SAINT
Software & Systems Architecture & Integration Team

S3EL
Service-Oriented Software & Systems Engineering Lab

UCSD

California Institute for Telecommunications and Information Technology : Computer Science and Engineering, University of California, San Diego

A Rich Services Approach to CoCoME

Ingolf H. Krueger

Joint work with Barry Demchak, Vina Ermagan, Emilia Farcas, To-ju Huang, Massimiliano Menarini

CSE Department – Calit2
University of California, San Diego
http://sosa.ucsd.edu

UCSD
UCIrvine

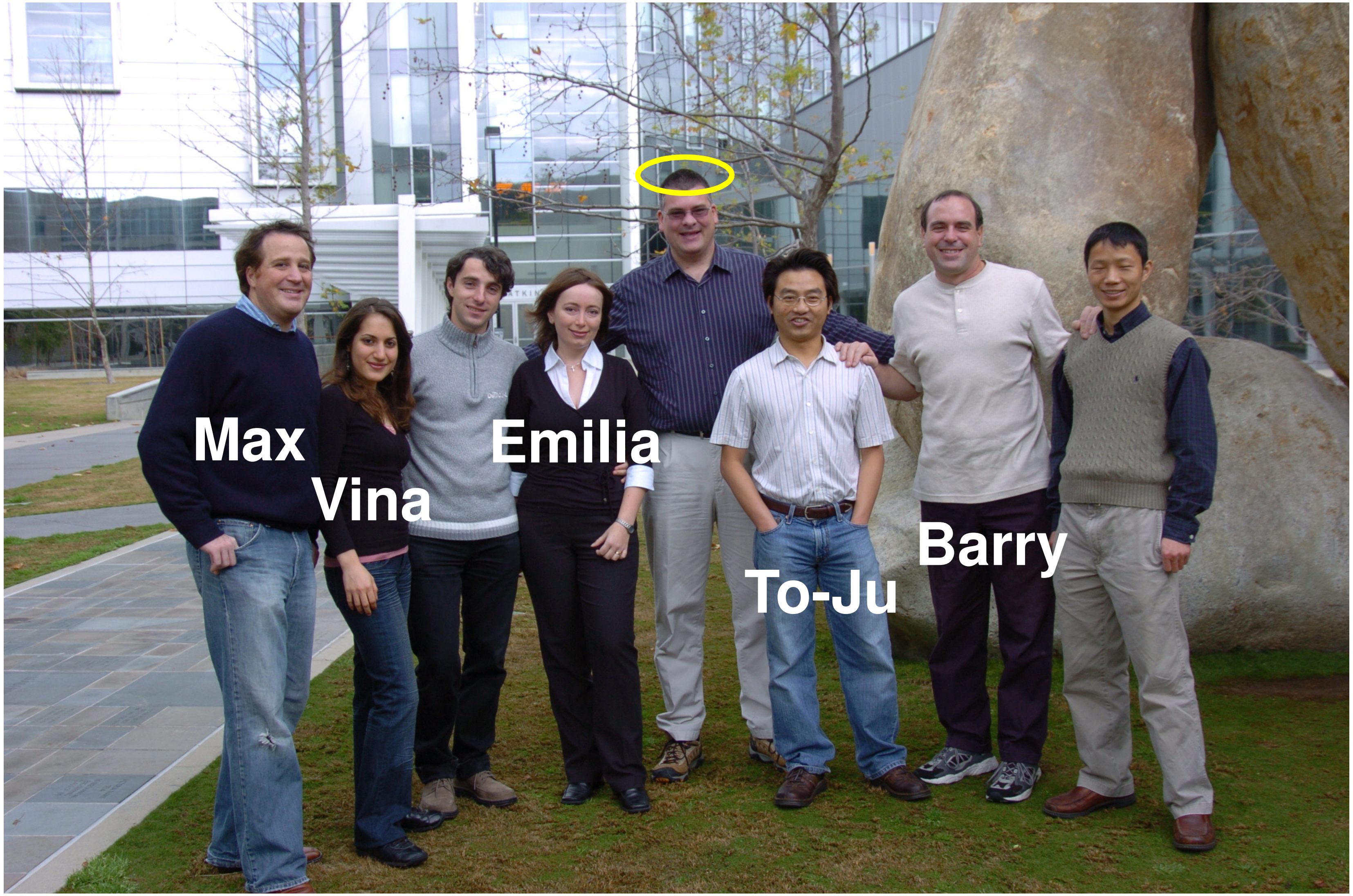
Team Introduction

- **Affiliation and Experience**
  - **UCSD, CSE Department: Service-oriented software & systems engineering laboratory (S3EL)**

    Research on innovative techniques for service-oriented software and systems engineering:
    - **service-oriented software architecture**
    - **tailored development process, methodology & tools**
    - **expressive description techniques**
    - **precise, methodological foundation**

  - **Calit2: Software and systems architecture and integration team (SAINT)**
    - **Successful application to large-scale software and systems integration projects within Calit2**
      - Ocean Observatories Cyberinfrastructure
      - Metagenomics, Bioinformatics
      - Automotive
      - Public Safety
      - Enterprise Chat
    - **Design and implementation of flexible and scalable solutions such as XML-based Web services, web-portals, message and enterprise service busses.**

Team Introduction



Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Technologies

- Traditional approaches
  - COTS
    - Standardization is a problem
    - Many unused features cause application bloat
  - CORBA
    - Heavyweight
- Promising approaches
  - Web Services
    - Several W3C standards backed by industry for the separation of concerns (HTTP/SOAP), data marshaling (XML), interface descriptions (WSDL)
    - Addressing cross-cutting concerns is a problem
  - Enterprise Service Bus
    - Message-oriented middleware (MOM)
    - Flexible plug-in architecture
    - Rich set of data adapters/connectors for rapid connections
    - Transition from logical architecture to ESB implementation is still ad-hoc

Challenges

- Address crosscutting architectural concerns
  - such as policy management, governance, and authentication
- Still maintain a lean implementation and deployment flavor?
- Horizontal: interplay at the same logical or deployment level of
  - application services
  - the corresponding crosscutting concerns
- Vertical: hierarchical decomposition into sub-services
  - the environment is shielded through encapsulation from
    - their structural and behavioral complexity
    - the form of their composition

Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

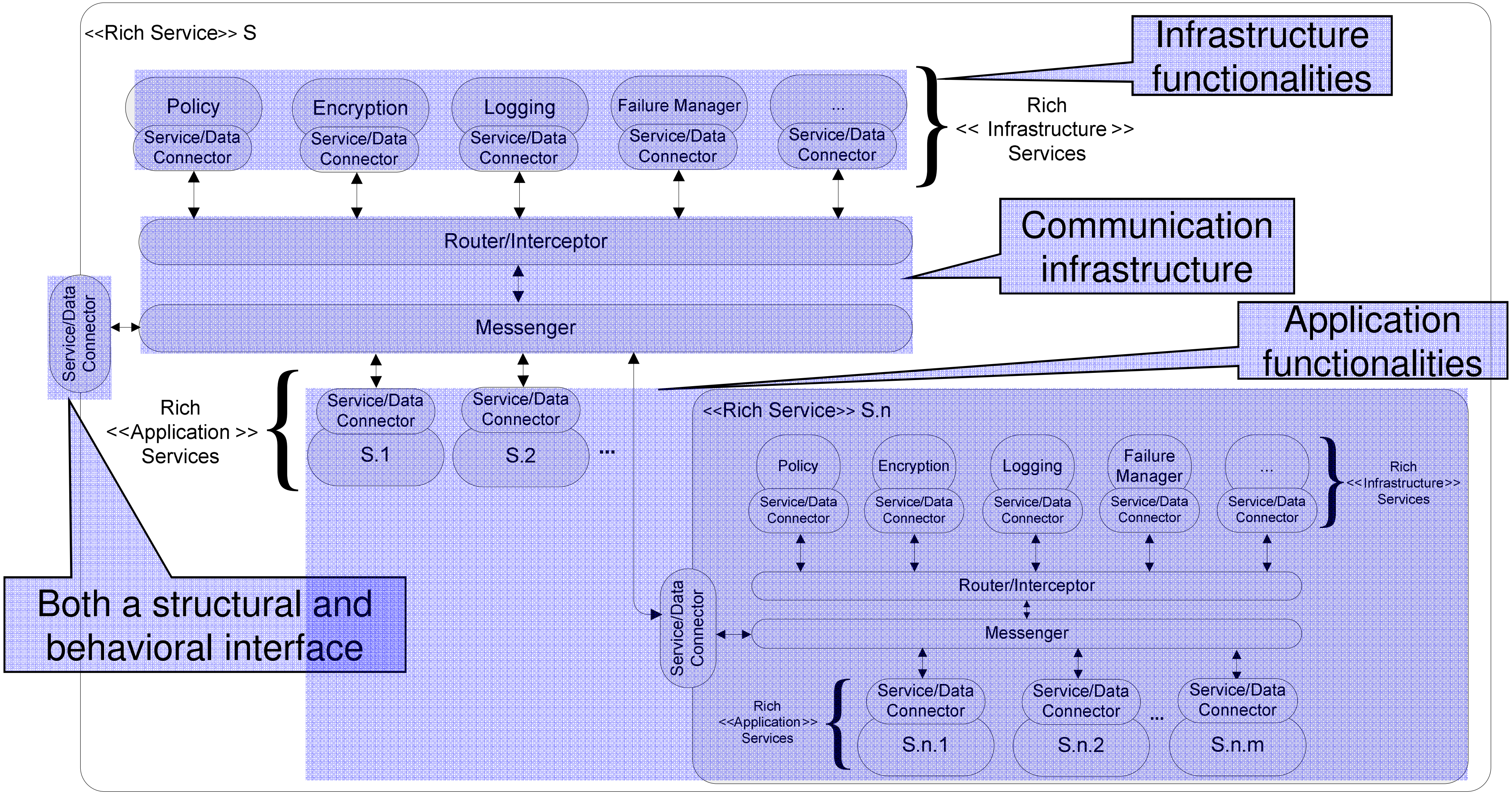
Rich Services – Why/What?

“To boldly go where no service has gone before”.

