Enhancing the Interoperability between Multiagent Systems and Service-Oriented Architectures through a Model-Driven Approach

Christian Hahn, Sven Jacobi, David Raber
Saarstahl AG, DFKI GmbH

MATES 2010 – Leipzig, 27.09.2010

Outline

- Introduction
  - Motivation
  - Service-Oriented Architecture
  - SHAPE Project
    - SoaML
    - PIM4Agents
- Saarstahl SoaML Model
- Model Transformation
  - Structural Concepts
  - Behavioral Concepts (Activities)
- Conclusion

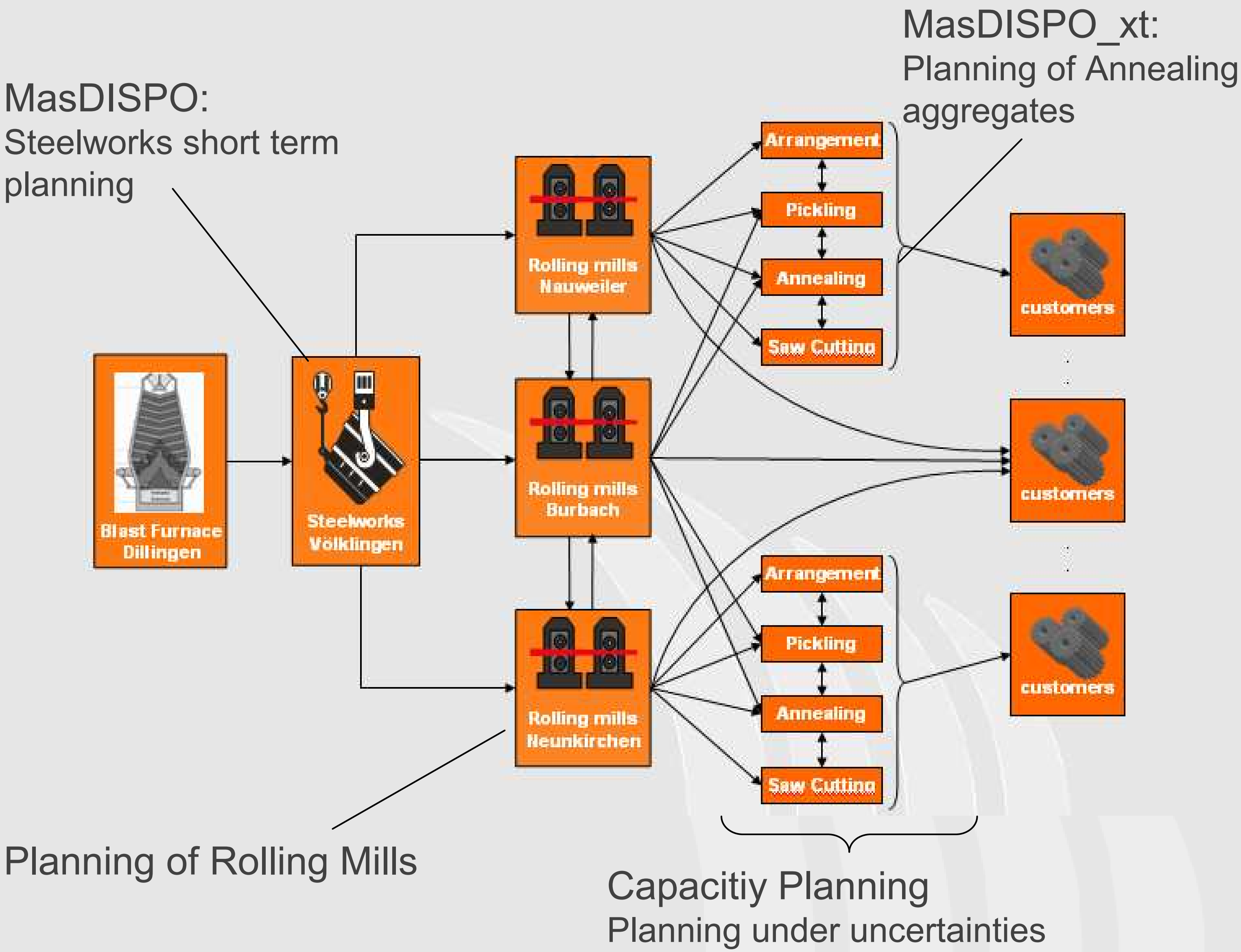
Motivation (Research point of view)

saarstahl

- The acceptance of multiagent systems (MASs) as a software engineering paradigm is increasing
- However there exist still several weaknesses limiting the impact of MASs in commercial applications
  - Most frameworks specialize on particular parts of the MAS
  - There exists no common standard
  - There exist only few development tools
- Solution:
  - Combine multiagent systems with service-oriented architectures (SOAs)

Motivation (Saarstahl point of view)

saarstahl



Service Oriented Architecture

- Technology independent architecture to solve interoperability issues
- **Idea:** Decompose a large system into single components (services)
  - Each service has a well defined interface
  - These Interfaces can be described, published, discovered and invoked over a network
  - The services are loosely coupled
- In a SOA, services can be either provided or required
- Interoperability is introduced via:
  - **Orchestration:** Expresses workflow of a single organisation and describes service composition and invocation
  - **Choreography:** Expresses public message exchange patterns and describes process logic between organisations thus enabling collaborations

Integration of SOA and MAS

- MASs and SOAs share several commonalities
- SOAs appear to be a natural environment in which agent technology can be exploited with significant advantages
- Agents can be used to encapsulate services

- Following the recent trend, principles of model-driven engineering (MDE) are used to integrate SOAs and MASs into an overall software engineering process

