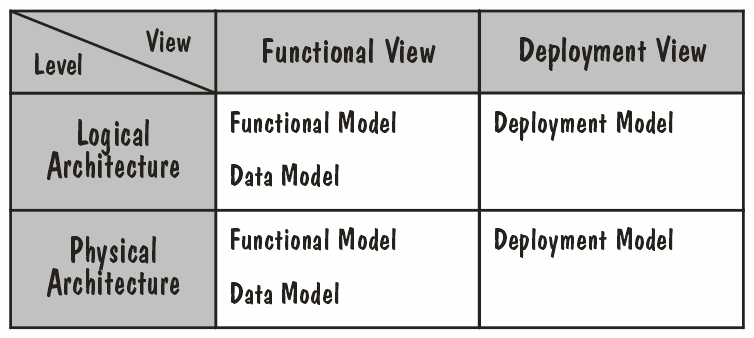
# Views, Diagrams and Models



Basic Views and Cross-Cutting Views



Views, Models and Levels of Realization



# Agenda

* + Introduction
  + Architecture, Architect, Architecting
  + Method fundamentals
  + Documenting a software architecture Reusable architecture assets

▪

* + A day in the life
  + Summary

# A Metamodel of Architecture Assets

Agenda

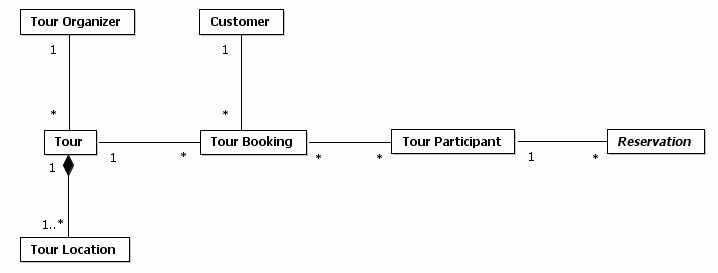
* + Introduction
  + Architecture, Architect, Architecting
  + Method fundamentals
  + Documenting a software architecture
  + Reusable architecture assets day in the life

▪

* + Summary

# Inputs

* + Business Entity Model
  + Business Process Model
  + Business Rules
  + Existing IT Environment
  + Vision



# Types of Requirements

* + Functional requirements

c*Describe the behaviors (functions or services) of the [IT] system that support user goals, tasks or activities. [Malan]*

* + Non-functional requirements

c*Non-functional requirements include constraints and qualities. [Malan]*

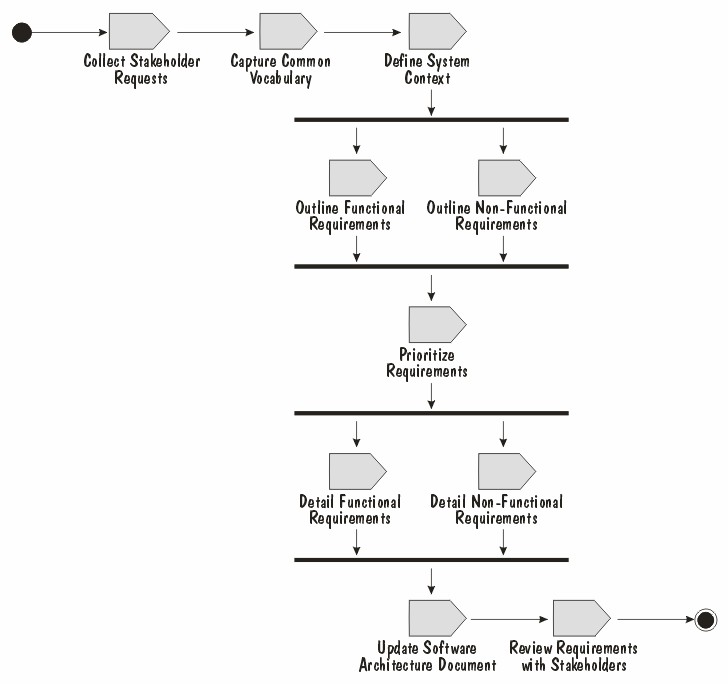
cConstraint

* + - *A constraint is a restriction on the degree of freedom we have in providing a solution. [Leffingwell]*

cQuality

* + - *[System] qualities are properties or characteristics of the system that its stakeholders care about and hence will affect their degree of satisfaction with the system. [Malan]*