- *an extension of the service notion, based on an architectural pattern*
- Manage the complexity of a system-of-systems
  - – decomposing into primary and crosscutting concerns
  - – providing flexible encapsulation for these concerns
  - – generating a model that can easily be leveraged into a deployment
- Workflow management
  - – Service choreography at the infrastructure or application level
- Dynamic adaptation
  - – new services can be introduced at runtime
  - – no need to change or adapt the implementation of existing services

Rich Services: Scalable Service Integration

From tightly to l o o s e l y coupled systems



<<Rich Service>> S
Infrastructure functionalities
Policy
Encryption
Logging
Failure Manager
...
Service/Data Connector
Rich << Infrastructure >> Services
Router/Interceptor
Messenger
Communication infrastructure
Application functionalities
Service/Data Connector
Rich <<Application >> Services
S.1
S.2
...
<<Rich Service>> S.n
Policy
Encryption
Logging
Failure Manager
...
Rich <<Infrastructure>> Services
Router/Interceptor
Messenger
Rich <<Application>> Services
S.n.1
S.n.2
S.n.m
Both a structural and behavioral interface

a hierarchically decomposed structure supporting “horizontal” and “vertical” service integration

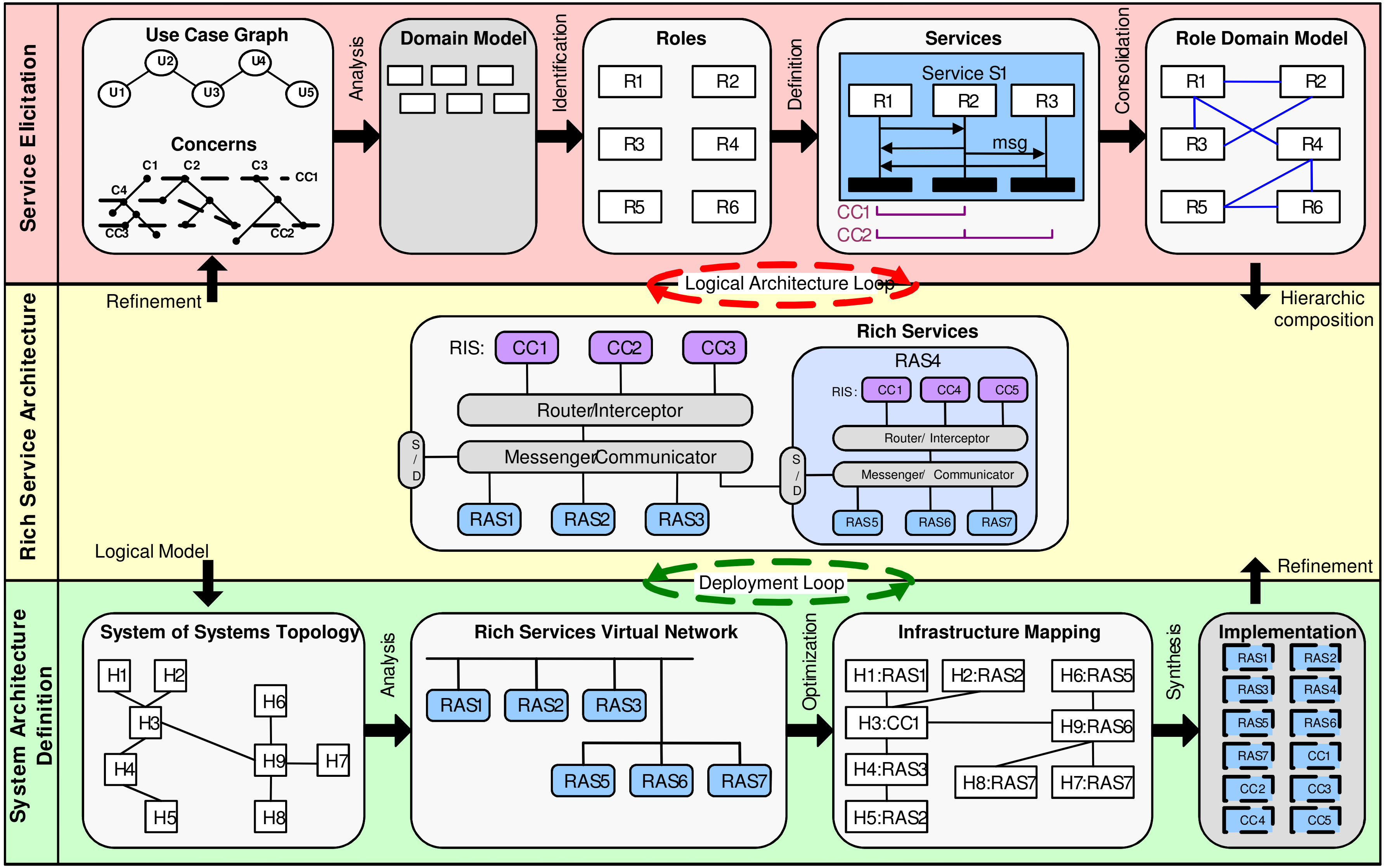
Rich Services – Core

- Main entities of the architecture blueprint
  - Service/Data Connector - interaction between the Rich Service and its environment
  - the Messenger and the Router/Interceptor - communication infrastructure
  - Rich Services - encapsulate various application and infrastructure functionalities

- Rich Application Services
  - interface directly with the Messenger
  - provide core application functionality

- Rich Infrastructure Services
  - interface directly with the Router/Interceptor
  - provide infrastructure and crosscutting functionality
  - Examples: policy monitoring/enforcement, encryption, authentication

Rich Services – Development Process

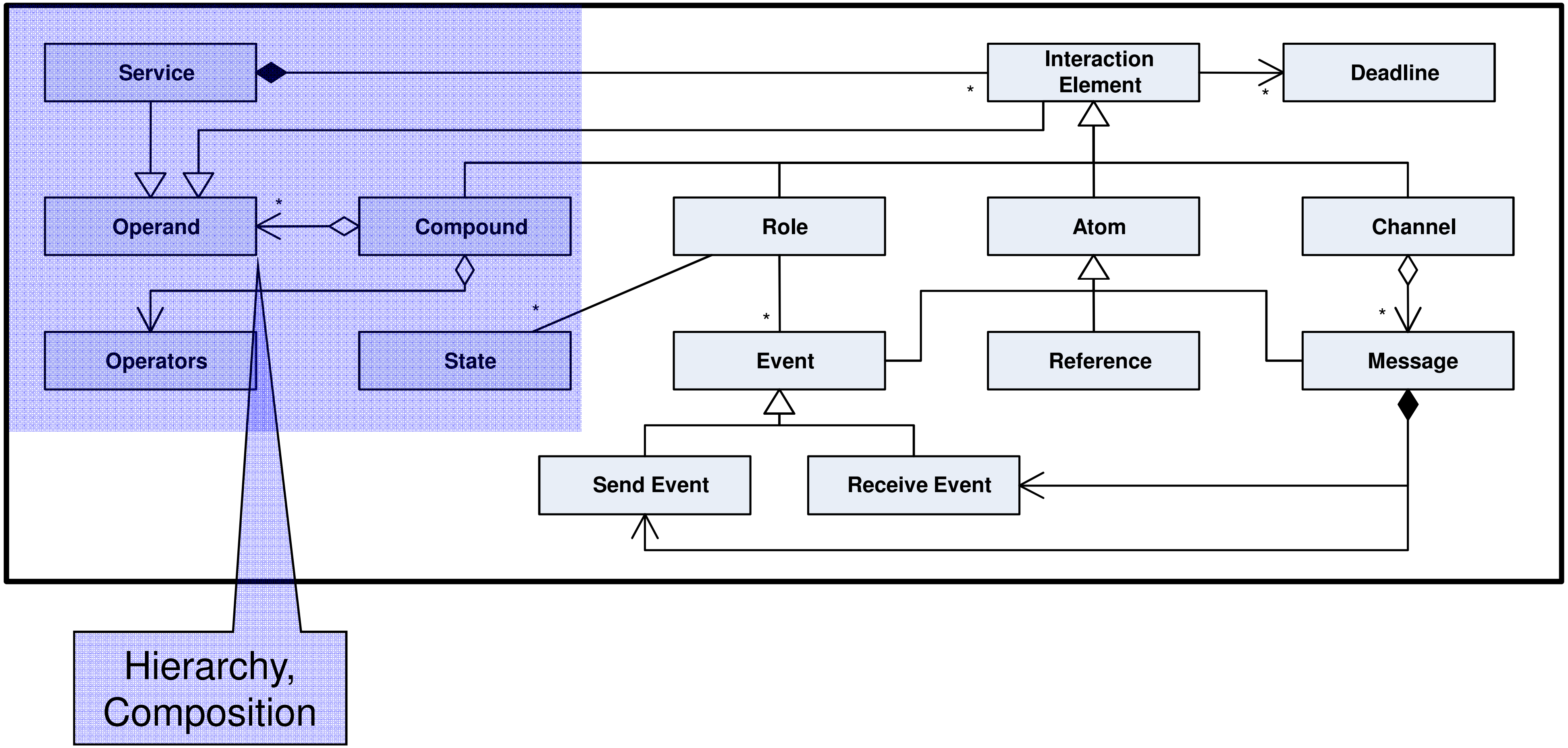


🟥 Service Elicitation	🟨 Rich Services Architecture	🟩 System Architecture Definition

