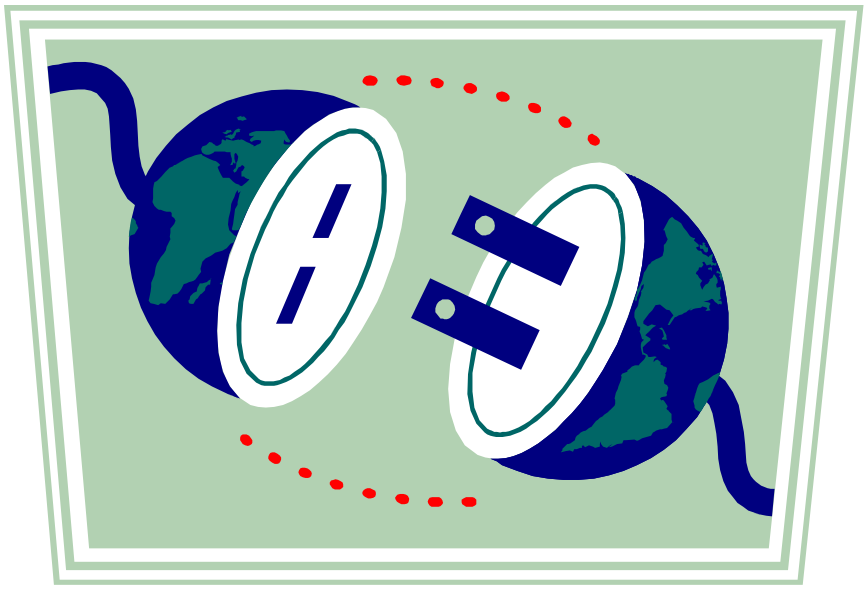
MAP
Model Driven Approach
An Overview



Kinetium Inc.

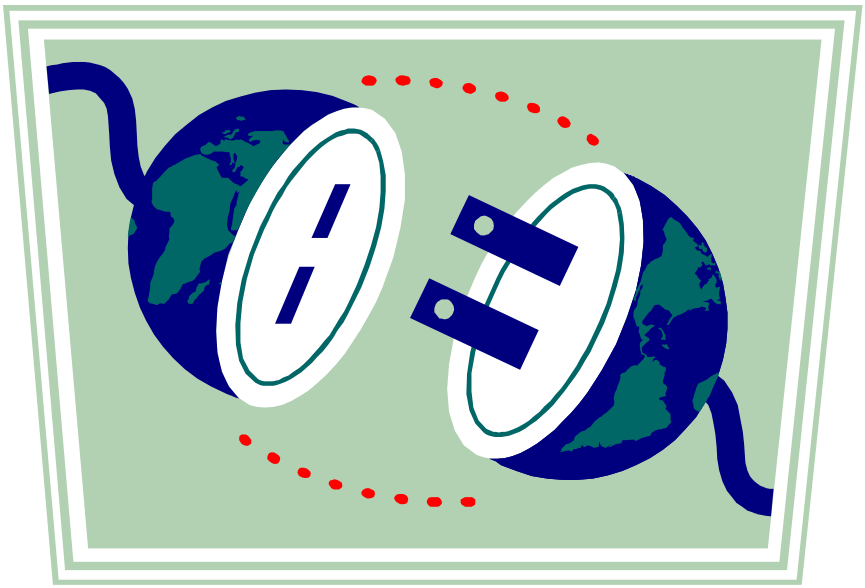
Version 2.1

(Back of cover sheet)

(1st page after cover sheet)

MAP

An Overview



Kinetium Inc.

Course Mechanics

- Let's start with introductions, background, expectations
- Please fill out the course sign-up sheet, put your name on name cards and on your course notes
- Let's agree planned classroom timings, breaks, etc.
- What are the facilities around the classroom?
- Some slides instructor will fill in the blanks
  - By editing into the slide itself, on a flip chart, etc.
  - Please fill important information into your course notes slide
- The style for class labs is
  - Work in a team for every lab
  - Mix business, application, infrastructure people in each team
- Please fill the course evaluations before leaving

Course Objectives and Approach

- **Objectives**
  - Obtain a good end-to-end view of MAP
  - Understand MAP architecture description and process
  - Be familiar with the kinds of deliverable produced
  - Understand the kind of flow and appropriate planning

- **Non-Objectives**
  - Make you an expert in architectural design
  - Teach project management and lifecycle aspects
  - Cover broad range of architectural styles and patterns

- **Approach**
  - Start with basics of MAP
  - Get high-level overview of the method and modeling
  - See our toolkit to model different architectural views
  - Work out a few “list-level” models as exercises

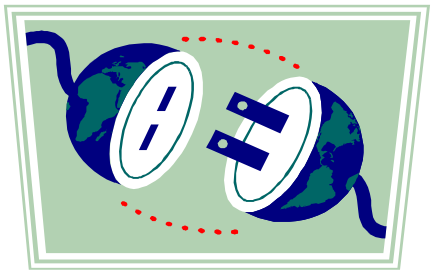
Where does this course fit in?

- Recommended path to applying MAP includes
  - Project planning and definition including MAP tailoring
  - MAP overviews (2 hour – 1 day)
  - MAP in-depth training course (4-5 days)
  - Project focused workshop (1-2 days)
  - MAP mentor working with team on their first project

- Other resources on (or on their way to) the Kinetium site include
  - Method guide and templates
  - Summary reference sheets
  - Detailed documentation
  - Tool customizations

MAP (for Roadmaps)

An Overview



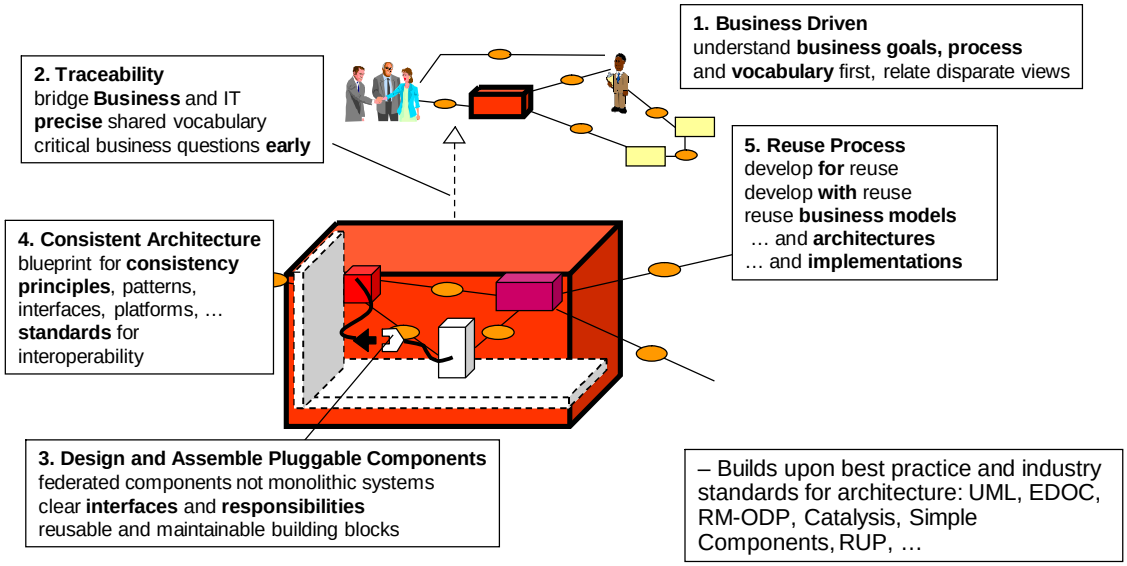
Kinetium Inc.

Outline

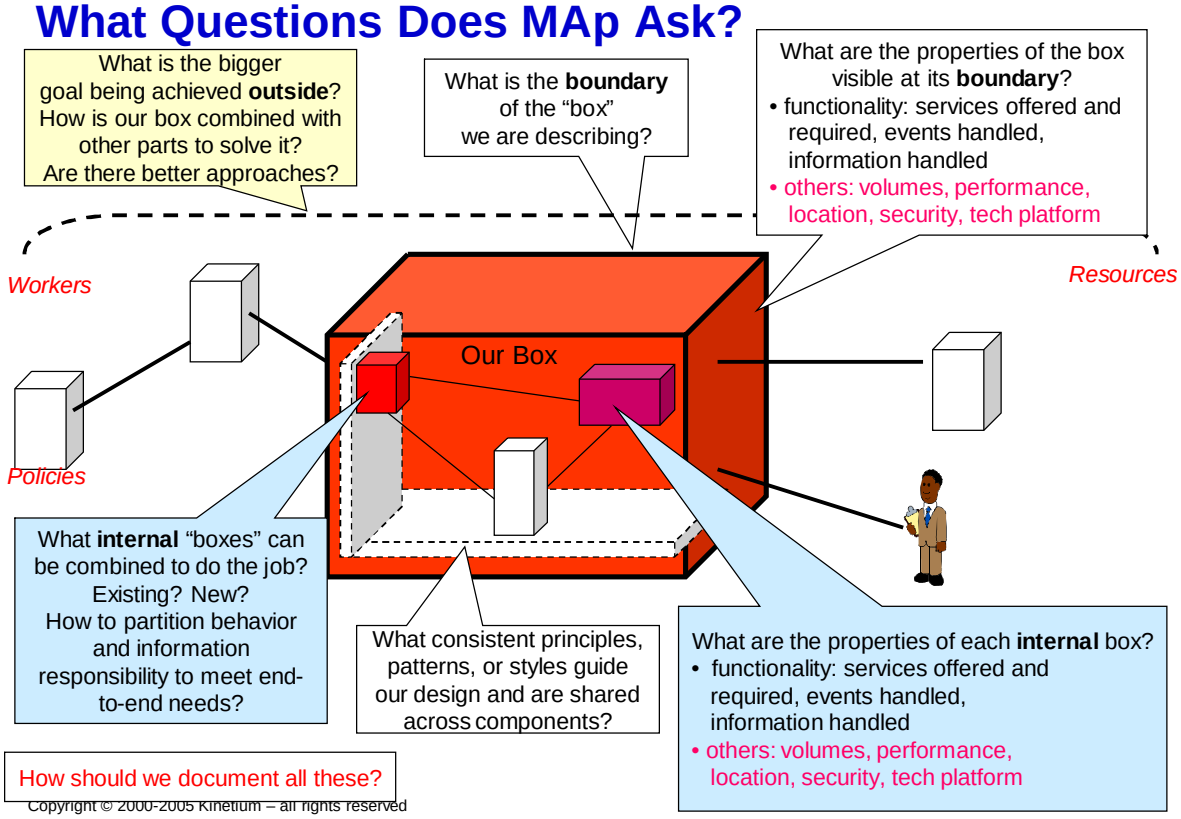
- **Introduction to MAP: What it is, Process and Artifacts**
- Models, Architecture, and Objects
- Goals and Domain Model
- Current State Architecture
- Target State Architecture
- Migration Plan
- Project Management

What is MAP?

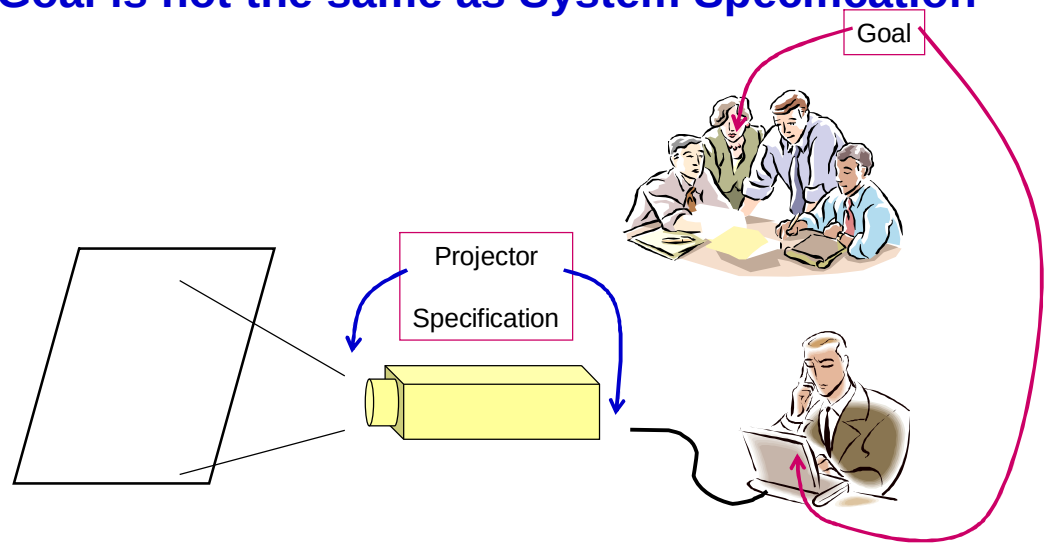
- MAP – Model-Driven Approach
- MAP is a systematic approach to **architect**, plan, develop, and evolve software systems
  - Clearly separated viewpoints and models to design or analyze an architecture



What Questions Does MAp Ask?

