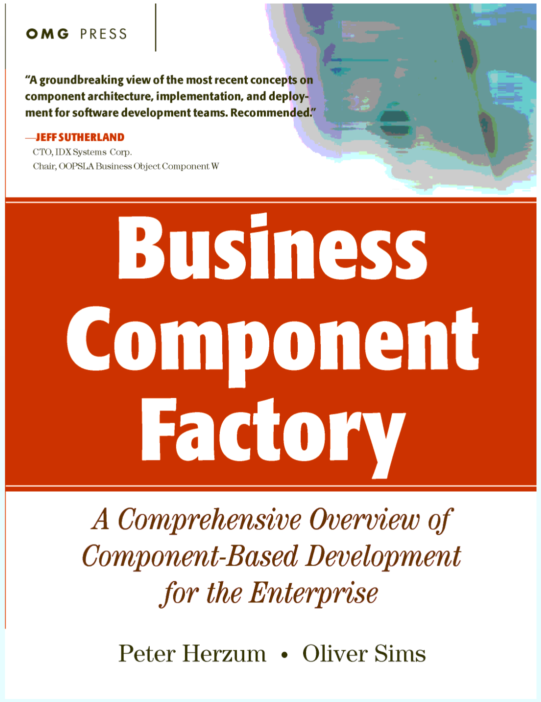
HERZUM SOFTWARE

COSM™: An Approach to Service Oriented Architectures and Federations of Business Systems

Peter Herzum

CTO & Software Ecologist

Herzum Software



OMG PRESS

"A groundbreaking view of the most recent concepts on component architecture, implementation, and deployment for software development teams. Recommended."

—JEFF SUTHERLAND
CTO, IDX Systems Corp.
Chair, OOPSLA Business Object Component W

Business Component Factory

A Comprehensive Overview of Component-Based Development for the Enterprise

Peter Herzum • Oliver Sims

HERZUM SOFTWARE

Presentation Material Legal Copyright Notice

Business Perspective: Examples

- **Intra-Enterprise Integration: Large companies looking into webservices as a way of reducing the costs and complexity of their systems portfolio**
  - **E.g. American Express, Boeing, …**
- **Integration centered around a single enterprise: Large organizations or portals can drive their partners in the value chain to align themselves with a common webservices architecture and approach**
  - **E.g. Amazon.com, General Motors, IBM**
  - **Can include Value Added Services (VAS) provided by an "integrator enterprise" or a portal based on services provided by many 3rd parties.**
    - ***Basic service providers do not necessarily know who provides the VAS, and yet need to be "good citizens" of the federation. Alternatively, basic service providers target a specific portal or integrator***
    - ***E.g. F.E.T.I.S.H. (Federated European Tourism Information System Harmonization) project***
- **Multi-Enterprise Collaboration of smaller or medium enterprises (SMEs): Clusters of SMEs in various forms of collaboration**
- **Product Suite Development: Large product vendors are looking into webservices and components as one way of reducing software supply chain costs and reducing time to market**
- **Each business case may require different architectural approaches, standards, methodologies, organizations, technologies, …**

About this Presentation

- **Objective: share with audience our experience extending COSM™ to federation of business systems (an ever-evolving process…)**
- **Many simplifications, including**
  - **Focus on functional services (not on platform services such as basic registry services)**
  - **Ignoring the important aspects of business strategy and business modeling**
  - **Focus on enterprise-level, ignoring user interface aspects and portal aspects**

Intra-Enterprise Integration: Methodology Requirements

- **Characteristics: Possible (but complex) enterprise-level strategic direction, architectural decision making, architectural management, enterprise-level integration, metrics, etc.**
- **Must have strong focus on enterprise-level architecture, architecture processes, and architectural management**
- **Must address legacy systems**
- **Must address both interoperability architecture and internal system architecture**
- **Must support enterprise-level dependency modeling**
- **Must address whole software supply chain**
- **Overall data architecture of critical importance**
- **(Webservices only one of many tools the architects can use to achieve their objectives)**

HERZUM SOFTWARE

Product-Suite Development: Methodology Requirements

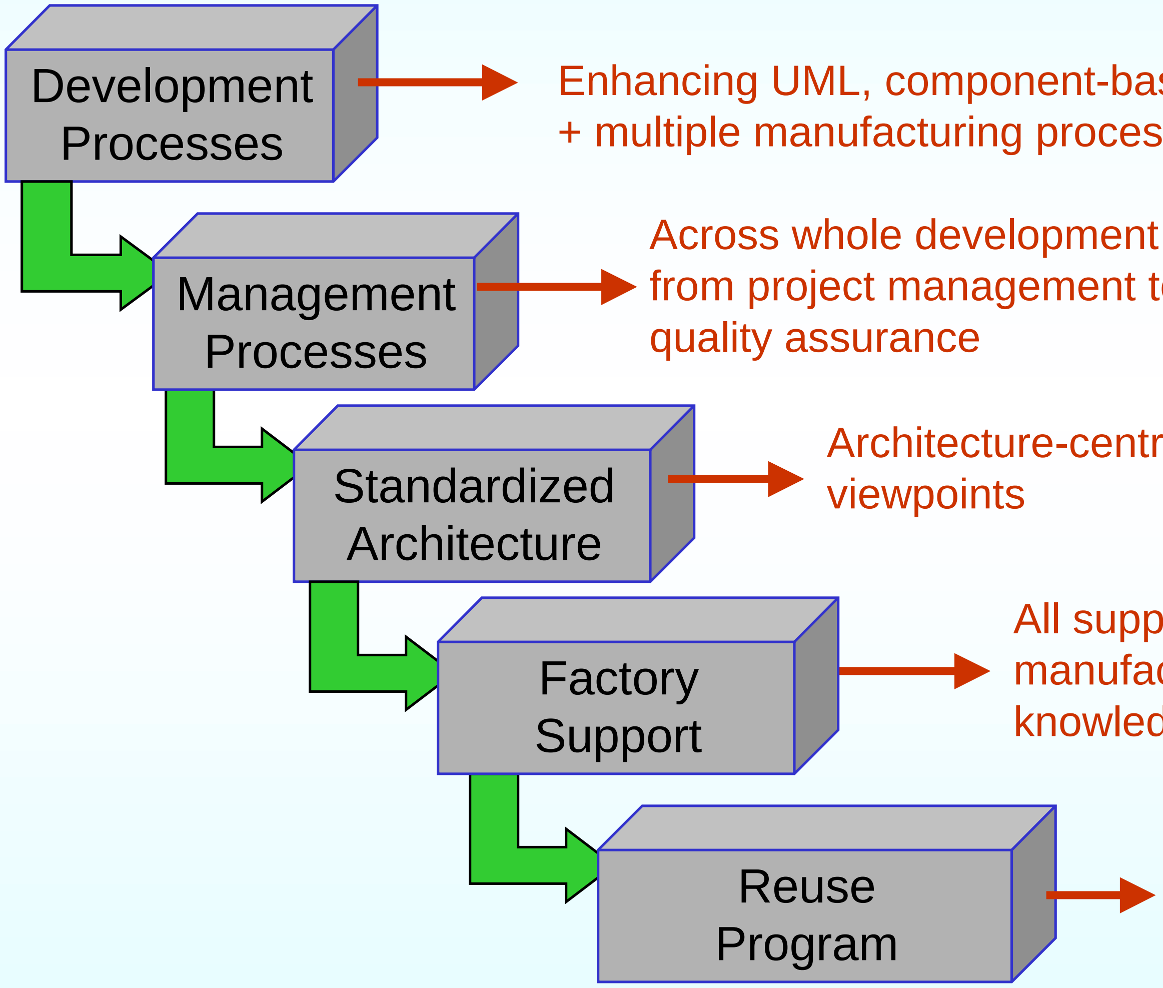
- Characteristics: (similar to intra-enterprise) Possible (but very complex) enterprise-level strategic direction, architectural decision making, architectural management, enterprise integration, metrics, etc.
  - Everything is much more complex because it is like building a <u>generic</u> set of interoperable systems
- Must have strong focus on enterprise-level product architecture, architecture processes, and architectural management
- Must address legacy products
- Must address both interoperability architecture and internal system architectures
- Must support enterprise-level dependency modeling
- Must address whole software supply chain
- Must address migration strategies and backward compatibility. Strong emphasis
- (Webservices only one of <u>many</u> tools the architects can use to achieve their objectives)

