Component Architectures

Some meeting points of practice, trend, and theory

Desmond D’Souza
Kinetium

desmond.dsouza@kinetium.com
Introduction
Introduction

Components and Objects are part of state-of-the-practice in SWE
Introduction

- Components and Objects are part of state-of-the-practice in SWE
- Formal methods bring powerful tools to bear … wherever they go
Introduction

- Components and Objects are part of state-of-the-practice in SWE
- Formal methods bring powerful tools to bear … wherever they go
- This conference is about bringing these two together
Introduction

- Components and Objects are part of state-of-the-practice in SWE
- Formal methods bring powerful tools to bear … wherever they go
- This conference is about bringing these two together

- “So, what is someone as informal as I doing addressing you?”
Introduction

Components and Objects are part of state-of-the-practice in SWE
Formal methods bring powerful tools to bear … wherever they go
This conference is about bringing these two together

“So, what is someone as informal as I doing addressing you?”
– Adapted from Michael Jackson at VDM-91, but …
Introduction

- Components and Objects are part of state-of-the-practice in SWE
- Formal methods bring powerful tools to bear … wherever they go
- This conference is about bringing these two together

“So, what is someone as informal as I doing addressing you?”
- Adapted from Michael Jackson at VDM-91, but …
- … I am even further out towards the “not-that-formal” end of the spectrum
Introduction

Components and Objects are part of state-of-the-practice in SWE
Formal methods bring powerful tools to bear … wherever they go
This conference is about bringing these two together

“So, what is someone as informal as I doing addressing you?”
– Adapted from Michael Jackson at VDM-91, but …
– … I am even further out towards the “not-that-formal” end of the spectrum

I want to share some things I have learned in component modeling and methods through industrial practice and industrial teaching, within my own focuses and biases, and outline my ideas about some meeting points of practice, trend, and research … many simply as questions.
More than you ever wanted to know …
More than you ever wanted to know … about me
More than you ever wanted to know … about me

Desmond D’Souza is President of Kinetium. He is co-author and developer of the CATALYSIS method, published by Addison Wesley in 1998, and is a respected authority and speaker at companies and conferences internationally. He was previously senior vice president of component-based development at Platinum Technology and at Computer Associates, working on methods, tools, and architectures for component-based development. He founded ICON Computing, an object and component technology methods and services company that was acquired by Platinum in 1998. Mr. D’Souza has worked with object and component technology since 1985.

Kinetium is an Austin, Texas company that provides solutions for model-driven component-based development and integration, including architectures, methods, training, and tools. To learn more about the how model-driven architecture, component- or service-based development, and enterprise architecture fit together, and how Kinetium can help your company, contact Mr. D’Souza at desmond . dsouza @ kinetium . com
Outline

- Practice, Trend, and Theory
- MDA – Model Driven Architecture
- General Components and Composition
- Architecture and Architecture Style
- Architecture Viewpoints and Concerns
- Refinement
- Requirements in Context
- Enterprise Architectures
Practice, Trend, and Theory
Practice, Trend, and Theory

- Practice: hacking, XP, UML, increasing use of tools and generators
Practice, Trend, and Theory

- Practice: hacking, XP, UML, increasing use of tools and generators
- Trend: kinder gentler XP, more **MDA**, more scalable and fractal
Practice, Trend, and Theory

- Practice: hacking, XP, UML, increasing use of tools and generators
- Trend: kinder gentler XP, more MDA, more scalable and fractal
- Theory: formalize what is pressing / current / interesting / funded
Practice, Trend, and Theory

- Practice: hacking, XP, UML, increasing use of tools and generators
- Trend: kinder gentler XP, more MDA, more scalable and fractal
- Theory: formalize what is pressing / current / interesting / funded

I will use MDA as an anchor for some promising meeting points

- MDA = Model Driven Architecture (Model of <x> = Description of relevant bits of <x>)
- Driven be quite real needs and problems in the industry
- Use platform-independent descriptions (of requirements, designs…)
- Generate platform-specific implementations and scaffolding
- An OMG initiative, leverages other OMG standards like UML, MOF, QVT
Why MDA -- ++ as an anchor point?
Why MDA -- ++ as an anchor point?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
Why MDA -- ++ as an anchor point?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
- Multiple modeling languages, fundamentally object / component based
  - UML, CWM (data warehouse), …
  - Use the one that works best for your problem, even domain-specific ones
Why MDA -- ++ as an anchor point ?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
- Multiple modeling languages, fundamentally object / component based
  - UML, CWM (data warehouse), …
  - Use the one that works best for your problem, even domain-specific ones
- Formal relationships between models, transformations, language definitions
  - Aspects, Refinement, others
  - Broader than just platform-specific code generation
  - To define transformations, perform generation, etc. you need more than pretty pictures
Why MDA -- ++ as an anchor point?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
- Multiple modeling languages, fundamentally object / component based
  - UML, CWM (data warehouse), …
  - Use the one that works best for your problem, even domain-specific ones
- Formal relationships between models, transformations, language definitions
  - Aspects, Refinement, others
  - Broader than just platform-specific code generation
  - To define transformations, perform generation, etc. you need more than pretty pictures
- Potentially large leverage for tools development
  - Model representation in repositories, interchange across tools, etc.
Why MDA -- ++ as an anchor point?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
- Multiple modeling languages, fundamentally object / component based
  - UML, CWM (data warehouse), …
  - Use the one that works best for your problem, even domain-specific ones
- Formal relationships between models, transformations, language definitions
  - Aspects, Refinement, others
  - Broader than just platform-specific code generation
  - To define transformations, perform generation, etc. you need more than pretty pictures
- Potentially large leverage for tools development
  - Model representation in repositories, interchange across tools, etc.
- “A” = Architecture for models, model of architectures
  - Components, connectors, configurations, and more
Why MDA -- ++ as an anchor point?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
- Multiple modeling languages, fundamentally object / component based
  - UML, CWM (data warehouse), …
  - Use the one that works best for your problem, even domain-specific ones
- Formal relationships between models, transformations, language definitions
  - Aspects, Refinement, others
  - Broader than just platform-specific code generation
  - To define transformations, perform generation, etc. you need more than pretty pictures
- Potentially large leverage for tools development
  - Model representation in repositories, interchange across tools, etc.
- “A” = Architecture for models, model of architectures
  - Components, connectors, configurations, and more
- Large-scale and long-lived systems
  - Approach has to be scalable, demands fractal-like composition properties
  - Architecture evolution, system integration problems, roadmaps
Why MDA -- ++ as an anchor point?