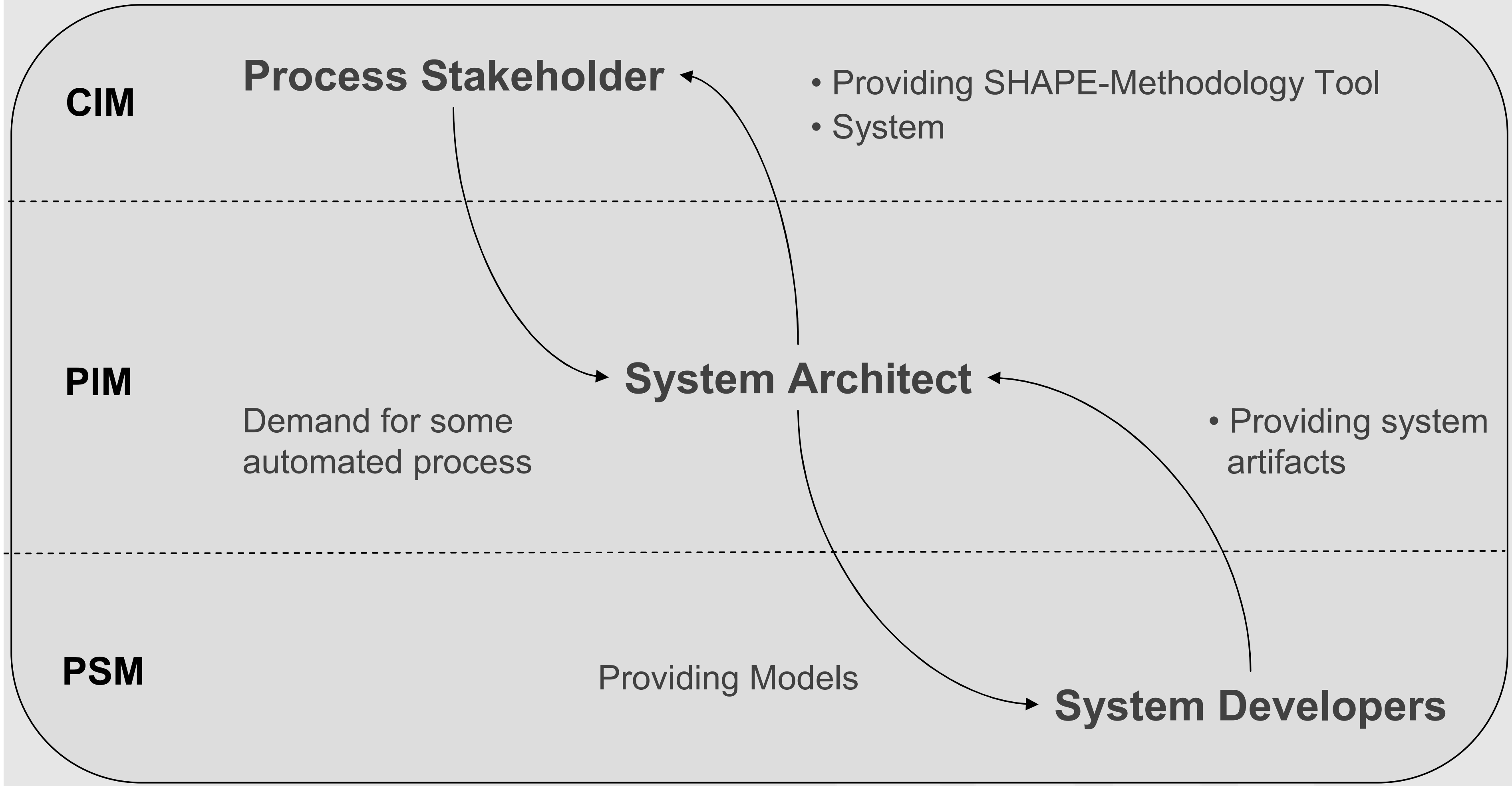
SHAPE Project

- SHAPE: Semantically-enabled Heterogeneous Service Architecture (SHA) and Platforms Engineering
- Consortium:
  - ESI, Softeam, SAP, Statoil, Saarstahl, DFKI, STI, SINTEF
- Main objectives:
  - develop a MDE tool-supported methodology
  - Help in standardization of metamodels and languages for SHA
- Main challenges:
  - How to map the flow of business logic and data to services and the functionality in a platform-independent way?
  - How to integrate the various models of processes, requirements, services and functions in a common model?

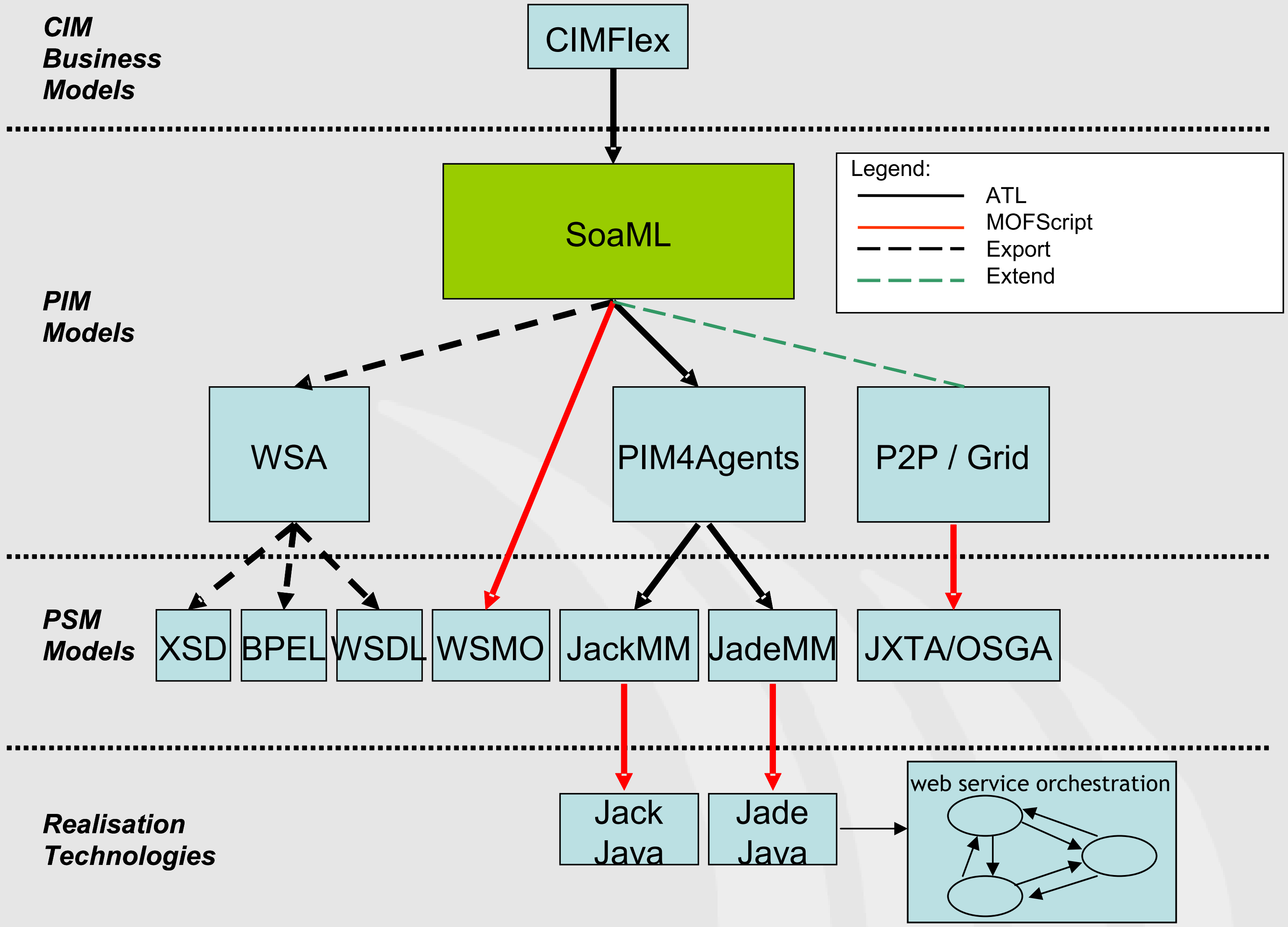
SHAPE Project

- Further Saarstahl specific objectives:
  - Evaluation of MDA and SOA in Saarstahl context
    - » Wrapping legacy systems behind services
    - » Transformation of business processes down to executable code
  - Proof of concept:
    - » Possibility of how to manage processes in future
  - Ease and increase interoperability of agent-based manufacturing execution systems (MES)

