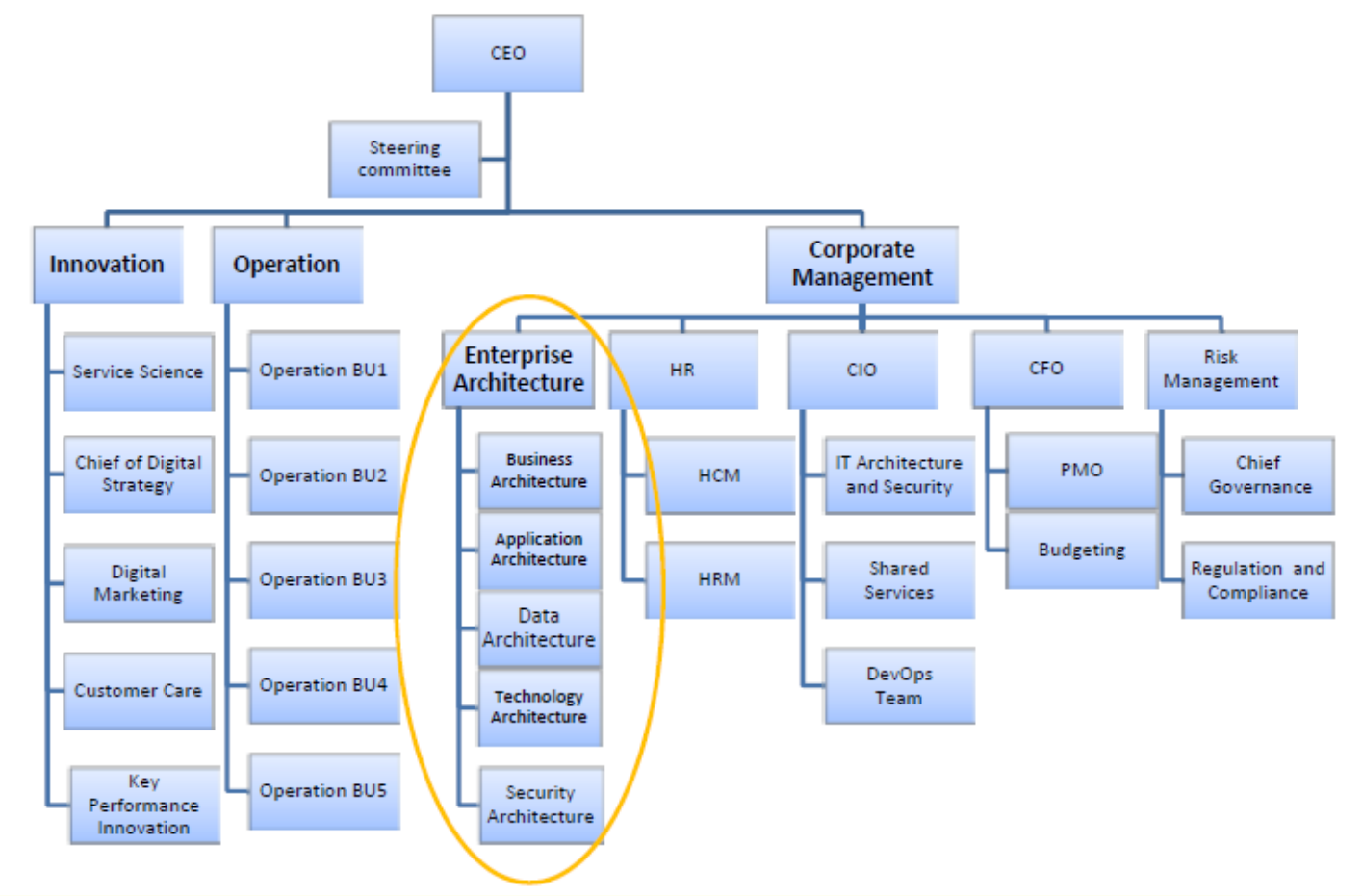
Agenda

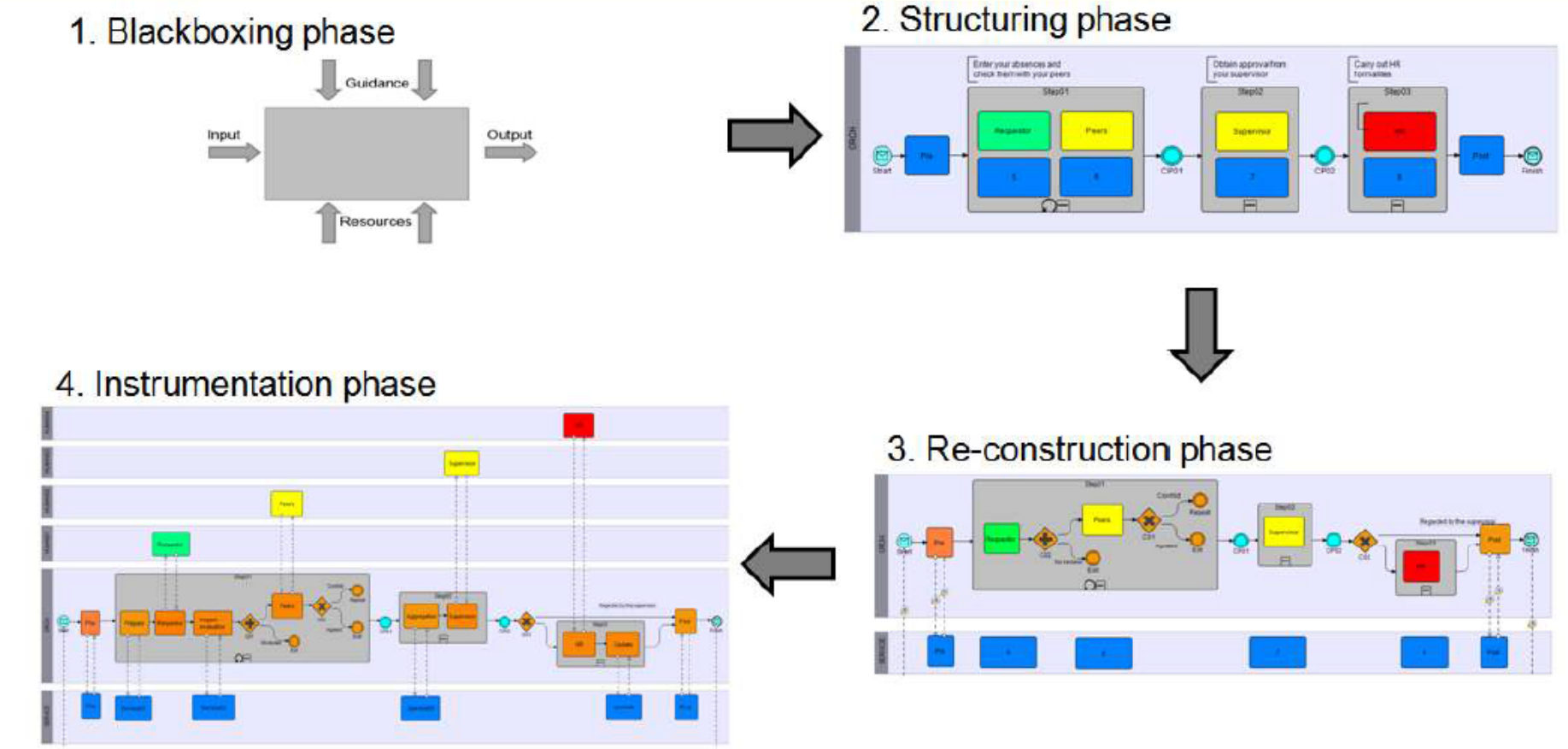
- Enterprise Architecture – Definition & Goals
- Building Enterprise Architecture
- Business Architecture
- Application Architecture
- Data Architecture
- Technology Architecture

Digital Organization Chart



Business Architecture and BPMN

4 phases of business process development



BPMN 2.0 - Business Process Model and Notation

http://bpmb.de/poster

Activities

Task — A **Task** is a unit of work, the job to be performed. When marked with a [+] symbol it indicates a **Sub-Process**, an activity that can be refined.