- Model-driven
  - More abstract descriptions of requirements, architecture, maybe pictures of code too
- Multiple modeling languages, fundamentally object / component based
  - UML, CWM (data warehouse), …
  - Use the one that works best for your problem, even domain-specific ones
- Formal relationships between models, transformations, language definitions
  - Aspects, Refinement, others
  - Broader than just platform-specific code generation
  - To define transformations, perform generation, etc. you need more than pretty pictures
- Potentially large leverage for tools development
  - Model representation in repositories, interchange across tools, etc.
- “A” = Architecture for models, model of architectures
  - Components, connectors, configurations, and more
- Large-scale and long-lived systems
  - Approach has to be scalable, demands fractal-like composition properties
  - Architecture evolution, system integration problems, roadmaps

- So Why The “- - “?
  - Take a look at the size of UML 2.0!
Model Driven Architecture (-- ++)

UML, Corba, .NET, J2EE, ebXML, and all …
UML, Corba, .NET, J2EE, ebXML, and all …
There are a huge amount of software descriptions – UML, IDL, Java, XML, …

- Many different subject areas or domains
- Many levels of detail, separating and mixing of concerns, diverse languages
- Many overlaps and relationships between these
UML, Corba, .NET, J2EE, ebXML, and all …

- There are a huge amount of software descriptions – UML, IDL, Java, XML, …
  - Many different subject areas or domains
  - Many levels of detail, separating and mixing of concerns, diverse languages
  - Many overlaps and relationships between these

- No clear Overarching Architecture
There are a huge amount of software descriptions – UML, IDL, Java, XML, …

- Many different subject areas or domains
- Many levels of detail, separating and mixing of concerns, diverse languages
- Many overlaps and relationships between these

No clear Overarching Architecture
MDA brings Consistent Separation
MDA brings Consistent Separation

- Clear separation of different kinds of computation- and platform-independent models
  - CIM – computation independent model
  - PIM – platform independent component model
  - PSM – platform specific component model
MDA brings Consistent Separation

- Clear separation of different kinds of computation- and platform-independent models
  - CIM – computation independent model
  - PIM – platform independent component model
  - PSM – platform specific component model
MDA brings Consistent Separation

- Clear separation of different kinds of computation- and platform-independent models
  - CIM – computation independent model
  - PIM – platform independent component model
  - PSM – platform specific component model
MDA brings Consistent Separation

- Clear separation of different kinds of computation- and platform-independent models
  - CIM – computation independent model
  - PIM – platform independent component model
  - PSM – platform specific component model

![Diagram](image)

MDA brings Consistent Relationships

Service Provisioning
- Component Independent Business Model
- Platform Independent Component view
- Platform Specific

Billing
- Component Independent Business Model
- Platform Independent Component view
- Platform Specific
MDA brings Consistent Relationships

- Clear relationship relating different models
  - Platform independent components to platform specific models – a refinement ("vertical") mapping
  - Business model to Platform independent components – a refinement mapping
  - Business area A to Business area B to model the overlap – a “horizontal” model integration
  - Interface A to interface B of the same component – a “horizontal” model integration
MDA brings Consistent Relationships

- Clear relationship relating different models
  - Platform independent components to platform specific models – a refinement (“vertical”) mapping
  - Business model to Platform independent components – a refinement mapping
  - Business area A to Business area B to model the overlap – a “horizontal” model integration
  - Interface A to interface B of the same component – a “horizontal” model integration

MDA brings Consistent Relationships

- Clear relationship relating different models
  - Platform independent components to platform specific models – a refinement (“vertical”) mapping
  - Business model to Platform independent components – a refinement mapping
  - Business area A to Business area B to model the overlap – a “horizontal” model integration
  - Interface A to interface B of the same component – a “horizontal” model integration

MDA brings Consistent Relationships

- Clear relationship relating different models
  - Platform independent components to platform specific models – a refinement (“vertical”) mapping
  - Business model to Platform independent components – a refinement mapping
  - Business area A to Business area B to model the overlap – a “horizontal” model integration
  - Interface A to interface B of the same component – a “horizontal” model integration
MDA brings Consistent Relationships

- Clear relationship relating different models
  - Platform independent components to platform specific models – a refinement (“vertical”) mapping
  - Business model to Platform independent components – a refinement mapping
  - Business area A to Business area B to model the overlap – a “horizontal” model integration
  - Interface A to Interface B of the same component – a “horizontal” model integration
MDA brings Standard Mappings and Portability

Service Provisioning

- Business Model
- Platform Independent
- Platform Specific CORBA

Billing

- Business Model
- Platform Independent
- Platform Specific CORBA

MDA brings Standard Mappings and Portability

- Standardized mappings and patterns for platform-specific model generators
  - These generic mappings and patterns are defined in shared packages
  - A specific application uses (perhaps is even generated from) the shared package
MDA brings Standard Mappings and Portability

- Standardized mappings and patterns for platform-specific model generators
  - These generic mappings and patterns are defined in shared packages
  - A specific application uses (perhaps is even generated from) the shared package

MDA brings Standard Mappings and Portability

- Standardized mappings and patterns for platform-specific model generators
  - These generic mappings and patterns are defined in shared packages
  - A specific application uses (perhaps is even generated from) the shared package
MDA brings Standard Mappings and Portability

- Standardized mappings and patterns for platform-specific model generators
  - These generic mappings and patterns are defined in shared packages
  - A specific application uses (perhaps is even generated from) the shared package
MDA brings Standard Mappings and Portability

- Standardized mappings and patterns for platform-specific model generators
  - These generic mappings and patterns are defined in shared packages
  - A specific application uses (perhaps is even generated from) the shared package
MDA brings Standard Mappings and Portability

- Standardized mappings and patterns for platform-specific model generators
  - These generic mappings and patterns are defined in shared packages
  - A specific application uses (perhaps is even generated from) the shared package
MDA brings Shared Standard Patterns

Service Provisioning
- Business Model
- Exchange service request and confirmation
- Platform Independent
- Platform Specific

Billing
- Business Model
- Exchange bill and payment
- Platform Independent
- Platform Specific
MDA brings Shared Standard Patterns

- Common patterns at any level of modeling
  - The idea of common patterns applies to models of business, components, platforms, mappings

Service Provisioning

- Business Model
  - Exchange service request and confirmation

Platform Independent

- Platform Specific

Billing

- Business Model
  - Exchange bill and payment

Platform Independent

- Platform Specific

MDA brings Shared Standard Patterns

- Common patterns at any level of modeling
  - The idea of common patterns applies to models of business, components, platforms, mappings

“Exchange” business pattern

Service Provisioning
- Platform Independent
  - Business Model
    - Exchange service request and confirmation
- Platform Specific

Billing
- Platform Independent
  - Business Model
    - Exchange bill and payment
- Platform Specific
MDA brings Shared Standard Patterns

- Common patterns at any level of modeling
  - The idea of common patterns applies to models of business, components, platforms, mappings

Service Provisioning
- Business Model
  - Exchange service request and confirmation
- Platform Independent
- Platform Specific

Billing
- Business Model
  - Exchange bill and payment
- Platform Independent
- Platform Specific
MDA enables Model-Driven Integration

- Service Provisioning
  - Business Model
  - Platform Independent
  - Platform Specific

- Billing
  - Business Model
  - Platform Independent
  - Platform Specific
MDA enables Model-Driven Integration
MDA enables Model-Driven Integration

- Design-time or Run-time mediator has access to model structure
  - Utilize inter-model refinement mappings from concrete up to more abstract levels
  - Works with given inter-model relationships or "horizontal" mappings at abstract level
  - Generate data, communication protocol, service, and business process adaptors
MDA enables Model-Driven Integration

- Design-time or Run-time mediator has access to model structure
  - Utilize inter-model refinement mappings from concrete up to more abstract levels
  - Works with given inter-model relationships or “horizonatal” mappings at abstract level
  - Generate data, communication protocol, service, and business process adaptors
MDA enables Model-Driven Integration

- Design-time or Run-time mediator has access to model structure
  - Utilize inter-model refinement mappings from concrete up to more abstract levels
  - Works with given inter-model relationships or "horizontal" mappings at abstract level
  - Generate data, communication protocol, service, and business process adaptors
MDA enables Model-Driven Integration

- Design-time or Run-time mediator has access to model structure
  - Utilize inter-model refinement mappings from concrete up to more abstract levels
  - Works with given inter-model relationships or “horizontal” mappings at abstract level
  - Generate data, communication protocol, service, and business process adaptors
Exploit Meta-Data !!!