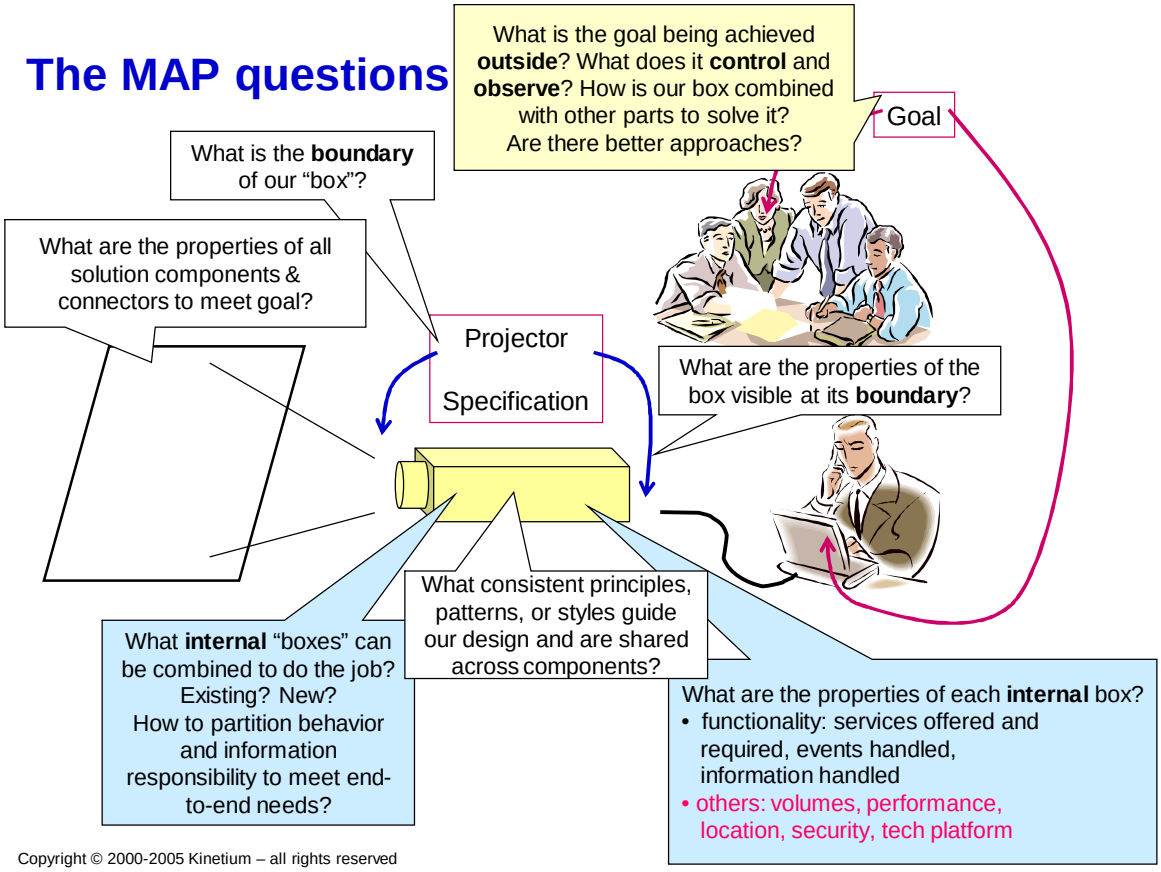


Goal is not the same as System Specification

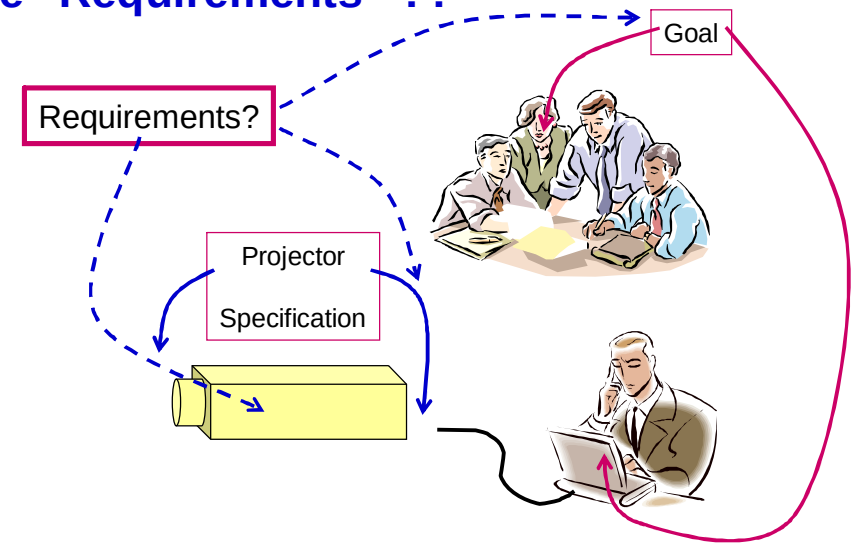


- Goals are typically 1, 2, or more levels "removed" from the system
  - Goal **observes** certain things in the "problem domain" to **control** certain other things
- Target system + other elements + domain properties needed to meet the goal
  - Target system = projector
  - Other elements = screen, cables, laptop, speaker, audience, room
  - Domain properties = audience co-located with speaker, room size adequate, …

The MAP questions



Where are “Requirements” ??



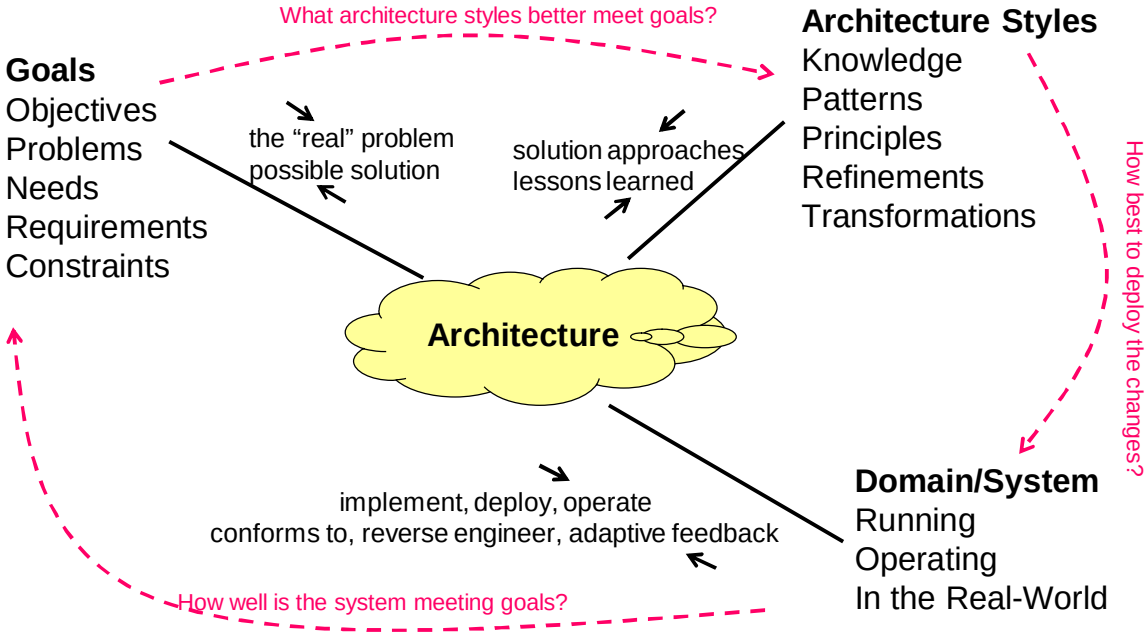
- “Requirements” is frequently used to cover a variety of different things
- Most frequently used for “Specification” of one system
- MAp recommendation: be clear which one you mean

7

Exercise 1

8

Architecture in Context



The architecture of a system is a key determinant of how well a running system will meet business goals

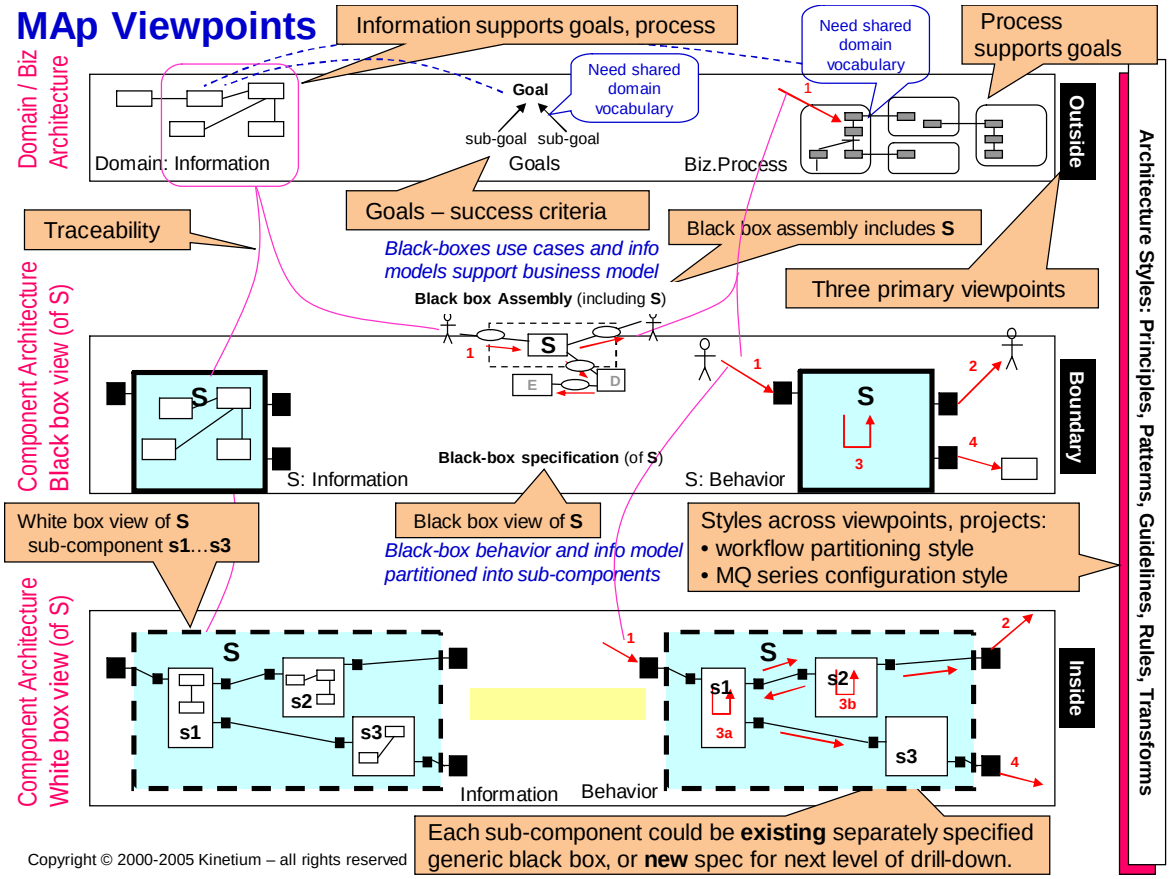
Two Aspects of the MAP Framework

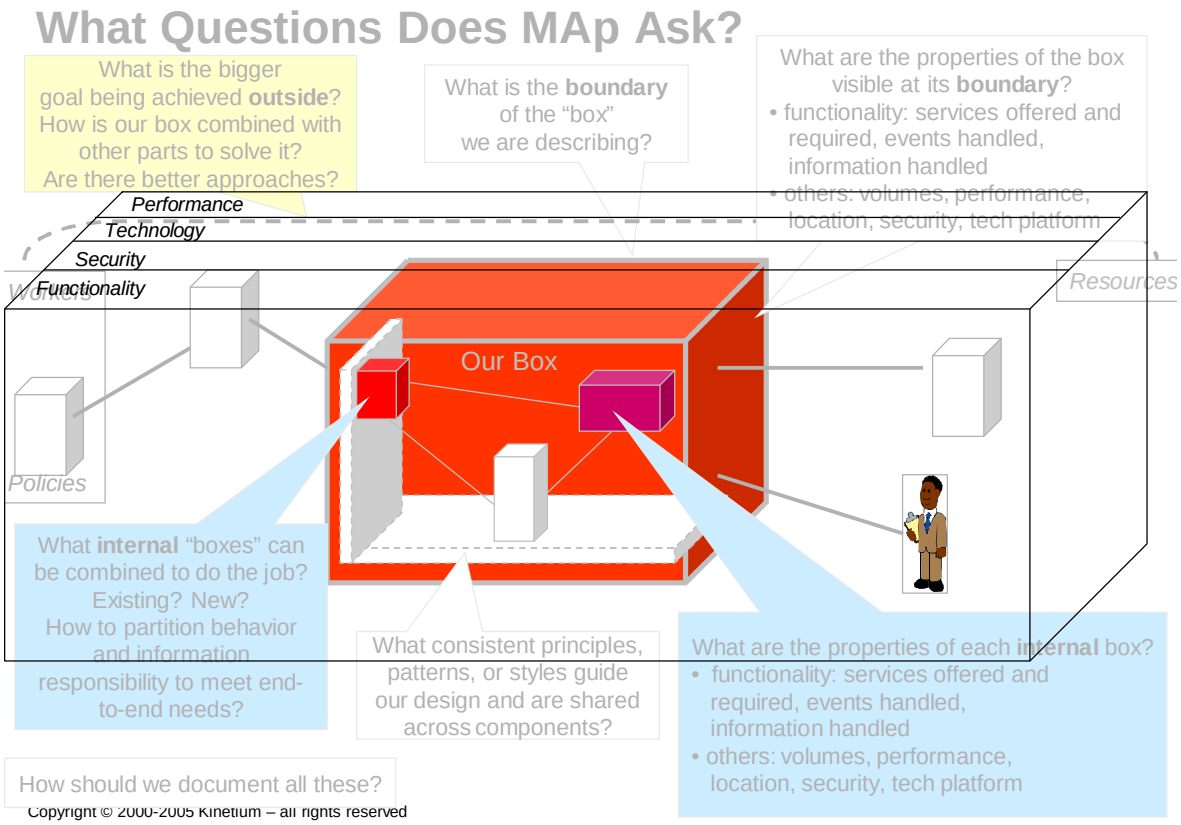
Formal Architecture Framework

- Goals
- Viewpoints
- Concerns
- Models
- Refinement
- Architecture Styles

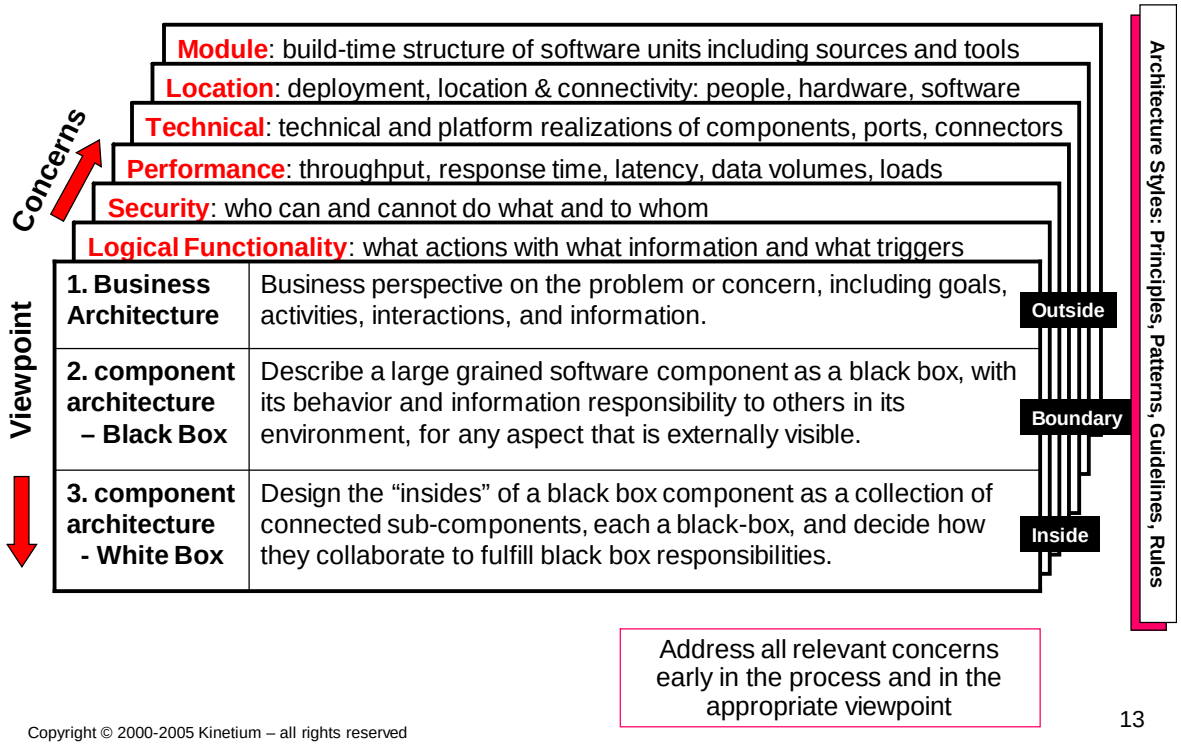
Process Pragmatics

- Routes: Roadmap, Construction, …
- Different Entry Points
- Iterative and Incremental
- List, Draft, or Dressed Templates
- Guidelines