Multi-Enterprise Collaboration: Methodology Requirements

- Characteristics: Assuming a set of loosely-connected small enterprises in some form of partnership to provide some business service, it is very difficult to have common strategic direction, architectural decision making, common implementation technologies, architectural management, integration, metrics, etc.
  - Easier if assuming one integrator or portal
- "Architecture by committee"
- Focus on interface or software contract specification, possibly on interoperability architecture
- Overall information exchange of critical importance
- Security aspect of critical importance
- (Webservices becoming a prominent tool for achievement of architectural objectives)

Agile Software Manufacturing

> Successful software manufacturers optimize these elements as a whole



Enhancing UML, component-based, methodology framework + multiple manufacturing processes, formal, multiples routes

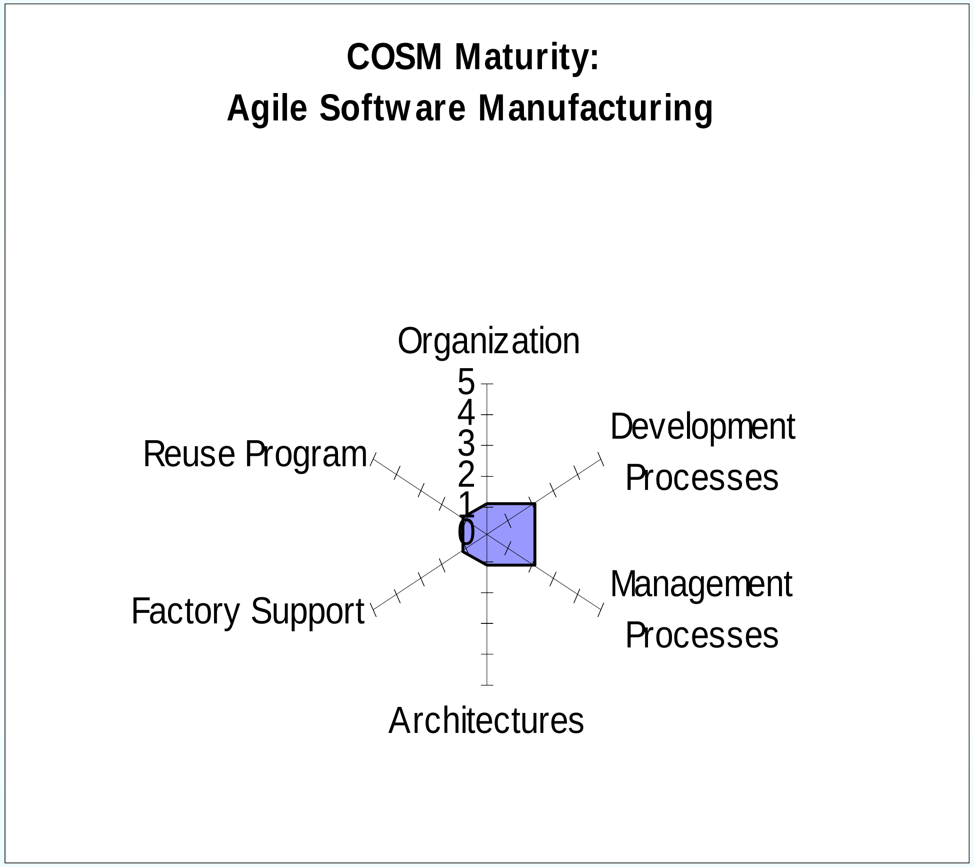
Across whole development lifecycle, CMM-inspired, from project management to process engineering through quality assurance

Architecture-centric approach across all viewpoints

All support required by an agile software manufacturing capability: management of knowledge, issues, time, education,...