Model-Driven Architecture:
Formal models define access functionality

1. Physical
2. Virtual
3. Relate
4. Deploy
5. Access

Enterprise Information Consumers (EICs)
- Business Intelligence
- Call Center
- CRM
- Risk Systems

Design-Time Metadata

Run-Time Metadata

Virtual Database

Enterprise Information Systems (EISs)
Exploit Meta-Data !!!

0. A high-level “dashboard” query: “What is our current share of Exxon’s wallet”? 

Model-Driven Architecture:
Formal models define access functionality

Enterprise Information Consumers (EICs)

- Business Intelligence
- Call Center
- CRM
- Risk Systems

Design-Time Metadata

Virtual Database

Run-Time Metadata

Deploy

Relate

Model Virtual

Model Physical

Access

Access

Enterprise Information Systems (EISs)
Exploit Meta-Data !!!

0. A high-level “dashboard” query: “What is our current share of Exxon’s wallet”?

1. Deployment information:
   Cust. Purchase intelligence (Oracle) at M/c-1
   Sales in Excel spreadsheets @ at M/c-2
Exploit Meta-Data !!!

0. A high-level “dashboard” query: “What is our current share of Exxon’s wallet”?

1. Deployment information:
   - Cust. Purchase intelligence (Oracle) at M/c-1
   - Sales in Excel spreadsheets @ at M/c-2

2,3. Mapped to business attributes:
   - customer.share of wallet
   - spend, revenue, our sales
   - all customer purchases …

Mapped to local components:
- A does cust. purchase intelligence,
- B does sales …

Mapped to platform specifics:
- Purchase intelligence in Oracle
- sales in Excel spreadsheets
Exploit Meta-Data !!!

0. A high-level “dashboard” query: “What is our current share of Exxon’s wallet”?

1. Deployment information:
   - Cust. Purchase intelligence (Oracle) at M/c-1
   - Sales in Excel spreadsheets @ at M/c-2

2,3. Mapped to business attributes:
   - customer.share of wallet
   - spend, revenue, our sales
   - all customer purchases …

Mapped to local components:
- A does cust. purchase intelligence,
- B does sales …

Mapped to platform specifics:
- Purchase intelligence in Oracle
- sales in Excel spreadsheets
Exploit Meta-Data !!!

0. A high-level “dashboard” query: “What is our current share of Exxon’s wallet”?

1. Deployment information:
   - Cust. Purchase intelligence (Oracle) at M/c-1
   - Sales in Excel spreadsheets @ M/c-2

2,3. Mapped to business attributes:
   - customer.share of wallet
   - spend, revenue, our sales
   - all customer purchases …

4. Made available for run-time use

5. … is transformed into a complex distributed heterogeneous join

2. Model-Driven Architecture:
   Formal models define access functionality

Exploit Meta-Data !!!

0. A high-level “dashboard” query: “What is our current share of Exxon’s wallet”?

1. Deployment information:
   Cust. Purchase intelligence (Oracle) at M/c-1
   Sales in Excel spreadsheets @ at M/c-2

2,3. Mapped to business attributes:
   - Customer.share of wallet
   - Spend, revenue, our sales
   - All customer purchases …

   Mapped to local components:
   - A does cust. purchase intelligence,
   - B does sales …

   Mapped to platform specifics:
   - Purchase intelligence in Oracle
   - Sales in Excel spreadsheets

4. Made available for run-time use

5. … is transformed into a complex distributed heterogeneous join

6. Result is monitored against goal “Maximize share of wallet for tier-1 customers”, and its sub-goals
Three Dimensions for Integration
Three Dimensions for Integration

1: Horizontal

- marketing
- engineering
- sales
- IT
- service
Three Dimensions for Integration

Different subjects: Integrate understanding, vocabulary?
Three Dimensions for Integration

Different subjects: Integrate understanding, vocabulary?
Three Dimensions for Integration

Different subjects: Integrate understanding, vocabulary?

1: Horizontal

- marketing
- engineering
- sales
- IT
- service

e-Boundaries?
Three Dimensions for Integration

1: Horizontal
Different subjects: Integrate understanding, vocabulary?

2: Vertical
marketing
engineering
sales
IT
service

e-Boundaries?
Three Dimensions for Integration

1: Horizontal
Different subjects: Integrate understanding, vocabulary?

2: Vertical

e-Boundaries?
Three Dimensions for Integration

1: Horizontal
Different subjects: Integrate understanding, vocabulary?

1: Vertical
e-Boundaries?


Different levels of abstraction: PIM, PSM, biz process, goals, policy? Are they consistent? How do they relate?
Three Dimensions for Integration

1: Horizontal
Different subjects: Integrate understanding, vocabulary?

2: Vertical

Different levels of abstraction: PIM, PSM, biz process, goals, policy? Are they consistent? How do they relate?

3: Variants
Time, Evolution
Product Line

e-Boundaries?
Foundation – Fractal Inter-Model Relationships
Foundation – Fractal Inter-Model Relationships

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.
  - Can reason about the composition
Foundation – Fractal Inter-Model Relationships

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.
  - Can reason about the composition
Foundation – Fractal Inter-Model Relationships

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.
  - Can reason about the composition

Diagram:

- Package P1
  System from viewpoint 1

- Package P2
  System from viewpoint 2

- Package P3
  Interrelationship between P1 and P2

- Two different levels of abstraction of the same system
  - The refinement relationship between levels is explicit
Foundation – Fractal Inter-Model Relationships

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.
  - Can reason about the composition

- Two different levels of abstraction of the same system
  - The refinement relationship between levels is explicit
Foundation – Fractal “Zooming” in and out
Foundation – Fractal “Zooming” in and out

- Abstract multiple objects as a single larger-grained object … and its properties
Foundation – Fractal “Zooming” in and out

Abstract multiple objects as a single larger-grained object … and its properties

(a) Zooming in/out – objects
Foundation – Fractal “Zooming” in and out

- Abstract multiple objects as a single larger-grained object … and its properties

  (a) Zooming in/out – objects

- Abstract detailed interaction protocols as a single action … and its properties
Foundation – Fractal “Zooming” in and out

- Abstract multiple objects as a single larger-grained object … and its properties

(a) Zooming in/out – objects

- Abstract detailed interaction protocols as a single action … and its properties

(b) Zooming in/out – interactions
Foundation – Fractal Assembly by Composition

Uniform construction of assemblies from other parts
- Component, port, connector, assembly for all variety of design artifacts
- Separate external specification of each port, composition of ports, internal design
- Utilize architecture kit that includes variety of connector types for useful composition
Foundation – Fractal Assembly by Composition

- Uniform construction of assemblies from other parts
  - Component, port, connector, assembly for all variety of design artifacts
  - Separate external specification of each port, composition of ports, internal design
  - Utilize architecture kit that includes variety of connector types for useful composition
Foundation – Fractal Assembly by Composition

- Uniform construction of assemblies from other parts
  - Component, port, connector, assembly for all variety of design artifacts
  - Separate external specification of each port, composition of ports, internal design
  - Utilize architecture kit that includes variety of connector types for useful composition
Foundation – Fractal Assembly by Composition

- Uniform construction of assemblies from other parts
  - Component, port, connector, assembly for all variety of design artifacts
  - Separate external specification of each port, composition of ports, internal design
  - Utilize architecture kit that includes variety of connector types for useful composition
Next – Compose Business Specifications?