The „Big Picture“



The „Big Picture“



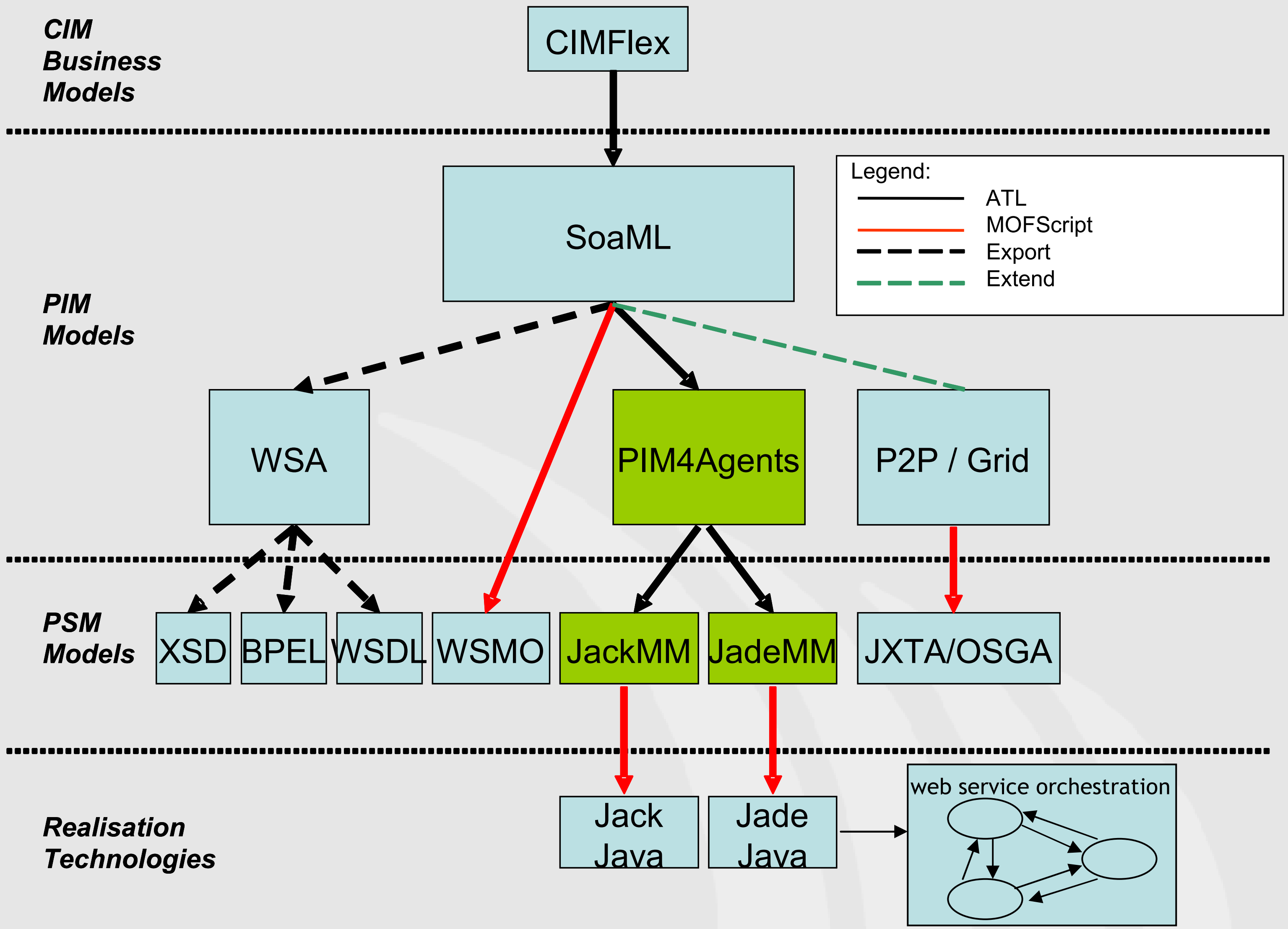
SoaML

saarstahl

- Service oriented architecture modeling language (OMG)
- Describes a UML profile and metamodel for the design of services within a SOA
- Extends UML2 only where it is necessary to accomplish the goals and requirements of service modeling
- Important SoaML concepts:
  - **Participant:** Provides or consumes services, has ports for communicating
  - **ServiceContract:** Defines terms, conditions, interfaces, choreography
  - **Service/RequestPoint:** Connection point through which a participant provides/requires a service
  - **ParticipantArchitecture**
  - **ServiceArchitecture**