Multiple Concerns, Styles, across Viewpoints

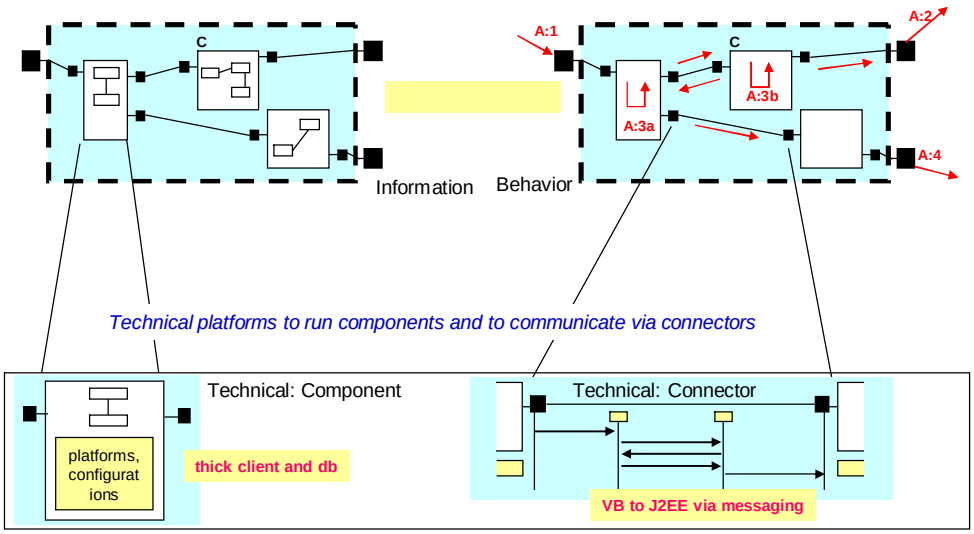


Address all relevant concerns early in the process and in the appropriate viewpoint

13

Technical Architecture – Complex "Overlay" at any Level

White-Box: Logical Components and Connectors



White-Box: Technical Realization of Components and Connectors

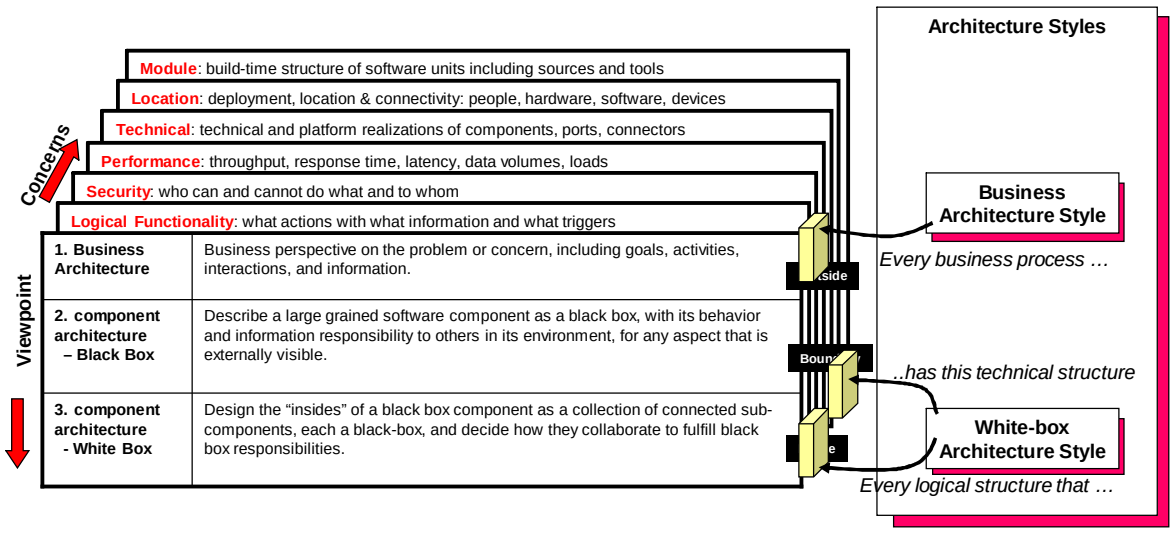
14

Architecture Principles and Styles

- Architecture styles define patterns of preferred or allowed architectures
- One common form: **Wherever** you have … **solve it by** …
- Very wide spectrum in (and across) all viewpoints
  - Earliest point of data capture
    - **Wherever** any process needs some information at multiple points …
    - Capture the information at the first opportunity in process. Never require it again.
  - MQ Series Configuration Style for Co-located Queues
    - **Wherever** co-located components need a messaging connector …
    - For a source S of events and a destination D that will be co-located on a single machine, use an MQ Series queue named <S><D><…>, configured as <….>, and set up the S and D ends of the queue as <….>
  - XML encoding styles
    - Use nested entities for …; use attributes for …; compress names as follows …
  - Integration styles: point-to-point, desktop, gateway, …

15

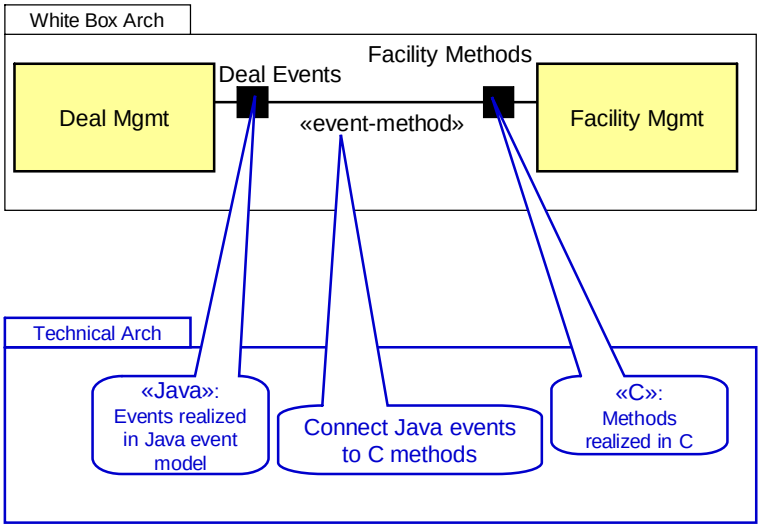
Architecture Styles can Span Viewpoints, Concerns



- Each style can be about just business, or white-box logical, or …
- … or about correspondence or even transformations across viewpoints

16

Technical Architecture Style – cross-concern pattern



White Box Arch Styles
An **event** is specified by:
- event name and event information
- condition for event to be raised

An **event method** connector:
- takes a raised event **e (T)**
- takes a corresponding method **m(x,y)**
- takes a mapping from **T** to **x, y**
- ensures **m** invoked when **e** is raised

Technical Arch Styles
To realize event **S::e (T)** in Java:
- method **S**::add_**e**_Listener (x)
- method **S**::remove_**e**_Listener (x)
- call **S**::notify_**e** (T)

To realize method **A::m (x, y)** in C:
- procedure **m** (**A**_id, **x, y**)
- Map ids, parameter types: …

To connect Java event to C method:
- connect Java event to Java method
- call C via Java Native Interface

- Map component spec to platform: platform mapping of methods? procedures? mainframe transactions? relational data?
- Map operations, parameters, failures to platform naming, parameter passing, and exceptions: refs, structs, XML, files?

Specify Vs. Design … "Turtles All The Way Down"

- At the top: models of highest-level goals and "intrinsic" domain model
- "Business-Design" starts with sub-goals
  - Subgoals may involve choice among alternative solution approaches
  - Business processes partition and co-ordinate responsibilities
- "Architectural-Design" starts with top-level components and connectors
  - Largest-grained components own or need different parts of the domain model
  - Continues to sub-components, sub-sub-components, etc.
  - Logical view vs. Technical realization at any level

Two Aspects of the MAP Framework

Formal Architecture Framework

- Goals
- Viewpoints
- Concerns
- Models
- Refinement
- Architecture Styles

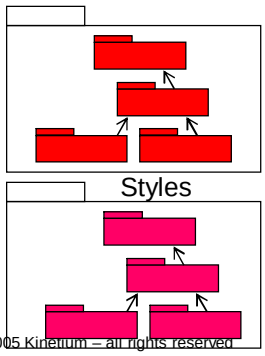
Process Pragmatics

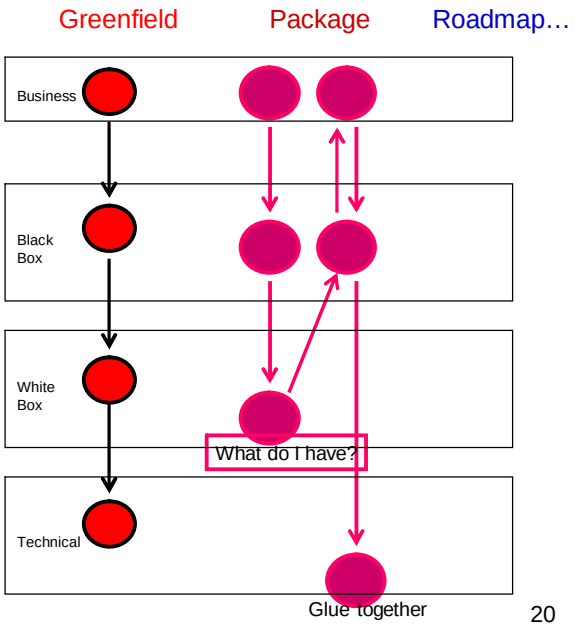
- Routes: Roadmap, Construction, …
- Different Entry Points
- Iterative and Incremental
- List, Draft, or Dressed Templates
- Guidelines

19

Different Project "Routes"

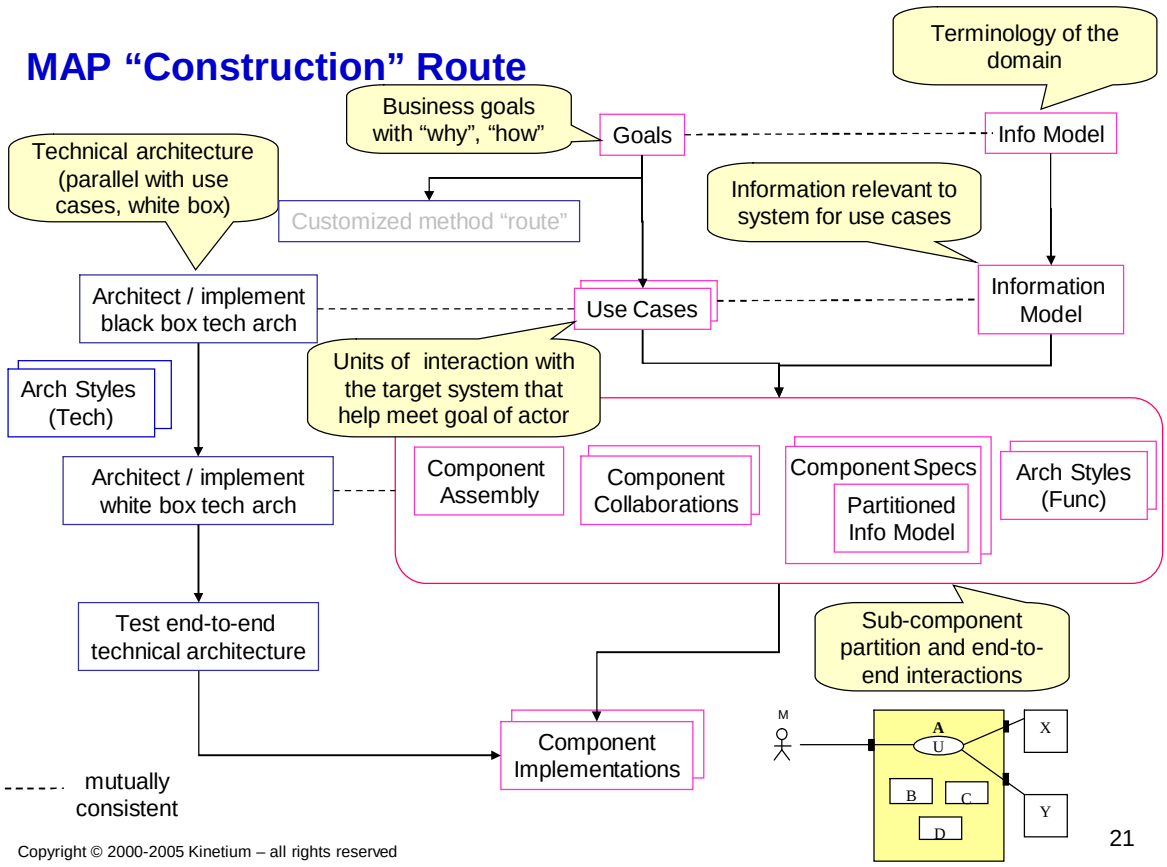
- Business, Black box, White box, Technical, … Architectures
  - For a project, which deliverables developed when and how far?
- Different projects, different "routes"
  - Different paths through deliverables
- Different architecture **styles**
  - … applicable for different projects