# Define Requirements

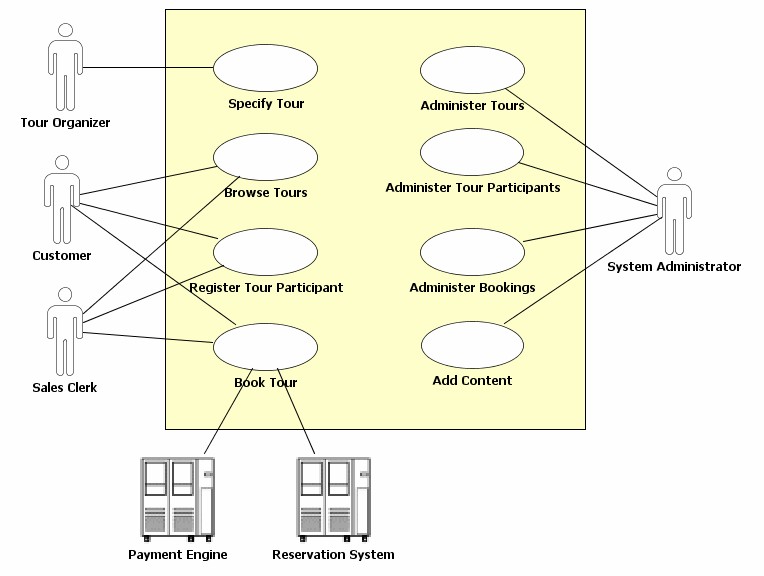


Task: Collect Stakeholder Requests

* + Pitfall: Treating Requests as Requirements
  + Pitfall: The Shopping Cart Mentality
  + Pitfall: The Questions are too Technical
  + Pitfall: Requests Are Too General
  + Pitfall: Requests Are Not Measurable
  + Pitfall: Talking with the Wrong People
  + Pitfall: All Requests Are Equal