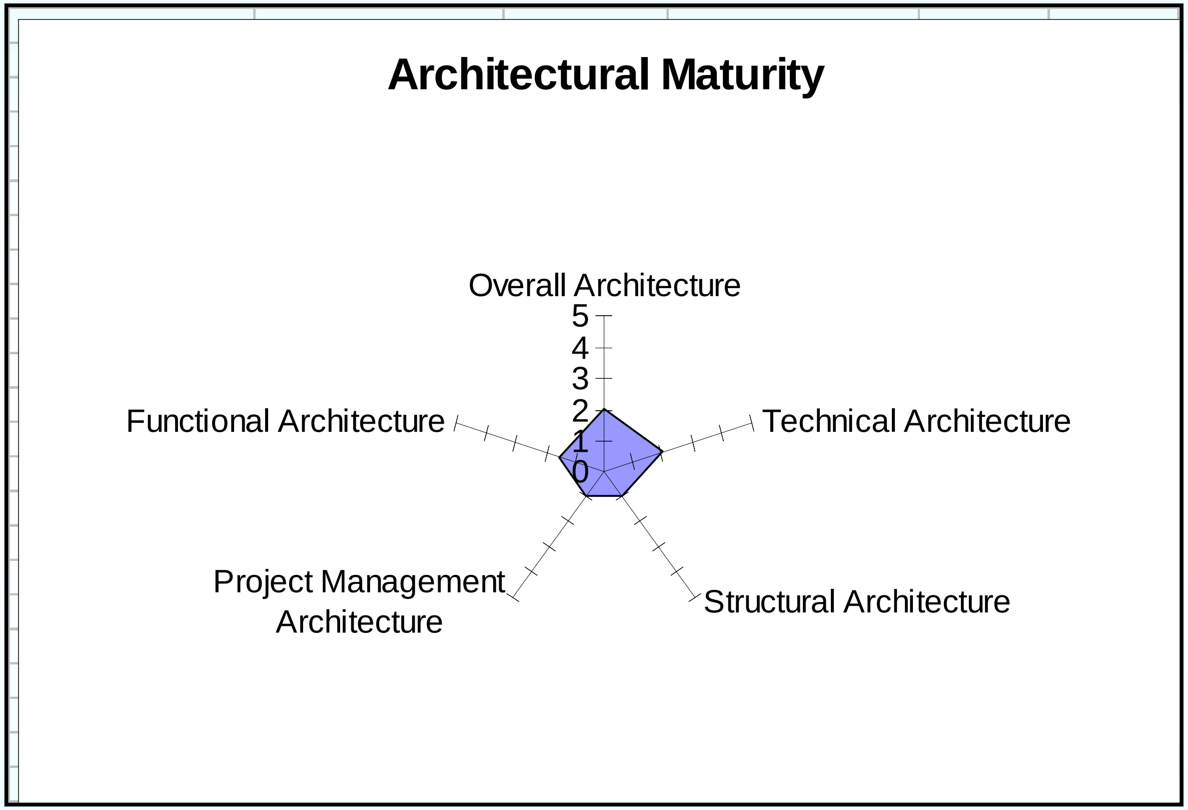
Reuse of components, frameworks, solutions across the lifecycle and the architectural viewpoints

COSM Maturity Model



- **Extends SEI CMM as a superset. SEI CMM corresponds to the <u>Management Processes</u> axis**
- **Can be applied to individual organizations or enterprise-wide**
- **Can assume specific standards (such as UML): it is more specific (hence more useful) that SEI assessment**

COSM Maturity Model: Architecture



HERZUM SOFTWARE

COSM™: Component-Oriented Software Manufacturing

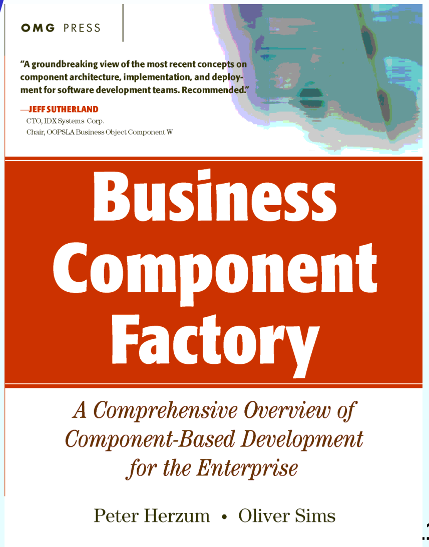
Scalable at runtime and development-time

Interoperability Reference Model

Agile & scalable, holistic, architecture-centric approach to large distributed system development, integration, and evolution
- entirely based on services and components
- standards-aligned
- 75% technology-independent
- focus on autonomy and collaborations
- naturally reuse-centric
- addressing systems and federations of business systems, webservices, product-line developments

MDA, UML, J2EE, CORBA, .Net, Webservices standards

Manufacturing, supply chain management, finance, banking, telecom, ERP


OMG PRESS

"A groundbreaking view of the most recent concepts on component architecture, implementation, and deployment for software development teams. Recommended."

—JEFF SUTHERLAND
CTO, IDX Systems Corp.
Chair, OOPSLA Business Object Component W

Business Component Factory

A Comprehensive Overview of Component-Based Development for the Enterprise

Peter Herzum • Oliver Sims

COSM™ has been applied and evolved since **1992**. It supports solution development, solutions integration, and legacy systems.

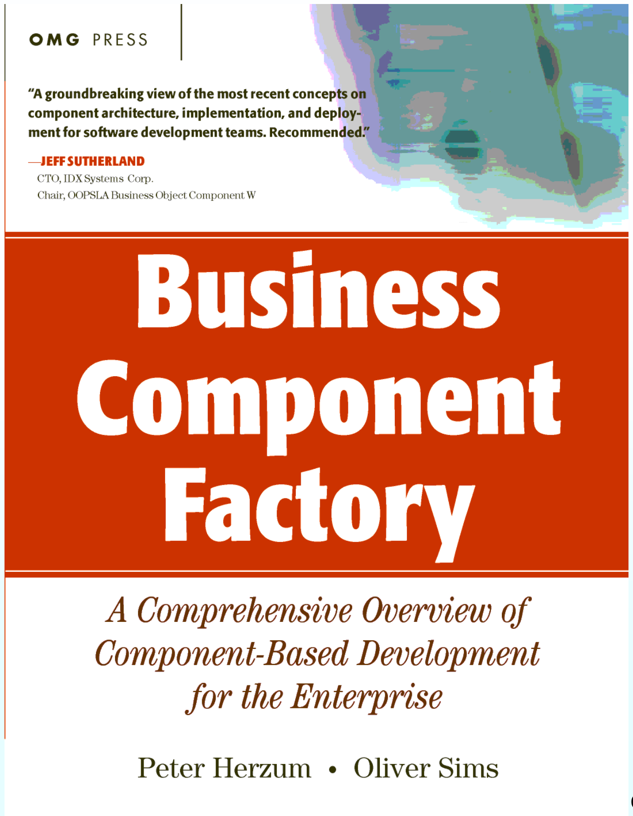
Also applied to real-time, embedded systems

HERZUM SOFTWARE

The Book

- Title: "Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise"
- Authors: Peter Herzum (principal author), Oliver Sims
- Editor: John Wiley & Sons
- Published: January 2000
- ISBN: 0-471-32760-3

www.ComponentFactory.org


OMG PRESS

"A groundbreaking view of the most recent concepts on component architecture, implementation, and deployment for software development teams. Recommended."