Outline

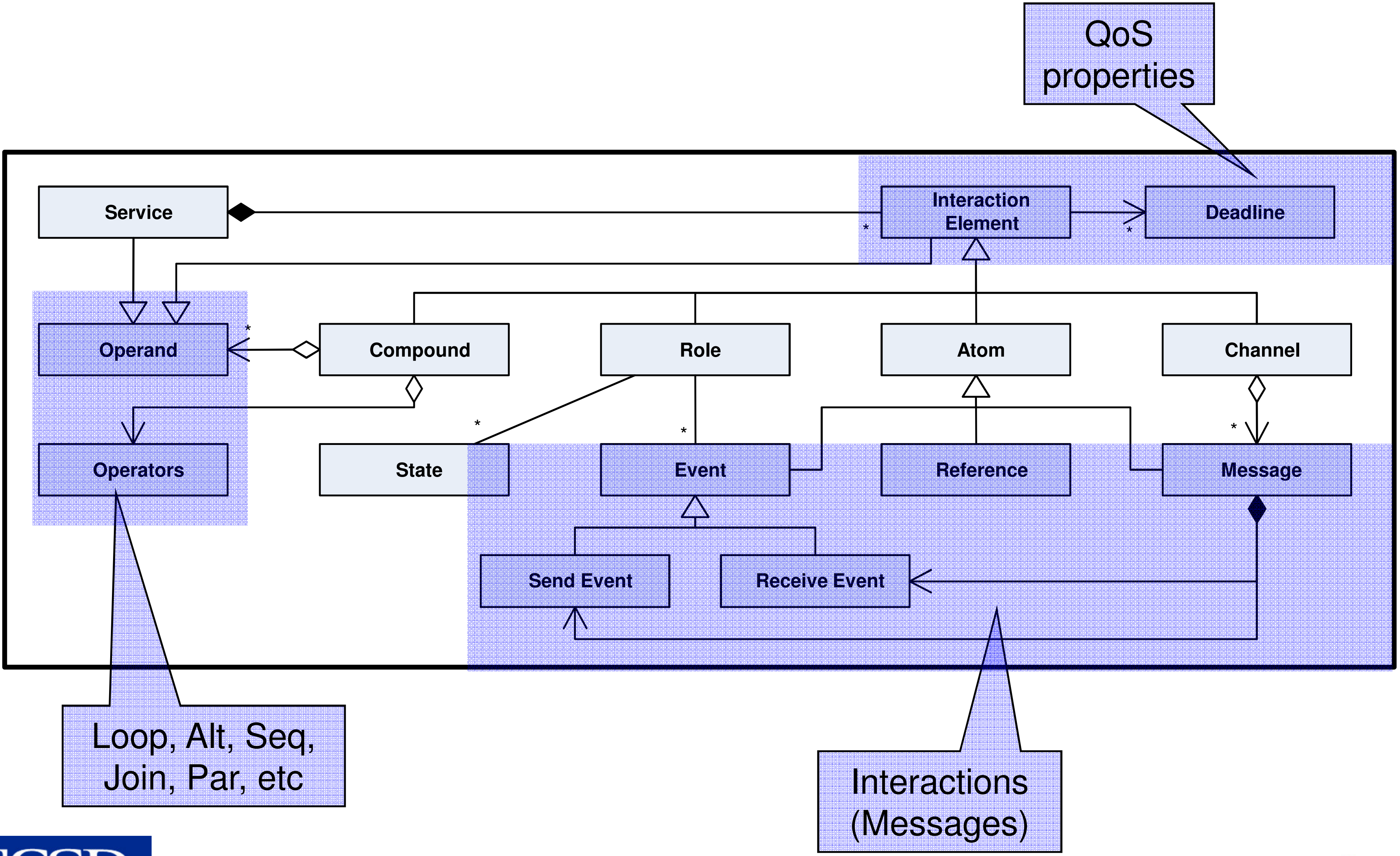
- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Message Sequence Chart Service Model