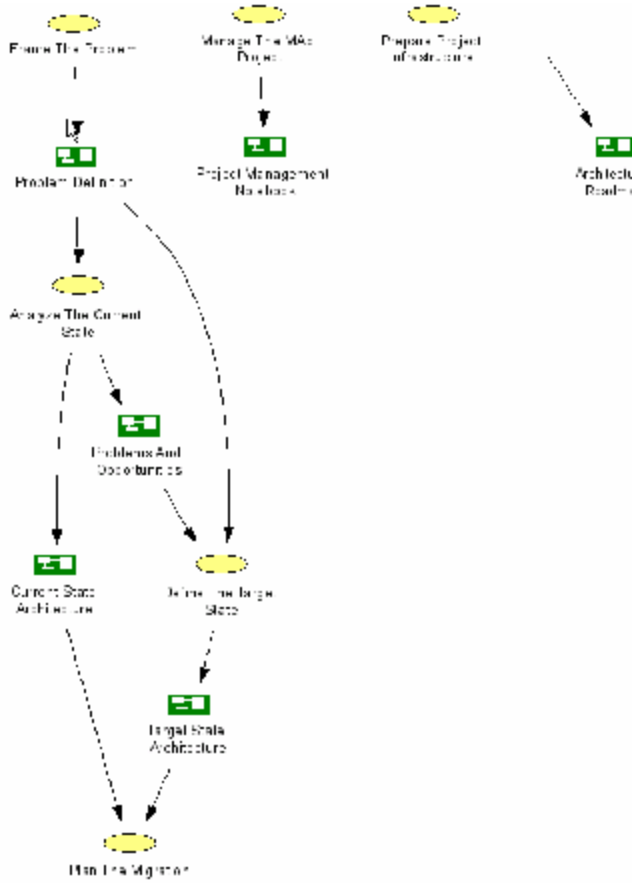
20

MAP "Construction" Route



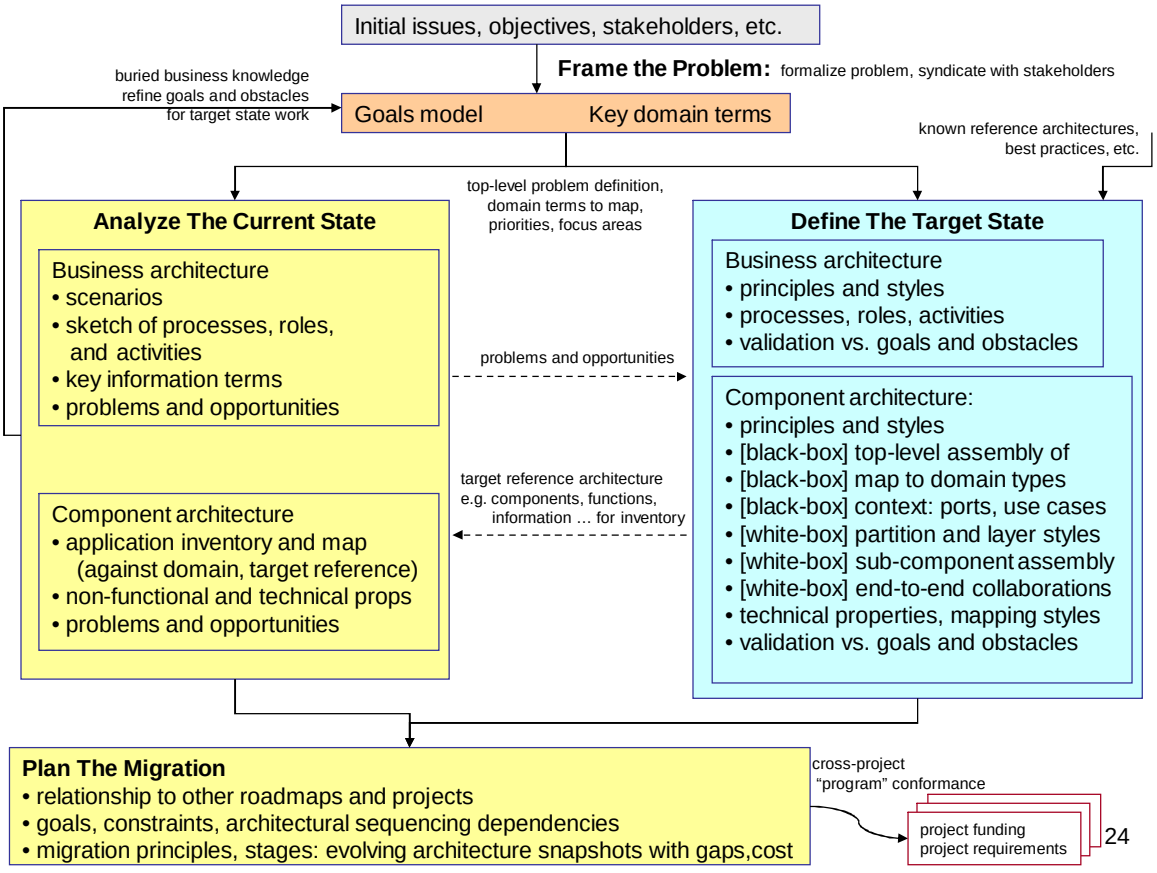
MAP "Roadmap" Route

- Overall flow shown
- Multiple entry points
- Iterative
- Incremental



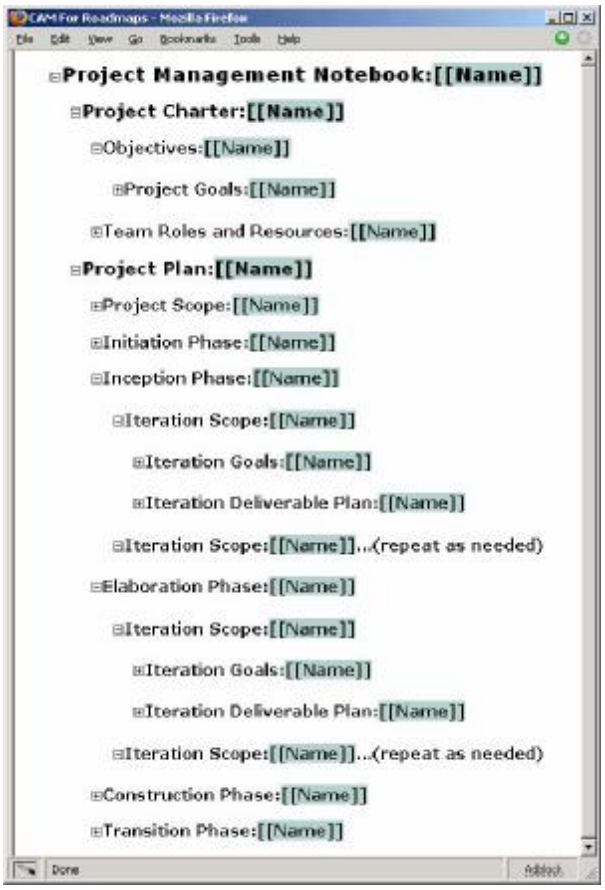
MAP Template

- Overall structure shown
- Model-driven
- Models can be list, draft, dressed



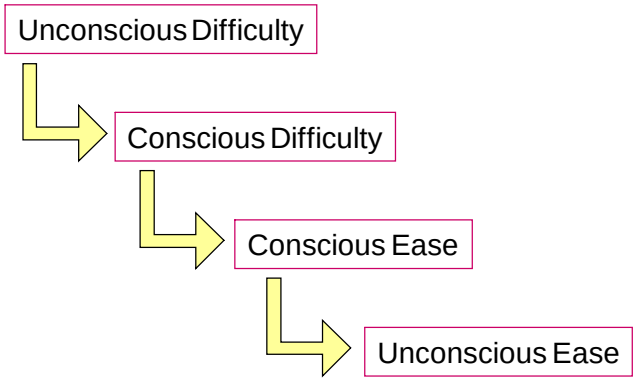
MAP Planning

- Iterative, not waterfall
- Emphasis shifts across phases
- List **à** draft **à** dressed artifacts
- High-risk items in early iteration



Migrating to MAP

- MAP can be applied as rigorously or lightly as appropriate
- The underlying concepts of MAP are few and simple
- Applying it effectively takes a focused mindset
- Classic migration path …

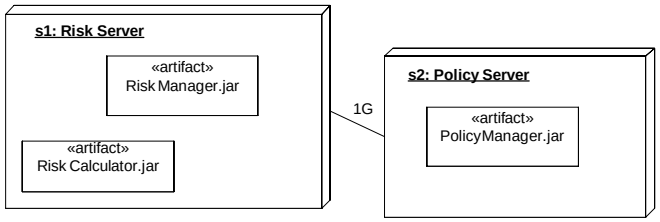


Exercise 2

Outline

- Introduction to MAP: What it is, Process and Artifacts
- **Models, Architecture, and Objects**
- Goals and Domain Model
- Current State Architecture
- Target State Architecture
- Migration Plan
- Project Management

What is a Model?



Nodes
s1: Risk Server
s2: Policy Server

Artifacts

Risk Manager.jar	s1
Risk Calculator.jar	s1
Policy Manager.jar	s2

Links

s1:s2	1G

- **Model**: A structured **description** of selected properties of a **domain** or **system**, using some combination of textual or graphical formal **language** with informal natural language

- There can be more than one notation for the same abstract language
  - Nodes have
    - artifacts deployed on them
    - communication links with bandwidth

Observations about Models

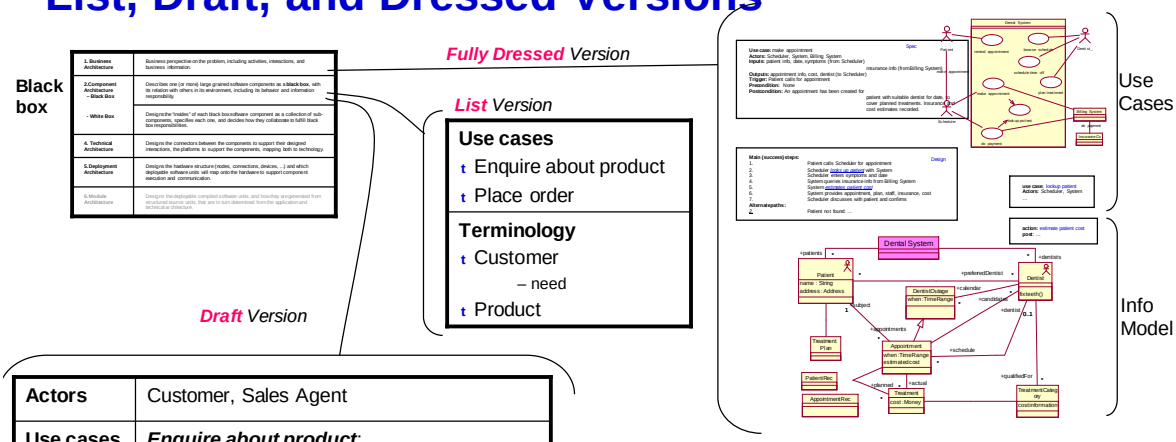
- Model ≠ Picture
  - Structured text, tabular presentations very common and very useful

- Models include formal, semi-formal, and informal pictures, text, …

- The model can exist in many forms:
  - paper and pencil
  - a modeling tool artifact
  - a word processor or xml document
  - a white board
  - spoken words
  - in someone's head (a bit of a stretch!)

- We specifically want models to progress through:
  - **List:** bulleted lists of names, optionally with nested bullets for properties
  - **Draft:** tabular information with name + description of each element
  - **Dressed:** full diagrams with accompanying narrative text

List, Draft, and Dressed Versions

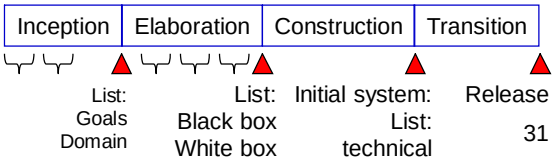


Actors	Customer, Sales Agent
Use cases	***Enquire about product***: **Goal:** The system has recorded customer's need for product. Callback has been scheduled, cross-sell analysis initiated.
	Place Order: ….
Glossary	**Customer**: a legal entity who … *need:* the set of likely future purchases **Product**: A unit of sales to customer …

- Depth, detail, and form to suit context
- Don't produce fully dressed all at once

Iterative MAP

Iterations within phases



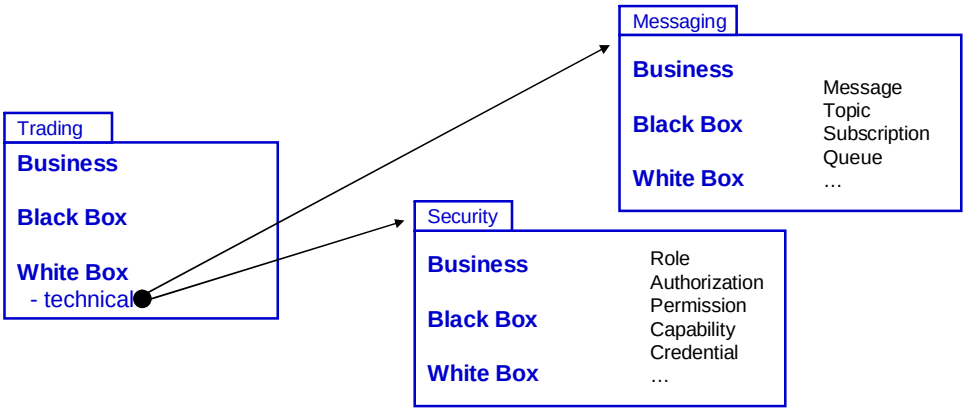
31

What is a Domain or System?

- **Domain / System:** Some part of the world (as-is, or could-be) relevant to understanding or solving a problem; the subject matter of our problem or goals.
- The "Domain" or "System" of interest could be any of:
  - A software application or suite + relevant aspects of its environment
    - *The Risk Management Application*
  - A business unit or process + relevant aspects of its environment
    - *The overall Risk Management process, people, organization, roles, and policies*
    - *The crisis_management phase of the overall Risk Management process*
  - A technical infrastructure environment
    - *The Network domain of nodes, connections, routers, firewalls, zones*
  - A collection of components and their intra- and inter-component interactions
    - *How the Policy Manager, Risk Calculator, and Hedge Manager collaborate*
- **Scope of domain** (e.g. for the "projector")
  - Where: how "far out" are the things we must understand and address to meet goals
  - When: which parts of the lifecycles do we care about
  - What: what aspects or concerns are relevant

32

Technical Domains and Business Domains

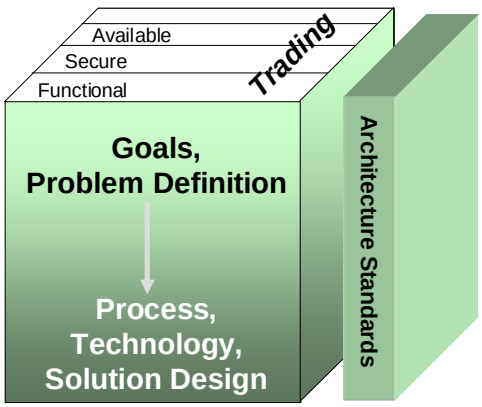


- Infrastructure or technical domains (e.g. Security, Messaging) have their own "business" or domain models, black box, and white box models
- These can be developed, to some extent, in parallel with the primary business domain (e.g. Trading) … but remembering that usage defines requirements
- When Trading is designed or implemented in technical or platform terms, that platform is defined by the technical domains

33

The MAP Approach To A Domain

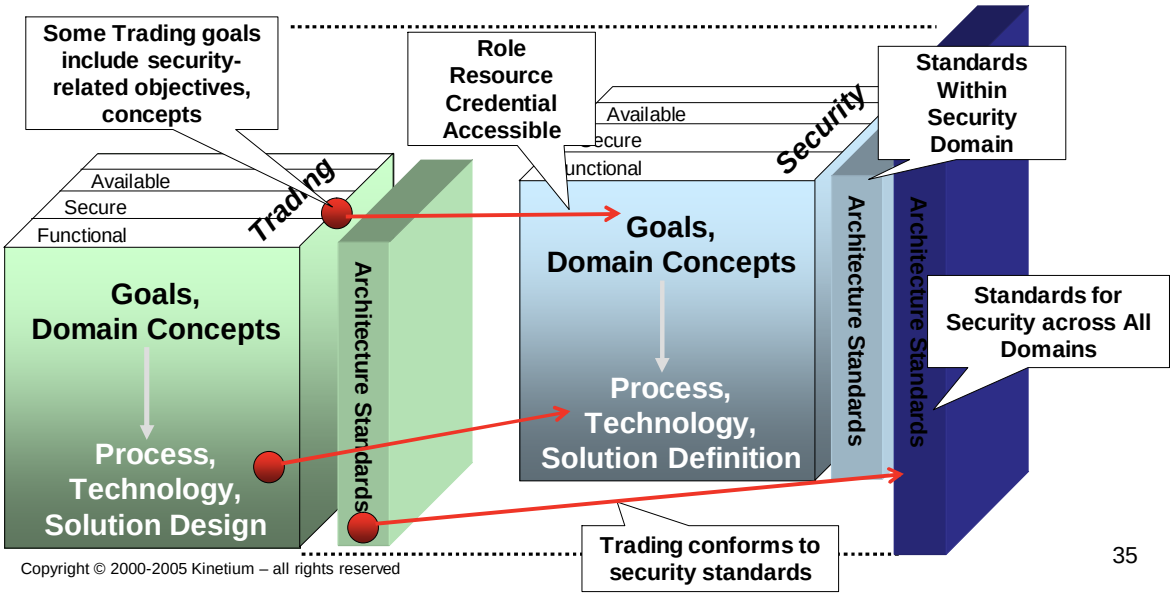
- **MAP** describes a domain across different architecture viewpoints and concerns
  - Goals are an explicit and essential part of the architecture
  - Both problem domain and solution architectures are precisely defined
- Architecture styles define
  - acceptable architectures
  - conformance
  - targets for governance



34

Domains Leverage Supporting Domains

- One Domain uses all the knowledge of other specialized domains to:
  - Define its problem: precise **Security** goals help define security goals of **Trading**
  - Design its solution: standard **Security** services used in **Trading** implementation
- Architecture styles and standards at different levels span domains



35

Example: MAP on SOA Domain

Goals and Problem Definition

- Define SOA and its goals
- Clearly define SOA concepts

Solution: Processes

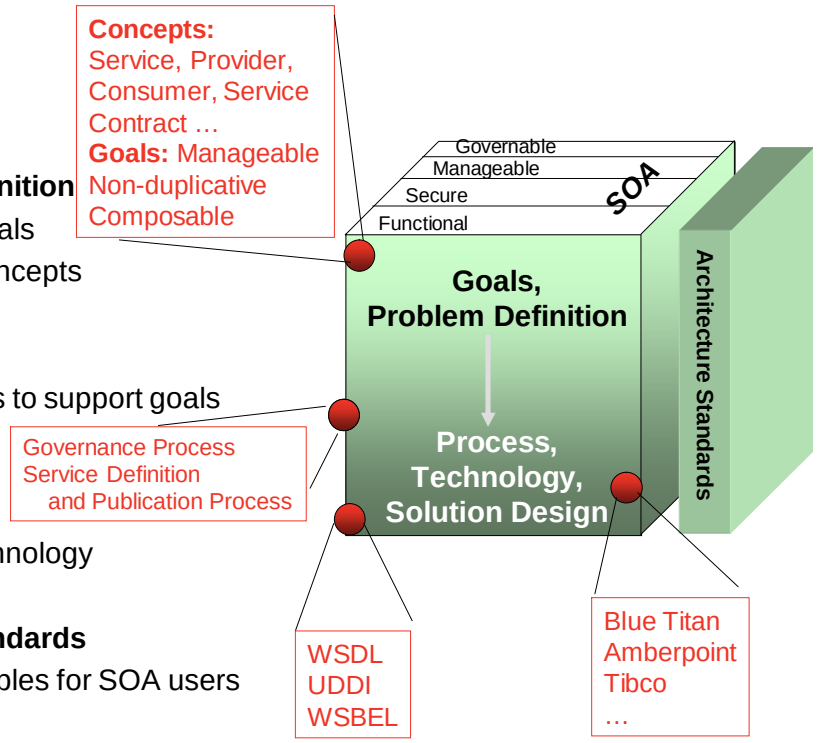
- Define SOA processes to support goals

Solution: Technology

- Design supporting technology

Define Architecture Standards

- Patterns, styles, principles for SOA users



36

What is Architecture?