SOAML

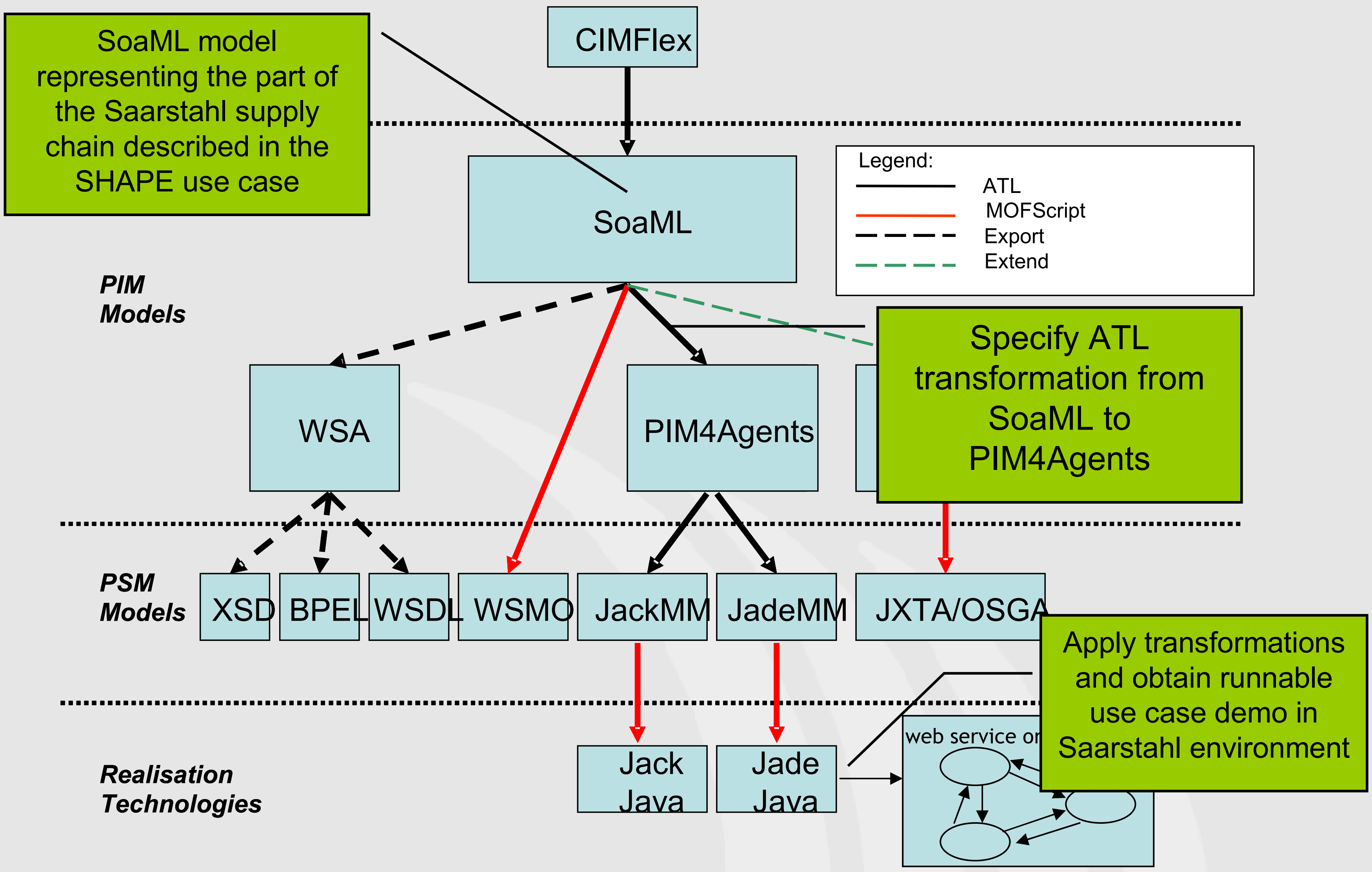
The „Big Picture“



PIM4Agents

- Platform independent metamdel for MAS (PIM4Agents)
- Several views allow modeling of MAS:
- **Agent view** describes single autonomous entities and the capabilities each can possess to solve tasks within an agent system
- **Organization view** describes how single autonomous entities collaborate within MASs and how complex organizational structures can be defined
- **Role view** covers feasible specializations and how they could be related to each role type
- **Interaction view** describes how the interaction between autonomous entities or organizations take place. **Behavioral view** describes how plans are composed by complex control structures and simple atomic tasks
- **Environment view** contains any kind of resource dynamically created, shared, or used by agents or organizations