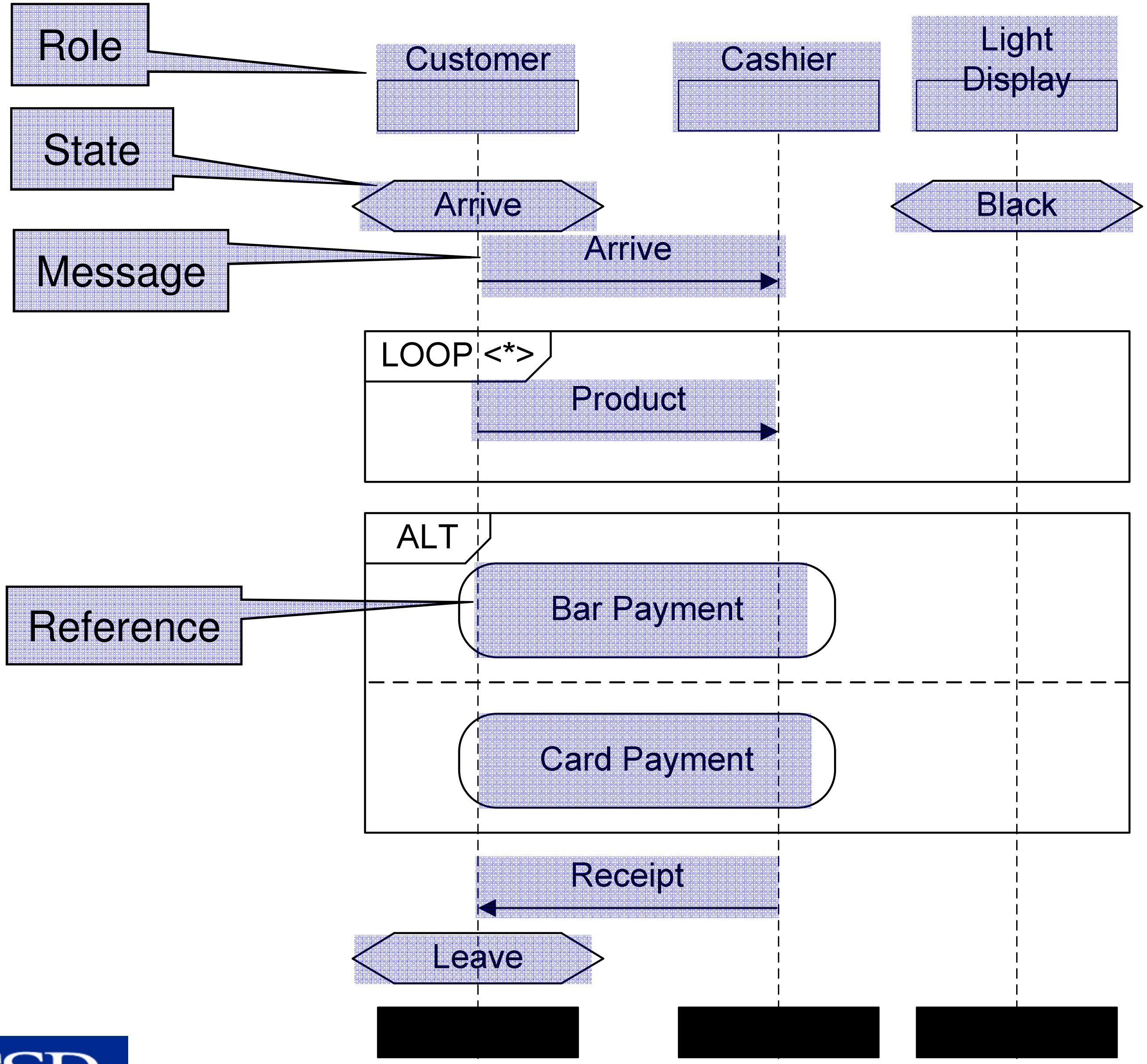


Hierarchy, Composition

Message Sequence Chart Service Model



Example MSCs



Role

State

Message

Reference

Customer

Cashier

Light Display

Arrive

Black

Arrive

LOOP <*>

Product

ALT

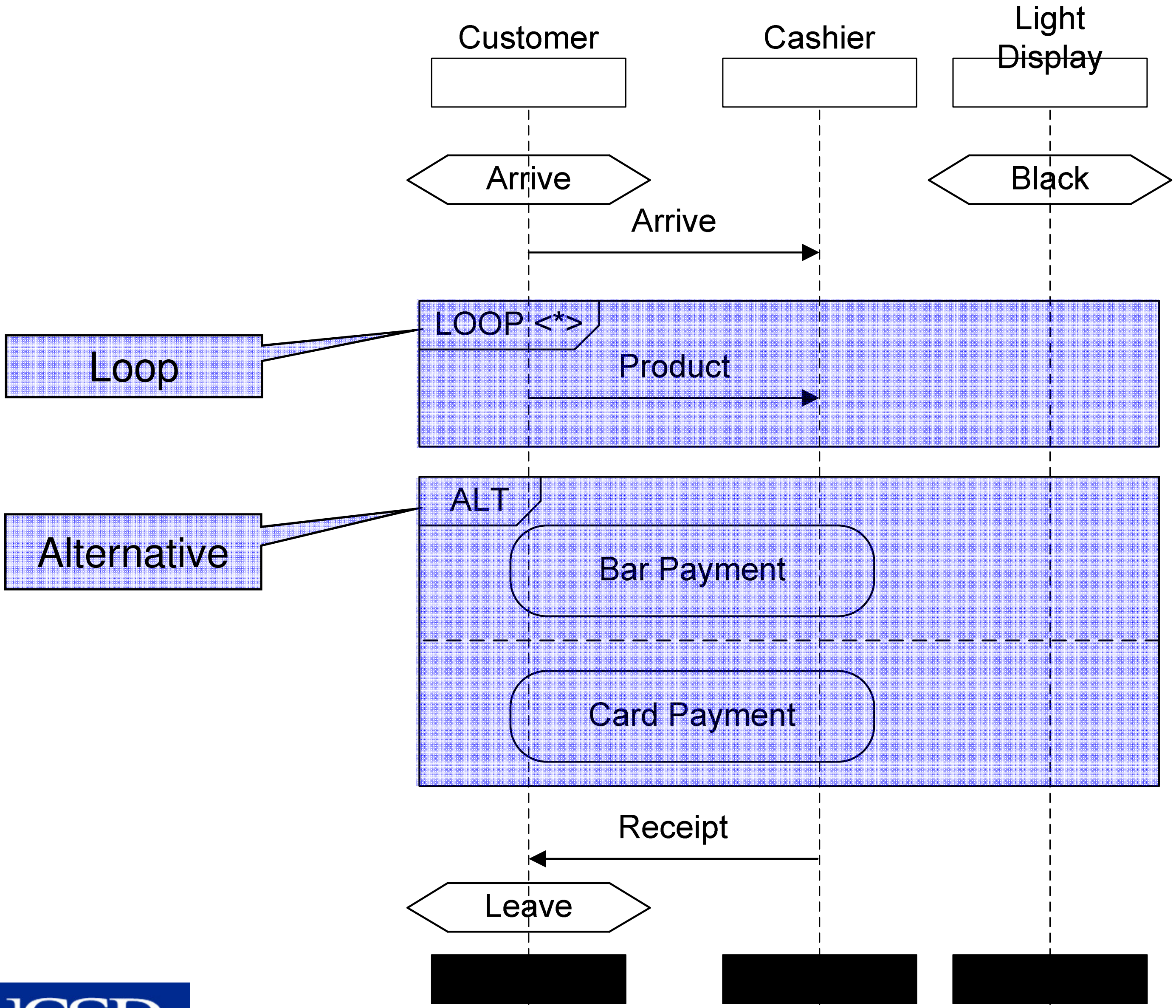
Bar Payment

Card Payment

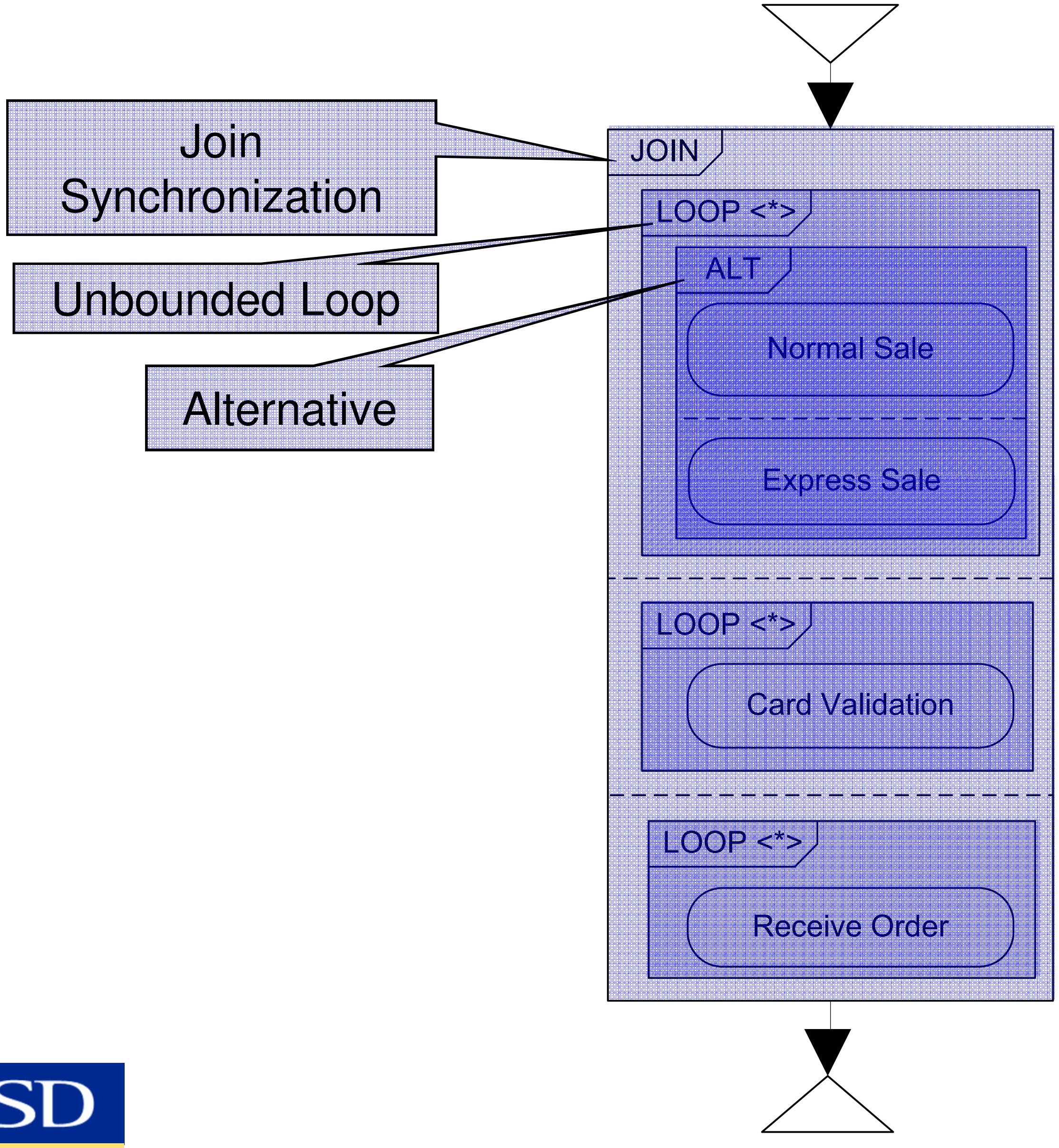
Receipt

Leave

Example MSCs

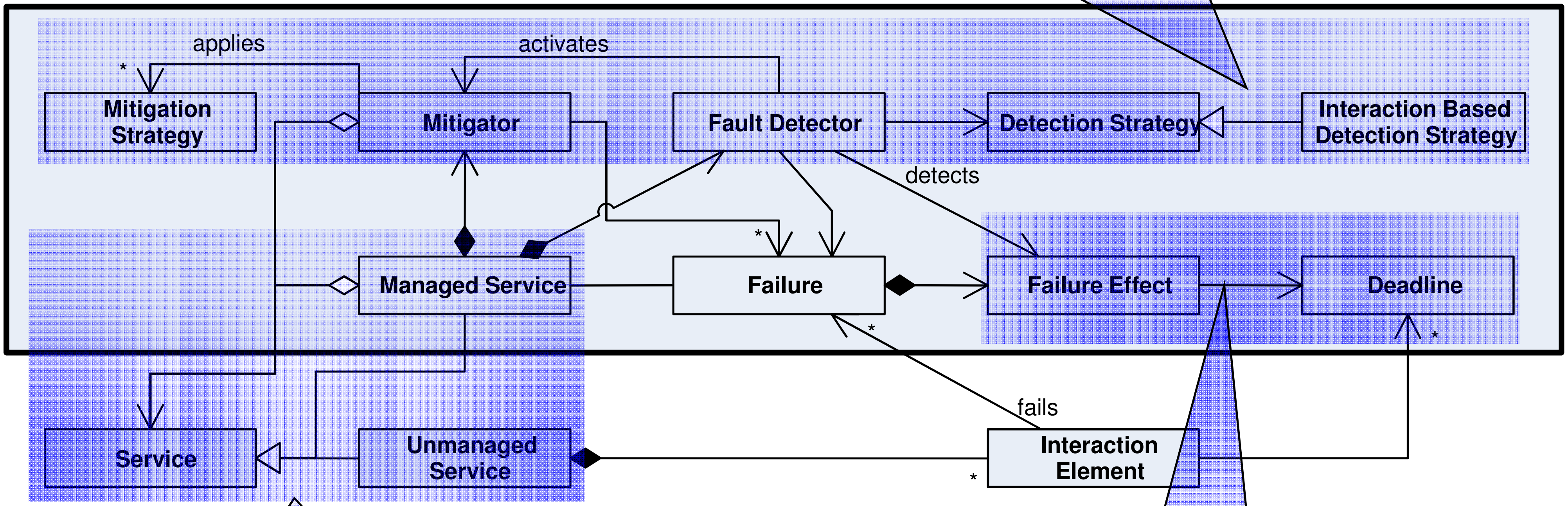


Example HMSC



Service Model for Failure Management

Detection Strategy can be based on Interactions or State Machines.



A Managed Service wraps another service, adding failure management mechanisms to it.

Failure Effect: Missed Deadline, …

Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Modeling the CoCoME

- **Which parts of CoCoME have we modeled?**
  - Static view – logical Rich Service architecture
  - Behavioral view – interactions in the logical model
  - QoS properties
  - Policy enforcement – encryption
  - Failure Management
  - Deployment choices with ESBs

Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

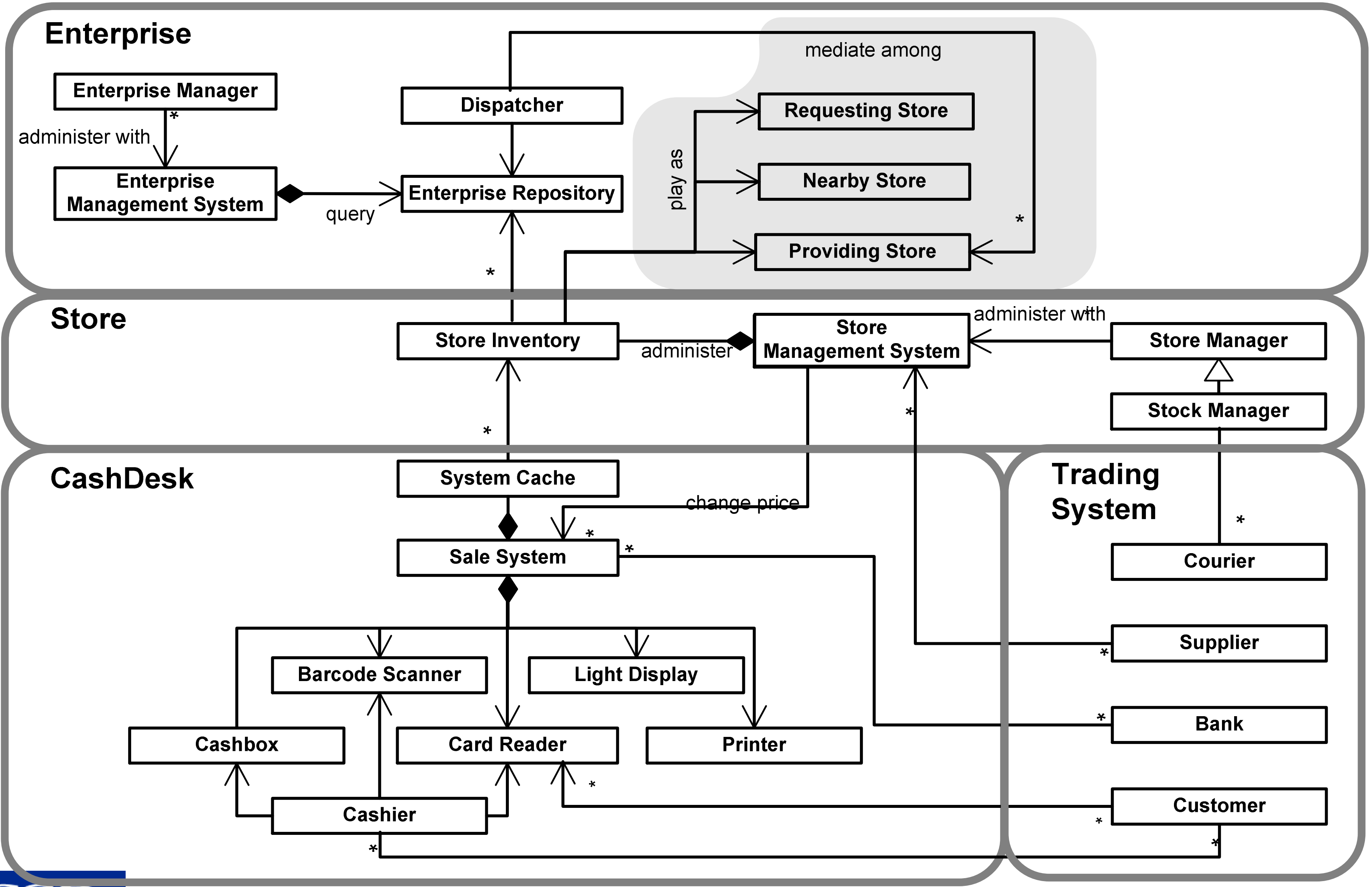
Modeling the CoCoME

- Domain Model

- Decomposition of the system based on *business manageability* as the most important concern in the system
  - Other candidate concerns:
    security, workflow optimization, policies, …