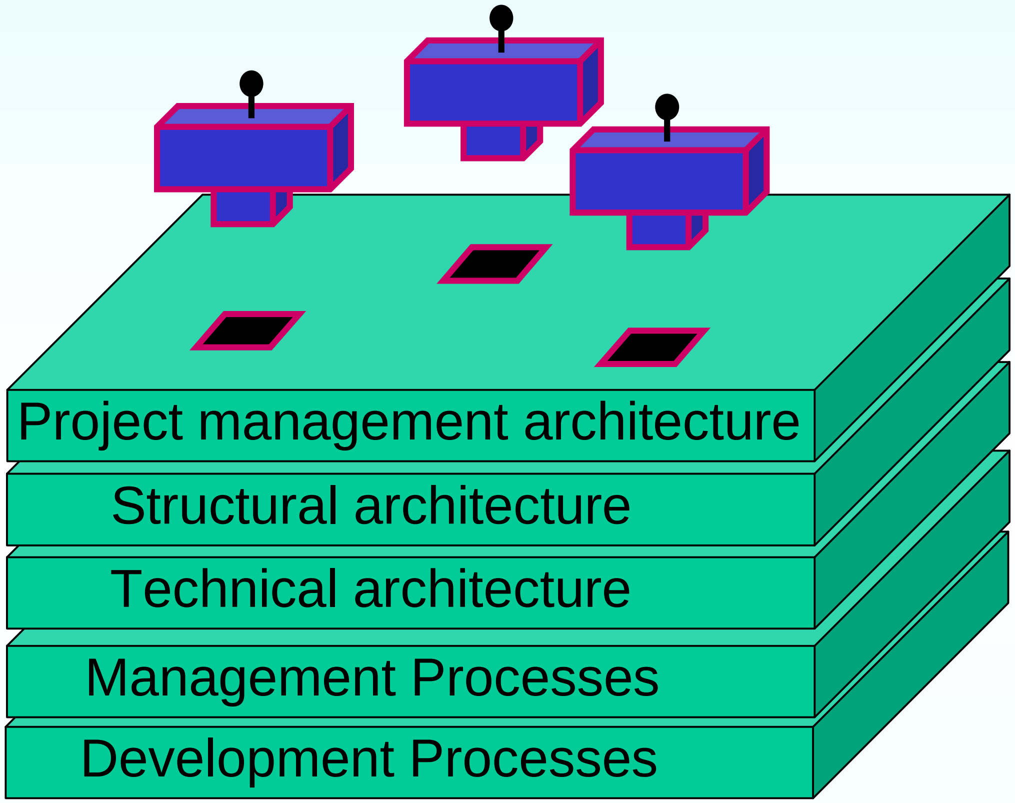
—JEFF SUTHERLAND
CTO, IDX Systems Corp.
Chair, OOPSLA Business Object Component W

Business Component Factory

A Comprehensive Overview of Component-Based Development for the Enterprise

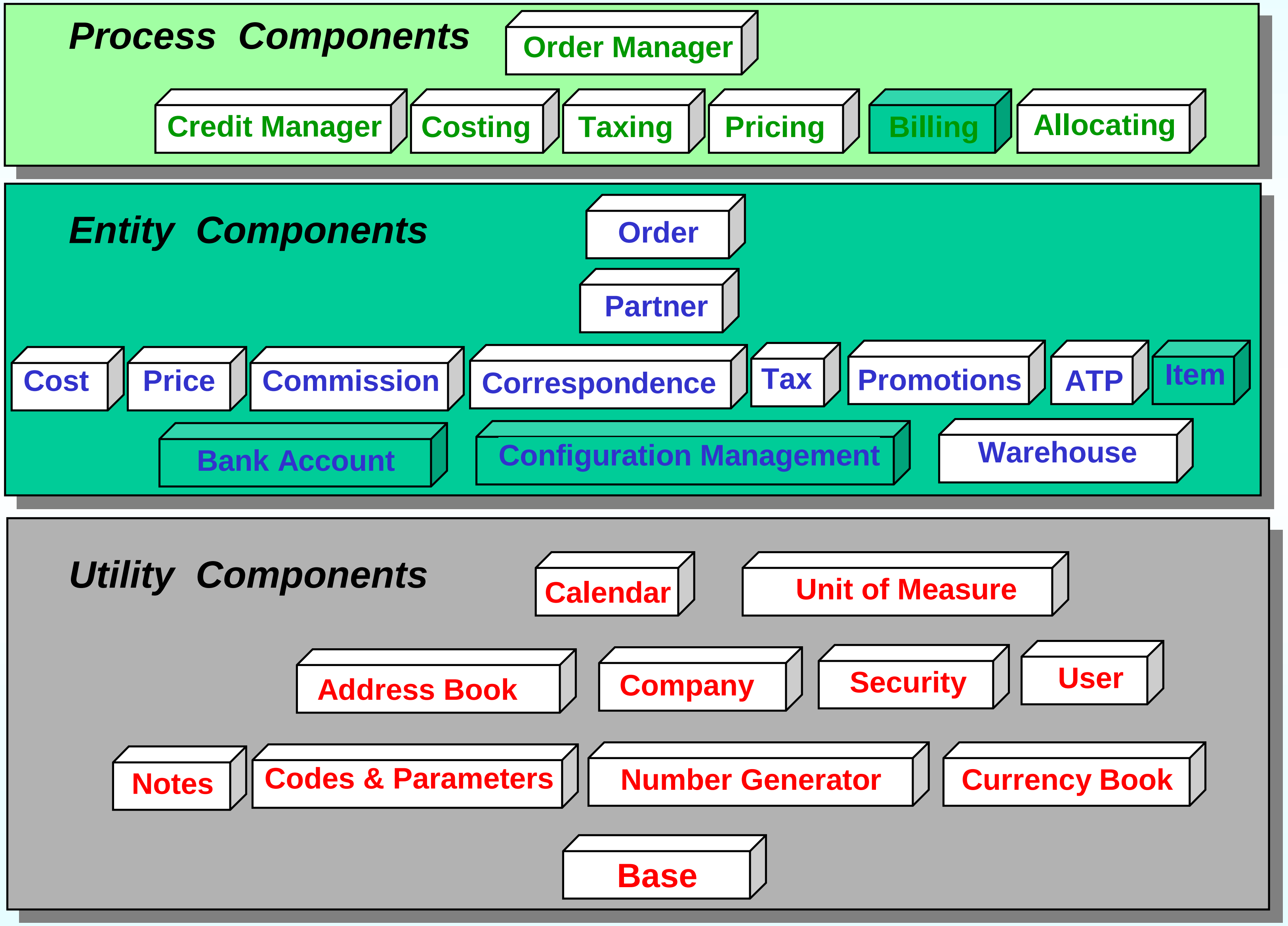
Peter Herzum • Oliver Sims

The Component “Plug”