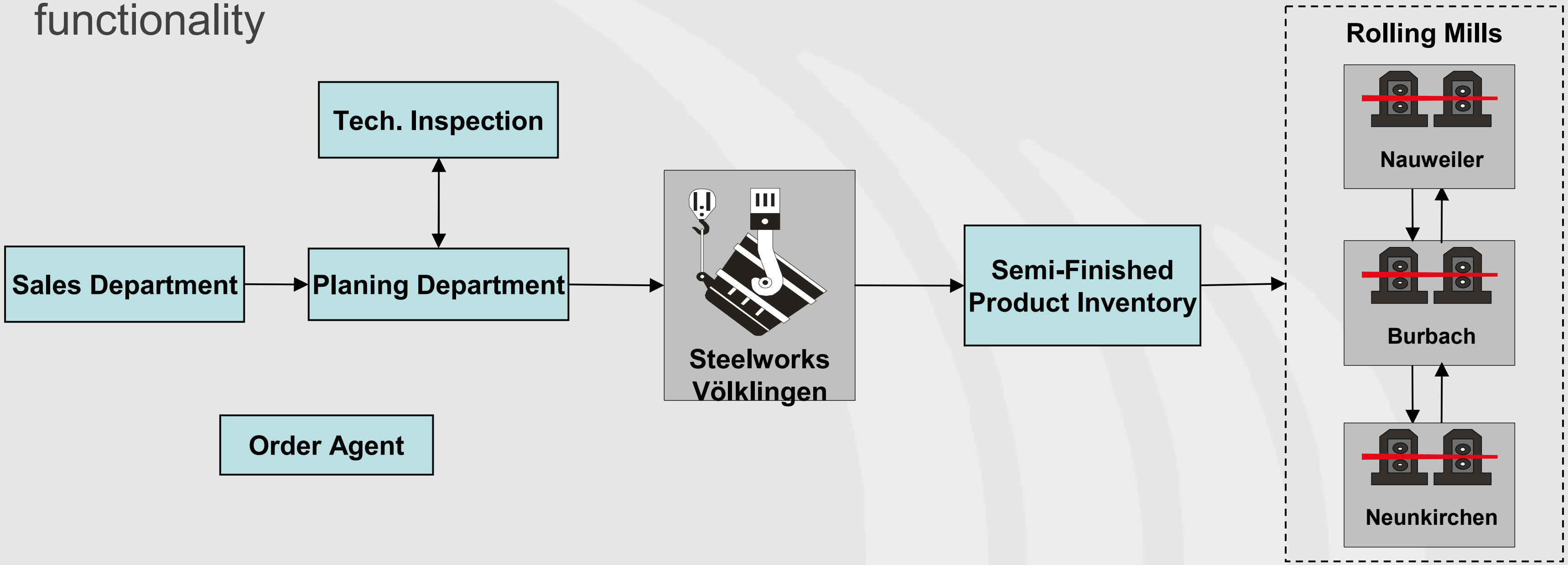
Problem Description - Overview



Saarstahl Scenario

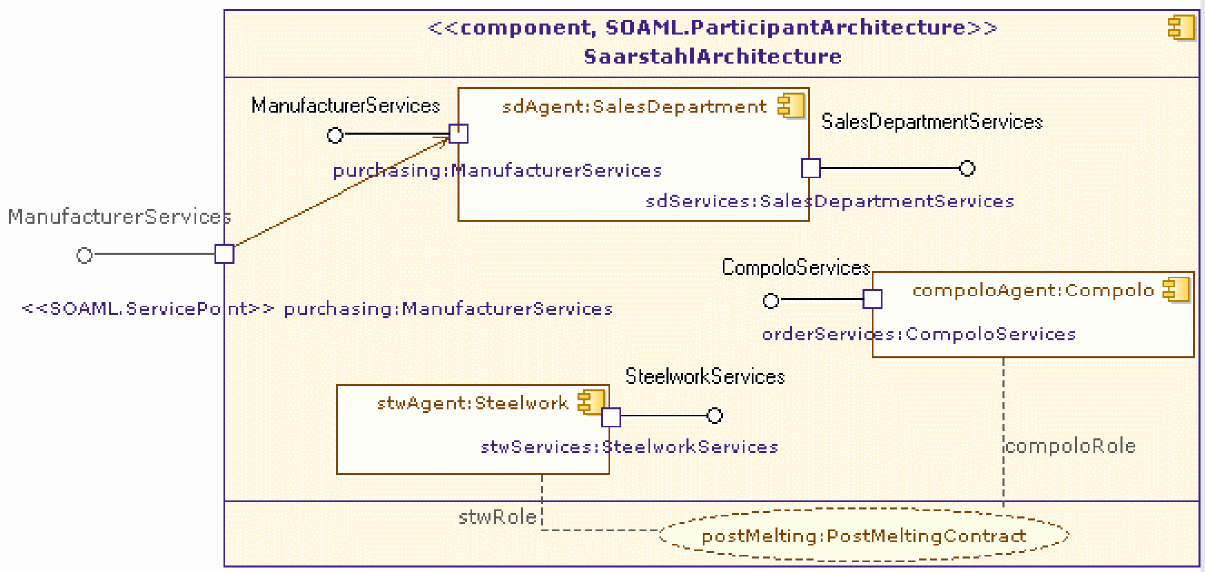
- Saarstahl system landscape consists of many running legacy systems
- Saarstahl demands a flexible business process management and high-levels of interoperability between these legacy systems to stay competitive
- Use SHAPE approach to wrap legacy systems behind services

- A segment of the supply chain has to be modeled in SoaML: From order entry to rolling
- 7 participants have been identified
- Each participant provides and requires services, wrapping the legacy functionality



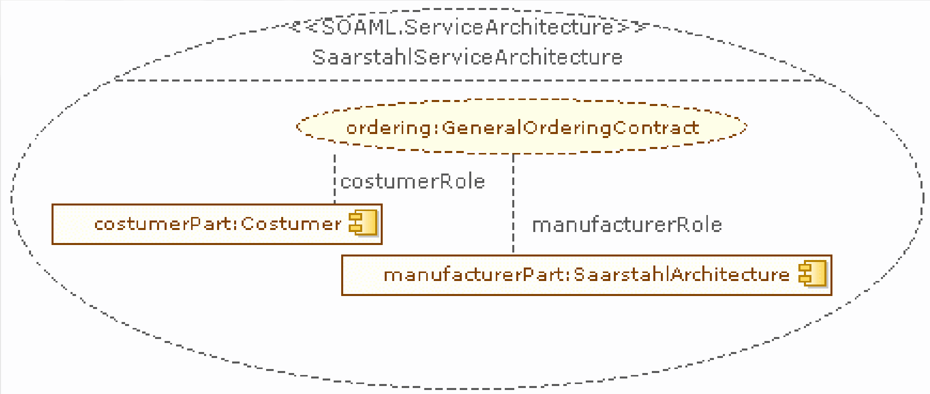
Saarstahl Model - Excerpt

Internal architecture of Saarstahl



The concept **ParticipantArchitecture** describes how internal participants work together for a purpose by providing and using services expressed as **ServiceContracts**

Collaboration Saarstahl - Costumer

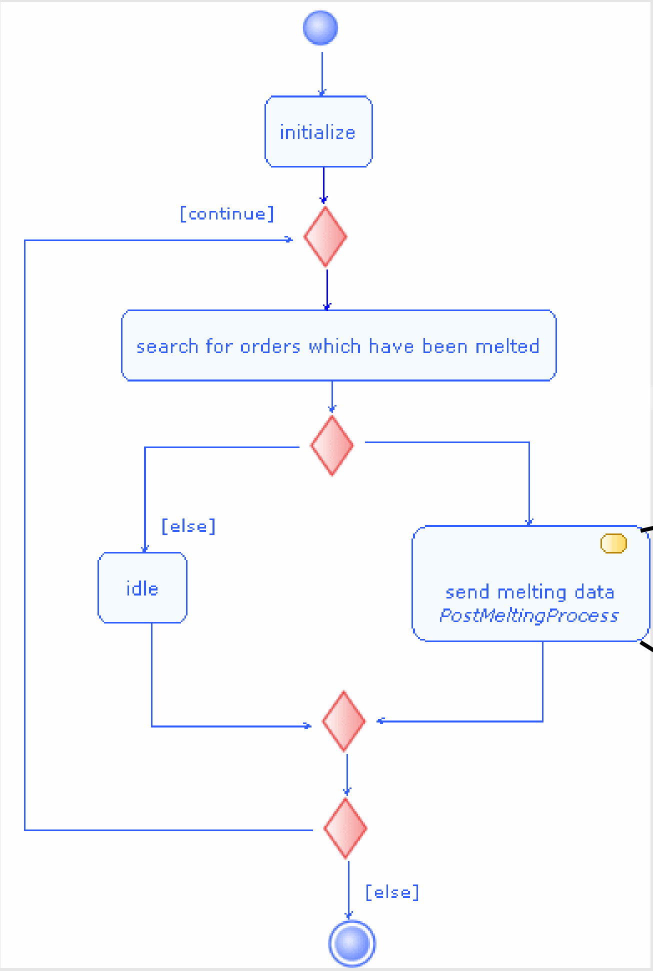


The concept **ServiceArchitecture** describes how participants work together for a purpose by providing and using services expressed as **ServiceContracts**