# Task: Define System Context

Task: Outline Functional Requirements

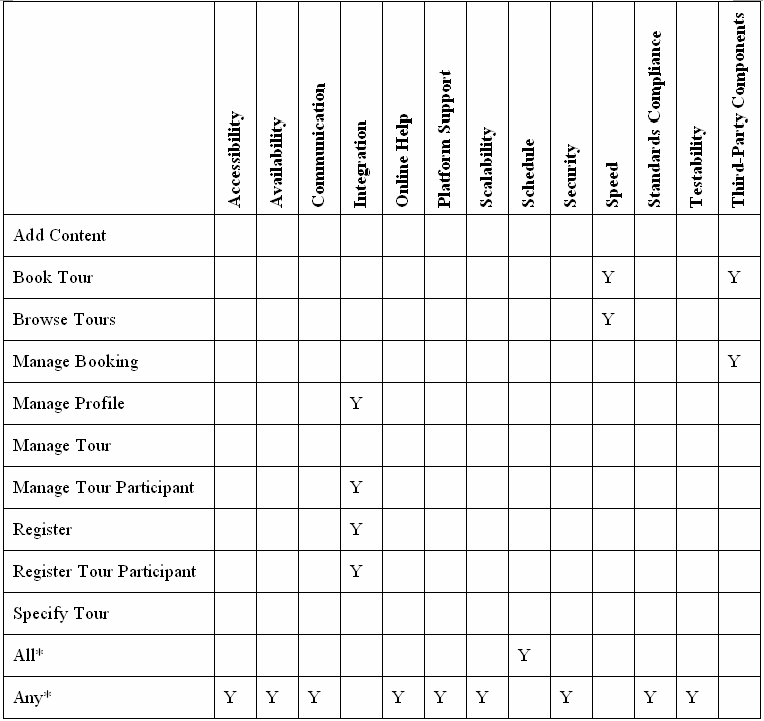


Task: Outline Non-Functional Requirements

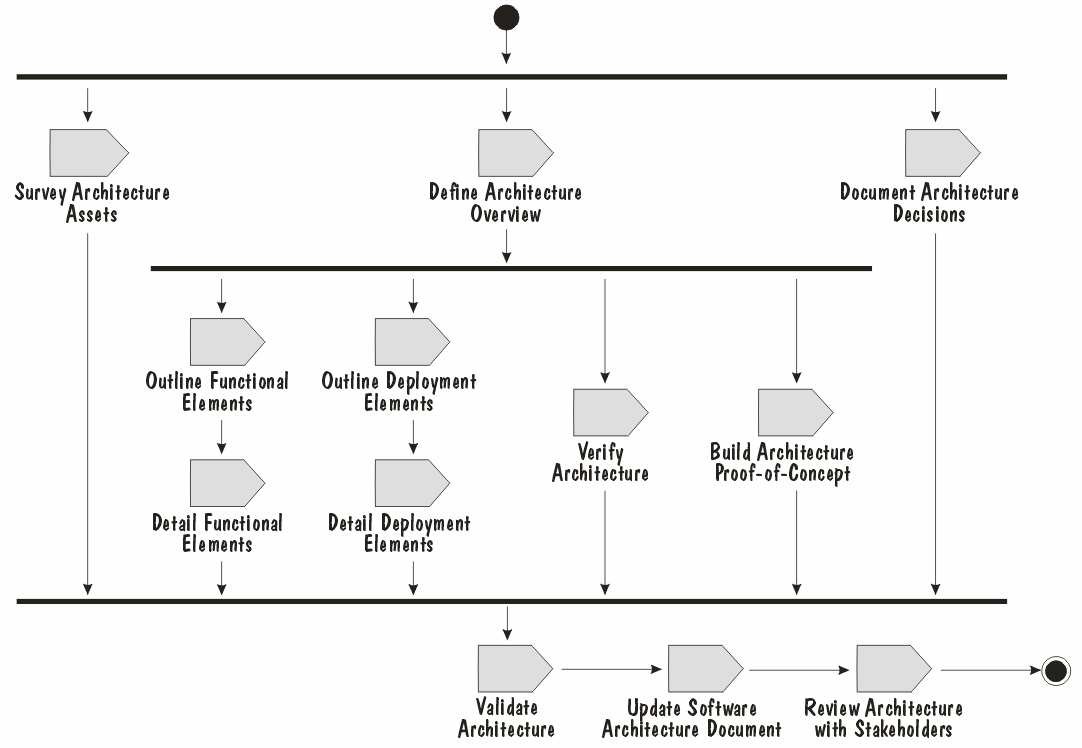
* + Usability Requirements
  + Reliability Requirements
  + Performance Requirements
  + Supportability Requirements
  + Constraints

cBusiness Constraints cArchitecture Constraints cDevelopment Constraints cPhysical Constraints

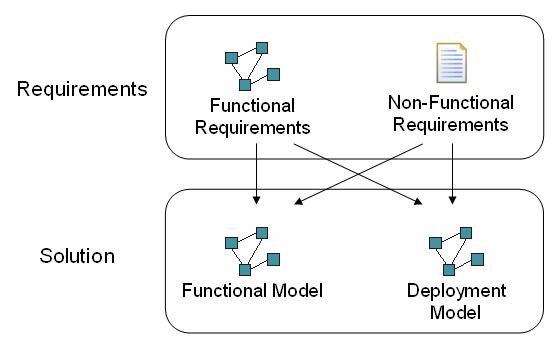
# Task: Prioritize Requirements



Create Logical Architecture



From Requirements to Solution



Approaches

* + Attribute Driven Design (ADD) Method

cDeveloped at the Software Engineering Institute cQuality attributes drive the derivation of the architecture cUnderpinned by architectural tactics and patterns

* + Siemens’ 4 Views (S4V) method

cDeveloped at Siemens Corporate Research

cStarts with a global analysis of the factors that influence the architecture

cIteratively addresses challenges across four views (conceptual, execution, module and code architecture)

* + The Rational Unified Process (RUP) cDeveloped at Rational Software (now IBM Rational) cDriven by architecturally-significant requirements

cEach iteration considers the key architectural elements of the solution, before realizing the requirements using these solution elements