- **Architecture:** An architecture of a system from a **viewpoint** is
  - a (more or less correct) **model** of that system
  - using the **language** of that viewpoint
  - that describes system properties that are of **concern** to that viewpoint
    - Properties explicitly described in the model, or
    - Properties that can be objectively inferred from the model

- Viewpoints are defined based on the kinds of questions they help answer
  - Services: what services are provided and required? interactions?
  - Deployment: where are deployable units run? what is connectivity?
  - Production: how is system installed? migrated? administered?

- The language is associated with the viewpoint
  - So all systems' deployment architectures are described in the same language

- Viewpoint definitions should be standardized across systems
  - Architecture styles take advantage of consistent viewpoint separation

- Architecture captures key decisions and eliminates needless creativity in the further design, implementation, and evolution of the system

Objects

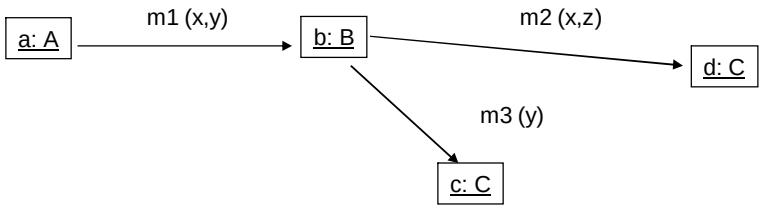
- Objects pervade our models, but not in the sense of OO-programming

  - An object is an individual thing we care about in some domain

- What we model about an object depends on:
  - Its nature
    - Active behavioral objects: software systems, components, roles in processes
    - Information objects: things stored or passed around between active objects

  - Our purpose
    - Are we focused on functionality?
    - Security?
    - Location and distribution?
    - Technology choices?
    - Product and tool choices?

Scenarios – Collaboration Diagrams

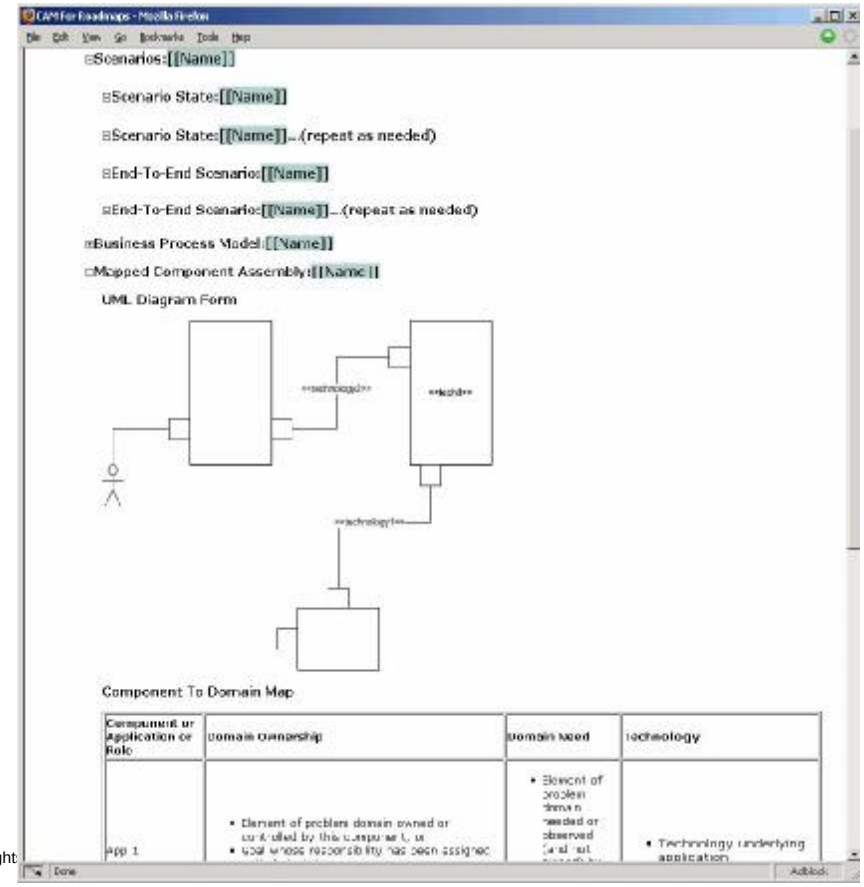


- a, b, c: objects “active” in the interaction
- x, y, z: objects manipulated or passed around

Exercise 3

Discuss

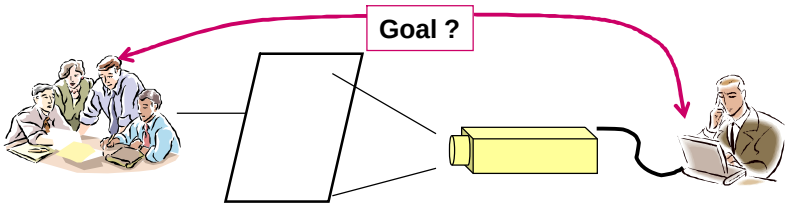
- Which parts of the MAP template have we just visited?
- Would this be useful for current-state? Target state?
- How would we know if we are exploring the right scenarios? Covering the right domain?



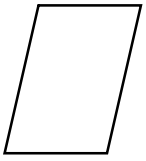
Viewpoints – What would a design review do?

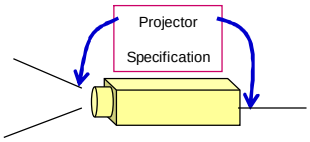
Problem Definition::
Goal +
Domain Model

Black-Box::
Assembly



Black-Box::
Specification (of one box)





Black-Box::
Specification (of one connector)

White-Box::
Internal Design as assembly of smaller parts

White-Box::
Do connectors have specs?
Implementations?

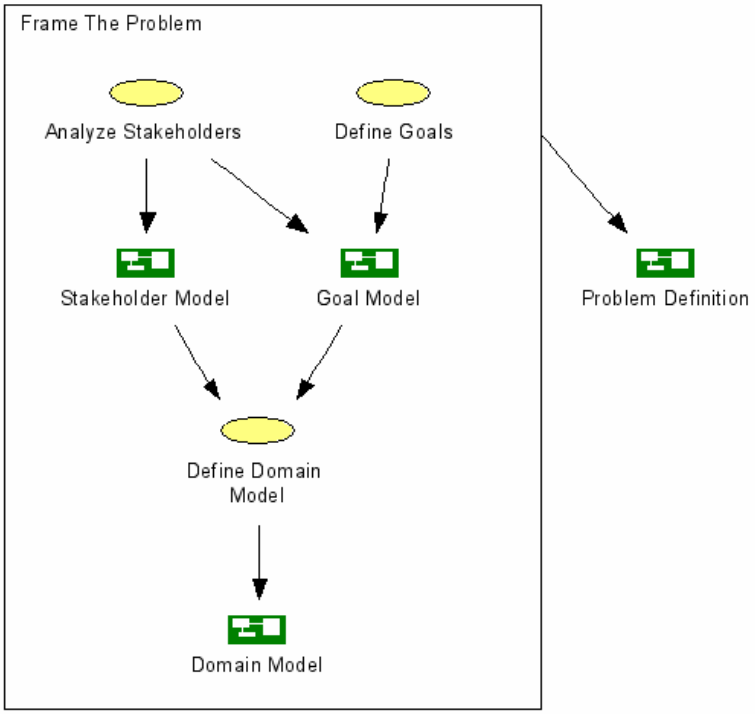
MAP Viewpoints: An Example Close To Home

- Let's take a look at **MAP** applied to ID&EM

Outline

- Introduction to MAP: What it is, Process and Artifacts
- Models, Architecture, and Objects
- **Goals and Domain Model**
- Current State Architecture
- Target State Architecture
- Migration Plan
- Project Management

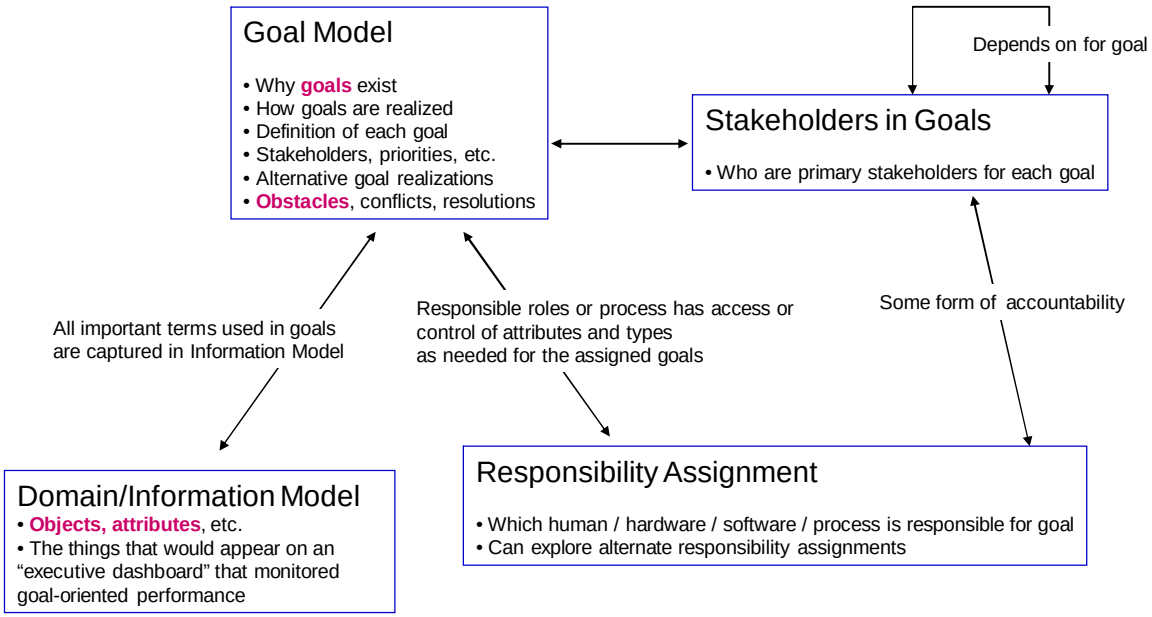
Frame The Problem



Main Elements of a Domain Model

- A **Domain Model** consists primarily of a model of
  - Types of objects
  - Their attributes, including relationship to other objects
  - Events involving those objects and attributes
  - Static + Dynamic constraints on the attributes and relationships
- An **Information Model** (types + attributes) often covers most of a Domain
  - One modeling technique
    - Model an event as an object
    - That object only appears in one snapshot: right after the event took place
- The basic approach to building an information model is
  - Consider domain terms used e.g. in goals, or scenarios, or constraints
  - Model those terms in the information model

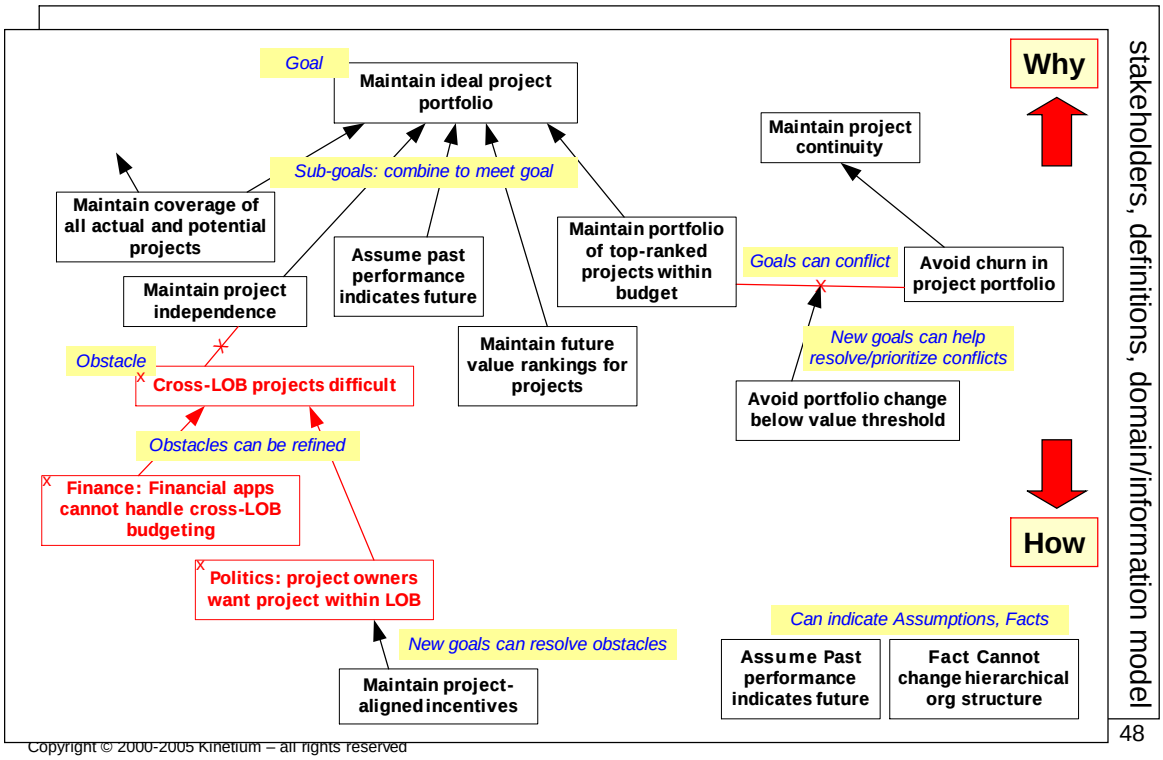
Main Elements of Goal Modeling



Light weight "List" and "Draft" versions are equally useful

47

A Goals Model

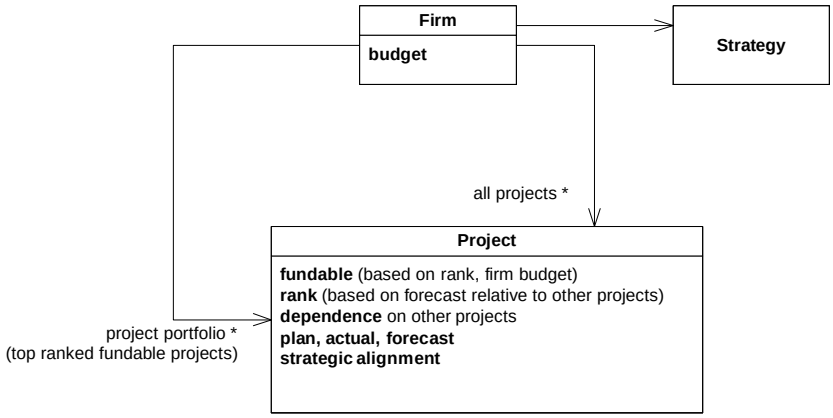


48

Goals and Domain Model

Goal Maintain ideal project portfolio level

At all times maintain the **firm's** top **ranked fundable projects** within the **project portfolio**. Rank is based on
§ **forecast** program value, from **plan**, **actual**, and **forecast** of all **projects**
§ it's overall **alignment** with firm strategy
§ it's minimal **dependence** on other projects