Saarstahl Model - Interactions

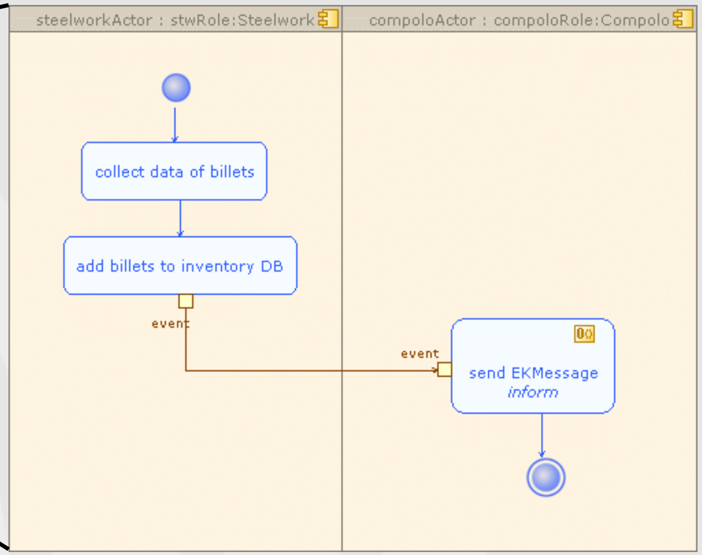
saarstahl

Behaviour of steelwork agent

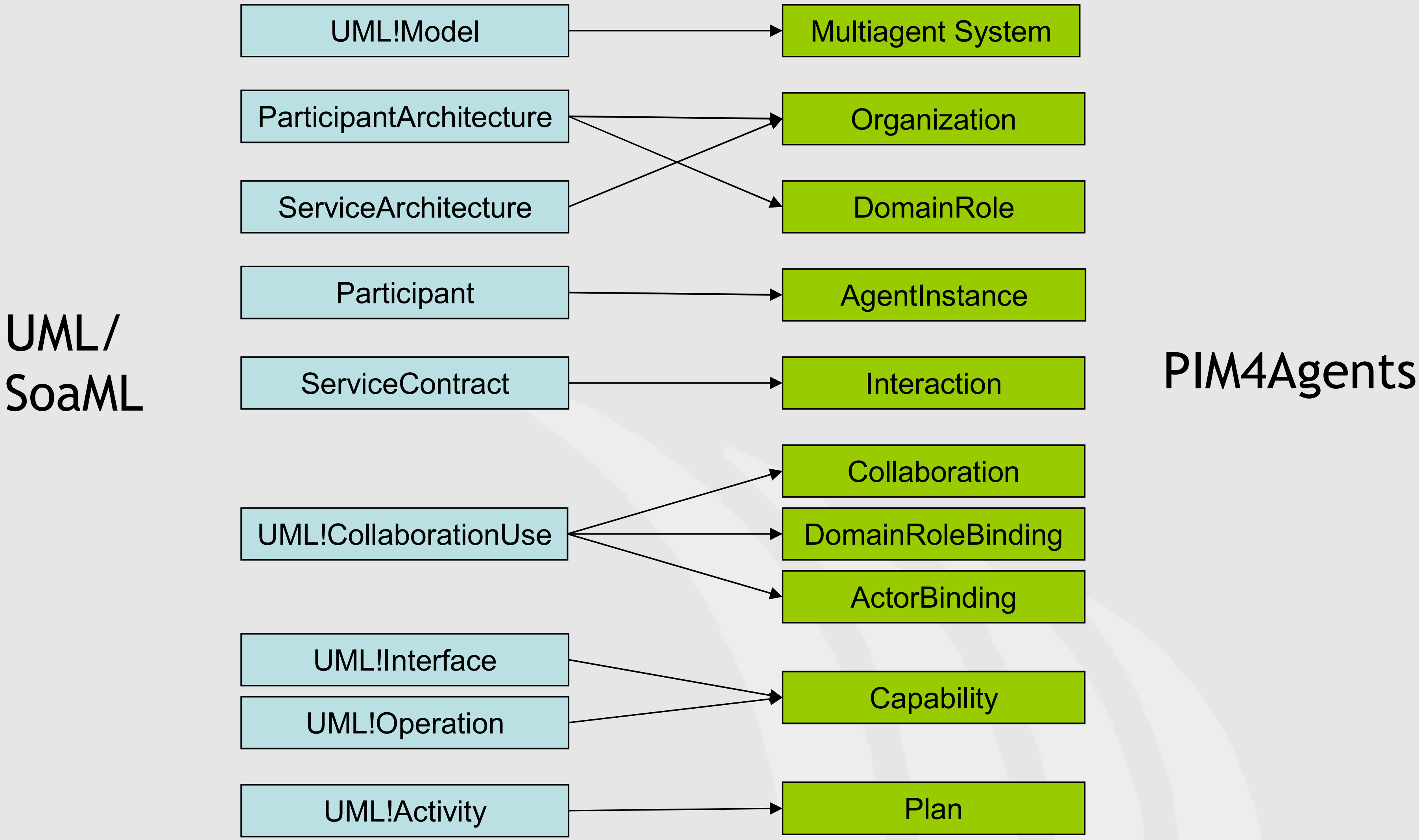


- Some participants got an owned behaviour
- Behaviours are modeled using activity diagrams
- Interaction points:
  - Contain a swimlane for every participant
  - Called by using CallBehaviorAction
- Service invocations:
  - Services are invoked by CallOperationAction