Example inspired by Kevin Tyson
Next – Compose Business Specifications?

Example inspired by Kevin Tyson
Next – Compose Business Specifications?

Example inspired by Kevin Tyson
Next – Compose Business Specifications?

Example inspired by Kevin Tyson
Next – Compose Business Specifications?

- Precise patterns of domains and processes
  - Commodities trading – orange juice, pork bellies, … electricity, drinking water?
  - Production – auto parts, … orange juice, pork bellies, … electricity, water water?
  - Degradation – auto parts, orange juice, pork bellies, … electricity, drinking water?
Next – Compose Business Specifications?

- Precise patterns of domains and processes
  - Commodities trading – orange juice, pork bellies, … electricity, drinking water?
  - Production – auto parts, … orange juice, pork bellies, … electricity, water water?
  - Degradation – auto parts, orange juice, pork bellies, … electricity, drinking water?

- Compose models, business specifics process and rules, patterns
  - Futures Trading [ commodity † electricity, payment † drinking water futures ]
    AND
    Consumption [ consumable † futures electricity purchase ]
Next – Compose Business Specifications?

Precise patterns of domains and processes
- Commodities trading – orange juice, pork bellies, … electricity, drinking water?
- Production – auto parts, … orange juice, pork bellies, … electricity, water water?
- Degradation – auto parts, orange juice, pork bellies, … electricity, drinking water?

Compose models, business specifics process and rules, patterns
- Futures Trading [ commodity ÷ electricity, payment ÷ drinking water futures ]
- AND
  Consumption [ consumable ÷ futures electricity purchase ]

Compose patterns of refinements / transformation / architectural styles
- Styles can combine aspect-based code weaving, reflection, schema merging, …
Next – Compose Business Specifications?

- Precise patterns of domains and processes
  - Commodities trading – orange juice, pork bellies, … electricity, drinking water?
  - Production – auto parts, … orange juice, pork bellies, … electricity, water water?
  - Degradation – auto parts, orange juice, pork bellies, … electricity, drinking water?

- Compose models, business specifics process and rules, patterns
  - Futures Trading [ commodity ↓ electricity, payment ↓ drinking water futures ]
    AND
    Consumption [ consumable ↓ futures electricity purchase ]

- Compose patterns of refinements / transformation / architectural styles
  - Styles can combine aspect-based code weaving, reflection, schema merging, …

- Compose directory services information on deployments, networks, database
Next – Compose Business Specifications?

- Precise patterns of domains and processes
  - Commodities trading – orange juice, pork bellies, … electricity, drinking water?
  - Production – auto parts, … orange juice, pork bellies, … electricity, water water?
  - Degradation – auto parts, orange juice, pork bellies, … electricity, drinking water?

- Compose models, business specifics process and rules, patterns
  - Futures Trading [ commodity ‡ electricity, payment ‡ drinking water futures ]
  - AND
  - Consumption [ consumable ‡ futures electricity purchase ]

- Compose patterns of refinements / transformation / architectural styles
  - Styles can combine aspect-based code weaving, reflection, schema merging, …

- Compose directory services information on deployments, networks, database
Next – Compose Business Specifications?

- Precise patterns of domains and processes
  - Commodities trading – orange juice, pork bellies, … electricity, drinking water?
  - Production – auto parts, … orange juice, pork bellies, … electricity, water water?
  - Degradation – auto parts, orange juice, pork bellies, … electricity, drinking water?

- Compose models, business specifics process and rules, patterns
  - Futures Trading [ commodity $\rightarrow$ electricity, payment $\rightarrow$ drinking water futures ]
    AND
    Consumption [ consumable $\rightarrow$ futures electricity purchase ]

- Compose patterns of refinements / transformation / architectural styles
  - Styles can combine aspect-based code weaving, reflection, schema merging, …

- Compose directory services information on deployments, networks, databases

- Implicitly integrate and evolve software components
  - Specialize and Configure
  - Generate plug-ins, bridges, adaptors
  - Connect together
  - Migrate data
Specifying a Component for its Client(s)
Specifying a Component for its Client(s)

Services Provided: Interface Spec  
e.g. pickup(goods)  
⇒ inventory decreased by quantity of goods
Specifying a Component for its Client(s)

Services Provided: Interface Spec
   e.g. pickup(goods)
   ⇒ inventory decreased by quantity of goods

Services Required: Interface Spec
Specifying a Component for its Client(s)

**Services Provided:** Interface Spec
- e.g. pickup(goods)
  - \( \Rightarrow \) **inventory** decreased by **quantity** of **goods**

**Services Required:** Interface Spec

**Events Raised:** Notifications of state changes
- e.g. **inventory** below **minimum**
  - \( \Rightarrow \) restock event raised
Specifying a Component for its Client(s)

Services Provided: Interface Spec
  e.g. pickup(goods)
  ⇒ inventory decreased by quantity of goods

Services Required: Interface Spec

Events Raised: Notifications of state changes
  e.g. inventory below minimum
  ⇒ restock event raised

Logical model of component state: state attributes for each interface
  The interface operations and events are specified based on this
  e.g. inventory and minimum attributes used to specify restock event
Specifying a Component for its Client(s)

**Services Provided**: Interface Spec
- e.g. pickup(goods)
  ⇒ inventory decreased by quantity of goods

**Services Required**: Interface Spec

**Events Raised**: Notifications of state changes
- e.g. inventory below minimum
  ⇒ restock event raised

Logical model of component state: state attributes for each interface
- The interface operations and events are specified based on this
  e.g. inventory and minimum attributes used to specify restock event

Result: Precise model of information exchanged, assumptions, guarantees
Client 1: Interface Client
Client 1: Interface Client

What does the *implementor* of this component need to know ...
Client 1: Interface Client

What does the implementor of this component need to know ...

... about this component?
Client 1: Interface Client

What does the implementor of this component need to know ...

… about this component?

Just this!
Client 1: Interface Client

What does the implementor of this component need to know ...

… about this component?

A client only knows about the relevant interface specification
- operations, events, logical state through that interface
- each interface has its own ops, events, logical model of state
  - warehouse inventory, staffing, storage maintenance: different views

Just this!
Client 2: Component Assembler
Client 2: Component Assembler
Client 2: Component Assembler

What does the creator of this component assembly need to know ...
Client 2: Component Assembler

What does the creator of this component assembly need to know ... 