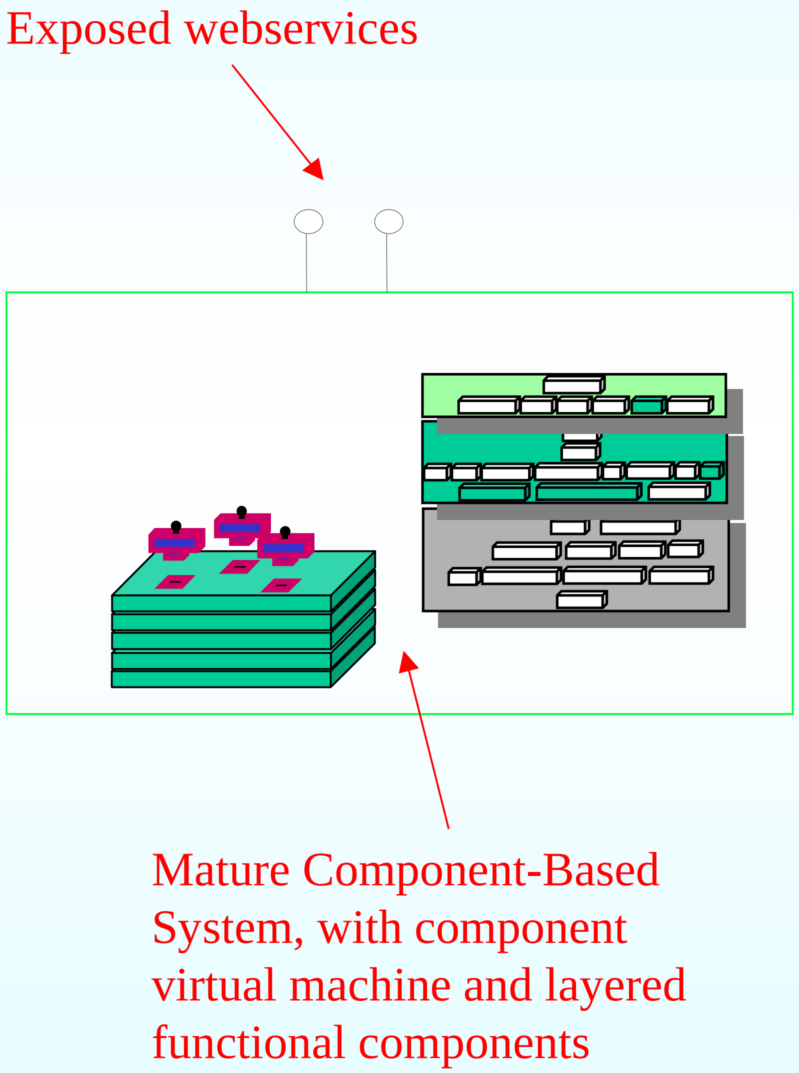


> The Component Plug is not only technical! It requires (and will require even more in the future) a set of protocols across multiple architectural viewpoints
> A “Component virtual machine” address much more than just technology

Functional Categories in CBD



Exposing Webservices


Exposed webservices
Mature Component-Based System, with component virtual machine and layered functional components

- "Somehow providing SOAP enabled interface, described in WSDL": very simple, can be automated, and has no particular methodology difficulty
  - Requires traditional specification of an interface
- But…a technology-based interface exposure will not profit of the many advantages of Service Oriented Architectures
- This is also the problem with a simplistic view of MDA approach

Service as “good citizens” of a larger community

- Shift from “providing individual services“ to “providing federation-ready services within a given functional reference model and federation ecology”
  - From “autonomous services” to “federation-aware services”
- Example: Geographical Information
  - Different providers of geographical information can use over 20 different ways of providing geographical coordinates
  - It would be beneficial for the federation to specify a few engagement rules, such as “all geographical coordinates need to be provided as absolute coordinates using standard XY”
  - Example: definitions of the common business datatype 'Address‘: are City names be used in the local language or in english? Does Venice really exists in Italy, or is it Venezia? Allowing the two names to exist open issues about naming mappings. Searching for the geografical localization of "Venezia" in Italy could return nothing

Example: Describing Webservices

In reality, what is desirable is:

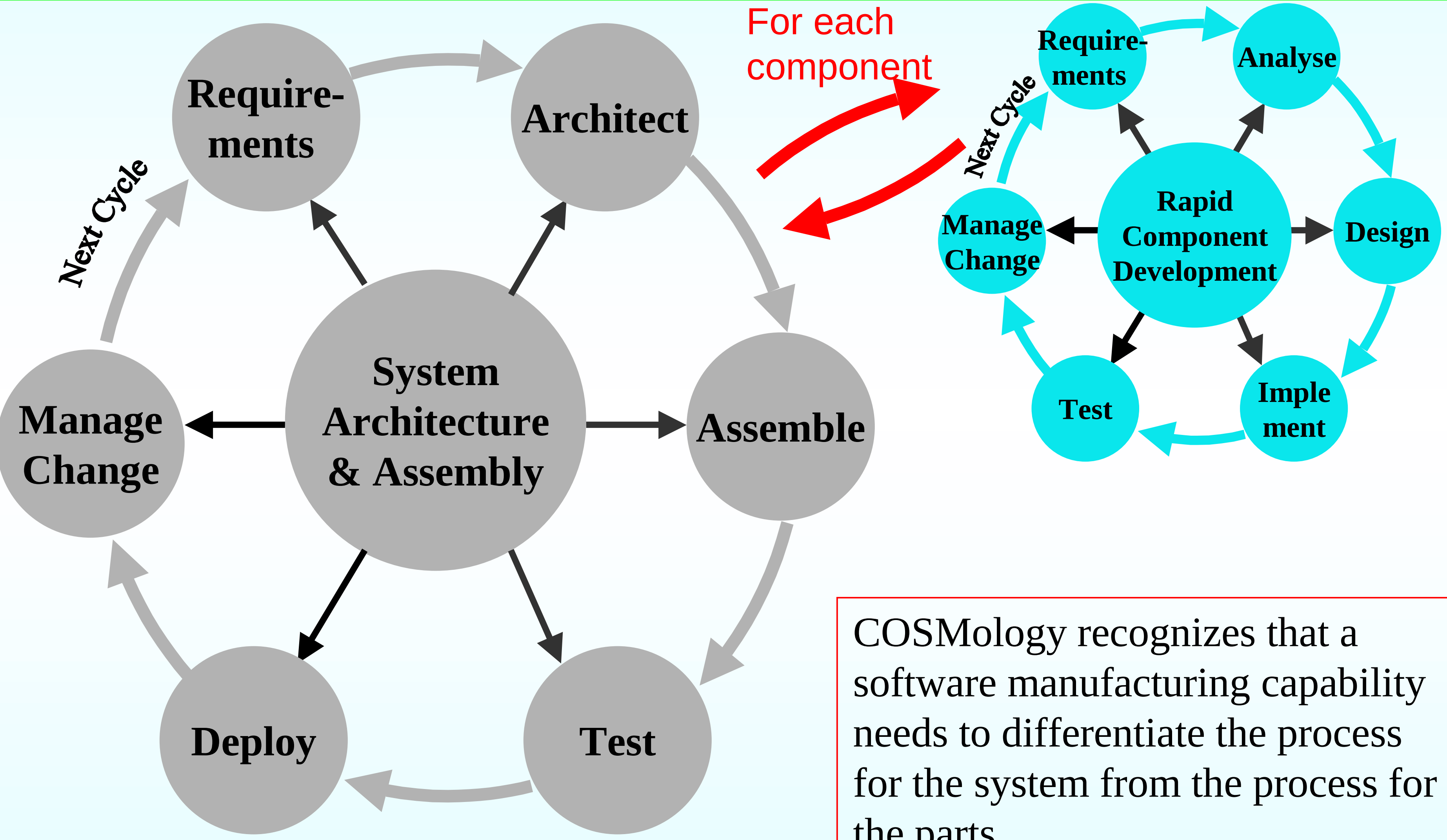
- **Pre-condition to specification: existence of reference models for:**
  - **Typical structural frameworks, such as security, error handling, transactions (!), …**
  - **Interface standards (use of specific UML profile, or WSDL profile)**
  - **Functional reference model (see later. Includes ontology)**
- **Contract specification, to include:**
  - **Interface**
  - **Context Model, context, profile (technical, business criteria, QoS, cost, …)**
  - **Pre-,Post-conditions**
  - **Contract-specific semantic, structural information, provider information, …)**
- **Approach is different for basic webservices and value added webservices**
  - **Importance dependency model for run-time management, monitoring, and even business and legal**

HERZUM SOFTWARE

Components v/s Services

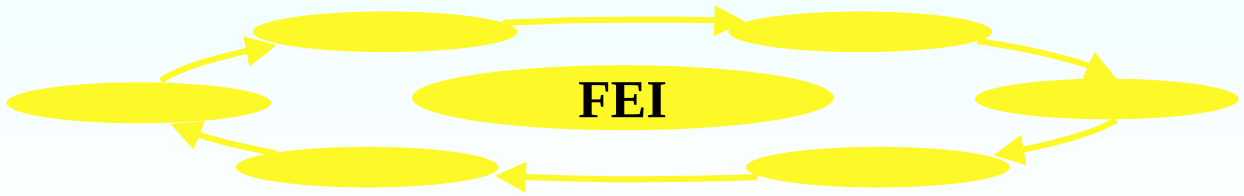
	Components	Services
Architectural Perspective	The internal asset of a system (not necessarily shown outside, externals can be webservices or not)	What is seen externally to a system (internals can be components or not)
Deployment Model	Software is physically deployed. “Install and use”	Software “exists” somewhere. “Connect and use”
Levels of Information Exchange	Mostly within enterprise	Mostly across enterprises
Coupling	Fairly loose. Based on internal standards	Very loose. Based on industry standards
Communication	Enterprise-based Protocols (like IIOP and messaging)	Internet-based Protocols (like XML over SOAP)

System Architecture & Assembly (SAA) Vs Rapid Component Development (RCD)

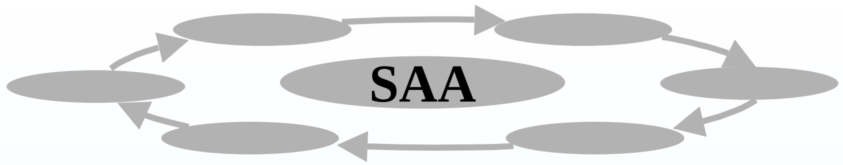


COSMology recognizes that a software manufacturing capability needs to differentiate the process for the system from the process for the parts

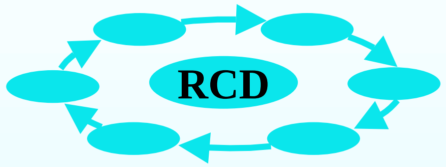
Agile Software Manufacturing Processes



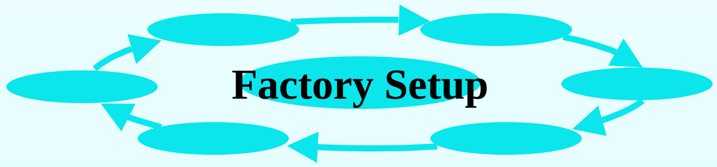
<u>Federation Ecology and Interoperation</u>, performed by *software ecologists* and *federation integrators* with eclectic functional and architectural background, working at the enterprise-level



<u>System Architecture and Assembly</u>, performed by *system architects* and *system-level developers* with eclectic functional background