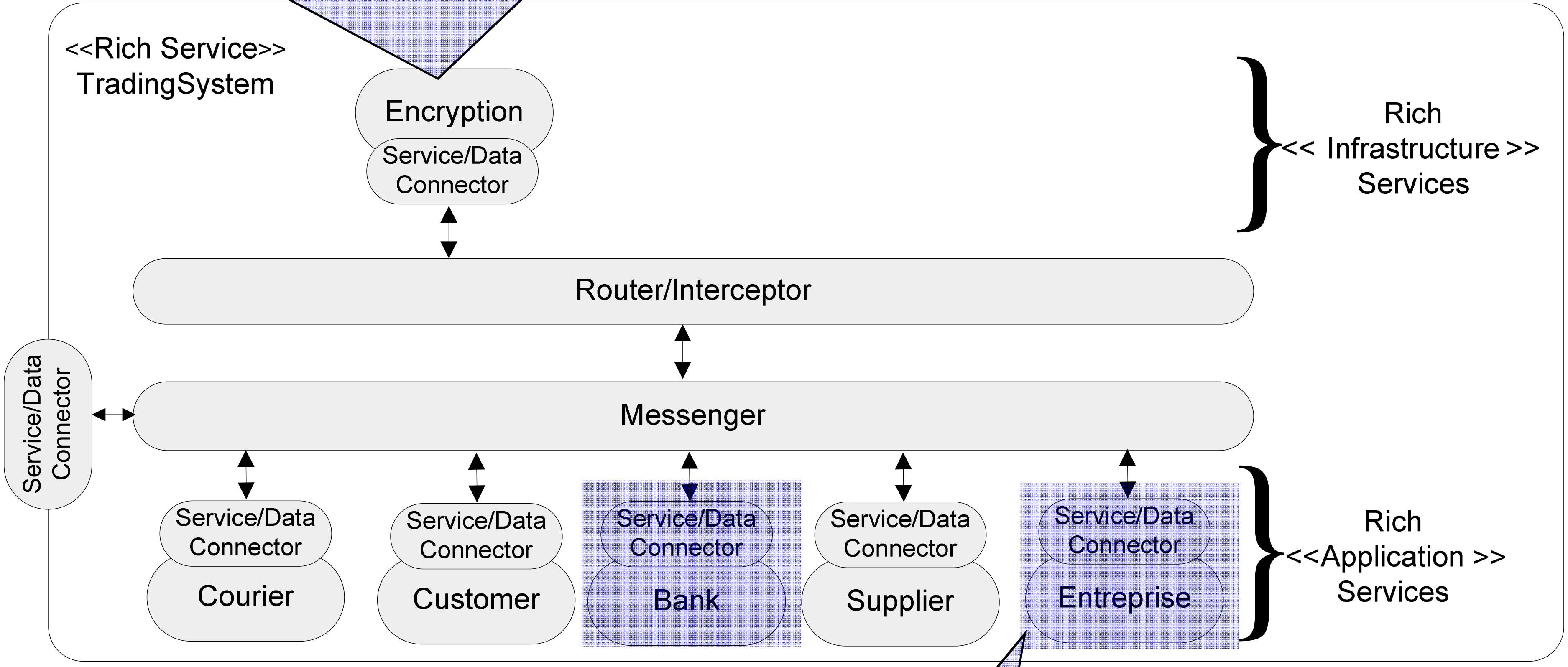
- Arranging the services in the service repository to match the Rich Service decomposition
  - Trading System Rich Service
  - Enterprise Rich Service
  - Store Rich Service
  - Cash Desk Rich Service

CoCoME – Domain Model



Trading System Rich Service

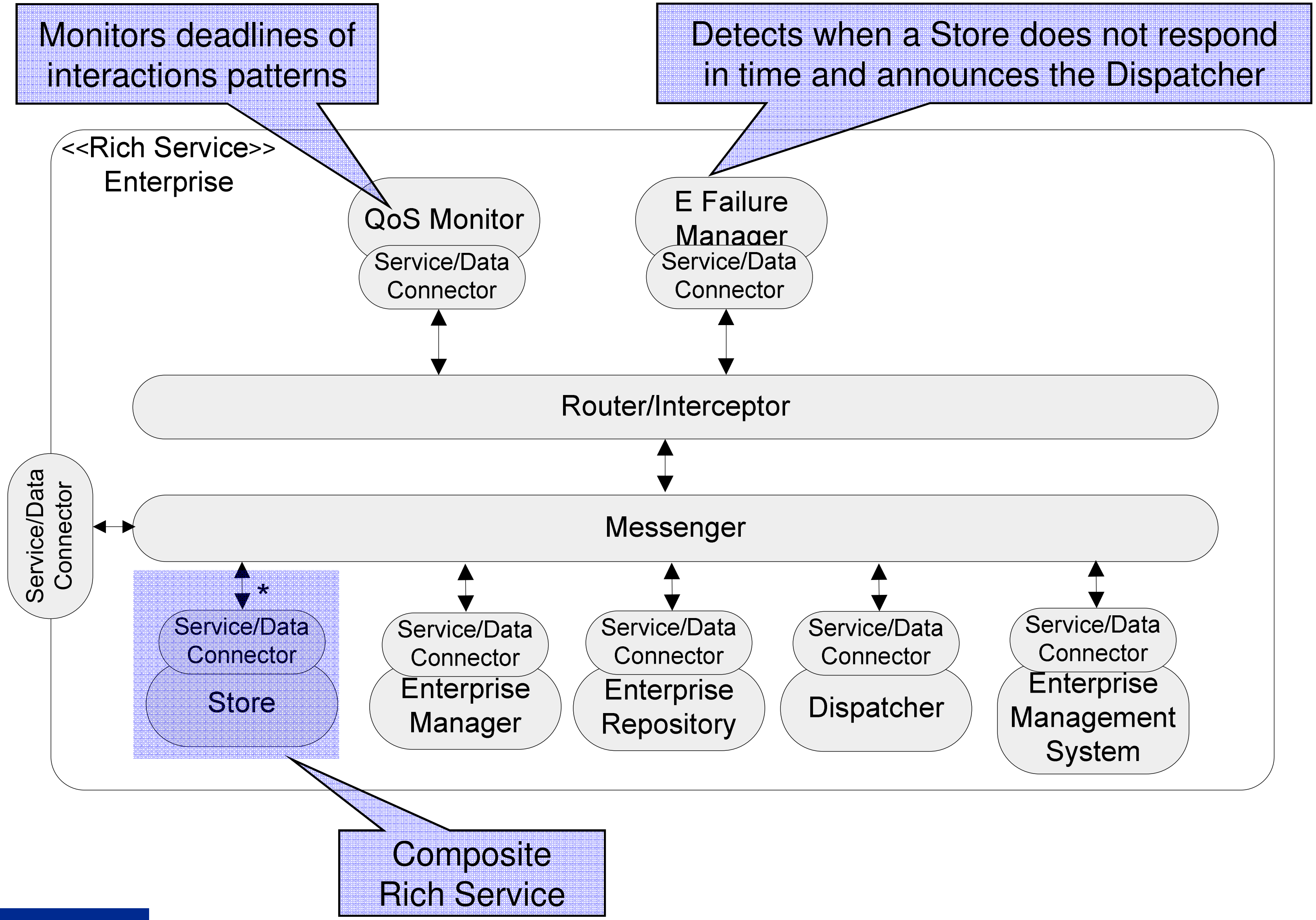
responsible for decrypting every message from the Enterprise to the Bank and encrypting messages from the Bank to the Enterprise



Composite Rich Service

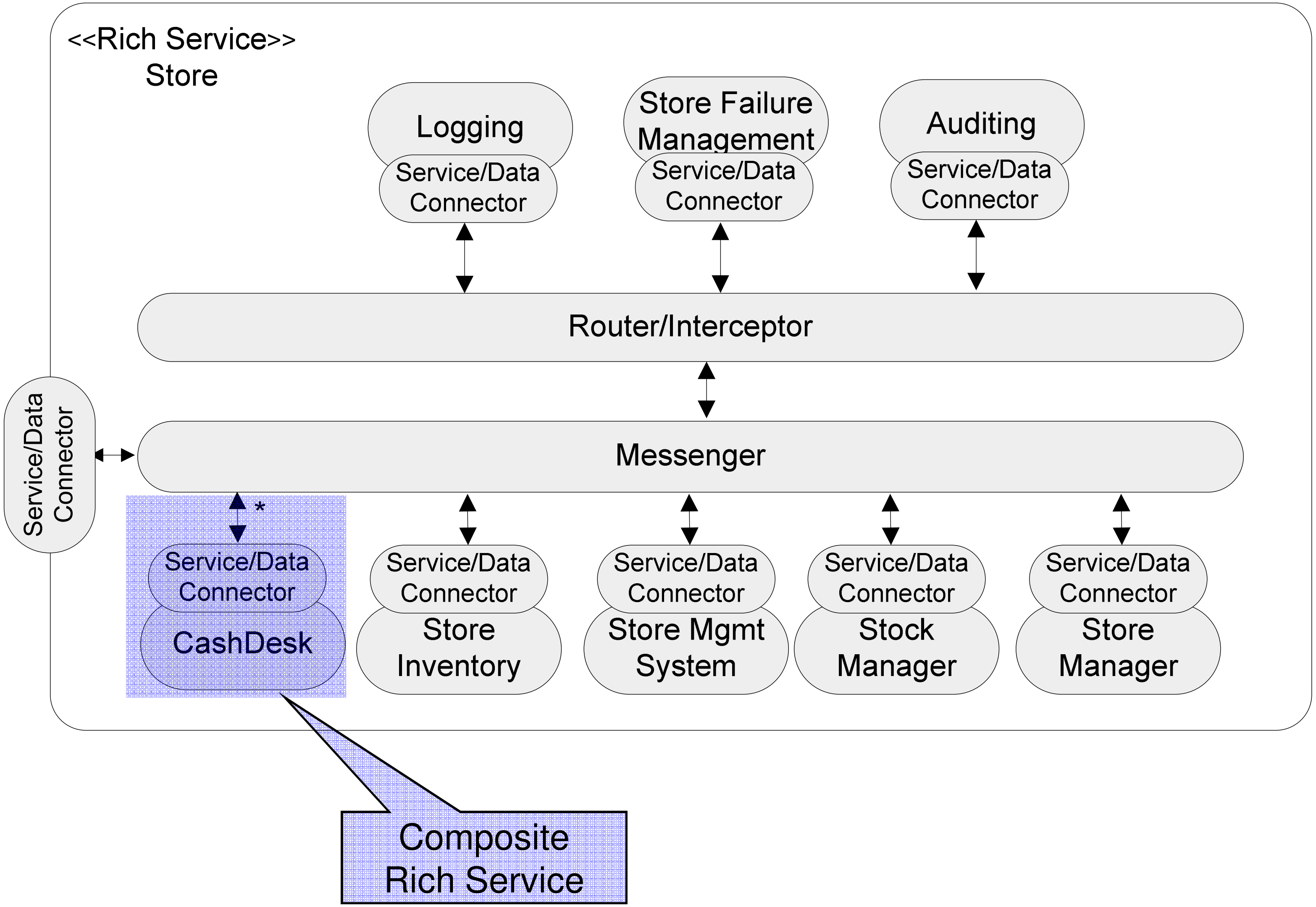
Service Elicitation | **Rich Services Architecture** | System Architecture Definition