Transaction — A **Transaction** is a set of activities that logically belong together; it might follow a specified transaction protocol.

Event Sub-Process — An **Event Sub-Process** is placed into a Process or Sub-Process. It is activated when its start event gets triggered and can interrupt the higher level process context or run in parallel (non-interrupting) depending on the start event.

Call Activity — A **Call Activity** is a wrapper for a globally defined Task or Process reused in the current Process. A call to a Process is marked with a [+] symbol.

Activity Markers

Markers indicate execution behavior of activities:

- Sub-Process Marker
- Loop Marker
- Parallel MI Marker
- Sequential MI Marker
- Ad Hoc Marker
- Compensation Marker

Task Types

Types specify the nature of the action to be performed:

- Send Task
- Receive Task
- User Task
- Manual Task
- Business Rule Task
- Service Task
- Script Task

Sequence Flow — defines the execution order of activities.

Default Flow — is the default branch to be chosen if all other conditions evaluate to false.

Conditional Flow — has a condition assigned that defines whether or not the flow is used.

Gateways

Exclusive Gateway — When splitting, it routes the sequence flow to exactly one of the outgoing branches. When merging, it awaits one incoming branch to complete before triggering the outgoing flow.

Event-based Gateway — Is always followed by catching events or receive tasks. Sequence flow is routed to the subsequent event/task which happens first.

Parallel Gateway — When used to split the sequence flow, all outgoing branches are activated simultaneously. When merging parallel branches it waits for all incoming branches to complete before triggering the outgoing flow.

Inclusive Gateway — When splitting, one or more branches are activated. All active incoming branches must complete before merging.

Exclusive Event-based Gateway (instantiate) — Each occurrence of a subsequent event starts a new process instance.

Complex Gateway — Complex merging and branching behavior that is not captured by other gateways.

Parallel Event-based Gateway (instantiate) — The occurrence of all subsequent events starts a new process instance.

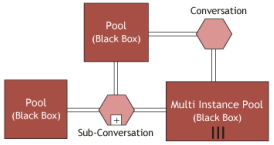
Conversations

A **Conversation** defines a set of logically related message exchanges. When marked with a [+] symbol it indicates a **Sub-Conversation**, a compound conversation element.

A **Call Conversation** is a wrapper for a globally defined Conversation or Sub-Conversation. A call to a Sub-conversation is marked with a [+] symbol.

A **Conversation Link** connects Conversations and Participants.

Conversation Diagram



Choreographies

Participant A / Choreography Task / Participant B — A **Choreography Task** represents an Interaction (Message Exchange) between two Participants.

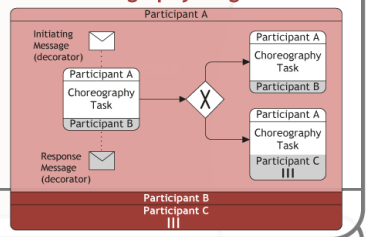
Participant A / Sub-Choreography / Participant B / Participant C — A **Sub-Choreography** contains a refined choreography with several Interactions.

Participant A / Call Choreography / Participant B — A **Call Choreography** is a wrapper for a globally defined Choreography Task or Sub-Choreography. A call to a Sub-Choreography is marked with a [+] symbol.

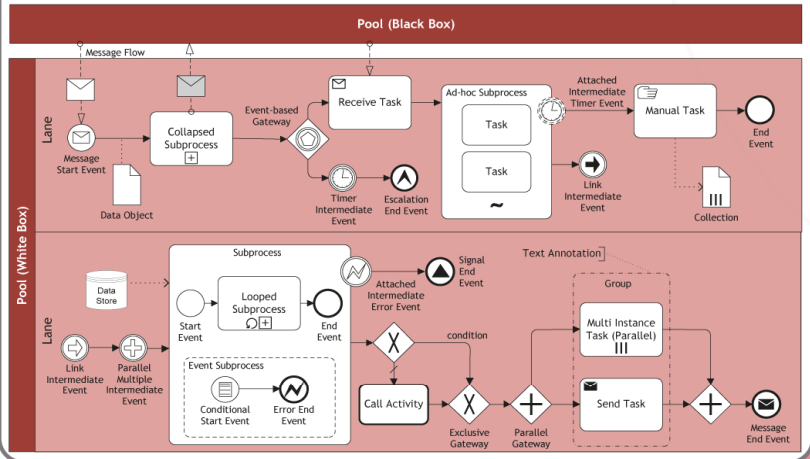
Multiple Participants Marker — denotes a set of Participants of the same kind.

Message — a decorator depicting the content of the message. It can only be attached to Choreography Tasks.

Choreography Diagram



Collaboration Diagram



Events

	Start: Standard	Start: Event Sub-Process Interrupting	Start: Event Sub-Process Non-Interrupting	Intermediate: Catching	Intermediate: Boundary Interrupting	Intermediate: Boundary Non-Interrupting	Intermediate: Throwing	End: Standard
None: Untyped events, indicate start point, state changes or final states.	○						○	○
Message: Receiving and sending messages.	✉	✉	✉	✉	✉	✉	✉	✉
Timer: Cyclic timer events, points in time, time spans or timeouts.	🕒	🕒	🕒	🕒	🕒	🕒		
Escalation: Escalating to an higher level of responsibility.		⅄	⅄		⅄	⅄	⅄	⅄
Conditional: Reacting to changed business conditions or integrating business rules.	☰	☰	☰	☰	☰	☰		
Link: Off-page connectors. Two corresponding link events equal a sequence flow.				⇨			➡	
Error: Catching or throwing named errors.		⚡			⚡			⚡
Cancel: Reacting to cancelled transactions or triggering cancellation.					✕			✕
Compensation: Handling or triggering compensation.		◁◁			◁◁		◀◀	◀◀
Signal: Signalling across different processes. A signal thrown can be caught multiple times.	△	△	△	△	△	△	▲	▲
Multiple: Catching one out of a set of events. Throwing all events defined	⬠	⬠	⬠	⬠	⬠	⬠	⬟	⬟
Parallel Multiple: Catching all out of a set of parallel events.	✚	✚	✚	✚	✚	✚		
Terminate: Triggering the immediate termination of a process.								●

Data

Data Object — A **Data Object** represents information flowing through the process, such as business documents, e-mails, or letters.