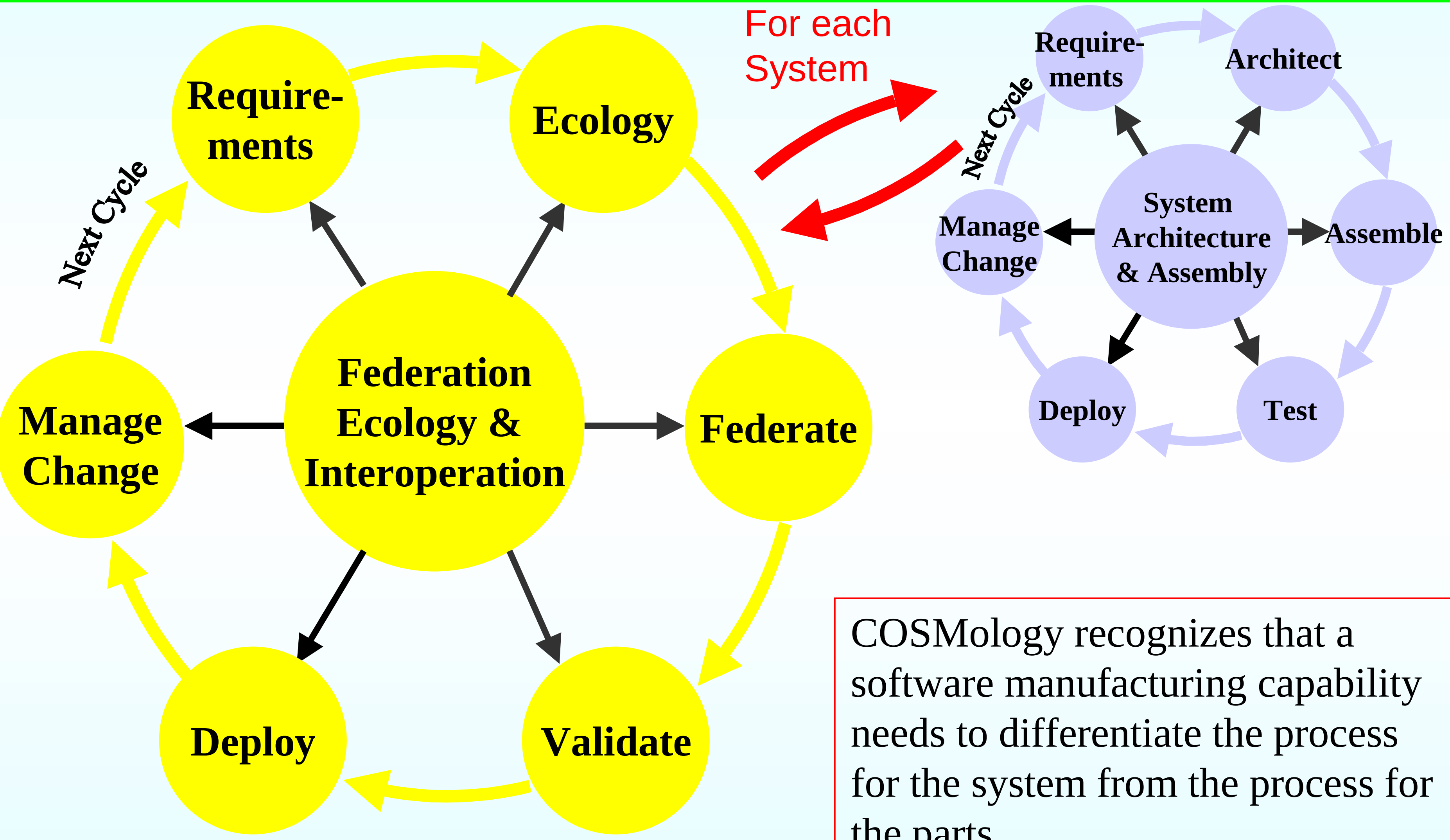


<u>Rapid Component Development</u>, performed for each component by a small team of *component designers* and *functional developers* with topic-specific functional background



<u>Factory Setup</u>, performed by *Factory Setup Specialists* (factory setup architects, skilled technical developers) working at the enterprise-level

Federation Ecology & Interoperation (FEI) Vs System Architecture & Assembly (SAA)



COSMology recognizes that a software manufacturing capability needs to differentiate the process for the system from the process for the parts

Example: Requirements

System Level: Main Deliverables

- Features Catalog
- Requirements Model (includes as a by product a glossary)
- Use Case Model
- User Interface Prototype

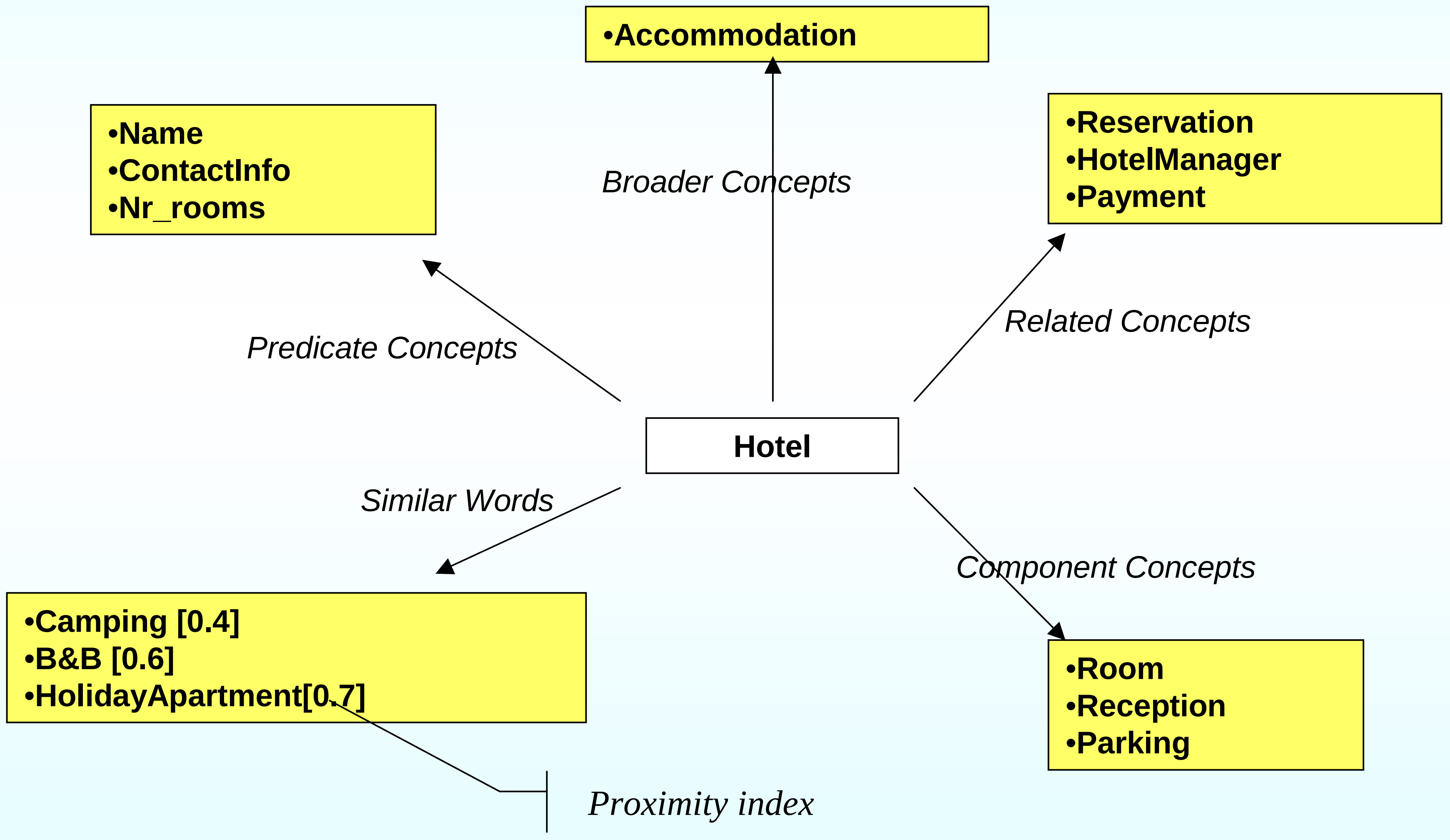
Federation Level: Main Deliverables

- Feature Catalog
- Requirements Model
- Ontology
- Business Process Model and/or Workflow Model
- Use Case Model
- (User Interface Prototype)