… about this component?
Client 2: Component Assembler

What does the creator of this component assembly need to know ...

… about this component?
Client 2: Component Assembler

What does the creator of this component assembly need to know ...

… about this component?

Assembler needs to know full component specifications
- all interfaces provided and required
- how the logical state models (and events, operations) are related
  e.g. storage out in IMaintenance ⇒ less space available for IDelivery

What Can Appear in a Port?

- Outbound Requests
- Inbound Requests
- Outbound Events
- Inbound Events
- Accessible Attributes
- In or Out Data Flows
- Exception Signals
- Exception Handlers
- Role in a Use-Case / Joint Action
- Combinations of these

... others as needed

Information model for all data exchanged and for state attributes of port

Port types made useful by standard connector types between ports
Connectors Assemble Components via Ports
Connectors Assemble Components via Ports

Deal Origination

~manage deal

Deal Management

manage deal
Connectors Assemble Components via Ports

Deal Origination

~manage deal

“Simple” connector

manage deal

Deal Management
Connectors Assemble Components via Ports

Deal Origination

~manage deal

“Simple” connector

manage deal

Deal Management

Transaction Management

manage transactions
Connectors Assemble Components via Ports

Deal Origination

~manage deal

“Simple” connector

manage deal

Deal Management

deal events

Transaction Management

manage transactions
Connectors Assemble Components via Ports

Deal Origination

~manage deal

“Simple” connector

manage deal

Deal Management

deal events

Event connector

manage transactions

Transaction Management
Connectors Assemble Components via Ports

Deal Origination

~manage deal

“Simple” connector

manage deal

Deal Management

Foreign Exchange Monitoring

Fx events

Event connector

Transaction Management

manage transactions

deal events
Connectors Assemble Components via Ports

- Deal Origination
  - "Simple" connector
    - manage deal

- Foreign Exchange Monitoring
  - Event connector
    - Fx events

- Deal Management
  - Event connector
    - manage transactions
    - deal events

- Transaction Management
Connectors Assemble Components via Ports

- Deal Origination
- Foreign Exchange Monitoring
- Deal Management
- Transaction Management

- "Simple" connector
- Event connector
- Manage deal
- Fx events
- Manage transactions
- Deal events
- Same Connector Type
Connectors Assemble Components via Ports

Deal Origination

Deal Management

Foreign Exchange Monitoring

Transaction Management

"Simple" connector

manage deal

~manage deal

manage deal

deal events

Fx events

manage transactions

Event connector

Event connector

Same Connector Type

Event Multicast Connector Type
Connectors Assemble Components via Ports

Given a source port
  - with a set of events and event data
And given some number of destination port
  - with methods, input args, data mappings
“Connect” the events to the methods
  - Logical connection or protocol
  - Platform / technical realization
Connectors Assemble Components via Ports

**Deal Origination**
- ~manage deal

**Foreign Exchange Monitoring**
- Fx events

**Deal Management**
- manage deal

**Transaction Management**
- manage transactions

**Event Multicast Connector Type**
- Given a source port
- with a set of events and event data
- And given some number of destination port
- with methods, input args, data mappings
- "Connect" the events to the methods
  - Logical connection or protocol
  - Platform / technical realization

**Replication Connector Type**
- Replicate source property
- to all destination properties

**Change Veto Connector Type**
- Given a source port property
- - Require a "propose-change" event
- - Add veto protocol from listeners
**Connectors Assemble Components via Ports**

- **Deal Origination**
  - manage deal
  - "Simple" connector

- **Foreign Exchange Monitoring**
  - Fx events

- **Deal Management**
  - manage deal

- **Transaction Management**
  - manage transactions

**Architecture Styles**

- **Replication Connector Type**
  - Replicate source property
  - to all destination properties

- **Change Veto Connector Type**
  - Given a source port property
  - Require a "propose-change" event
  - Add veto protocol from listeners

- **Event Multicast Connector Type**
  - Given a source port
    - with a set of events and event data
  - And given some number of destination port
    - with methods, input args, data mappings
  - "Connect" the events to the methods
    - Logical connection or protocol
    - Platform / technical realization
Connector Type defines Layered Protocol
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties

- Connector type = Co-ordination Protocol
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties

- Connector type = Co-ordination Protocol
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties

- Connector type = Co-ordination Protocol
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties

- Connector type = Co-ordination Protocol
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties

- Connector type = Co-ordination Protocol
Connector Type defines Layered Protocol

- Each connector type is specified as follows
  - Set of roles
  - Set of actions involving roles
  - Information model
    - for each role’s state and for information exchanged
  - Behavior model of action sequence

- Connector type can be layered
  - One specification
  - Multiple realizations

- E.g. Different ways to sync two properties

- Connector type = Co-ordination Protocol
Connector specifications

Property connectors: 2-way and 1-way

Event connectors:
Connector specifications

Property connectors: 2-way and 1-way

$$\text{inv} \ a.x = b.y$$

Event connectors:
Connector specifications

Property connectors: 2-way and 1-way

- **2-way** connection:
  - $a \xrightarrow{x} 2P \xrightarrow{y} b$
  - **inv** $a.x = b.y$

- **1-way** connection:
  - $a \xrightarrow{x} 1P \xrightarrow{y} b$
  - **inv effect** $a.x \leftrightarrow a.x@pre$
  - **implies** $b.y.equals(a.x)$

Event connectors:

Connector specifications

Property connectors: 2-way and 1-way

```
inv a.x = b.y
```

```
inv effect a.x <>a.x@pre
implies b.y.equals(a.x)
```

Event connectors:

```
inv effect a.x>a.z and a.x<=a.z@pre implies b.y=b.y@pre+1
```
Connector refinements
Connector refinements

The 2P connector can be realized as 2 1P connectors:
Connector refinements

The 2P connector can be realized as 2 1P connectors:

\[ \text{since } \text{inv effect } a.x <> a.x@pre \implies b.y.equals(a.x) \]
\[ \text{and } \text{inv effect } b.y <> b.y@pre \implies a.x.equals(b.y) \]

implies \text{inv } a.x.equals(b.y)
Connector refinements

The 2P connector can be realized as 2 1P connectors:

\[
\begin{align*}
\text{since} \\
\textbf{inv effect} & \quad a.x \leftrightarrow a.x@ \text{pre implies } b.y = a.x \\
\text{and} \\
\textbf{inv effect} & \quad b.y \leftrightarrow b.y@ \text{pre implies } a.x = b.y \\
\text{implies} & \quad \textbf{inv} \quad a.x = b.y
\end{align*}
\]

Many alternative realizations e.g. sharing

\[
\begin{align*}
\text{since } a.x &= b.y \\
\text{implies } &\quad a.x = b.y
\end{align*}
\]
Connector refinements

The 2P connector can be realized as 2 1P connectors:

- Since
  - \text{inv effect } a.x \not<=> a.x@pre \implies b.y.equals(a.x)
  - \text{inv effect } b.y \not<=> b.y@pre \implies a.x.equals(b.y)

implies \text{ inv } a.x.equals(b.y)