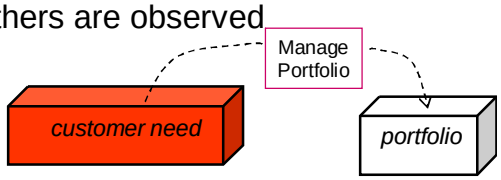
- Goals become clearer and fewer, goal structure better motivated

Why–How, Obstacles, and Domain Model of Projector

Exercise 4

From Goals to Top Level Components, Connectors

- Each Goal refers to certain elements of the Information Model
  - Manage Portfolio: Constantly evolve the *portfolio* of products to optimally meet *customer need*
- Some of these elements are controlled, others are observed
  - *customer need* - observed
  - *portfolio* – controlled



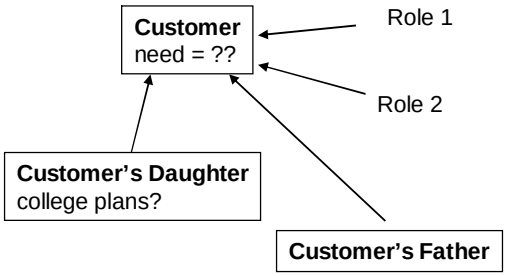
- These elements may be owned elsewhere in the enterprise
  - *customer need* – owed by Application-1
  - *portfolio* - owned by Application-2
- Essential connectivity between components spelled out by goal itself

Goals Drive Other Architecture Decisions

- **Goals modeling drives out other goals, assumptions**
  - Constantly evolve the *portfolio* of products to optimally meet *customer need*
  - **Why? à** higher level goal: provide optimally aligned customer-experience
  - **How? à** evolving portfolio of products is not itself sufficient to meet higher goal
- **Goals drive the top level architecture choices, alternatives, refinements**
  - What is the essential connectivity between top-level black boxes?
  - What changes to components and connectors help meet the goals?
- **Goals drive the broad structure and detailed specification of processes**
  - What should be essential processes themselves?
  - What should be the top-level boundary of each process?
  - What should be the top-level interactions between those processes?
  - What are the essential activities within each process?
  - What are the specifications of each activity i.e. when triggered, prohibited, etc.

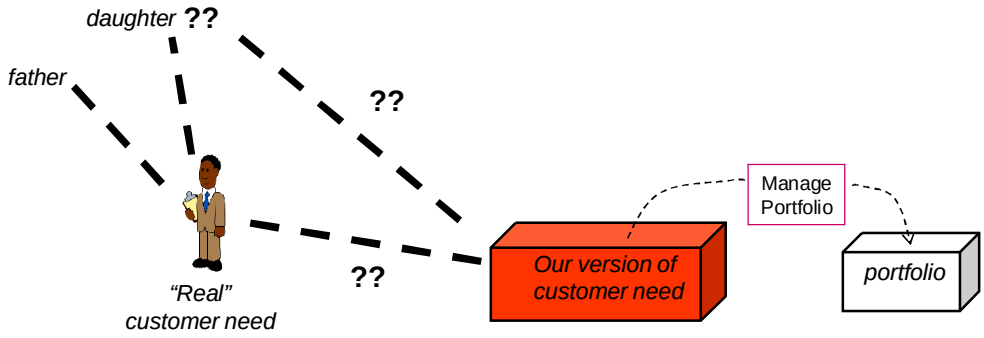
Clarified Goals lead to Better Architectures

- What constitutes “Customer Need”?
  - What are different roles a customer plays?
  - How do they contribute to his/her needs?
  - What different roles other objects related to Customer play?
  - How do *those* contribute to Customer’s need?



Domains to Architecture

- These are models of the *Problem Domain itself*
  - Not necessarily of any existing piece of software
  - Perhaps not directly of any to-be piece of software
  - Help understand what things in domain should be monitored, controlled
  - And, by refinement and architecture styles … what processes? Technologies?
- Domain Architecture: better coverage of need-related information
- Technical architecture: collect, correlate multiple sources of information



55

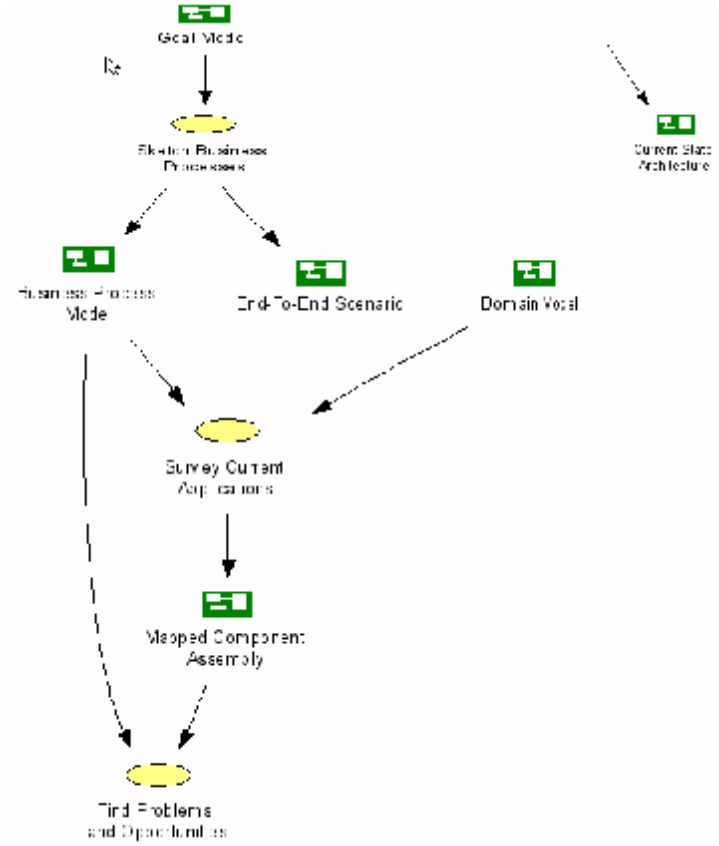
Outline

- Introduction to MAP: What it is, Process and Artifacts
- Models, Architecture, and Objects
- Goals and Domain Model
- **Current State Architecture**
- Target State Architecture
- Migration Plan
- Project Management

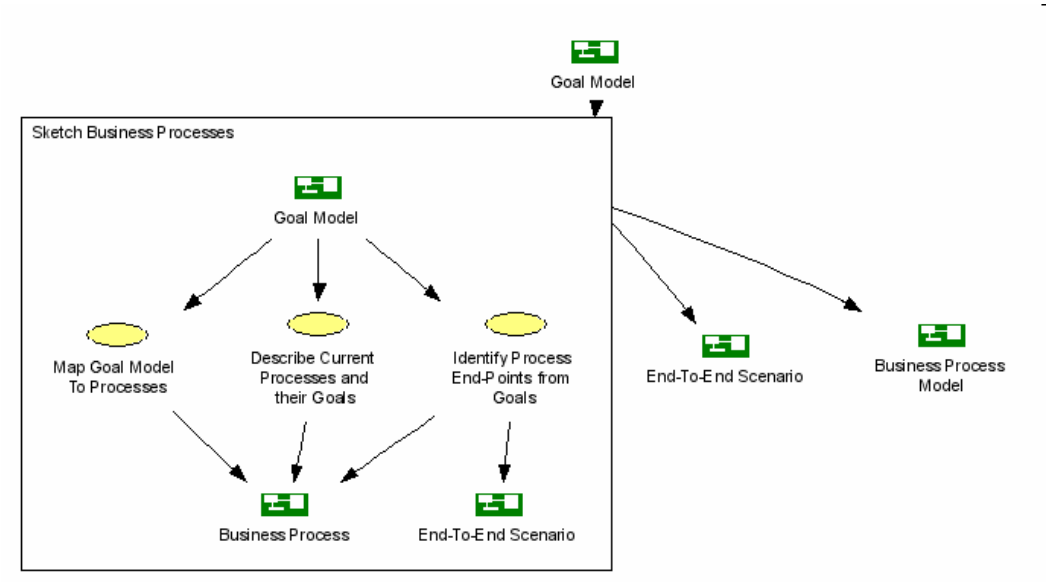
56

Current State

- Focus scoped by problem defn.
- Map applications to domain model
- Feed back obstacles to problem definition
- Feed forward both problems and opportunities to target state definition



Sketching Business Processes



- Identify Process End-Points From Goals
  - Process (and illustrative scenarios) should span all elements relevant to goal

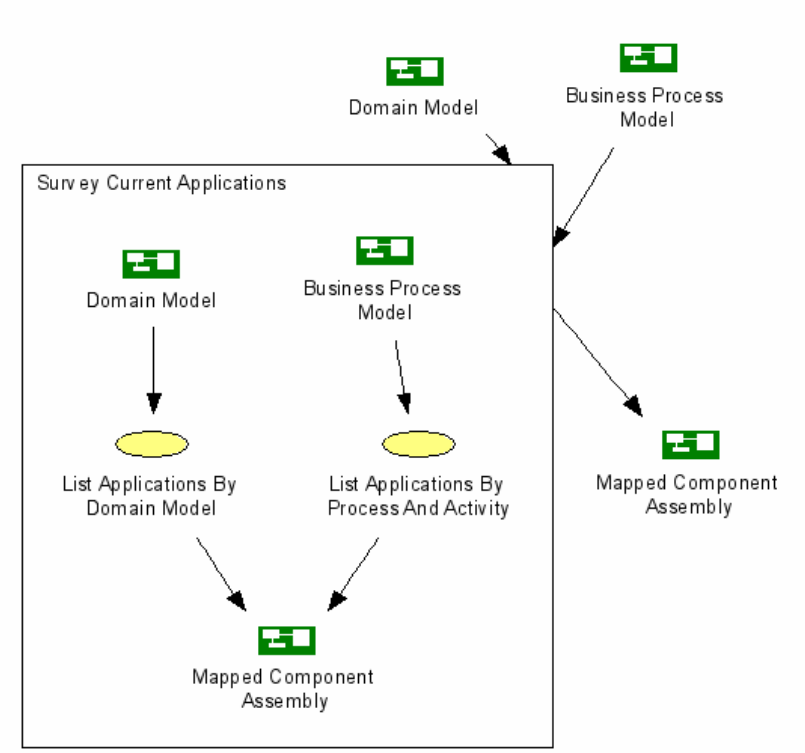
Finding Scenarios for a Goal

- A goal refers to certain elements (attributes, events, etc.) of the domain
- Hence certain events in the domain could cause the goal to be violated
  - Changes in observed attributes
  - Explicit request events
- We want to identify such events from the goal and domain definition
  - Part of our architecture will need to detect these events
  - Other parts will need to collaboratively react to these events
- These help us find scenarios to use to validate our architecture

End-To-End Scenarios from Goals: for Projector

Survey Current Applications

- Map applications (and roles) to their coverage of the domain model



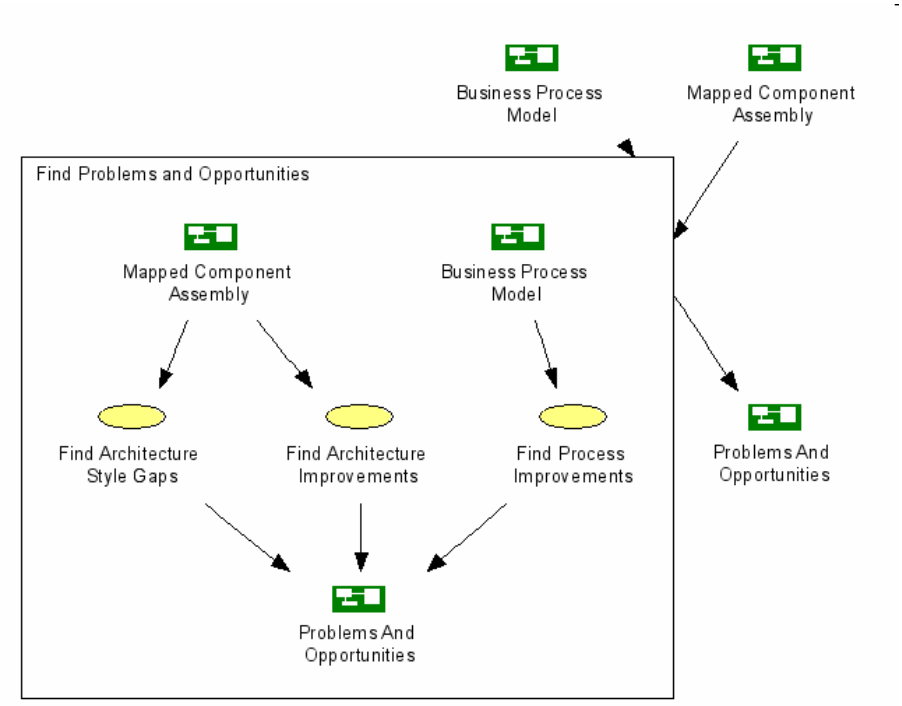
Mapped Components

Application	Domain Element Owned	Domain Element Needed	Technology
App1	Type or Attribute or Event (or Goal)	Type or Attribute or Event (or Goal)	
App2	…		

- If there is no clean allocation of ownership for some domain element:
  - Try refining it into finer-grained elements that have clear ownership
  - e.g. Gasoline station: sales transactions owned by pump and by station computer
    - Owned by pump during the filling
    - Owned by station computer after completion of filling

Exercise 5

Find Problems and Opportunities



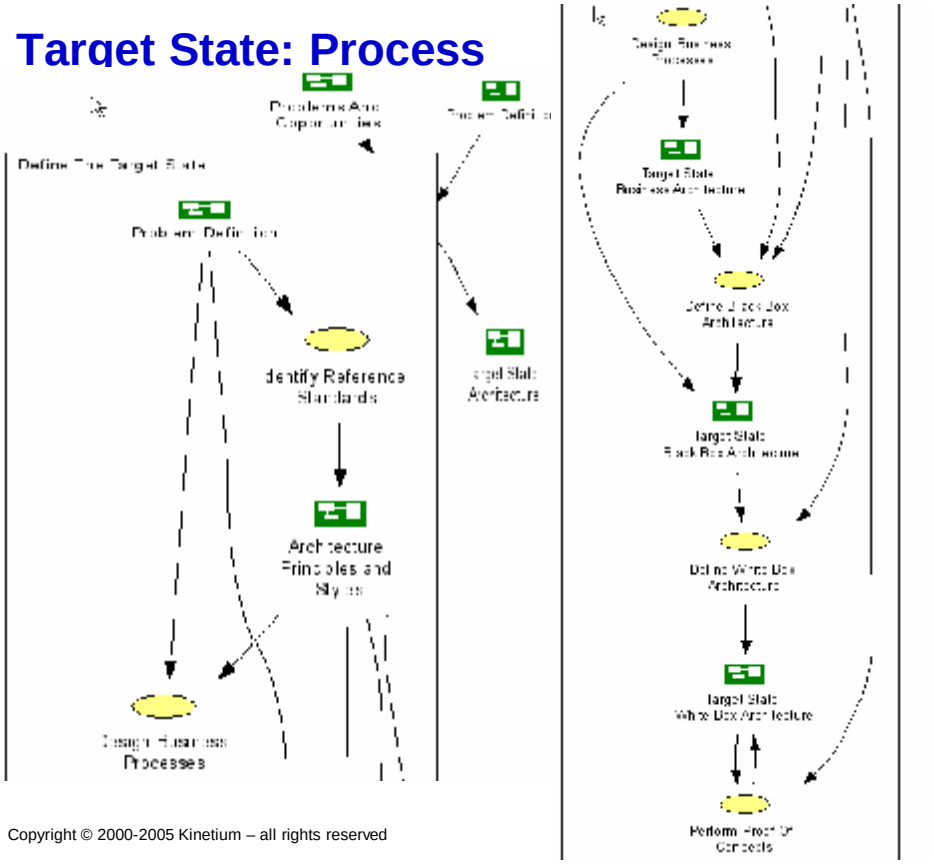
Problems and Opportunities: Projector

- Assume that live meetings using projectors are the current state. Let's come up with likely current-state problems and improvements in:
  - Business process
  - Software architecture
- Pick one current-state problem. Model it as an obstacle to some goals.