Collection Data Object — A **Collection Data Object** represents a collection of information, e.g., a list of order items.

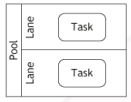
Input — A **Data Input** is an external input for the entire process.A kind of input parameter.

Output — A **Data Output** is data result of the entire process. A kind of output parameter.

A **Data Association** is used to associate data elements to Activities, Processes and Global Tasks.

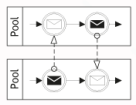
Data Store — A **Data Store** is a place where the process can read or write data, e.g., a database or a filing cabinet. It persists beyond the lifetime of the process instance.

Swimlanes



Pools (Participants) and **Lanes** represent responsibilities for activities in a process. A pool or a lane can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically.

Message Flow symbolizes information flow across organizational boundaries. Message flow can be attached to pools, activities, or message events. The Message Flow can be decorated with an envelope depicting the content of the message.

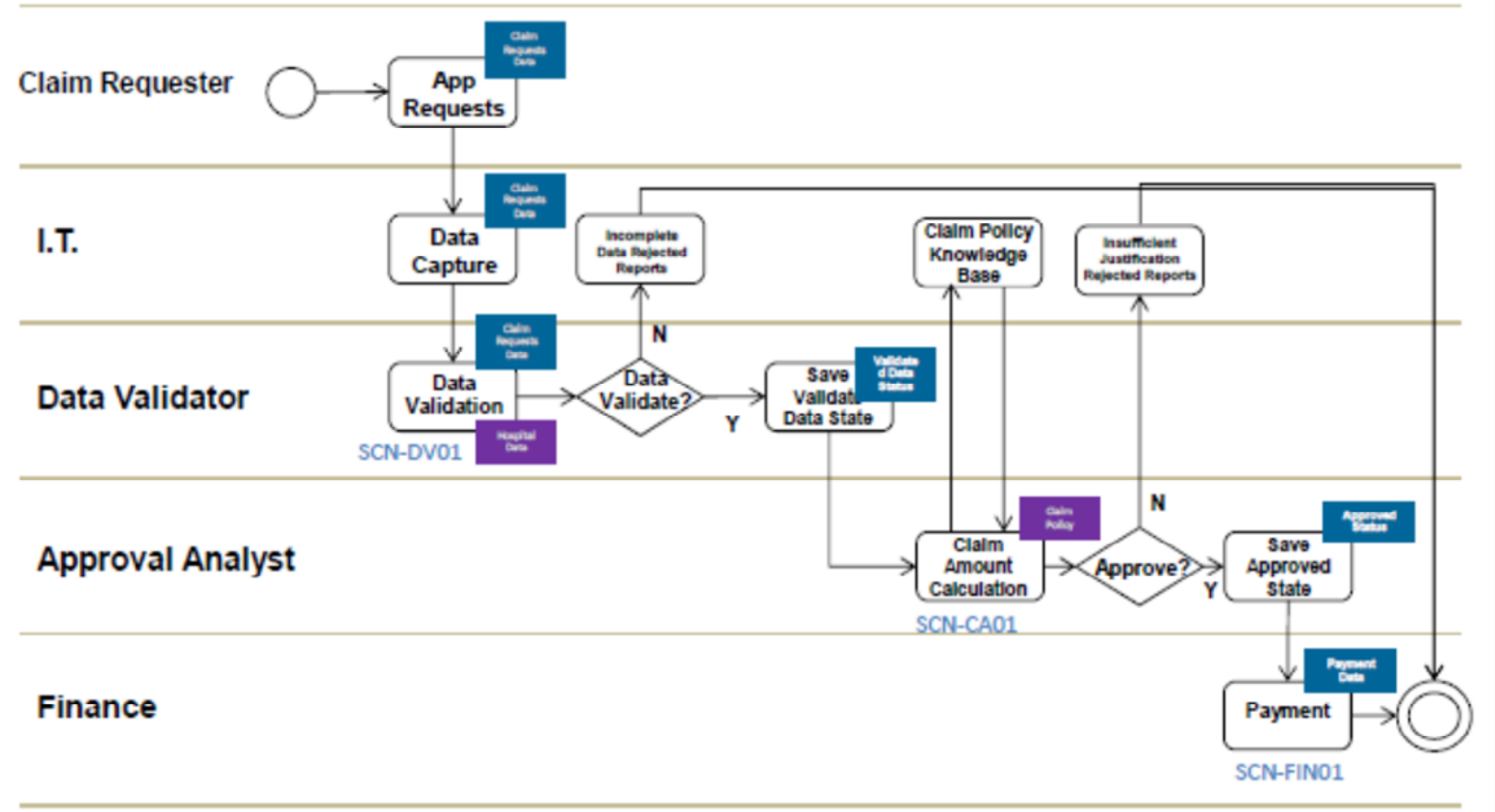


The order of message exchanges can be specified by combining message flow and sequence flow.

BPM Offensive Berlin · HPI Hasso Plattner Institut · Humboldt-Universität zu Berlin · camunda · inubit · SIGNAVIO

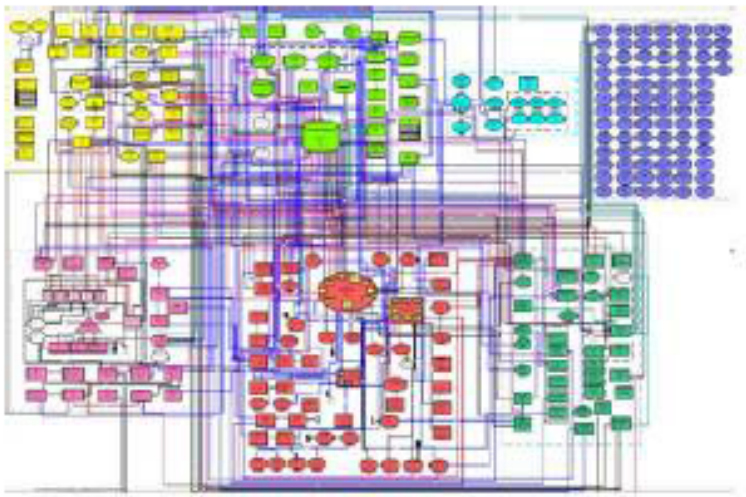
Business Process - Sample Case:

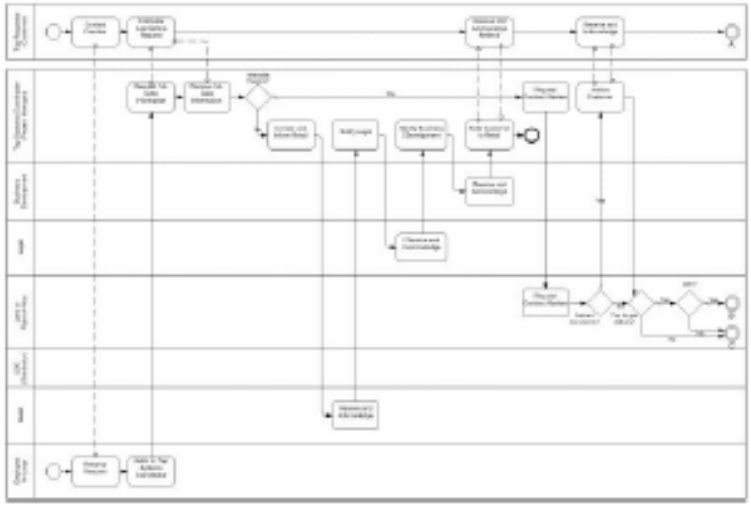
Request Approval Process



Tag Screen ID to the activities which required user interaction

Business Process - Current and Target State





Agenda

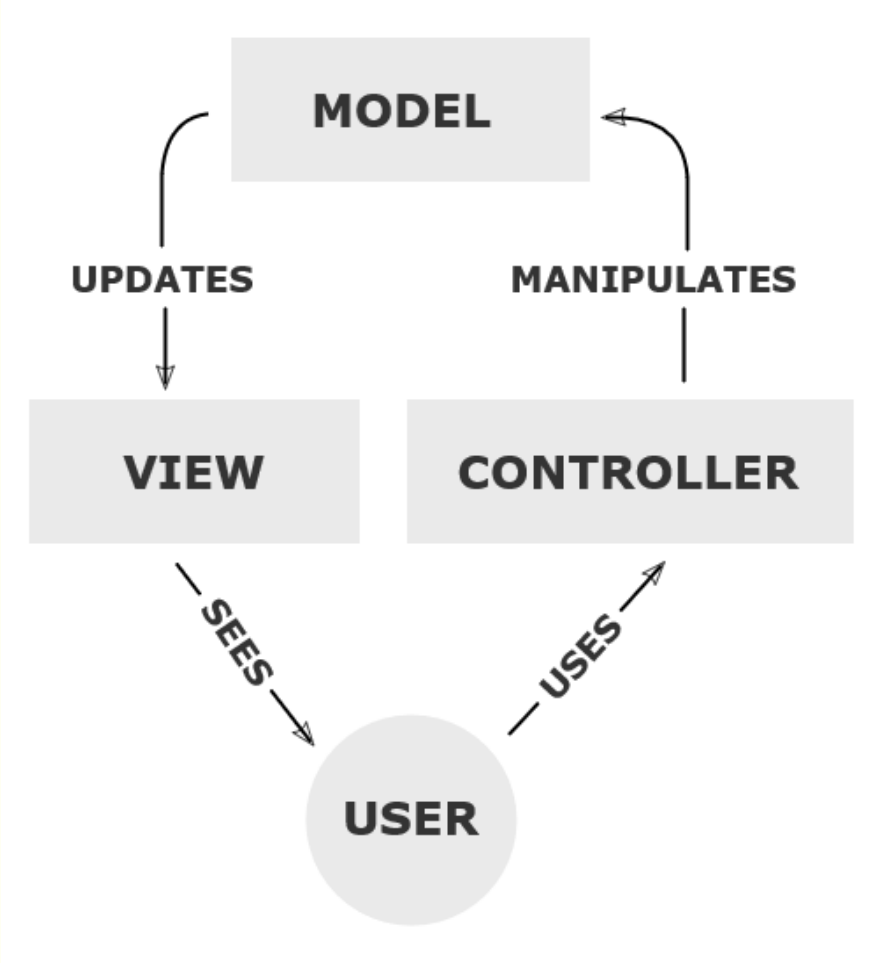
- Enterprise Architecture – Definition & Goals
- Building Enterprise Architecture
- Business Architecture
- Application Architecture
- Data Architecture
- Technology Architecture

Application Architecture - Design Patterns

MVC - Model View Controller

Divides a given software application into three interconnected parts so as to separate internal representations of information from the ways that information is presented to or accepted from the user:

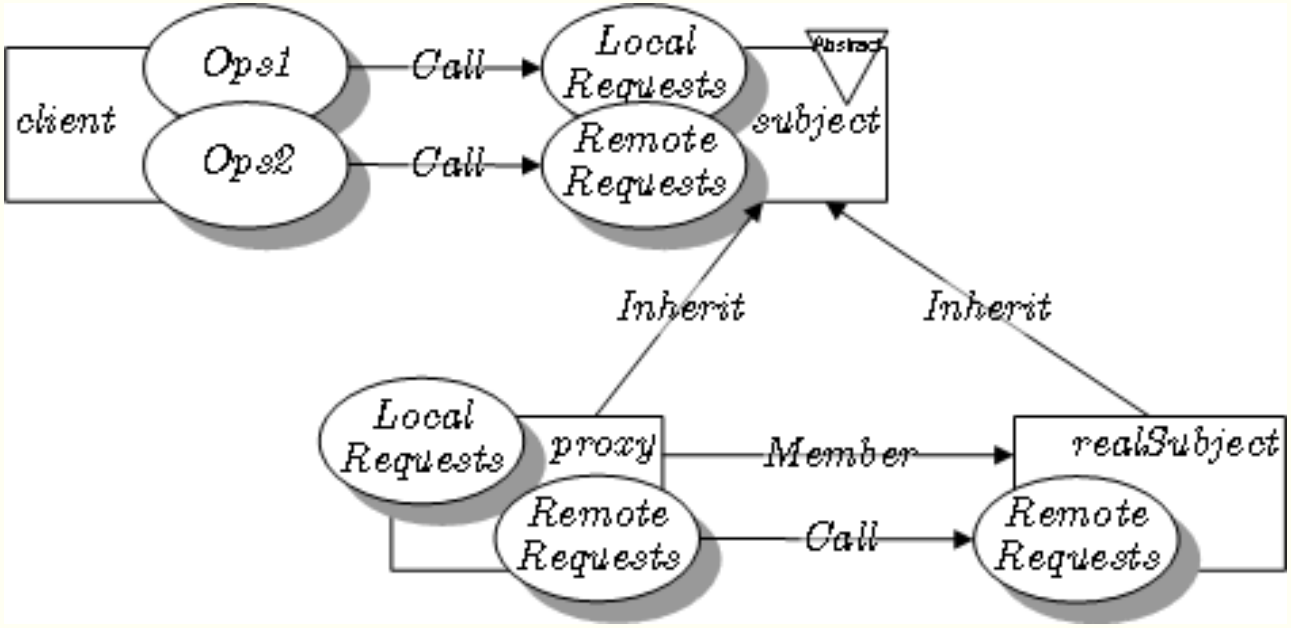
1. The **model** directly manages the data, logic, and rules of the application.
2. A **view** can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
3. The **controller**, accepts input and converts it to commands for the model or view.