Many alternative realizations e.g. sharing

- Since \( a.x = b.y \)
  - \text{implies } a.x.equals(b.y)

The 1P connector can be realized as an E connector:
Connector refinements

The 2P connector can be realized as 2 1P connectors:

\[
\text{since } \text{inv effect } a.x \leftrightarrow a.x@\text{pre} \implies b.y.\text{equals}(a.x) \text{ and } \text{inv effect } b.y \leftrightarrow b.y@\text{pre} \implies a.x.\text{equals}(b.y) \]

\[\implies \text{inv } a.x.\text{equals}(b.y)\]

Many alternative realizations e.g. sharing

\[
\text{since } a.x=b.y \implies a.x.\text{equals}(b.y)\]

The 1P connector can be realized as an E connector:

\[
\text{... and then there are MDA’s Platform-Specific Realizations}\]
Multi-Tiered Federated Components

Trade management

Risk Allocator

Trade management

Trade originator
Multi-Tiered Federated Components

User port

Trade management

UI Composition
And Scripting Port
(e.g. layout, UI events)

Business Logic Port
(e.g. events, methods)

Data Access Port
(e.g. SQL)

Risk Allocator

Trade management

Trade originator
Multi-Tiered Federated Components

User port

Trade management

Trade Browser
UI

Trade Manager
Biz

Trade Data
DB

UI Composition
And Scripting Port
(e.g. layout, UI events)

Business Logic Port
(e.g. events, methods)

Data Access Port
(e.g. SQL)

Risk Allocator

Trade management

Trade originator
Multi-Tiered Federated Components

A component may span multiple tiers
- Federated components manage their own data
- Components can include user-interface parts
Multi-Tiered Federated Components

A component may span multiple tiers
- Federated components manage their own data
- Components can include user-interface parts

Modeled and built as an assembly of smaller components
**Multi-Tiered Federated Components**

A component may span multiple tiers
- Federated components manage their own data
- Components can include user-interface parts

Modeled and built as an assembly of smaller components

Can design “assembly” of large-grained components that “conform” to spec
Component (Re)Configuration

User port

Trade Browser

UI Composition And Scripting Port

Business Logic Port (e.g. events, methods)

Data Access Port
Component (Re)Configuration

User port

Trade Browser

Trade Browser UI [Default] → Trade Manager Biz → Trade Data DB

UI Composition And Scripting Port
Business Logic Port (e.g. events, methods)
Data Access Port
Component (Re)Configuration

User port

Trade Browser

Trade Browser UI [Custom]
Trade Manager Biz
Trade Data DB

UI Composition And Scripting Port
Business Logic Port (e.g. events, methods)
Data Access Port

Component (Re)Configuration

User port

Trade Browser

Trade Browser UI [Custom]

User port

Trade Browser

Trade Data DB

UI Composition And Scripting Port

Business Logic Port (e.g. events, methods)

Data Access Port

Component (Re)Configuration

User port

Trade Browser

Trade Browser
UI [Custom]

UI Composition
And Scripting Port

Biz
Risk Allocator
[Custom]

Business Logic Port
(e.g. events, methods)

Trade Data
DB

Data Access Port
Many large components should be customizable to variable contexts.

- Customization by binding simple properties, or entire sub-components.
- Uses custom binding information in factory or creation operations.
- Language to express configuration or dynamic re-configuration and evolution.
Architecture and Style
Is This an “Architecture”?
Is This an “Architecture”?
Is This an "Architecture"?

- Button
  - pressed
  - start
  - stop
  - toggled

- Switch
  - reactor

- Thermometer
  - temp
  - value
  - in

- Threshold
  - limit
  - out
  - in

- Differentiator
  - gradient
  - in

- OR
  - in1
  - in2
  - out

- Alarm
  - position
  - limit

<<event>>
<<prop>>
Is This an “Architecture”? 

This is an abstract view of the implementation
This is an abstract view of the implementation

- It uses the language of properties, events, methods
  - With <<prop>>, <<event>> connectors between these "connection points"
This is an abstract view of the implementation

- it uses the language of properties, events, methods
  - with <<prop>>, <<event>> connectors between these “connection points”
- it maps to corresponding patterns in the Java code
Is This an “Architecture”?

This is an abstract view of the implementation
  - it uses the language of properties, events, methods
    • with <<prop>>, <<event>> connectors between these “connection points”
  - it maps to corresponding patterns in the Java code

This is an instance of the Java Beans architecture style covering design and code
Architecture and “Style”
Architecture and “Style”

Architecture Style
(design elements, rules, constraints)

specification

<<refines>>

realization
Architecture and “Style”

Architecture Style
(design elements, rules, constraints)

<<refines>>

<<conforms>>

specification

<<refines>>

realization
Architecture and “Style”

Architecture Style
(design elements, rules, constraints)

<<refines>>

<<conforms>>

specification

<<refines>>

realization
Architecture and “Style”

Architecture Style
(design elements, rules, constraints)

specification

<<conforms>>

realization

<<refines>>
Architecture and “Style”

Architecture Style
(design elements, rules, constraints)

<<conforms>>

specification

<<refines>>

realization
Architecture and “Style”

- Range of options for “conformance”
  - Fully defined translation (compiler)
  - Completely ad-hoc (or “creative”)
  - Some defined rules and constraints
Architecture and “Style”

- Range of options for “conformance”
  - Fully defined translation (compiler)
  - Completely ad-hoc (or “creative”)
  - Some defined rules and constraints

- Architecture style defined in separate package
- In general, realization **refines** specification in a way that **conforms** to the architecture style
- Style constrains realization and/or refinement
Conform vs. Refine : Form vs. Meaning

Specification

<<refine>>

Design / Impl
Conform vs. Refine : Form vs. Meaning

Architecture styles constrain the **form** of the architecture models and designs.
Conform vs. Refine: Form vs. Meaning

Architecture styles constrain the **form** of the architecture models and designs.

- Physical Architecture Style
- Process Architecture Style

<<conforms>>

<<refine>>

<<refine>>

<<refine>>

<<refine>>

Specification

Design / Impl
Conform vs. Refine: Form vs. Meaning

Architecture styles constrain the **form** of the architecture models and designs.