Outline

- Introduction to MAP: What it is, Process and Artifacts
- Models, Architecture, and Objects
- Goals and Domain Model
- Current State Architecture
- **Target State Architecture**
- Migration Plan
- Project Management

Target State: Process



- Establish architecture styles for goals
- Design ideal business processes for goals
- Identify black-box partition and assembly
- Design internal white-box assemblies
- Choose candidate products for POCs

What is an Architecture Style?

- **Architecture Style:** A definition of which **architectures** are acceptable (or preferred), within or across viewpoints, in the context of **goals** and constraints, including a possibly customized **language** with new element types and rules of usage.

- Possible usage includes
  - MDA-style transformation
  - design patterns
  - principles
  - shared code components

Style	Example	Generative	Automated	Checkable
Guidelines and patterns for the architect to follow	If high availability is needed use replication and heartbeats	X	-	X
A generator or transformer across architecture viewpoints	Generate EJB deployment descriptors, database schemas from source model	X	X	X
A predicate to use to check if an architecture is acceptable	Component count should be constant or increase very slowly with increase of users	-	-	X

Architecture Principles and Styles

- Architecture principles and styles define preferred architectures
- Very wide spectrum of styles in all viewpoints
  - Earliest point of data capture: capture data at earliest point in a process
  - Workflow coordinator style: partition components and co-ordinate flows
  - Coexistence Bridge: replacement and legacy apps coexist using replication
  - Replication and heartbeats styles: maintain availability through failures

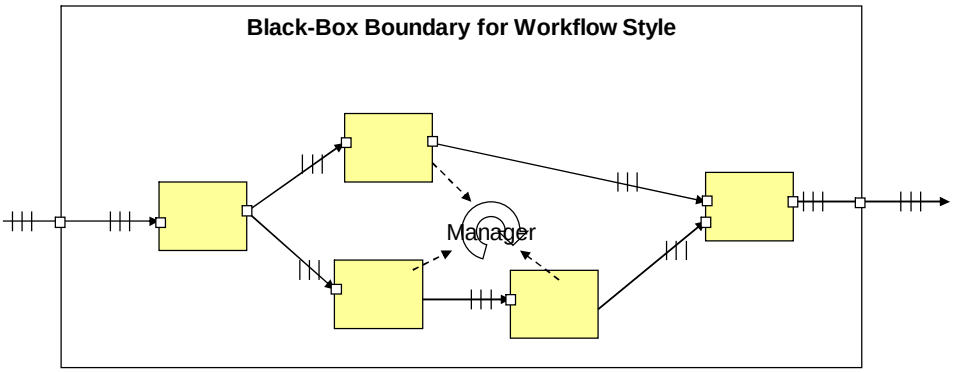
Style: Basic Workflow

= component corresponding to an activity in the lifecycle of some object

= workflow manager for object lifecycle

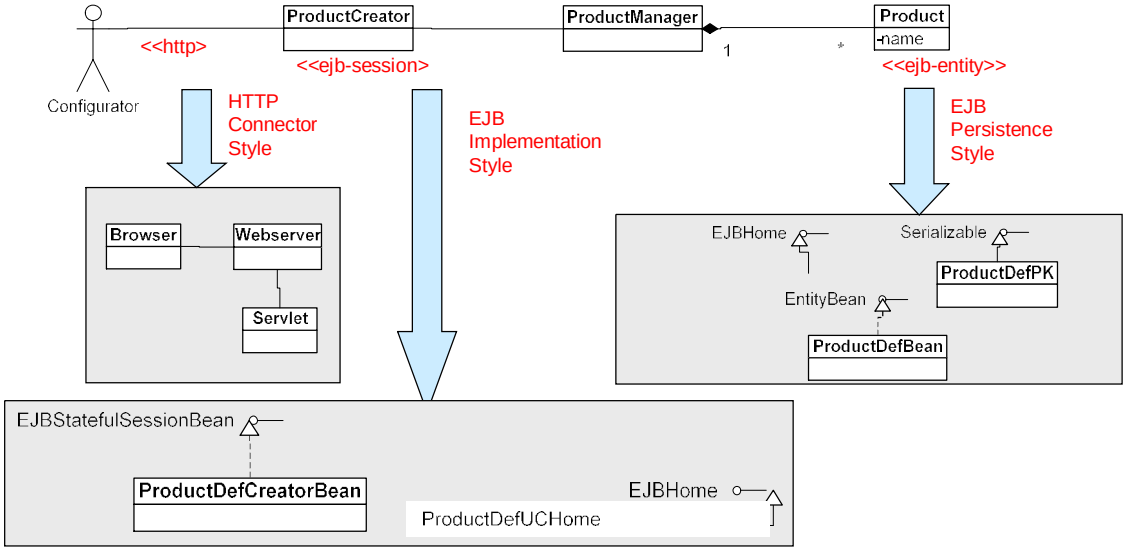
= event from component to / from workflow manager

= "virtual" connector between components, represented explicitly in workflow manager.
Connector variations include constructs for loops, sequence, conditions, forks, joins, …



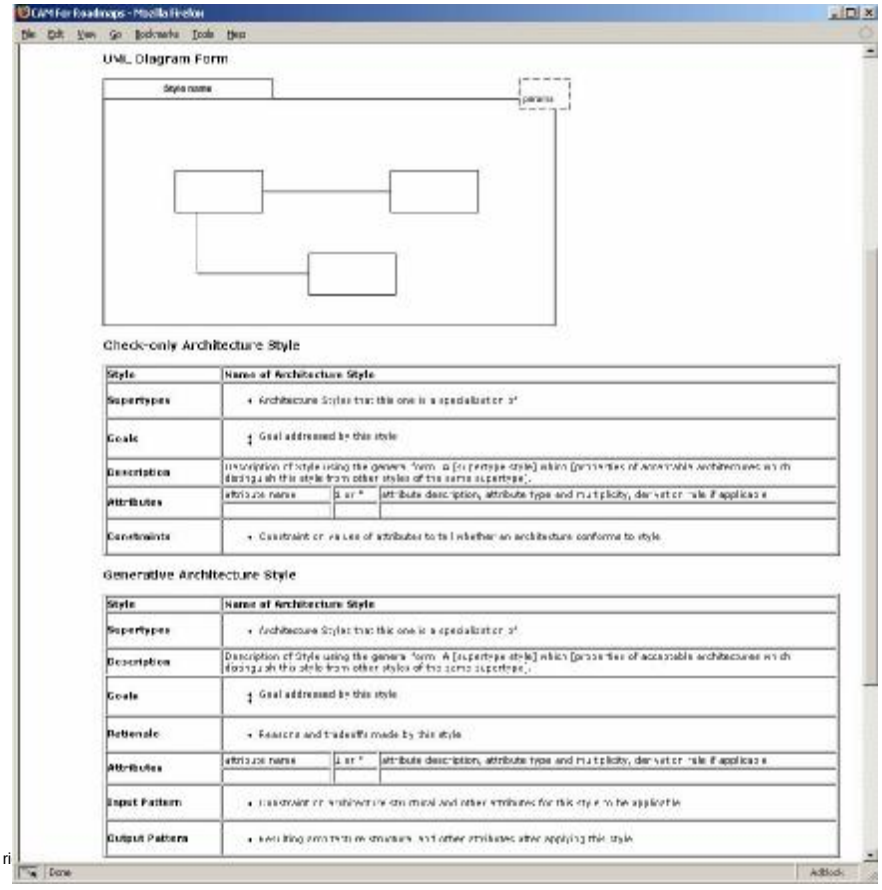
- Input architecture: pattern, assumptions …
- Output architecture: patterns, …
- Resulting architectural properties …

Style: Map Components/Connectors to J2EE



- Can define patterns to map to platform supported architecture
  - EJB Session beans (stateful for dialog; preferably stateless for server)
  - EJB Entity beans
  - Servelets and JSPs

Styles



How do goals drive Business Processes?

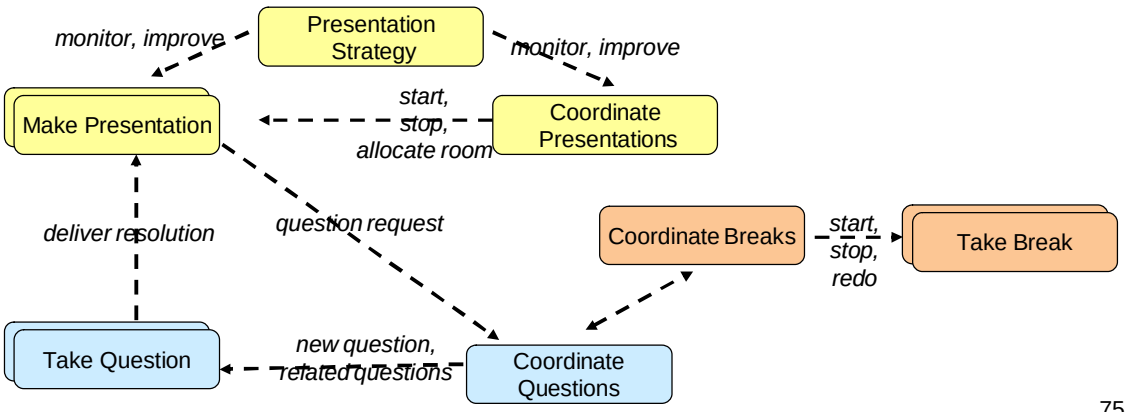
- Business process should have an objectively determined boundary
- Required process interactions and activities are determined by goals

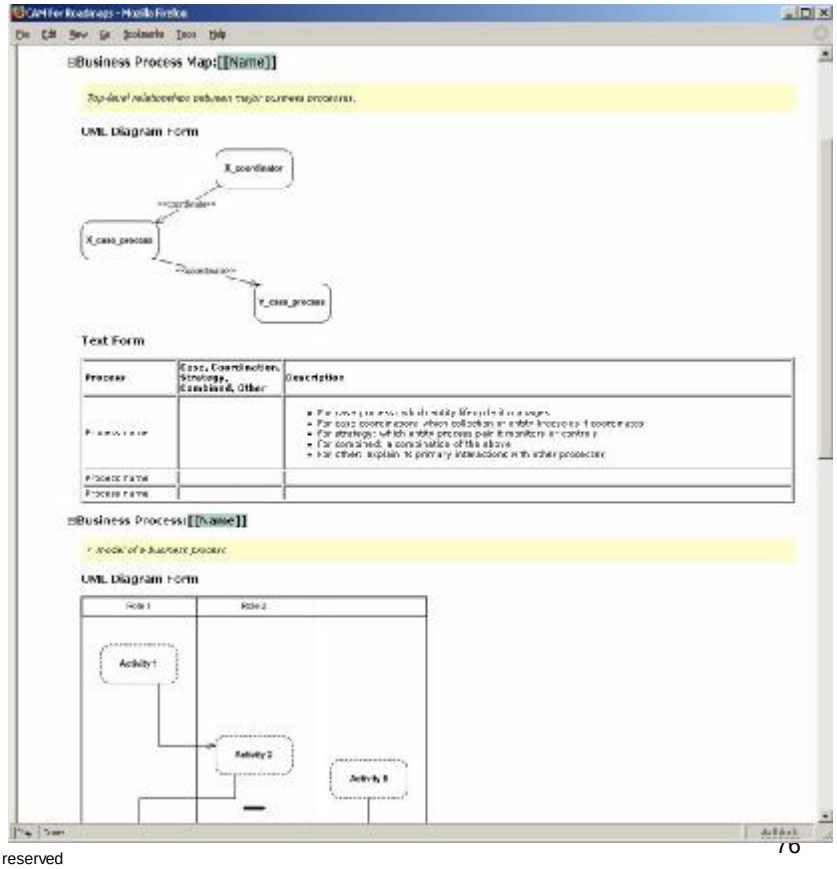
Target State Processes: Case, Coordination, and Strategy

- Identify long-lived client-facing Unit Of Work (customer-aligned process)
  - *e.g. Presentation*
- Ask the question: “Is This an Essential Business Entity”?
  - Test: eliminating it is not an option in your business
- Define “case” process: takes one instance through its lifecycle
  - *e.g.* ***Make Presentation***
- Find “generates” relations to other entities spawned off, with their own lifecycles
  - *e.g. Question*
- Define “case” process: takes one instance through its lifecycle
  - *e.g.* ***Take Question***
- Define “case coordination” process to start, stop, coordinate multiple “cases”
  - *e.g.* ***Coordinate Questions, Coordinate Presentations***
- Case coordination process does not have to be stand-alone
  - It could be a part of some larger case process
  - *e.g.* ***Make Presentation*** could include the coordination across all its ***Take Question***s
- Define dynamic relationships between Process instances

Refining Goals and Business Process Design

- "Essential" entities with interesting lifecycles in Presentation domain
  - Presentation, Question, …
- Corresponding reference processes
  - Make Presentation, Take Question
  - Coordinate Presentations, Coordinate Questions
  - Presentations Strategy, Questions Strategy
- Reference process architecture ties these together
  - Arrows do not imply sequence, but summarize inter-process interactions at lower level



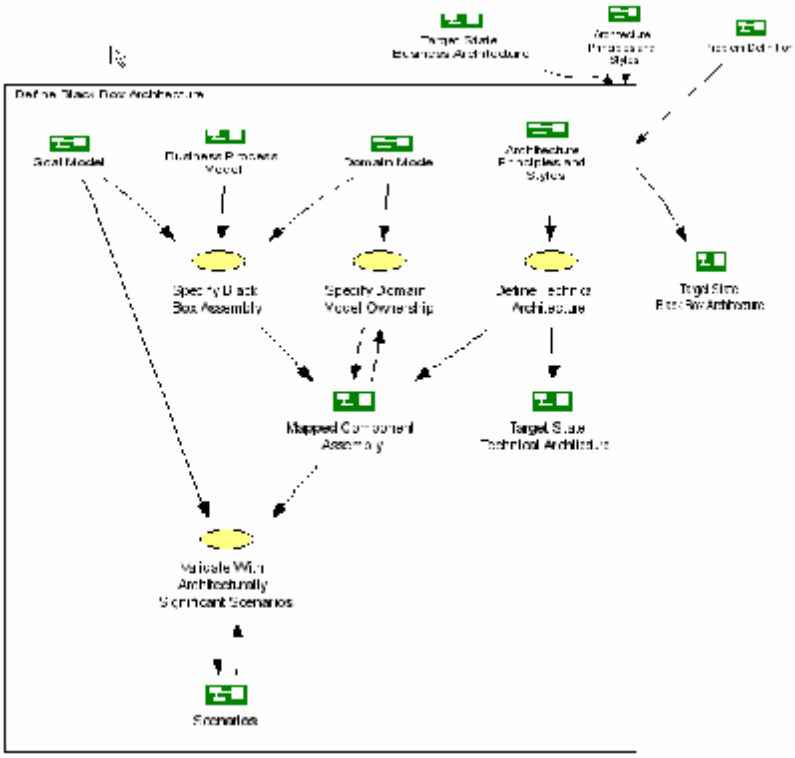


Discuss

- Case and Co-ordination processes for case study

Black-Box

- Choose top-level assembly of black-boxes and roles, using:
  - Goals
  - Domain model
  - Styles
- Map to domain model
- Validate with goal-based end-to-end scenarios



Template

Target State Black Box Architecture:[[Name]]

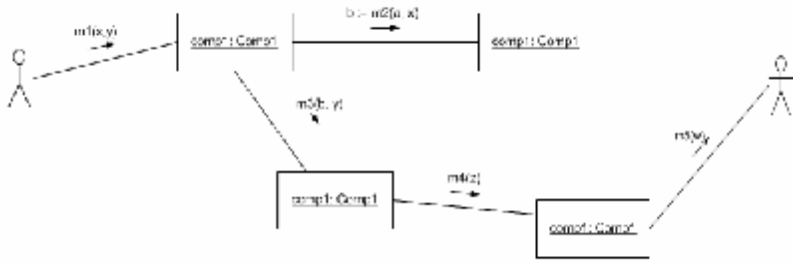
An assembly of interconnected large-grained black-boxes and roles, with the relevant sequences of each, that together meet required goals.