Ontologies

- How do ontologies fit into Agile Software Manufacturing?
  - Methodologies for ontologies: what is the best way to define them?
  - Relationship of methodologies for ontologies with business modeling, business process modeling, use cases, requirements management, etc.
  - Relationship of methodologies for ontologies with software development methodologies and management processes

OPAL Relations example



HERZUM SOFTWARE

Example: “Hotel” Concept

Hotel

Kind: Object
Definition: A place where a tourist can stay

XMLTag: <Hotel>
Code: htl

Broader: Accommodation (O)
Narrower: FarmHouse (O), Motel (O)
ComponentOf: ReceptivitySystem (O)
Component: ParkingFacilities (O), Restaurant (O), HotelRoom (O), Reception (O), Lobby(O)
Predicate: Name (IE), ContactInfo (IC), NrOfRooms (IE)

Related: Reservation (O), Payment (O), Deposit (O) HotelManager (A), Cashier (A) RoomService (A) Reserving (P), Paying (P) Billing (P)

Similar: B&B [0.6], Camping [0.4], HolidayApartment [0.7]

HERZUM SOFTWARE

Example: Architecture & Ecology

System Level: Main deliverables

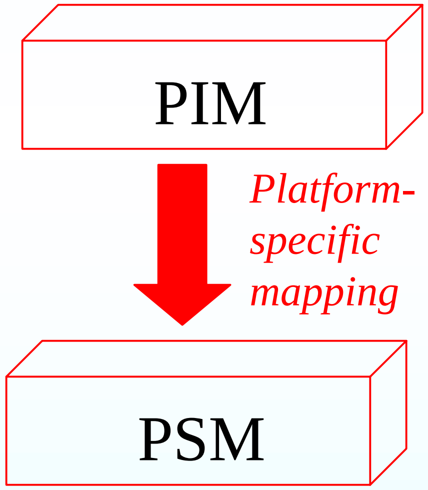
- Functional:
  - (System) Component Model
  - Enterprise Object Model
  - Persistence Model
  - Testing Model
- Technical:
  - Deployment Model

Federation Level: Main Deliverables

- Functional:
  - Functional Reference Model, Portfolio Model
  - Collaboration Models, Interoperability Models, Information Exchange Model
  - Ontology
  - Federation Object Model
  - Federation Testing Model
- Technical:
  - Repository and registry model
  - Deployment Model
- Fragmented Models…

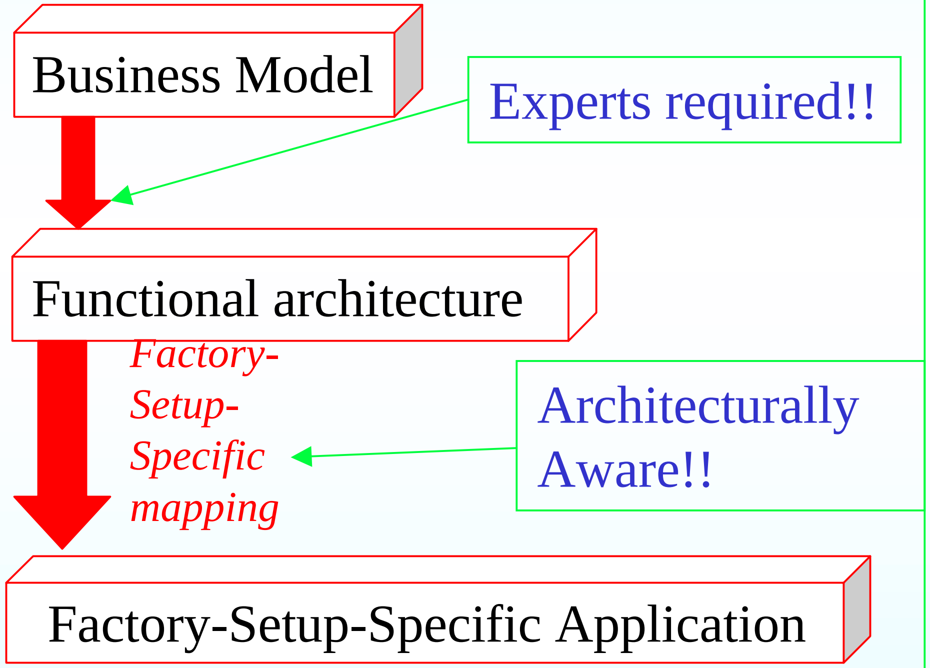
MDA Applied to different contexts

- MDA is relatively simple for small applications within one enterprise with non-stringent extra-functional requirements
- Most architectural styles will satisfy requirements



- Within single enterprise
- Less than 50 tables
- Non-stringent extra-functional requirements

- For complex situations, mapping (multiple stacks of) PIMs to the complex "platform" requires significant architectural experience



- Within single enterprise
- > 50 tables
- Stringent extra-functional requirements

Common Characteristics

- Strong focus on architectural processes
  - Definition of “setup architecture”: technical architecture, structural architecture, project management architecture
  - Definition of Functional Reference Models
- Focus on whole software supply chain, not only on development
- Collaborative software development across different (and potentially competing) organizations
- Role of outsourcing
- Everything is much more fragmented: no common architecture, no common testing, no common deployment, …
- Reduction of complexity becomes a major focus
- Webservice development as “micro-product development”

Examples of Differentiation Characteristics

- Webservices are provided by different enterprises, and the implementation could change at any time
  - The run-time for each webservices could be physically owned by and located on different servers
- Federation-level testing is nearly impossible
  - Even just organizationally
- Registry becomes a crucial architectural tool. The architectural process must include registry considerations
  - Patterns for clustering webservices into registry, combining webservices from different registries, dealing with registry security
- Security

HERZUM SOFTWARE

Time Permitting: Example of COSM™ Roadmaps for Systems and Federations