Application Architecture - Design Patterns

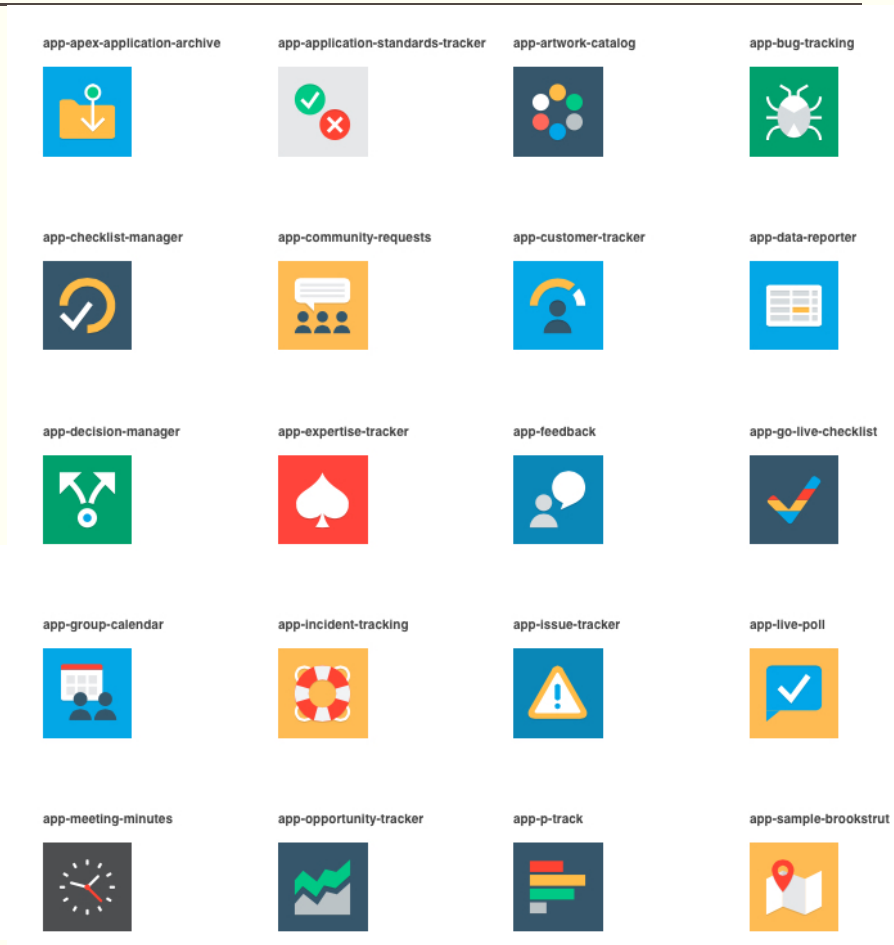
Proxy Pattern

A *proxy*, in its most general form, is a class functioning as an interface to something else.
The proxy could interface to anything: a network connection, a large object in memory, a file, or some other resource that is expensive or impossible to duplicate