Mapped Component Assembly:[[Name]]

Collaboration:[[Name]]

A text, sequence diagram, or collaboration diagram showing how all the components of the assembly collaborate to meet the role-based behaviors and other specifications

Collaboration



Collaboration:[[Name]]...(repeat as needed)

Black-Box Specification:[[Name]]

A specification of relevant external properties of an individual black-box

Black Box Behavior Model:[[Name]]

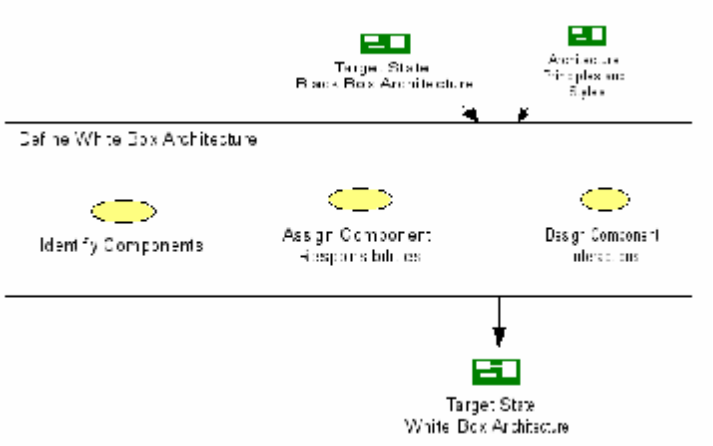
Black Box Information Model:[[Name]]

Black Box User Interface:[[Name]]

Black-Box Specification:[[Name]]...(repeat as needed)

White-Box

- White box partition driven by styles
- Produces component as
- Validate by scenarios



Template

Target State White Box Architecture:[[Name]]

A model of how a given black-box is realized by some combination of sub-components and connectors.

Target State White Box Assembly:[[Name]]

A model of the interconnected sub-components and how they map to the ports of the containing black box.

Mapped Component Assembly:[[Name]]

Target State White Box Collaborations:[[Name]]

A model of the interactions between the sub-components, via collaboration instances, showing how their properties combine to realize the containing black box or its responsibilities or goals.

Collaboration:[[Name]]

Collaboration:[[Name]]...(repeat as needed)

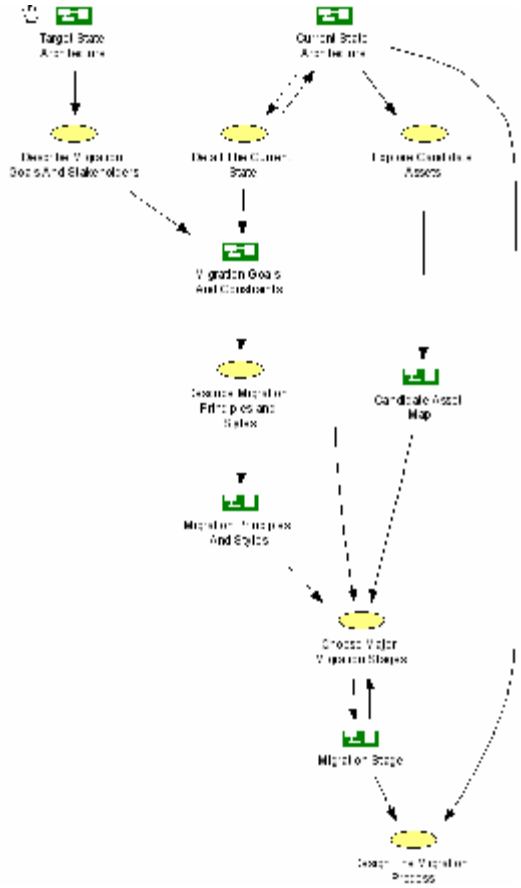
Target State Technical Architecture:[[Name]]

Target State Product Architecture:[[Name]]

Outline

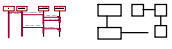
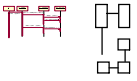
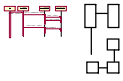
- Introduction to MAP: What it is, Process and Artifacts
- Models, Architecture, and Objects
- Goals and Domain Model
- Current State Architecture
- Target State Architecture
- **Migration Plan**
- Project Management

- For complex migrations:
  - Treat migration as miniature problem domain itself
  - Model migration-oriented goals and stakeholders
  - Define styles to guide migration design
  - Find candidate components or products to use
  - Choose stages (snapshots of architecture) for value and cost
  - Design detailed process to syndicate to stakeholders

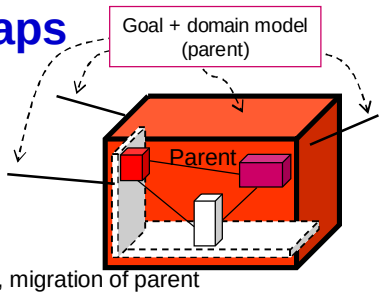


Road Maps and Migration Planning

- MAP can describe architectures at *a variable level of precision*
- MAP viewpoints apply to architecture migration plan

Stage Name	**Current State**	Interim 1	Interim 2	**Target State**
Stage Date	Feb 2002	Nov 2002	Sep 2003	
Stage Description	…	…	…	… including Target architecture style
Logical Assembly Connected components, optional key interactions, business processes				
Technical Architecture				
Changes, Gaps, Training required				
Costs, benefits, risks, rationale				
Conformance to Target state architecture style				

Relating Roadmap to other Roadmaps

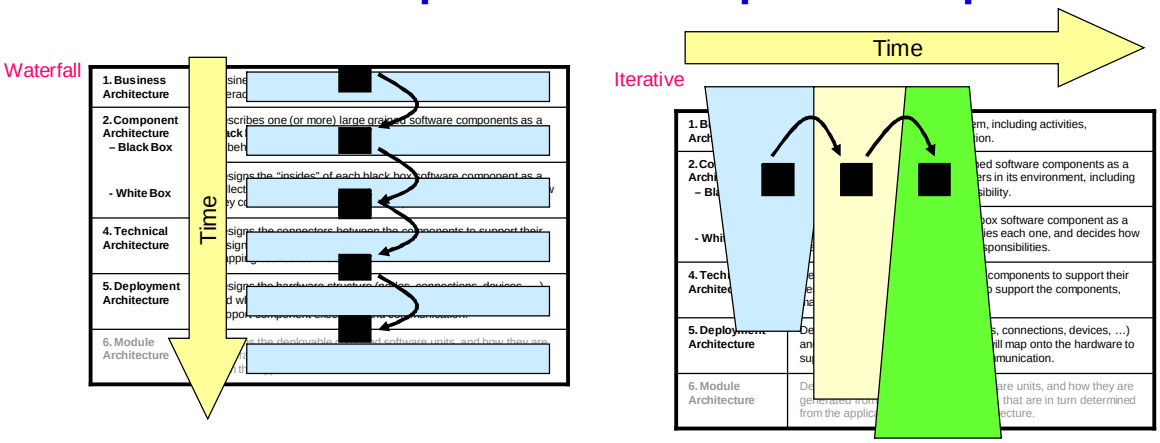


- Common approach for 2 kinds of relationships
  - Child roadmaps
  - Peer roadmaps
- Child roadmaps (e.g. ID&EM and ER)
  - Child roadmap mostly details the goals, white-box architecture, migration of parent
  - Need clear consistency relationship between the two, even if not enforced 100%
  - Child roadmaps may be focused on a sub-component, or a slice through all sub-components
- **Ownership**: parent roadmap owns child's interfaces, connections, cross-child domain model, architecture styles
- **Coverage:** parent **must** explore "architecturally significant" sub-goals scenarios
  - Scenarios must begin and end at the "end-points" of the outermost parent goals
  - Must include scenarios that exercise new paths through the next level architecture
  - Must include scenarios that cannot be localized within a single child
- Peer roadmaps (e.g. ID&EM and HR)
  - Demands communication with different stakeholder and owner group
  - Goals and information models must fit together for overall goal
  - Component architecture, information ownership, interaction must fit together
  - Migration plans and timelines must fit
  - **Note:** Peer roadmaps should also have a **notional "parent"** to own the connections

Outline

- Introduction to MAP: What it is, Process and Artifacts
- Models, Architecture, and Objects
- Goals and Domain Model
- Current State Architecture
- Target State Architecture
- Migration Plan
- **Project Management (*not* the focus of this course)**

Architecture Viewpoint [1] Development Sequence

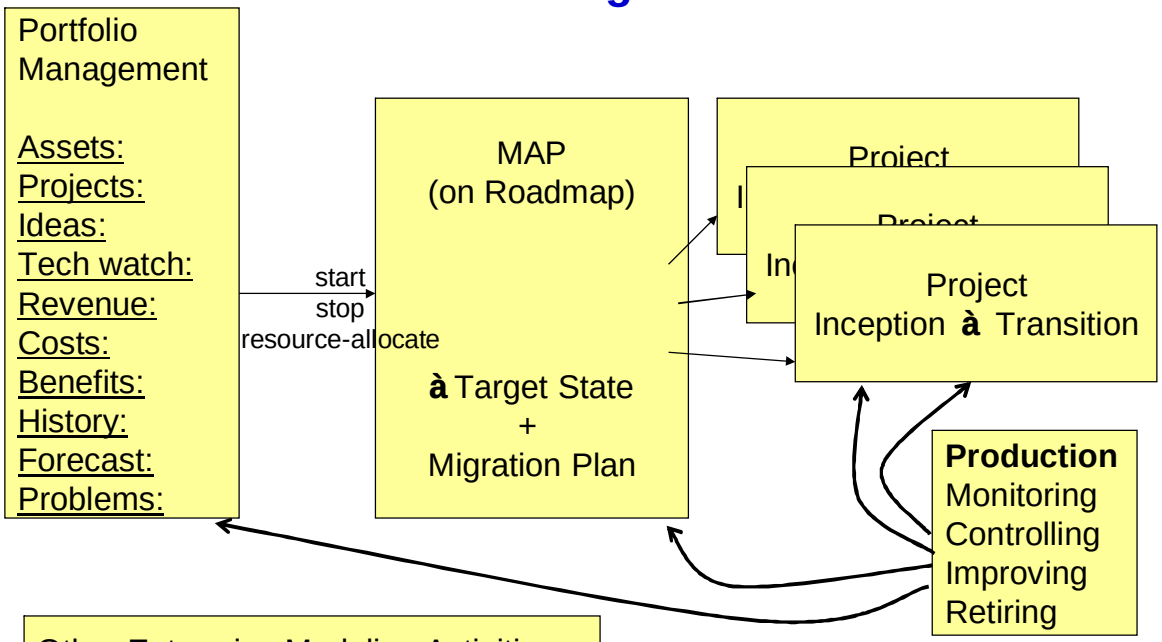


- Development often should not be waterfall i.e. not 1-viewpoint at a time
- Each iteration can cover multiple viewpoints
- Same deliverables is refined across iterations

- Each iteration balances project risk and delivering early value

87

The Role of Portfolio Management

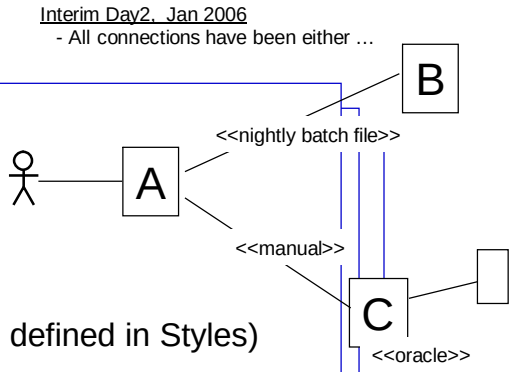


- So early MAP activity (e.g. Initiation Phase) corresponds to Portfolio Management activities

88

Defining Migration Stages

- Stage Name
- Date
- Summary description
- White box component assembly
  - Optional: key collaborations
- Product/technical overlay (overlay terms defined in Styles)
- The main changes from the previous stage
- For each target-state architecture principle or style
  - How well does this stage conform or not conform, and why
- For each leaf-level target-state business goal
  - How well does this stage meet that business goal, and why
- Rationale for this stage
- Cost of getting to this stage
- Benefits and risks of this stage

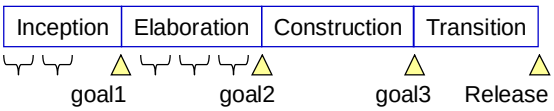


Architecture Sliced By Project Lifecycle

- MAP architecture viewpoints apply to different Project Lifecycles
- MAP deliverables are adapted to lifecycle
  - "List" vs. "Draft" vs. "Dressed" versions
  - As-is vs. To-be models

	Cycle 1	Cycle 2	Cycle 3
Business	list	draft	dressed
Black box	list	draft	dressed
White box		list	draft

RUP-like Lifecycle
Iterations within phases



Detailing Migration Plans to Project Plans

- Projects produce much more detailed descriptions than migration plans
- Detailed specifications and internal designs of components
- Styles of mapping components, connectors to platforms
  - Usage patterns and guidelines from logical architecture to platform facilities
- Incremental delivery
  - Typically scoped by use cases delivered in an increment
- Parallel work plans and dependency management
  - Share the portions of models e.g. overlapping business domains
  - Separate business and technical domains e.g. trade capture vs. security services
  - Continue to use external views of components versus their internal implementations
- Candidate components
  - Include your current state components in your application inventory
  - Consider finer-grained partitions of your current components that might be harvested and re-packaged for use
  - Include candidate third-party components, applications, and frameworks (both business and technical)
- If roadmap-scale effort leads to other roadmaps, MAP provides further guidance

91