# How Much Logical Architecture?

* + Minimizing Logical Architecture

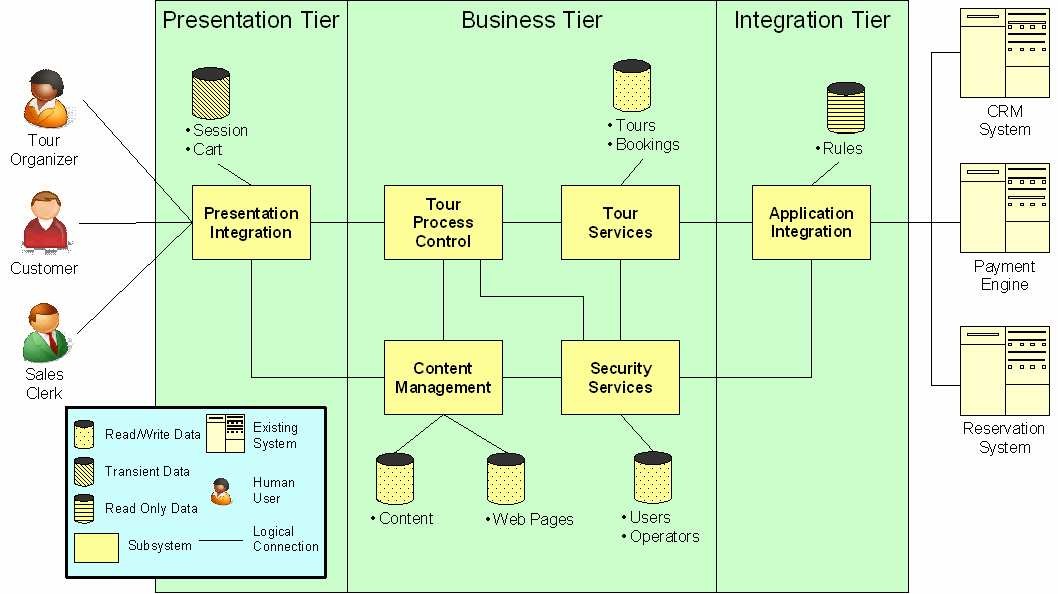
cThe logical architecture is simply a means of getting to a physical architecture as quickly as possible

cIn some cases, no logical architecture may be required at all

* + - E.g. The requirements for the system are similar to those of an existing system
    - E.g. We are using a packaged application or integrating with an existing system
  + Logical Architecture as an Investment

cA valuable asset if a technology change is anticipated at some point in the future

# Task: Define Architecture Overview



Task: Outline Functional Elements

* + Component identification

cBusiness Entity Model

* + - Clustering of related entities

cFunctional requirements

* + - Boundary, control and entity components

cNon-functional requirements

* + - Constraints
    - Components that address specific technical challenges (e.g. security)