Enterprise Rich Service

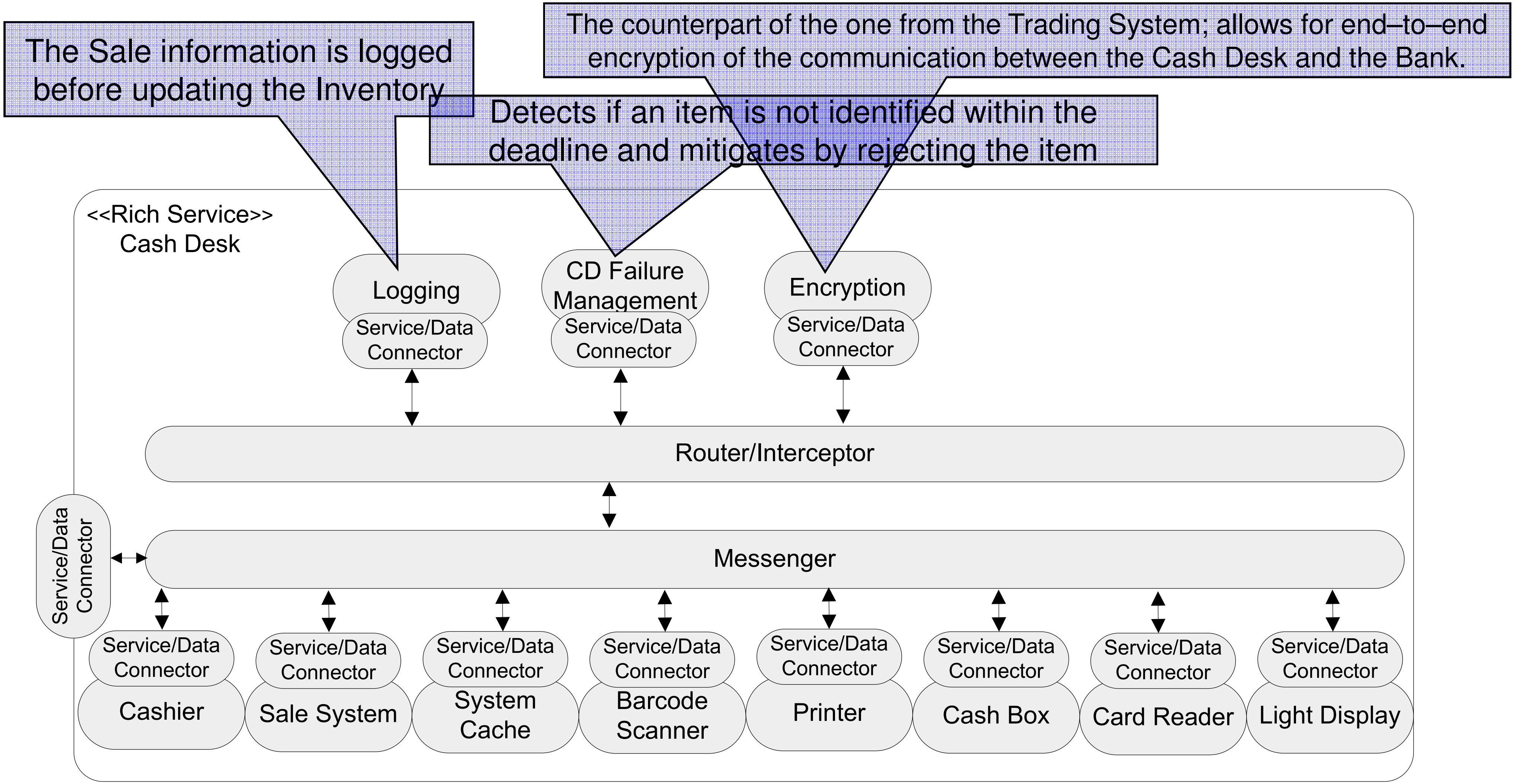


Service Elicitation | **Rich Services Architecture** | System Architecture Definition

Store Rich Service



Cash Desk Rich Service



UCSD
UCIrvine

Service Elicitation | **Rich Services Architecture** | System Architecture Definition

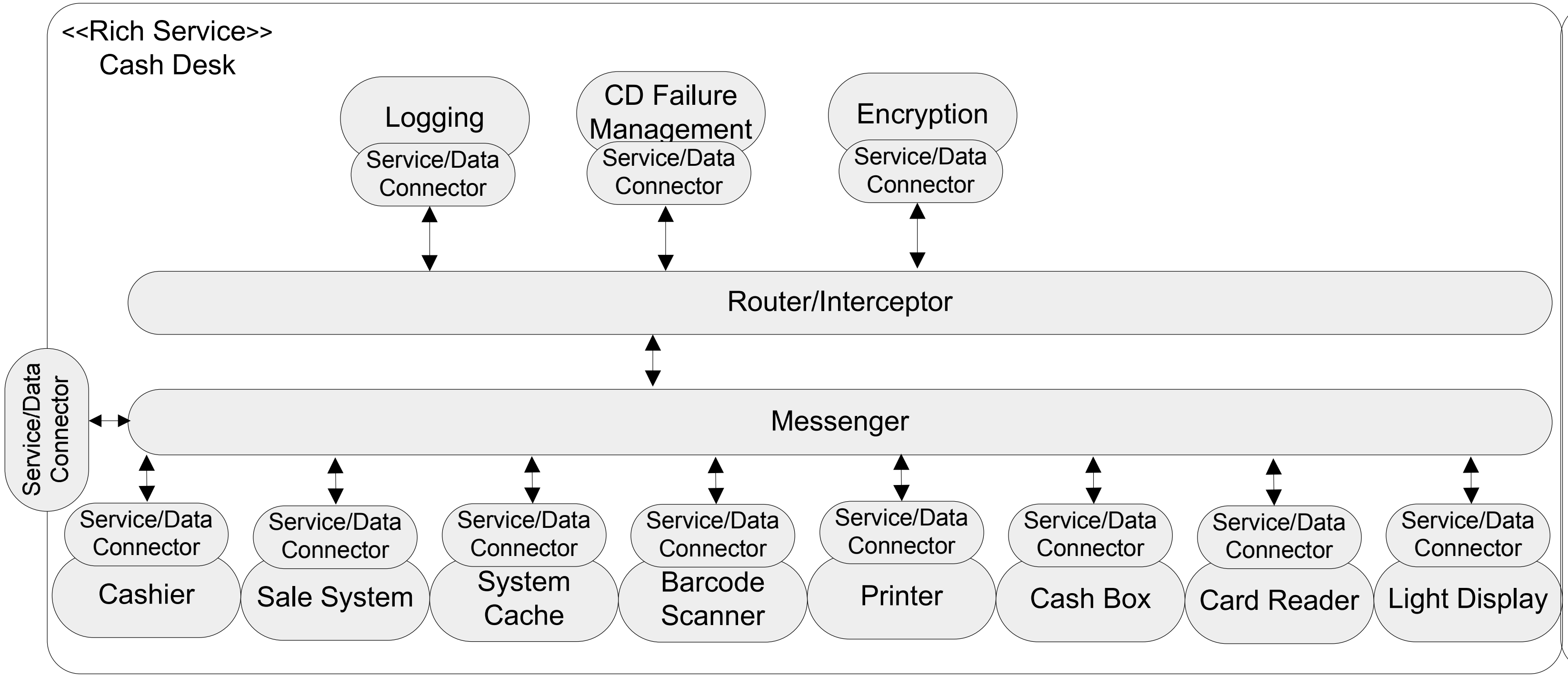
Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Cash Desk Rich Service

Static View

Behavioral View



<<Interactions>>
Cash Desk

CD-MSC

Normal Sale
Express Sale
Enter Express
Exit Express
Managed AddItem
AddItem Mitigate
Bar Payment
Card Payment
Update Cached Sales

Connector:
(Imports: Customer, Bank
Exports: Cashier, Card Reader, Light Display
MSCs: INormalSale, IExpressSale)
...