Specification and architecture models constrain the **meaning** of a design.
Viewpoints and Concerns
Viewpoints ‡ Interlinked meta-data

- **Settlement**
  - Software independent Business model
  - Platform independent Component model
  - Platform Specific

- **Deal Capture**
  - Software independent Business model
  - Platform independent Component model
  - Platform Specific

Goals: goal1, goal2, goal3
Viewpoints † Interlinked meta-data
Viewpoints ‡ Interlinked meta-data

1. How do deal-capture and settlement influence Goal: Max Share of Client Wallet

Settlement

- Software independent Business model
- Platform independent Component model
- Platform Specific

Deal Capture

- Software independent Business model
- Platform independent Component model
- Platform Specific

Deployment
Viewpoints † Interlinked meta-data

1. How do deal-capture and settlement influence Goal: Max Share of Client Wallet

2. How do the two processes and information models interact?

Deal Capture
- Software independent Business model
- Platform independent Component model
- Platform Specific

Settlement
- Software independent Business model
- Platform independent Component model
- Platform Specific

Deployment

1. How do deal-capture and settlement influence **Goal**: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?
Viewpoints † Interlinked meta-data

1. How do deal-capture and settlement influence Goal: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?

4. Which parts of the process and info are here?

Deployment

Software independent
Business model

Platform independent
Component model

Platform Specific

Settlement

Deal Capture

Goals

goal1

goal2

goal3

Viewpoints ‡ Interlinked meta-data

1. How do deal-capture and settlement influence **Goal**: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?

4. Which parts of the process and info are here?

5. How is that represented in a VSAM database?
Viewpoints † Interlinked meta-data

1. How do deal-capture and settlement influence Goal: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?

4. Which parts of the process and info are here?

5. How is that represented in a VSAM database?

6. How is that represented in an Oracle db?

Deployment

Settlement

Deal Capture

Software independent
Business model

Platform independent
Component model

Platform Specific

Viewpoints à Interlinked meta-data

Viewpoints † Interlinked meta-data

1. How do deal-capture and settlement influence **Goal**: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?

4. Which parts of the process and info are here?

5. How is that represented in a VSAM database?

6. How is that represented in an Oracle db?

7. Where is it deployed?

Viewpoints ‡ Interlinked meta-data

Clear relationship between viewpoints is the key
- Goal ‡ process ‡ logical component ‡ physical components

1. How do deal-capture and settlement influence Goal: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?

4. Which parts of the process and info are here?

5. How is that represented in a VSAM database?

6. How is that represented in an Oracle db?

7. Where is it deployed?
Viewpoints ‡ Interlinked meta-data

Clear relationship between viewpoints is the key
- Goal ‡ process ‡ logical component ‡ physical components

1. How do deal-capture and settlement influence Goal: Max Share of Client Wallet

2. How do the two processes and information models interact?

3. Which parts of the process and info are here?

4. Which parts of the process and info are here?

5. How is that represented in a VSAM database?

6. How is that represented in an Oracle db?

7. Where is it deployed?

8. “Horizontal” and “Vertical” integration are both necessary

Multiple Concerns across Viewpoints
## Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Business Architecture</strong></td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td><strong>2. Application Architecture – Black Box</strong></td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td><strong>3. Application Architecture – White Box</strong></td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>
Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>2. Application Architecture – Black Box</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture – White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>
## Multiple Concerns across Viewpoints

### Logical Functionality: what actions with what information

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Business Architecture</strong></td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td><strong>2. Application Architecture – Black Box</strong></td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td><strong>3. Application Architecture – White Box</strong></td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>
Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Concerns</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Security</td>
<td>who can and cannot do what and to whom</td>
</tr>
<tr>
<td></td>
<td>Logical Functionality</td>
<td>what actions with what information</td>
</tr>
<tr>
<td>2. Application Architecture</td>
<td></td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>– Black Box</td>
<td></td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture</td>
<td></td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
<tr>
<td>– White Box</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Concerns</th>
<th>Outside Boundary</th>
<th>Inside</th>
<th>1. Business Architecture</th>
<th>Business perspective on the problem or concern, including goals, activities, interactions, and information.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Application Architecture – Black Box</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Application Architecture - White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>

**Performance**: throughput, response time, latency, data volumes, loads

**Security**: who can and cannot do what and to whom

**Logical Functionality**: what actions with what information
# Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>2. Application Architecture – Black Box</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture – White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>

**Concerns**
- **Logical Functionality**: what actions with what information
- **Performance**: throughput, response time, latency, data volumes, loads
- **Security**: who can and cannot do what and to whom
- **Technical**: technical and platform realizations of components, ports, connectors

**Outside**

**Boundary**

**Inside**
# Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>2. Application Architecture – Black Box</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture – White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>

**Concerns**

- **Logical Functionality**: what actions with what information
- **Performance**: throughput, response time, latency, data volumes, loads
- **Security**: who can and cannot do what and to whom

**Technological**: technical and platform realizations of components, ports, connectors

**Deployment**: locations, connectivity of people, processes, hardware, software
Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>2. Application Architecture – Black Box</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture - White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>

- **Module**: build-time structure of units including sources and tools
- **Deployment**: locations, connectivity of people, processes, hardware, software
- **Technical**: technical and platform realizations of components, ports, connectors
- **Performance**: throughput, response time, latency, data volumes, loads
- **Security**: who can and cannot do what and to whom
- **Logical Functionality**: what actions with what information
Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>2. Application Architecture – Black Box</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture - White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>

- Address all relevant concerns
- Early in the process
- At the appropriate viewpoint and fractal level
- In a way that can be usefully composed
## Multiple Concerns across Viewpoints

<table>
<thead>
<tr>
<th>Viewpoint</th>
<th>Concerns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Business Architecture</td>
<td>Business perspective on the problem or concern, including goals, activities, interactions, and information.</td>
</tr>
<tr>
<td>2. Application Architecture</td>
<td>Describe a large grained software component as a black box, with its behavior and information responsibility to others in its environment, for any aspect that is externally visible.</td>
</tr>
<tr>
<td>3. Application Architecture - White Box</td>
<td>Design the “insides” of a black box component as a collection of connected sub-components, each a black-box, and decide how they collaborate to fulfill black box responsibilities.</td>
</tr>
</tbody>
</table>

- Address all relevant concerns
- Early in the process
- At the appropriate viewpoint and fractal level
- In a way that can be usefully composed
Foundation – Fractal Inter-Model Relationships
Foundation – Fractal Inter-Model Relationships

t  Two different viewpoints on the same system or component
    – Their overlap is modeled – common objects, events, attributes, etc.
Foundation – Fractal Inter-Model Relationships

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.

```
Package P1
System from viewpoint 1

Package P2
System from viewpoint 2

Package P3
Interrelationship between P1 and P2
```
Foundation – Fractal Inter-Model Relationships

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.

- Two different levels of abstraction of the same system
  - The refinement relationship between levels is modeled
**Foundation – Fractal Inter-Model Relationships**

- Two different viewpoints on the same system or component
  - Their overlap is modeled – common objects, events, attributes, etc.

- Two different levels of abstraction of the same system
  - The refinement relationship between levels is modeled
Multiple Concerns: Development Process

What are sensible development methods and processes when working with models of multiple viewpoints and concerns?
Refinement
A Missing Piece in Architecture: Rationale
A Missing Piece in Architecture: Rationale

- Rationale = WHY
A Missing Piece in Architecture: Rationale

• Rationale = WHY

• Question form:
  
  Why did I do <X>
A Missing Piece in Architecture: Rationale

- **Rationale = WHY**

- **Question form:**
  
  Why did I do \(<X>\)

- **Answer form:**

  Because if I do \(<X>\) (and \(<Y>\), and avoid doing \(<Z>\))
  
  then I will accomplish \(<A>\)

  (and because this works better on \(<NFR>\) than alternatives P, Q)
A Missing Piece in Architecture: Rationale

**Rationale** = WHY

**Question form:**

Why did I do <X>

**Answer form:**

Because if I do <X> (and <Y>, and avoid doing <Z>)
then I will accomplish <A>

(and because this works better on <NFR> than alternatives P, Q)

Rationale **R** is associated with Refinement
A Missing Piece in Architecture: Rationale

- Rationale = WHY

- Question form:

  Why did I do <X>

- Answer form:

  Because if I do <X> (and <Y>, and avoid doing <Z>)
  then I will accomplish <A>

  (and because this works better on <NFR> than alternatives P, Q)

- Rationale R is associated with Refinement

- There are (should be) many Refinement Patterns
A Missing Piece in Architecture: Rationale

- Rationale = WHY

- Question form:
  Why did I do <X>

- Answer form:
  Because if I do <X> (and <Y>, and avoid doing <Z>)
  then I will accomplish <A>

  (and because this works better on <NFR> than alternatives P, Q)

- Rationale R is associated with Refinement

- There are (should be) many Refinement Patterns
A Missing Piece in Architecture: Rationale

- Rationale = WHY

- Question form:
  Why did I do <X>?