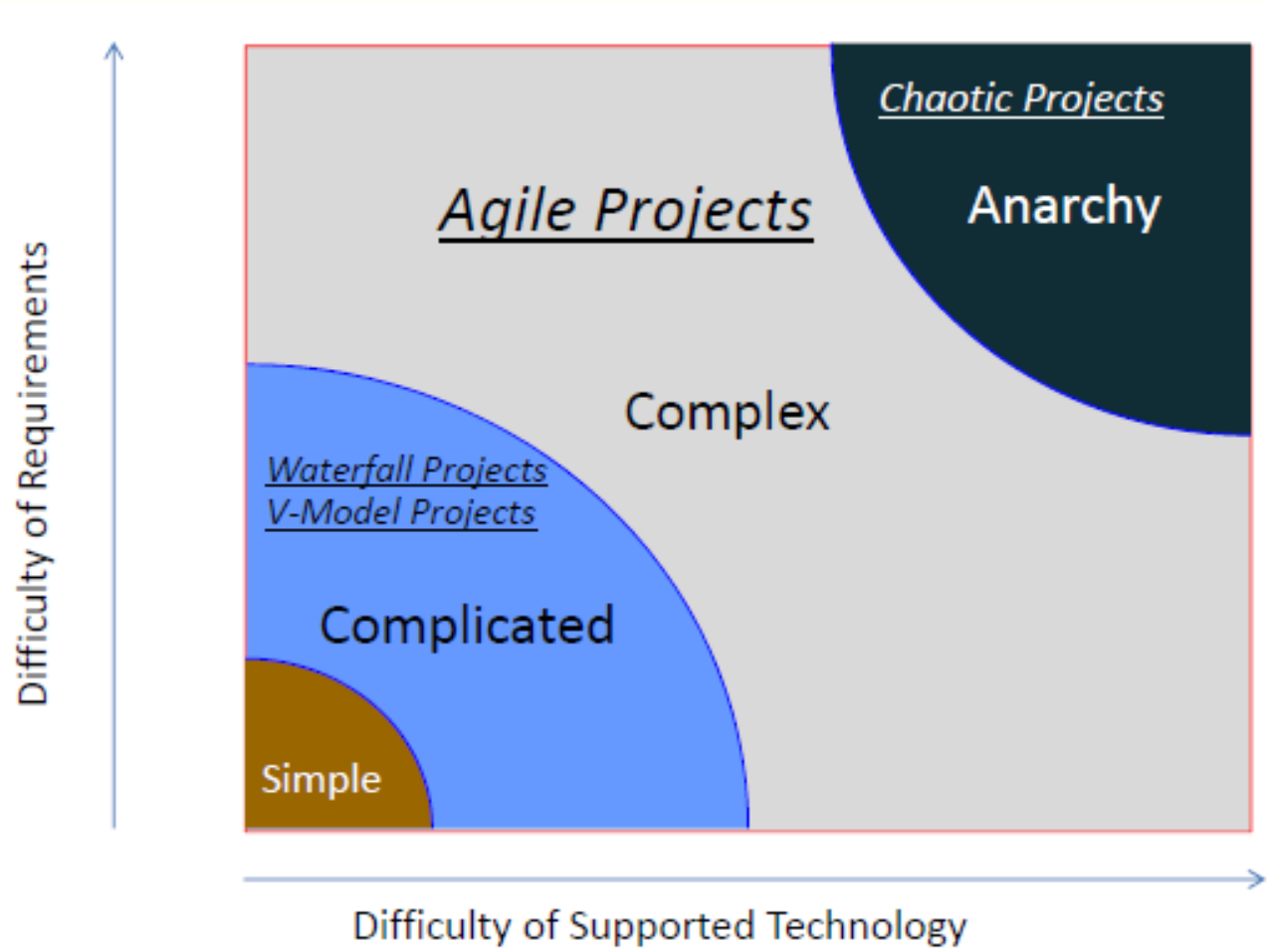


Application Architecture - Design Patterns

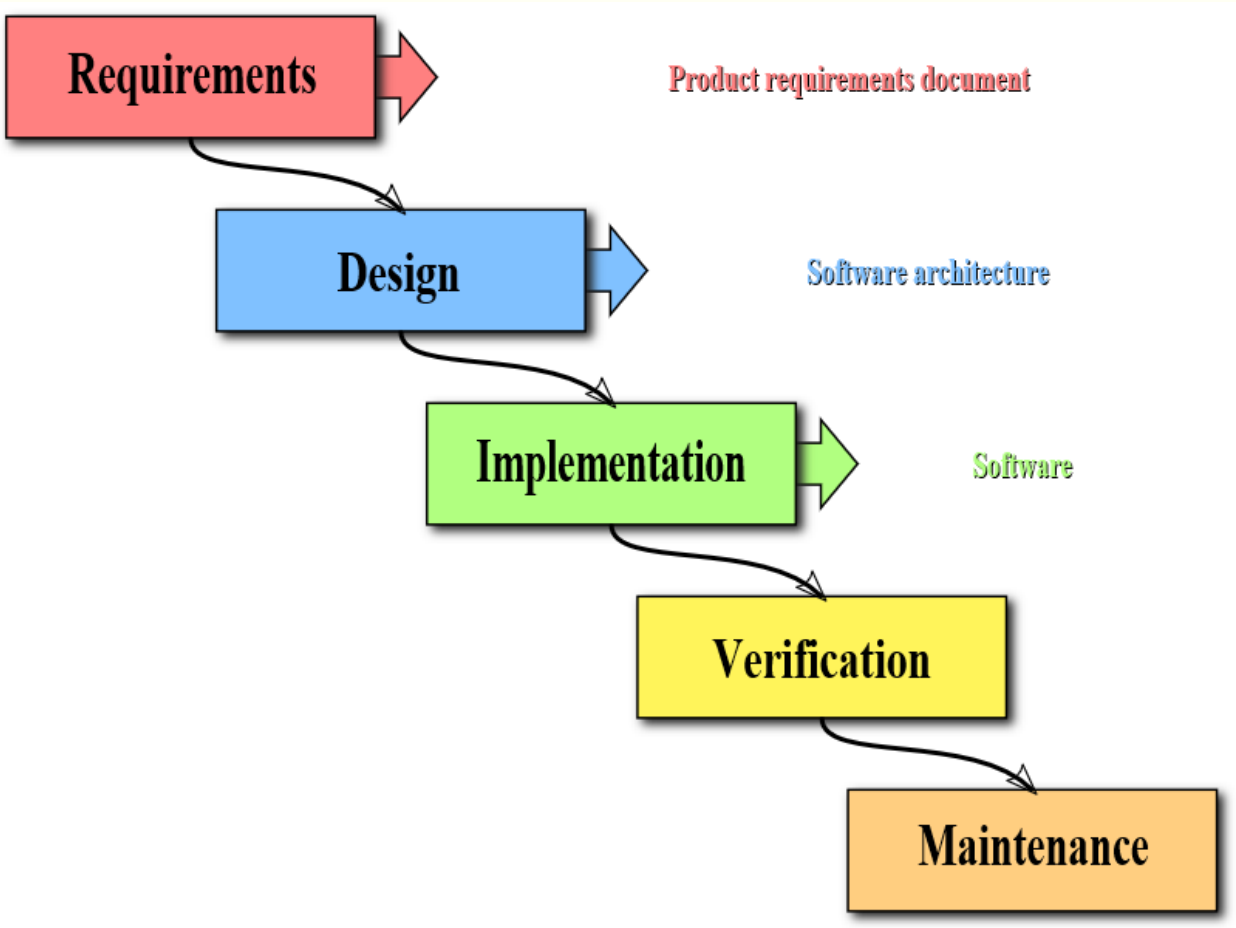
Packaged applications
A packaged application is a large-grained Commercial-Off-The-Shelf (COTS) product that provides a significant amount of capability (and reuse)
Enterprise resource planning (**ERP**) is a category of business-management software—typically a suite of integrated applications—that an organization can use to collect, store, manage and interpret data from many business activities,



Application Architecture - When to use which methodology



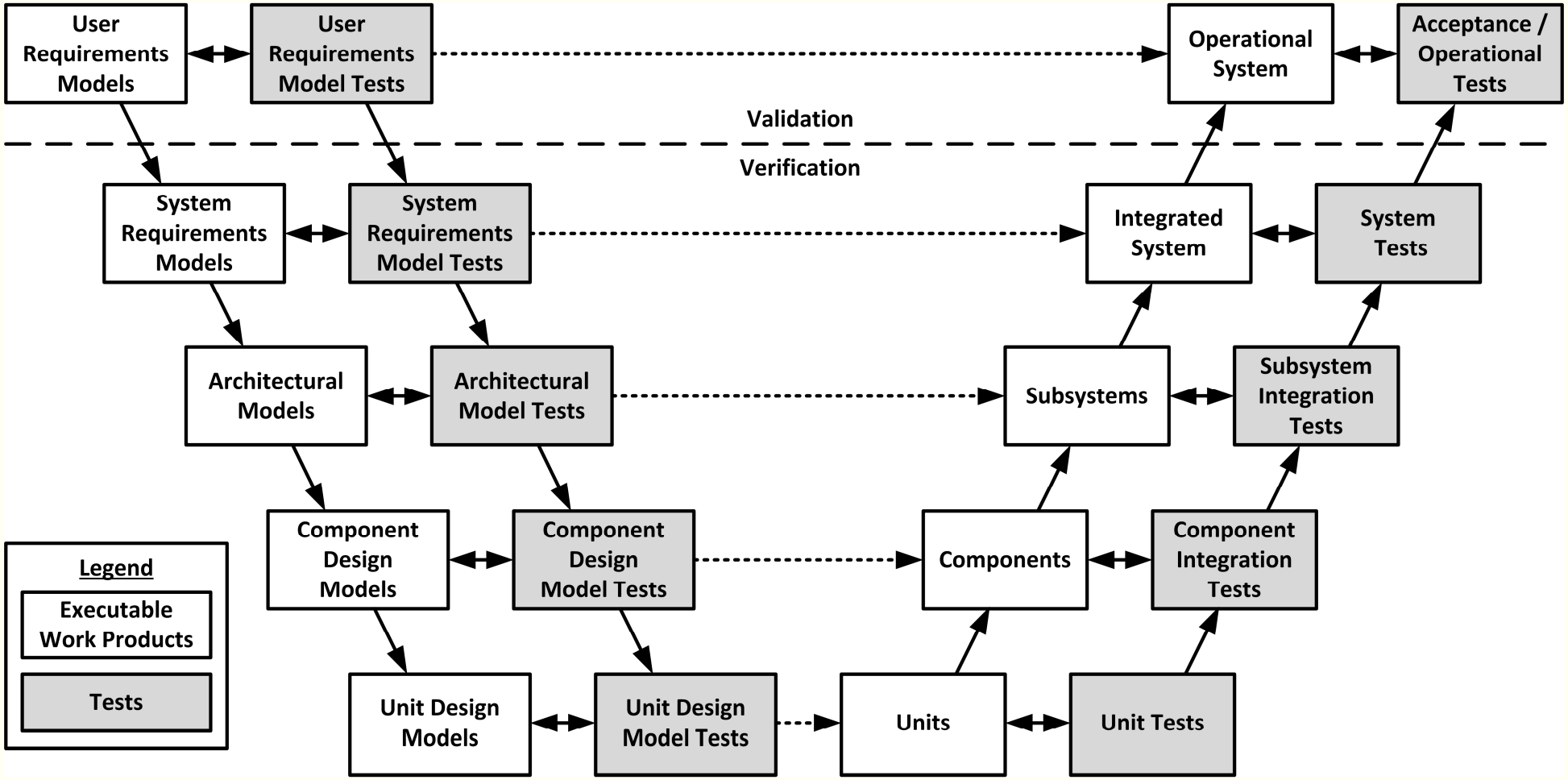
Application Architecture - Waterfall Model