- **Cash Desk Rich Service: the lowest level in hierarchy.**
- **Each Rich Service is associated with a behavioral view.**

Rich Service Behavioral View

- **References the services (MSCs) taking place at this level**
- **Has a main MSC which captures how the services in that Rich Service are composed together to form the overall behavior of that Rich Service**
- **Declares the imported and exported roles**
  - **Imported Roles:**
    - **Roles from higher levels of hierarchy that interact with internal roles of the Rich Service**
  - **Exported Roles:**
    - **Internal roles of the Rich Service that will interact with external roles.**
  - **Import/Export Interface:**
    - **The communication pattern (MSCs) between Imported and Exported roles = behavior protocol**

<<Interactions>>
Cash Desk

CD-MSC

Normal Sale
Express Sale
Enter Express
Exit Express
Managed AddItem
AddItem Mitigate
Bar Payment
Card Payment
Update Cached Sales

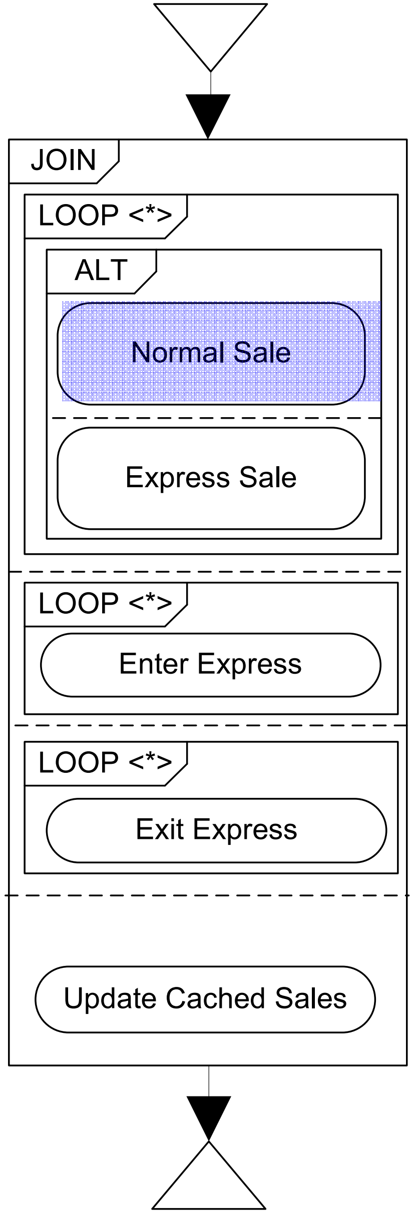
Connector:

(**Imports**: Customer, Bank
Exports: Cashier, Card Reader, Light Display
MSCs: INormalSale, IExpressSale)
...

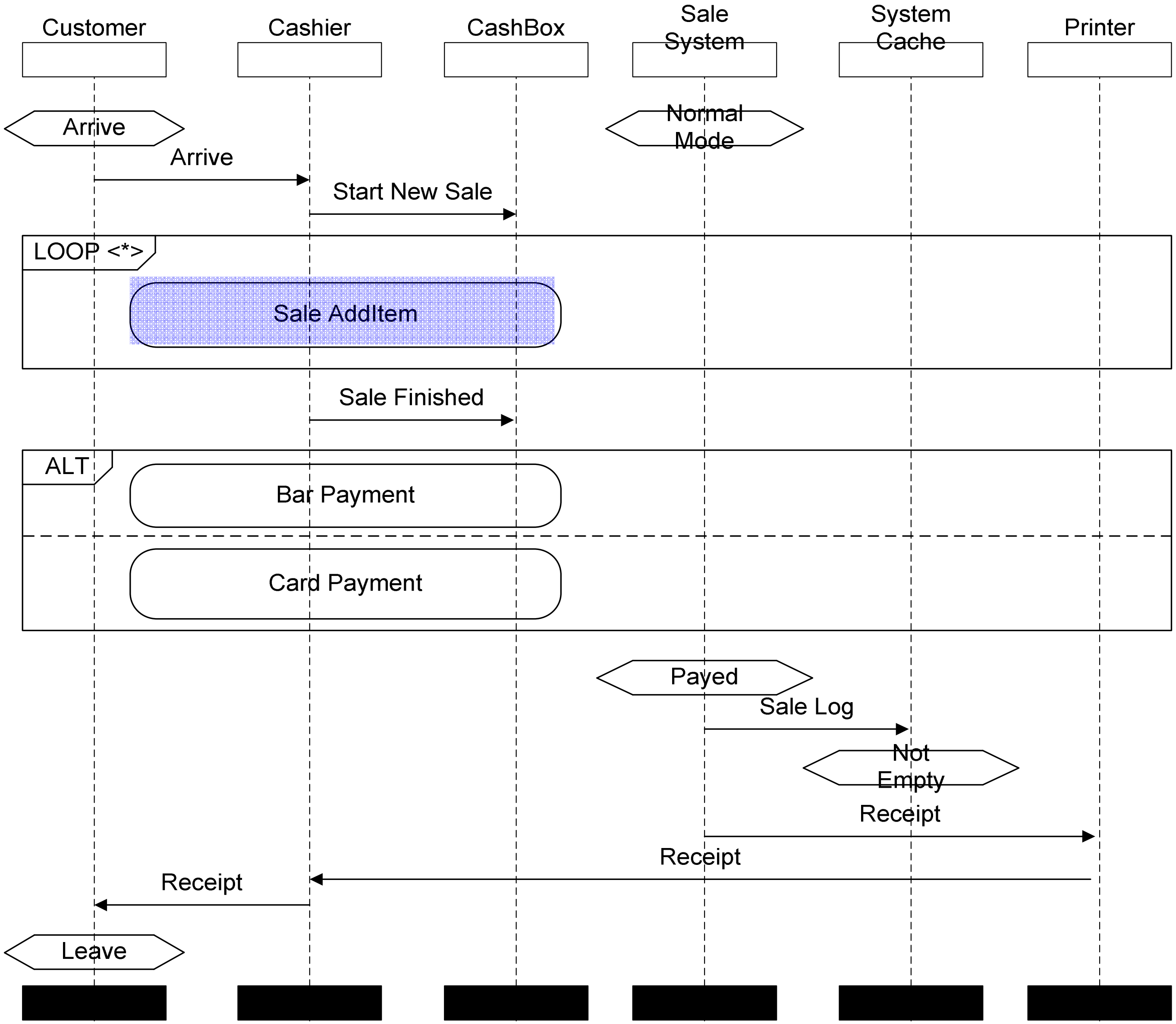
Cash Desk – Main MSC

- Only the services contained within, or interacting with, this Rich Service are captured in the main MSC of the Rich Service.

- JOIN operator: Composes the operand MSCs in parallel, synchronizing them on the common messages.



Cash Desk - Normal Sale



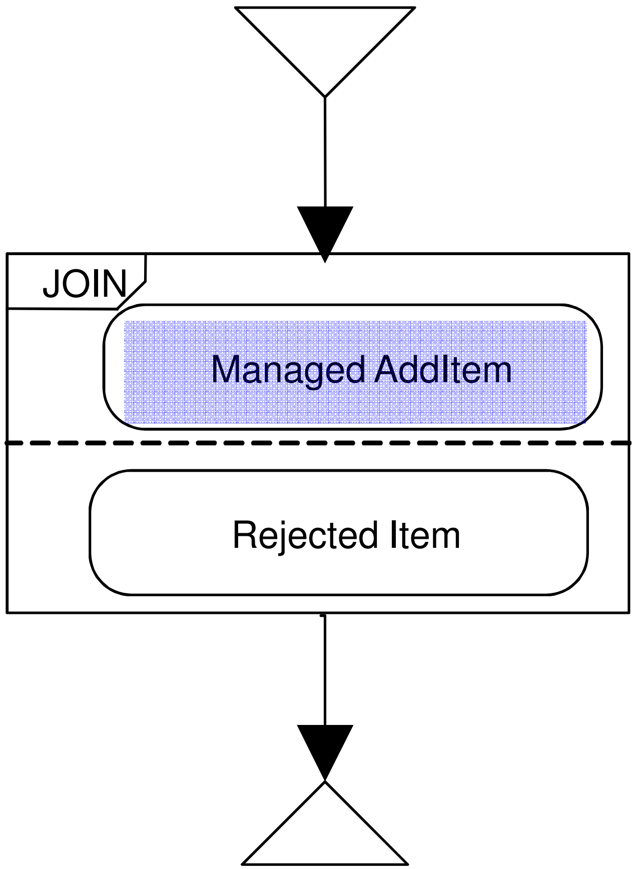
Service Elicitation | **Rich Services Architecture** | System Architecture Definition

Cash Desk – Sale AddItem MSC

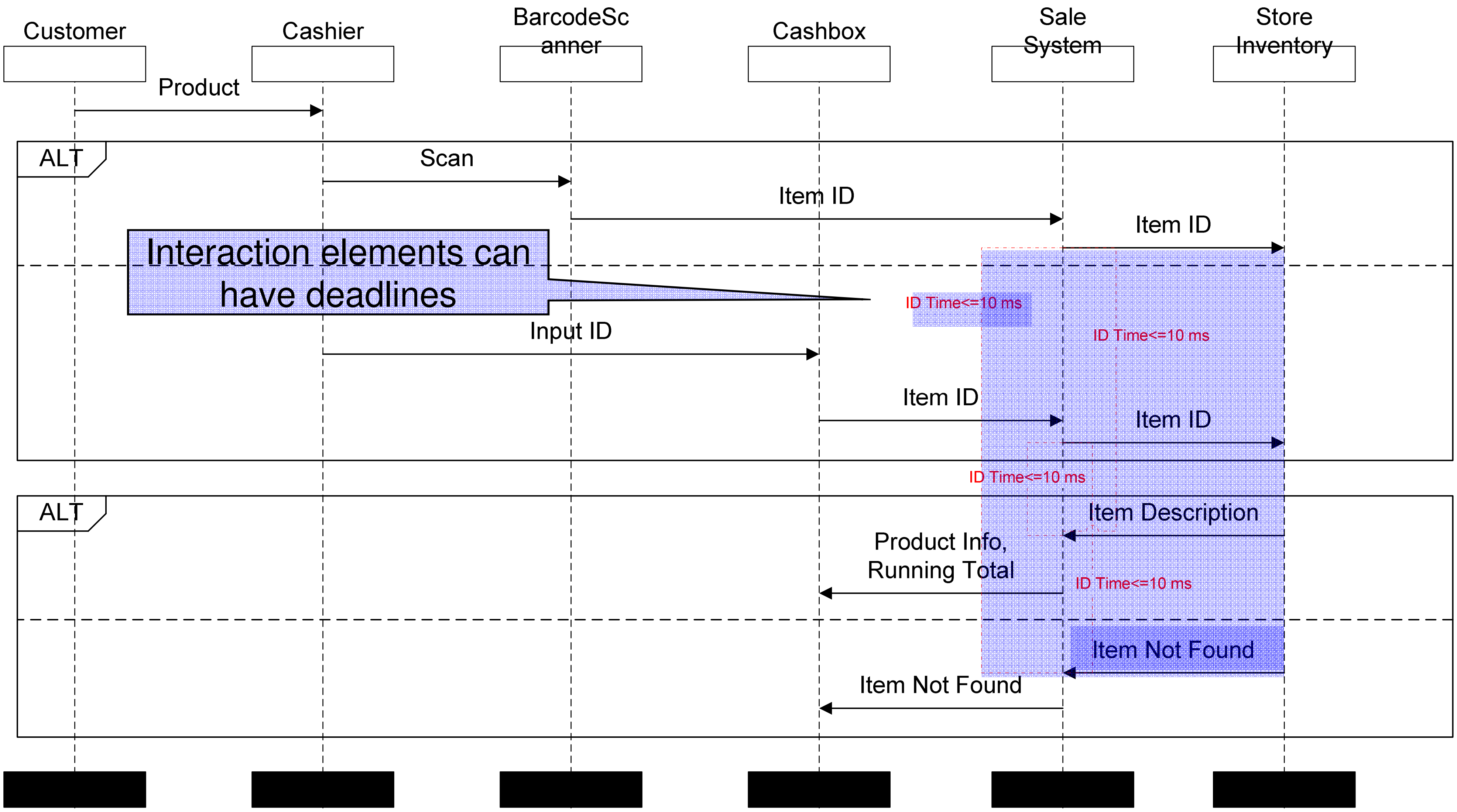
- Managed AddItem captures adding an item
  - The Item ID is sent to Store Inventory, which will provide the Item Description.
  - If the item is not recognized, the Store Inventory will reject the item
  - Shown on the next slide