Interaction point – PostMeltingProcess



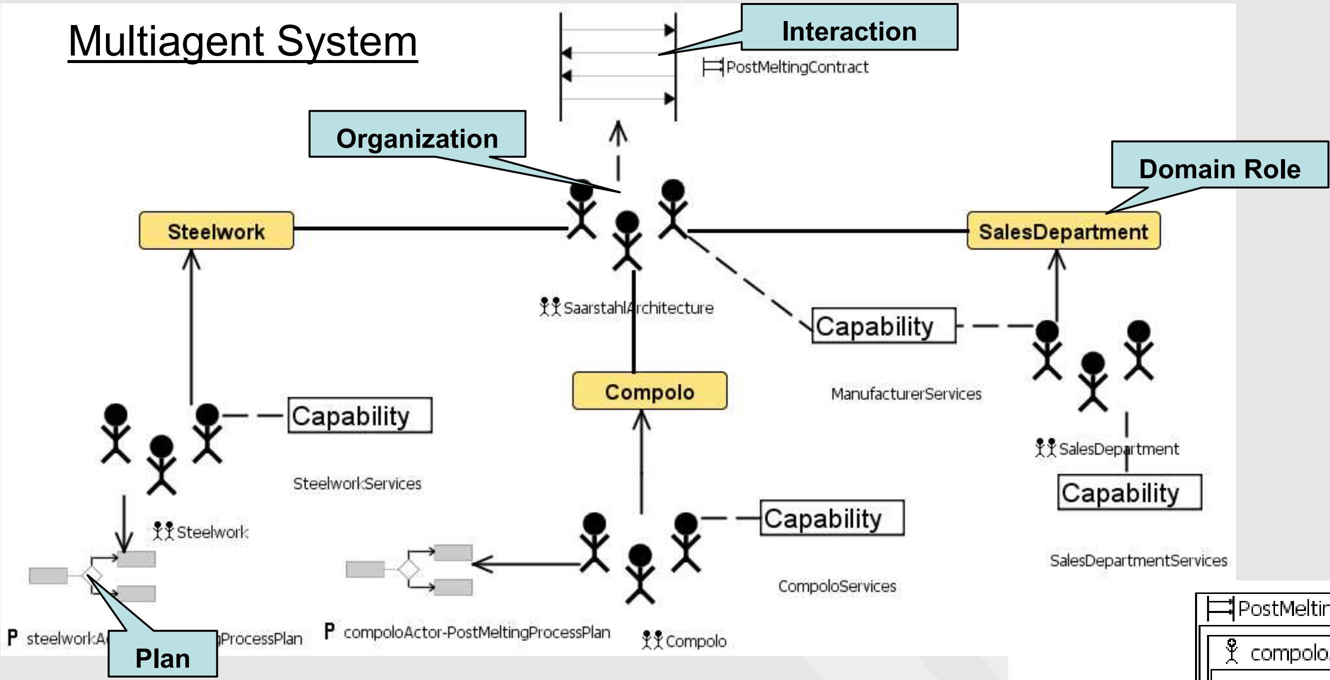
Transformation – Mappings Overview



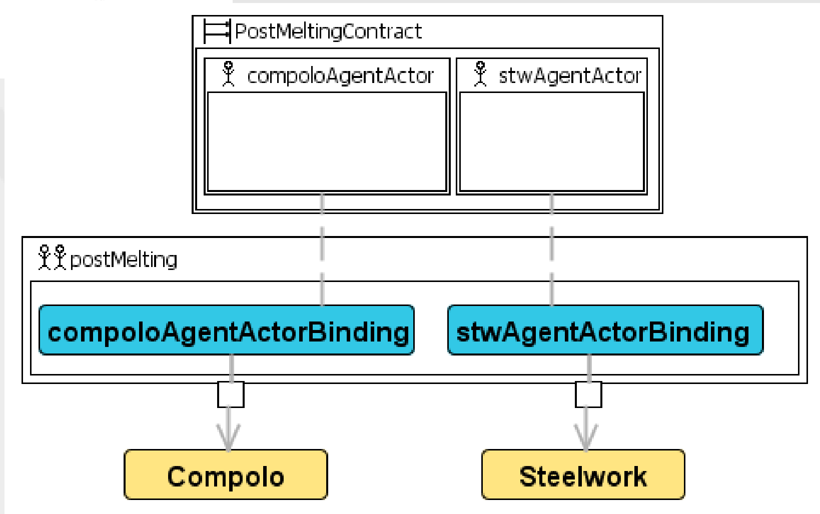
Transformation - Results

saarstahl

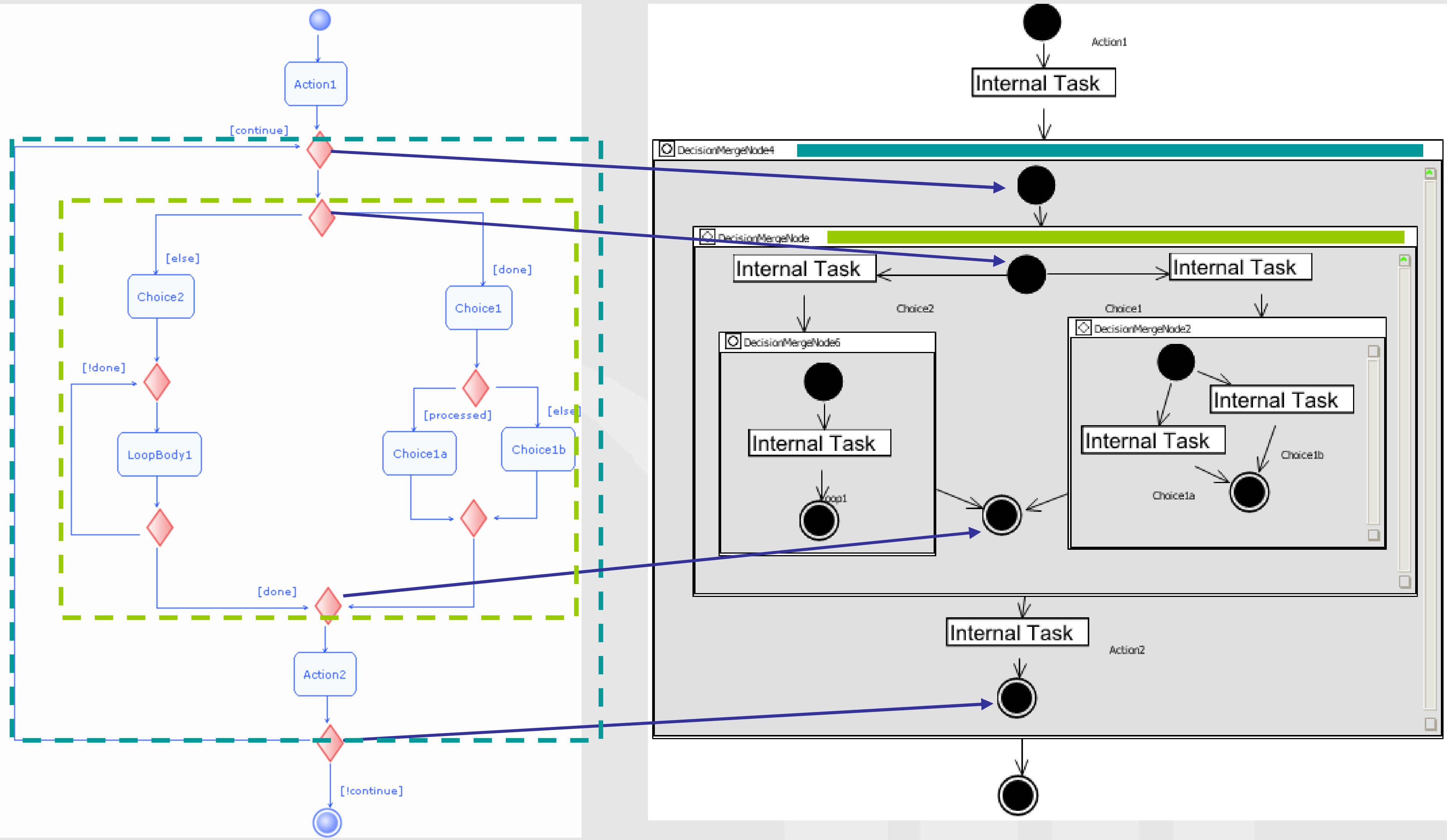
Multiagent System



Collaboration



Transformation - Results



Conclusion

saarstahl

- SHAPE approach has been introduced
- The transformation from SoaML to PIM4Agents offers the ability to combine SOAs and MASs on a platform independent level
- Demonstrated the modeling of a segment of the Saarstahl supply chain in SoaML
- Presented conceptual mappings and implementation of the transformation
- Evaluation of the Saarstahl use case shows that this approach can increase the potential of MASs for commercial applications
- The runnable demo is a proof of concept in a real business environment