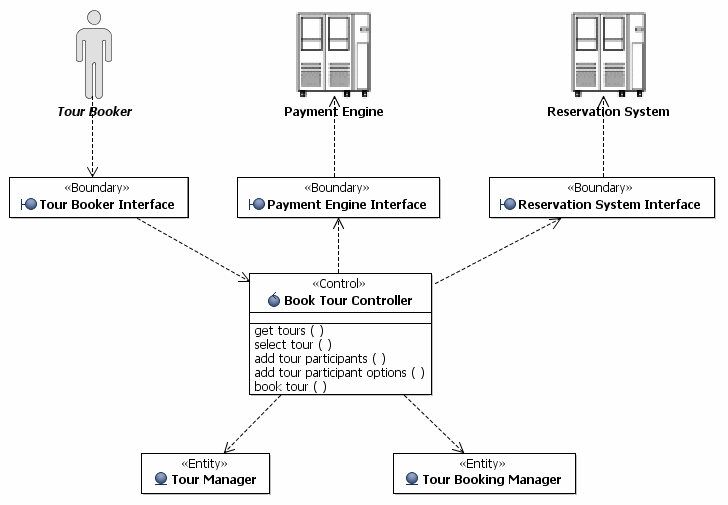
cBusiness rules

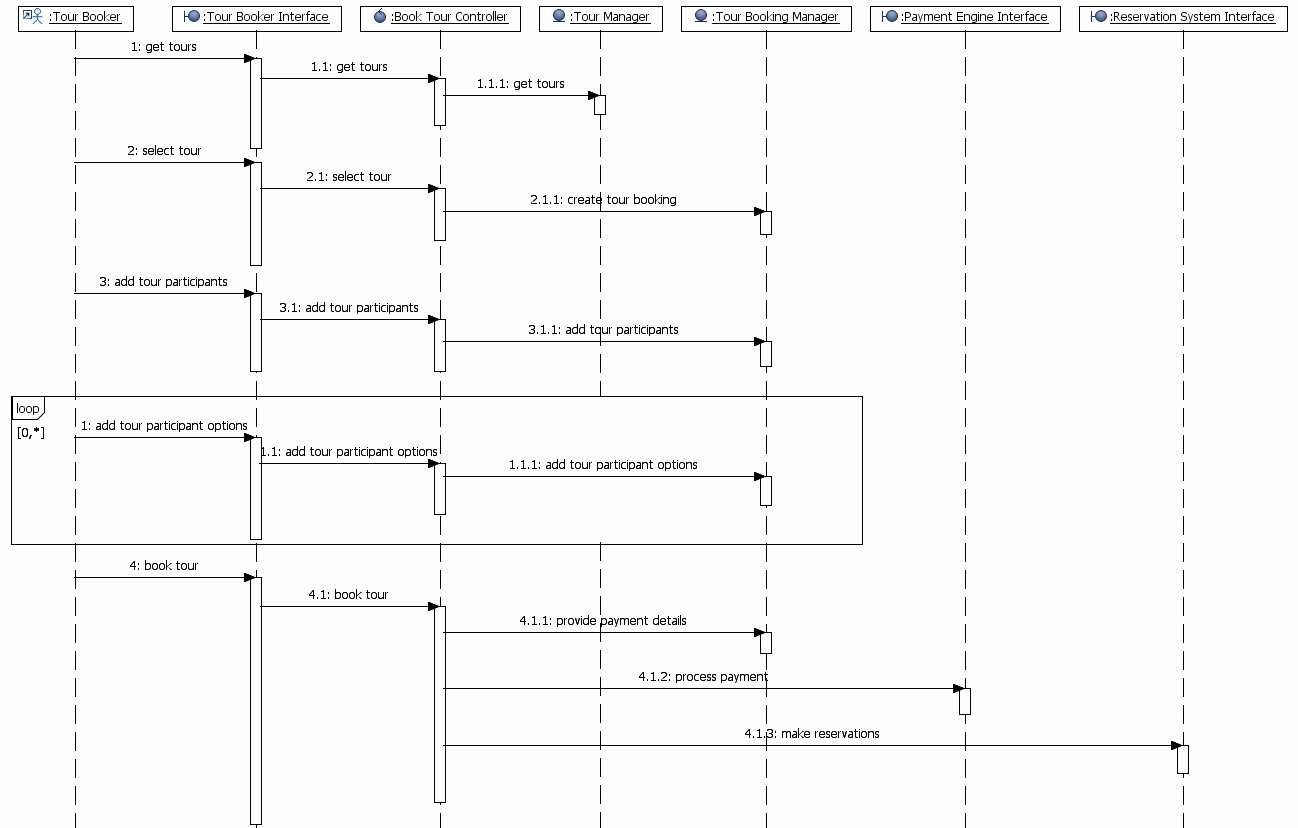
* + - Business rules component(s)

cArchitecture decisions

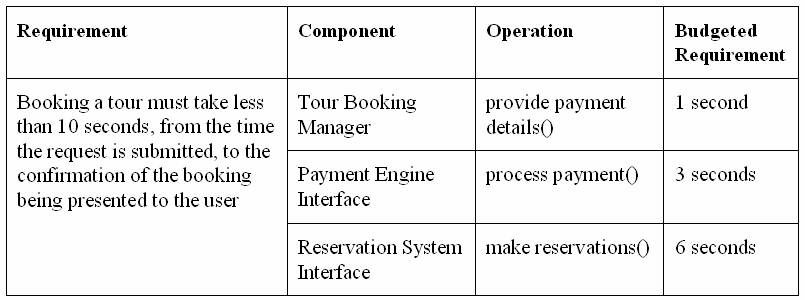
* + - Use of particular assets (e.g. packages, patterns)

# Task: Outline Functional Elements

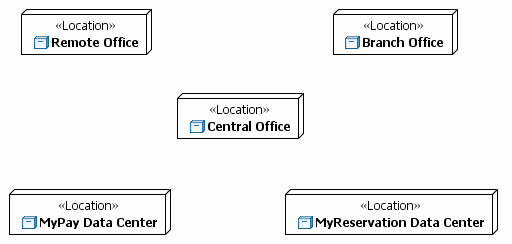


Task: Outline Functional Elements

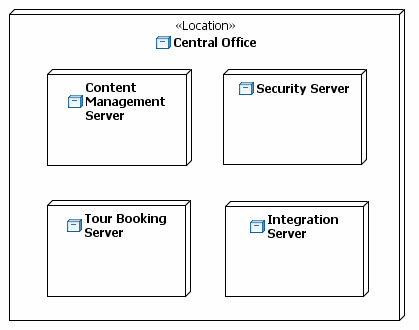
Assigning NFRs to Components



# Task: Outline Deployment Elements



Task: Outline Deployment Elements



Task: Outline Deployment Elements