- Answer form:
  Because if I do <X> (and <Y>, and avoid doing <Z>)
  then I will accomplish <A>

  (and because this works better on <NFR> than alternatives P,Q)

- Rationale R is associated with Refinement

- There are (should be) many Refinement Patterns
Foundation – Refinement of both data and protocol
Foundation – Refinement of both data and protocol

- Abstract multiple objects as a single larger-grained object
Foundation – Refinement of both data and protocol

- Abstract multiple objects as a single larger-grained object

(a) Zooming in/out – objects
Foundation – Refinement of both data and protocol

- Abstract multiple objects as a single larger-grained object

(a) Zooming in/out – objects

- Abstract detailed interaction protocols as a single action
Foundation – Refinement of both data and protocol

- Abstract multiple objects as a single larger-grained object

(a) Zooming in/out – objects

- Abstract detailed interaction protocols as a single action

(b) Zooming in/out – interactions
Platform Realization of Connected Components

```
Deal Mgmt  Deal Events  Facility Methods  Facility Mgmt
          «event-method»
```

Platform Realization of Connected Components

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
Platform Realization of Connected Components

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
- «Java»: Events realized in Java event model
- Connect Java events to C methods
- «C»: Methods realized in C

Deal Mgmt
Deal Events
Facility Methods
Facility Mgmt
«event-method»
Platform Realization of Connected Components

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

- «Java»: Events realized in Java event model
- «C»: Methods realized in C
- Connect Java events to C methods
Platform Realization of Connected Components

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
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?
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

To realize event \( S::e(T) \) in Java:
- method \( S::add_eListener(x) \)
- method \( S::remove_eListener(x) \)
- 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?
Requirements with Fractal Components
Requirements with Fractal Components

- Michael Jackson
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames
Michael Jackson
- Requirements ≠ interface specification
- Domains, shared phenomena
- One machine, requirements on machine
- Problem frames, composite frames

Figure 2. The commanded behaviour frame
Requirements with Fractal Components

Michael Jackson
- Requirements ≠ interface specification
- Domains, shared phenomena
- One machine, requirements on machine
- Problem frames, composite frames

KAOS
- Goal-based requirements
- Agents, elements controlled and observed by goals
- Multiple agents, allocation of responsibility to agents
- Goal refinements and refinement patterns

Figure 2. The commanded behaviour frame
Requirements with Fractal Components

- Michael Jackson
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames

- KAOS
  - Goal-based requirements
  - Agents, elements controlled and observed by goals
  - Multiple agents, allocation of responsibility to agents
  - Goal refinements and refinement patterns

Figure 2. The commanded behaviour frame
Requirements with Fractal Components

Michael Jackson
- Requirements ≠ interface specification
- Domains, shared phenomena
- One machine, requirements on machine
- Problem frames, composite frames

KAOS
- Goal-based requirements
- Agents, elements controlled and observed by goals
- Multiple agents, allocation of responsibility to agents
- Goal refinements and refinement patterns

Fractal agents, joint goals and actions, refinement patterns
Requirements with Fractal Components

- **Michael Jackson**
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames

- **KAOS**
  - Goal-based requirements
  - Agents, elements controlled and observed by goals
  - Multiple agents, allocation of responsibility to agents
  - Goal refinements and refinement patterns

- Fractal agents, joint goals and actions, refinement patterns
Requirements with Fractal Components

- **Michael Jackson**
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames

- **KAOS**
  - Goal-based requirements
  - Agents, elements controlled and observed by goals
  - Multiple agents, allocation of responsibility to agents
  - Goal refinements and refinement patterns

- Fractal agents, joint goals and actions, refinement patterns

---

Figure 2. The commanded behaviour frame
Requirements with Fractal Components

- Michael Jackson
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames

- KAOS
  - Goal-based requirements
  - Agents, elements controlled and observed by goals
  - Multiple agents, allocation of responsibility to agents
  - Goal refinements and refinement patterns

- Fractal agents, joint goals and actions, refinement patterns
Requirements with Fractal Components

- Michael Jackson
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames

- KAOS
  - Goal-based requirements
  - Agents, elements controlled and observed by goals
  - Multiple agents, allocation of responsibility to agents
  - Goal refinements and refinement patterns

- Fractal agents, joint goals and actions, refinement patterns

Figure 2. The commanded behaviour frame
Requirements with Fractal Components

- Michael Jackson
  - Requirements ≠ interface specification
  - Domains, shared phenomena
  - One machine, requirements on machine
  - Problem frames, composite frames

- KAOS
  - Goal-based requirements
  - Agents, elements controlled and observed by goals
  - Multiple agents, allocation of responsibility to agents
  - Goal refinements and refinement patterns

- Fractal agents, joint goals and actions, refinement patterns
Enterprise Architecture, Evolution, Roadmaps
The most financially massive decisions at the enterprise level are not about specifying or building a piece of software.
The most financially massive decisions at the enterprise level are not about specifying or building a piece of software.

Assess: What is the current state of the enterprise architecture? What are the goals and policies to be met?
Enterprise Architecture, Evolution, Roadmaps

- The most financially massive decisions at the enterprise level are *not* about specifying or building a piece of software.

- Assess: What is the current state of the enterprise architecture? What are the goals and policies to be met?

- Diagnose: what are the pain points, root causes of problems?
The most financially massive decisions at the enterprise level are not about specifying or building a piece of software.

- Assess: What is the current state of the enterprise architecture? What are the goals and policies to be met?
- Diagnose: what are the pain points, root causes of problems?
- Propose: what are potential solutions, build-or-buy, migration?
The most financially massive decisions at the enterprise level are not about specifying or building a piece of software.

Assess: What is the current state of the enterprise architecture? What are the goals and policies to be met?

Diagnose: what are the pain points, root causes of problems?

Propose: what are potential solutions, build-or-buy, migration?

Build: specify, build, integrate, test, deploy
Enterprise Architecture, Evolution, Roadmaps

- The most financially massive decisions at the enterprise level are not about specifying or building a piece of software.
- Assess: What is the current state of the enterprise architecture? What are the goals and policies to be met?
- Diagnose: what are the pain points, root causes of problems?
- Propose: what are potential solutions, build-or-buy, migration?
- Build: specify, build, integrate, test, deploy

- What is π-calculus for enterprise architecture problems?