Thanks for your attention

Transformation – Mappings (2)

- SoaML:ParticipantArchitecture → PIM4Agents:Organization
- Both concepts describe how entities collaborate inside some encapsulating component

Target	**Source**
RequiredRoles	OwnedAttributes which are of type ParticipantArchitecture
PerformedRoles	Every occurrence as an OwnedAttribute in some ParticipantArchitecture or ServiceArchitecture
OrganizationUse	Owned CollaborationUses
Interaction	Corresponding ServiceContracts to owned CollaborationUses

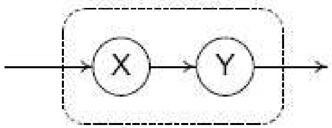
Transformation – Mappings (2)

- UML:CollaborationUse → PIM4Agents:Collaboration
- Collaboration:
  - Defines which organizational members are bound to which kind of Actor as part of an ActorBinding
- CollaborationUse:
  - Binds participants to roles of a ServiceContract in a specific situation
- To create a Collaboration, information from both CollaborationUse and corresponding ServiceContract are required

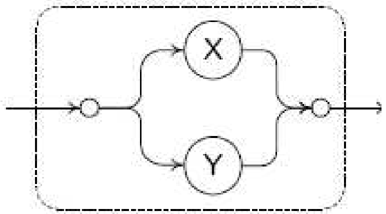
Target	Source
Organization	The containing ParticipantArchitecture or ServicesArchitecture
InteractionInstance	ServiceContract which types the CollaborationUse
Binding	Collection of DomainRoles which are required by the corresponding ServiceContract
ActorBinding	Collection of roles bound to this CollaborationUse

Transformation – Activities

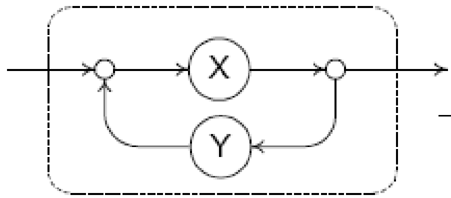
- Difficulties:
  - Some graph-oriented structures have to be transformed to block-oriented structures (decisions, loops)
  - Arbitrary workflow models
- Solution: Impose restrictions on activities
  - Use only intermediate activities (sufficient for proof of concept)
  - Activities must conform to the definition of structured workflow models (see below)
  - In loops, one edge has to connect the MergeNode to the DecisionNode



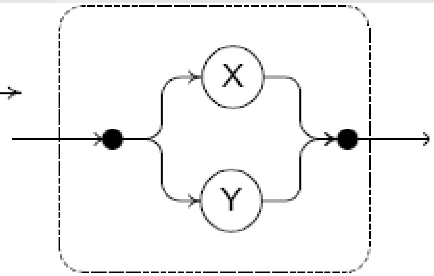
Sequence



Decision structure

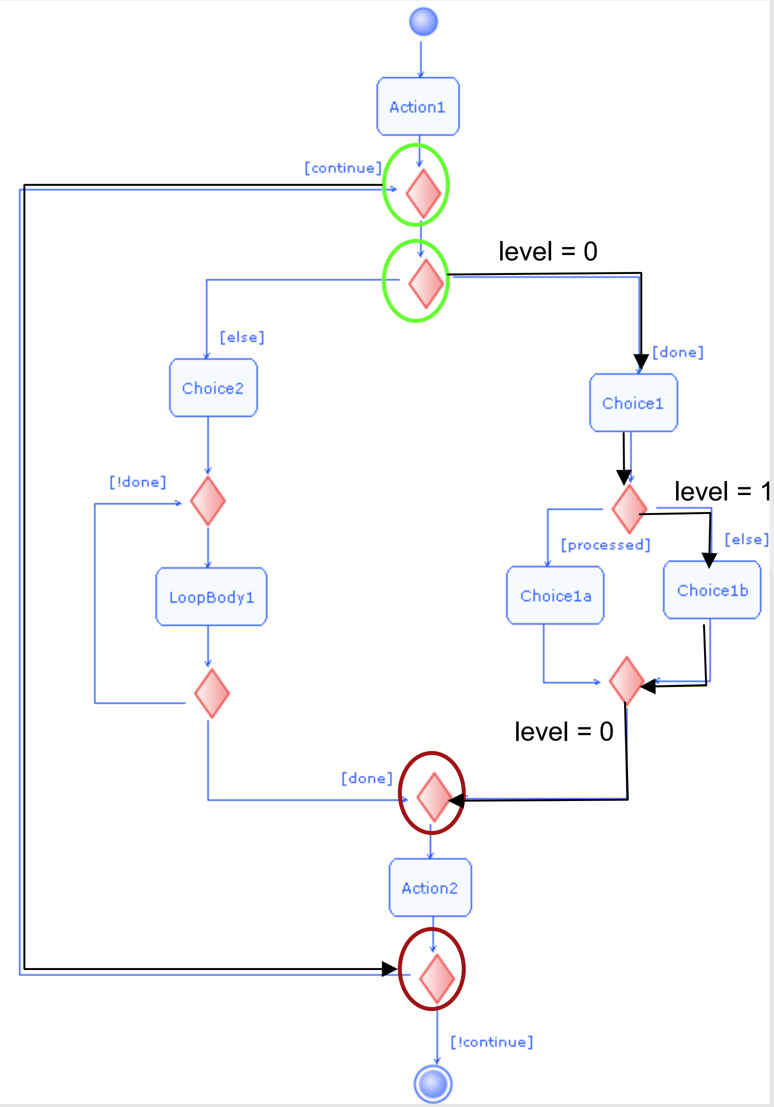


Structured loop



Parallel structure

Transformation – Activities Example



Pseudocode

```
Node getComplementaryLoopNode(visited:Sequence(Node)) {
    list = incomingEdges();
    list = list.select(DecisionNode);
    list = list.select(not in visited);
    return list.first();
}
```

```
Node getComplentaryDecisionNode(node:Node, level:int,
        visited:Sequence(Node)) {

    if (node  == MergeNode and isLoop())
        return gCDN(nextNode(), level, visited+node);
    if (node == MergeNode and not isLoop())
        if (level == 0) return node;
        else return gCDN(nextNode(), level-1, visited+node);
    if (node == DecisionNode)
        return gCDN(nextNode(), level+1, visited+node);
    else
        return gCDN(nextNode(), level+1, visited+node);
}
```