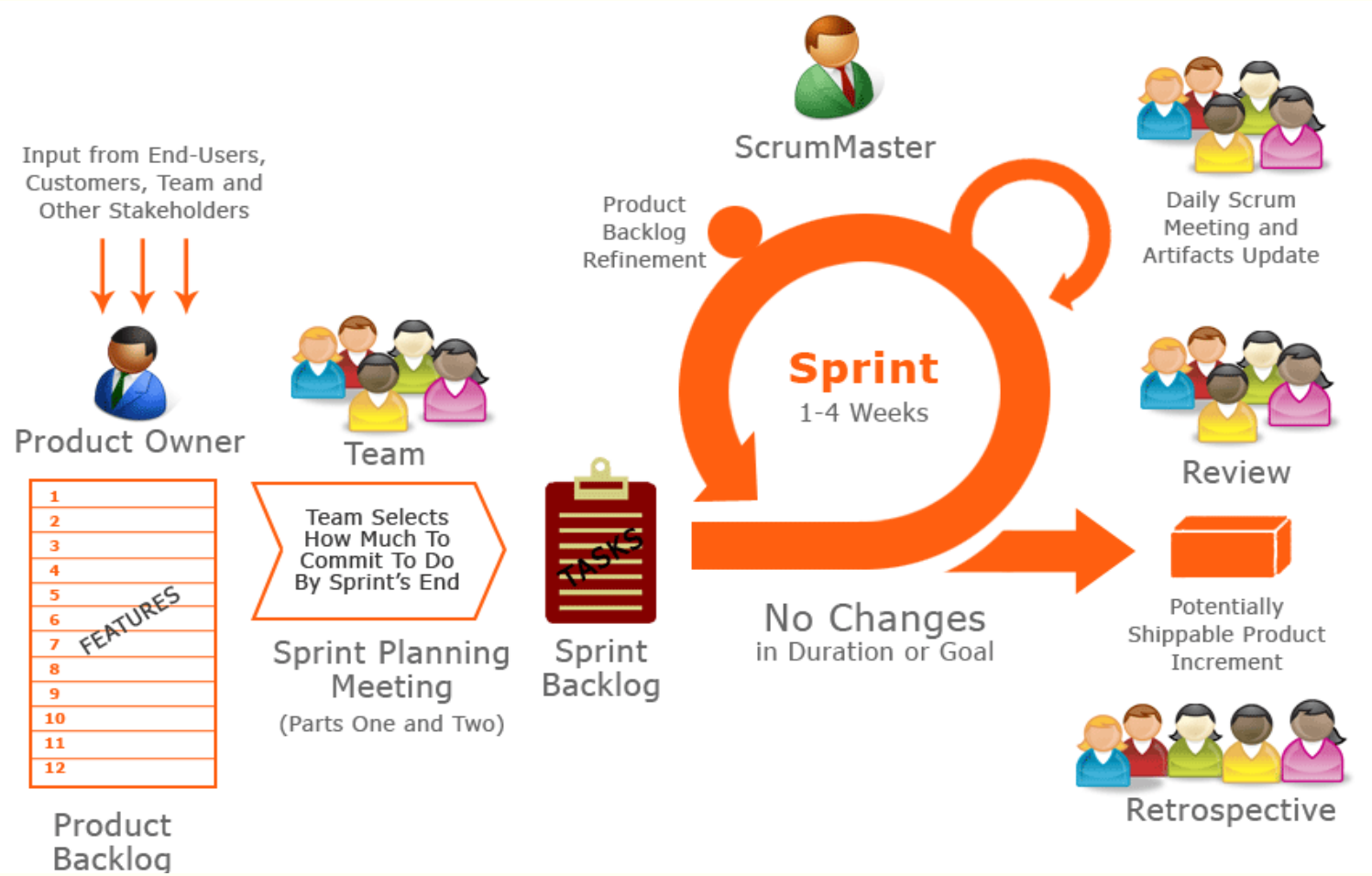
Sequential (non-iterative) design process

The waterfall development model originates in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible

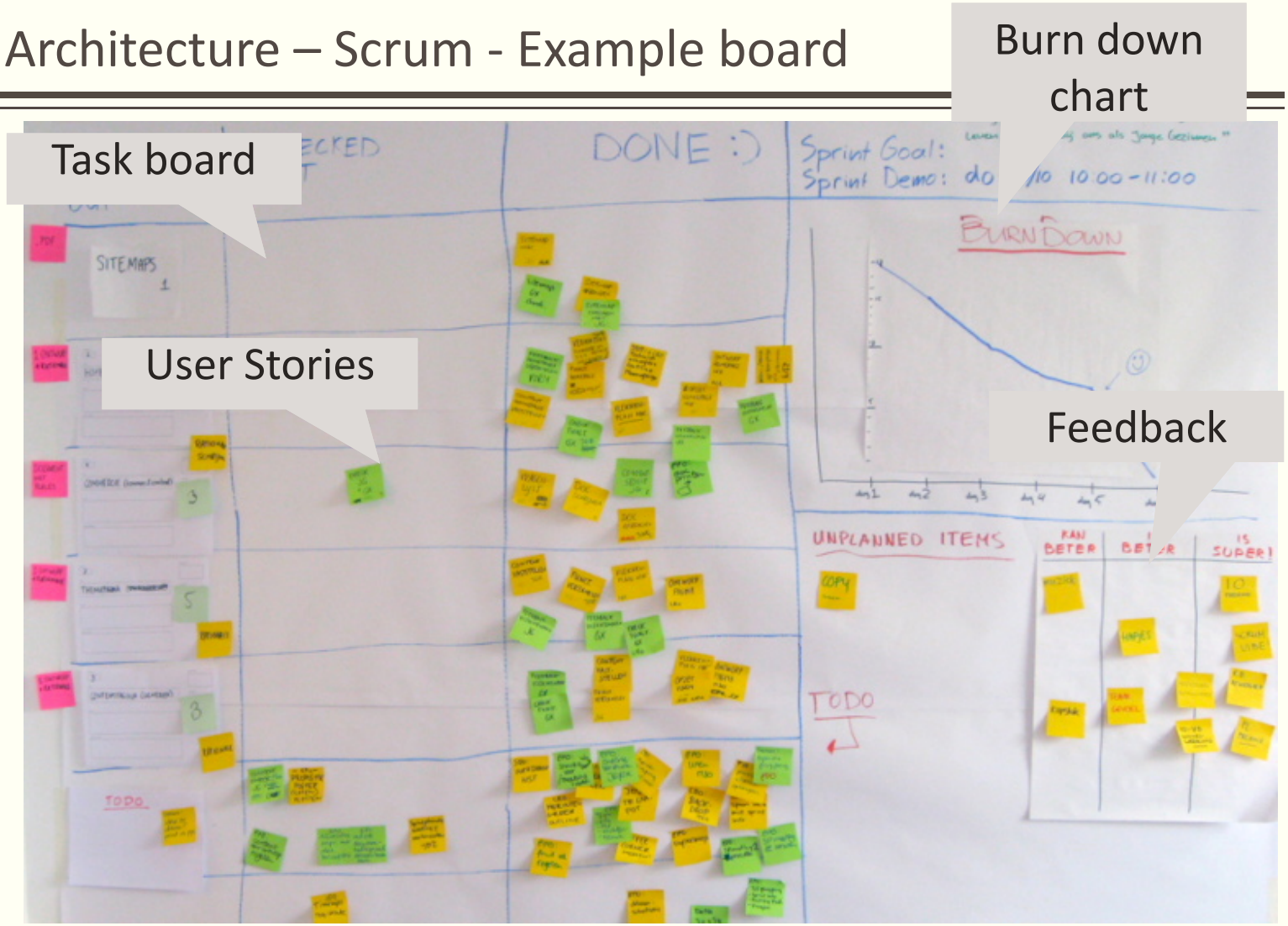
Application Architecture - V-Model



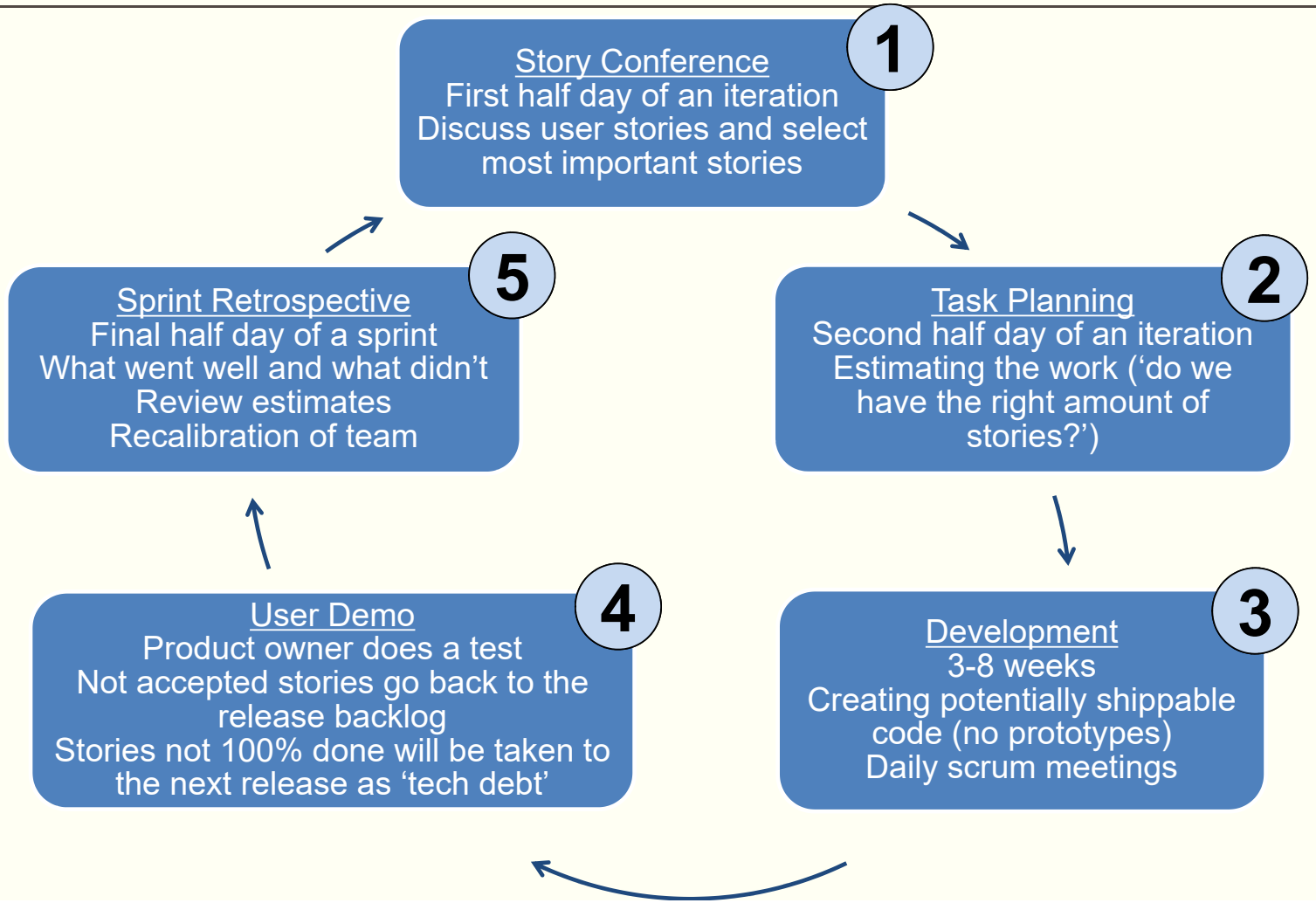
Application Architecture – Agile Model



Application Architecture – Scrum - Example board



Application Architecture – Scrum - Sprint cycle



Application Architecture - Anarchy

Application Architecture – ADM Maturity Model

0 Non-existent when

There is no process for designing and specifying applications. Typically, applications are obtained based on vendor-driven offerings, brand recognition or IT staff familiarity with specific products, with little or no consideration of actual requirements.

1 Initial/Ad Hoc when

There is an awareness that a process for acquiring and maintaining applications is required. Approaches to acquiring and maintaining application software vary from project to project. Some individual solutions to particular business requirements are likely to have been acquired independently, resulting in inefficiencies with maintenance and support.

2 Repeatable but Intuitive when

There are different, but similar, processes for acquiring and maintaining applications based on the expertise within the IT function. The success rate with applications depends greatly on the in-house skills and experience levels within IT. Maintenance is usually problematic and suffers when internal knowledge is lost from the organization. There is little consideration of application security and availability in the design or acquisition of application software.

Application Architecture – ADM Maturity Model

3 Defined when

A clear, defined and generally understood process exists for the acquisition and maintenance of application software. This process is aligned with IT and business strategy. An attempt is made to apply the documented processes consistently across different applications and projects. The methodologies are generally inflexible and difficult to apply in all cases, so steps are likely to be bypassed. Maintenance activities are planned, scheduled and coordinated.

4 Managed and Measurable when

There is a formal and well-understood methodology that includes a design and specification process, criteria for acquisition, a process for testing and requirements for documentation. Documented and agreed-upon approval mechanisms exist to ensure that all steps are followed and exceptions are authorized. Practices and procedures evolve and are well suited to the organization, used by all staff and applicable to most application requirements.

Application Architecture – ADM Maturity Model

5 Optimized when

Application software acquisition and maintenance practices are aligned with the defined process. The approach is component based, with predefined, standardized applications matched to business needs. The approach is enterprise wide. The acquisition and maintenance methodology is well advanced and enables rapid deployment, allowing for high responsiveness and flexibility in responding to changing business requirements. The application software acquisition and implementation methodology is subjected to continuous improvement and is supported by internal and external knowledge databases containing reference materials and good practices. The methodology creates documentation in a predefined structure that makes production and maintenance efficient.

Application Architecture – Testing