- The Rejected Item MSC captures the Cashier's response when an Item is not found:
  - either reject the item or
  - enter the human readable item ID

Sale AddItem MSC



Cash Desk - Managed AddItem



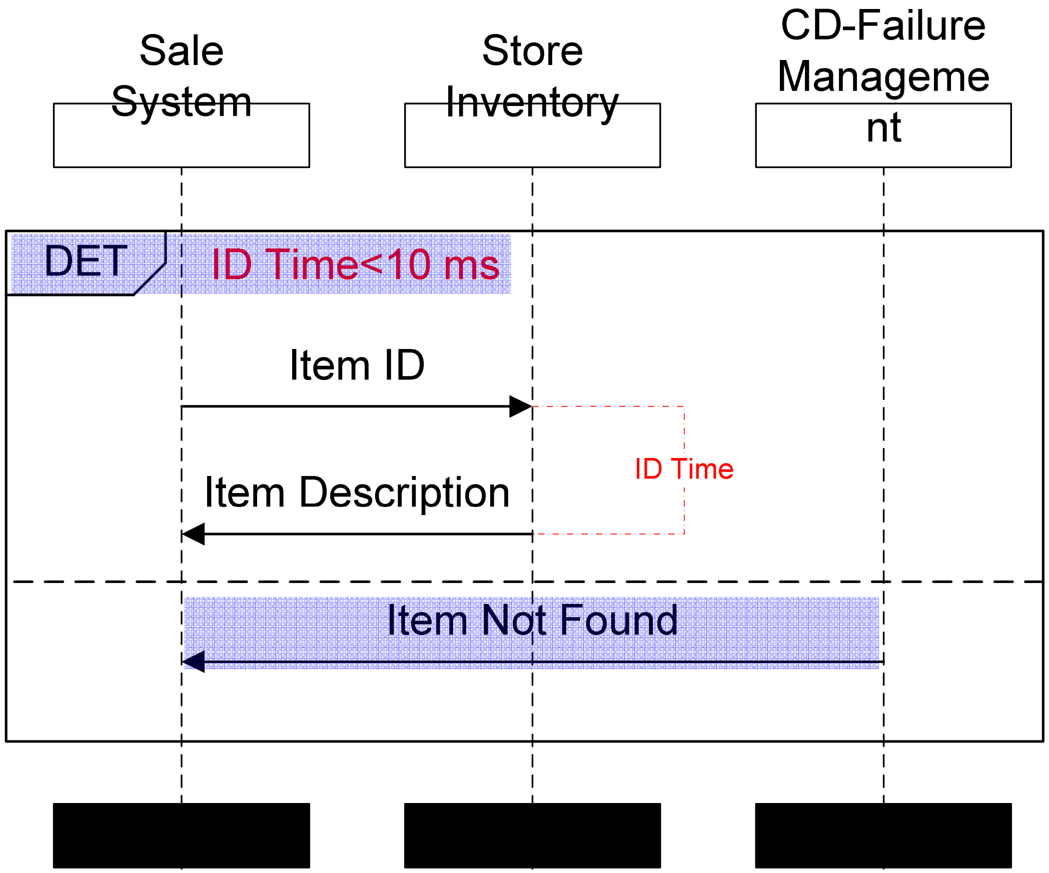
Interaction elements can have deadlines

Service Elicitation | **Rich Services Architecture** | System Architecture Definition

Cash Desk – Failure Management

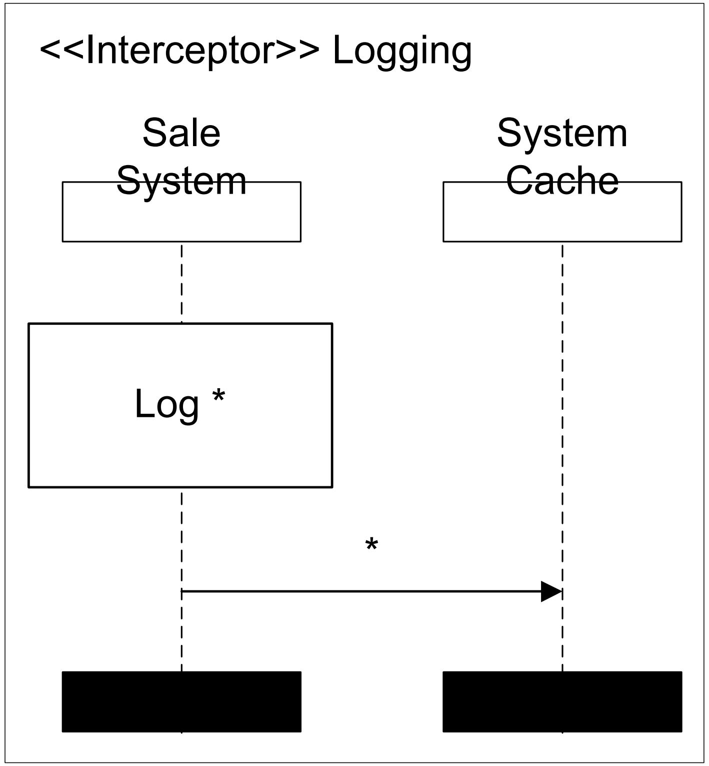
- If the Store inventory does not recognize the item in 10 ms, the CD-Failure Management will send an “Item Not Found” message on Store Inventory’s behalf.
- DET operator: Failure Manager monitors the interaction specified as the first operand. If the time condition is not met, the second operand interaction is activated.

AddItem FM MSC

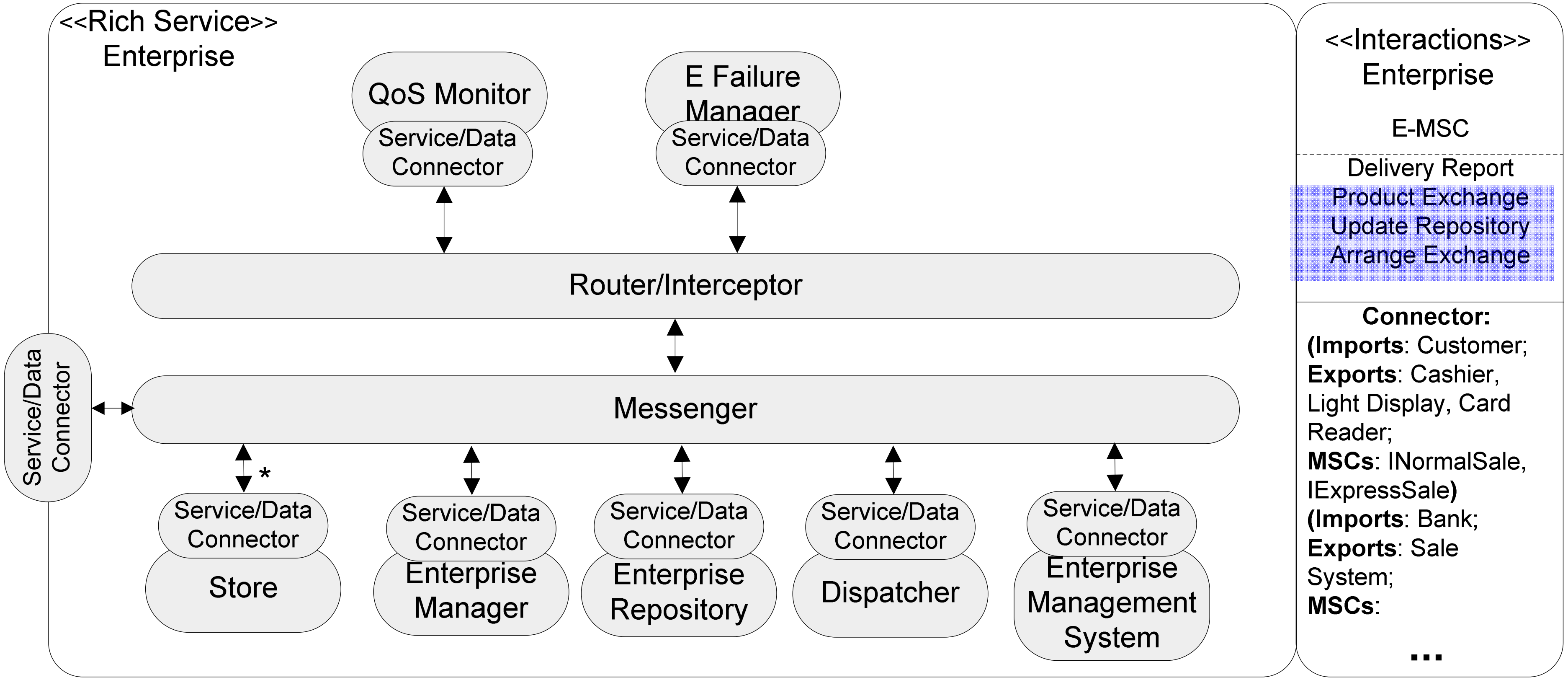


Logging Rich Infrastructure Service

- **Requirement: Each Sale must be logged before sending to Inventory for update.**
- **The Rich Infrastructure Service "Logging" intercepts every message sent by Sale System to System Cache to log the message.**
- **System Cache will periodically send its contents to Store Inventory if not empty.**



Enterprise Rich Service



<<Interactions>>
Enterprise

E-MSC

Delivery Report
Product Exchange
Update Repository
Arrange Exchange

Connector:
(Imports: Customer;
Exports: Cashier, Light Display, Card Reader;
MSCs: INormalSale, IExpressSale**)**
(Imports: Bank;
Exports: Sale System;
MSCs:

…

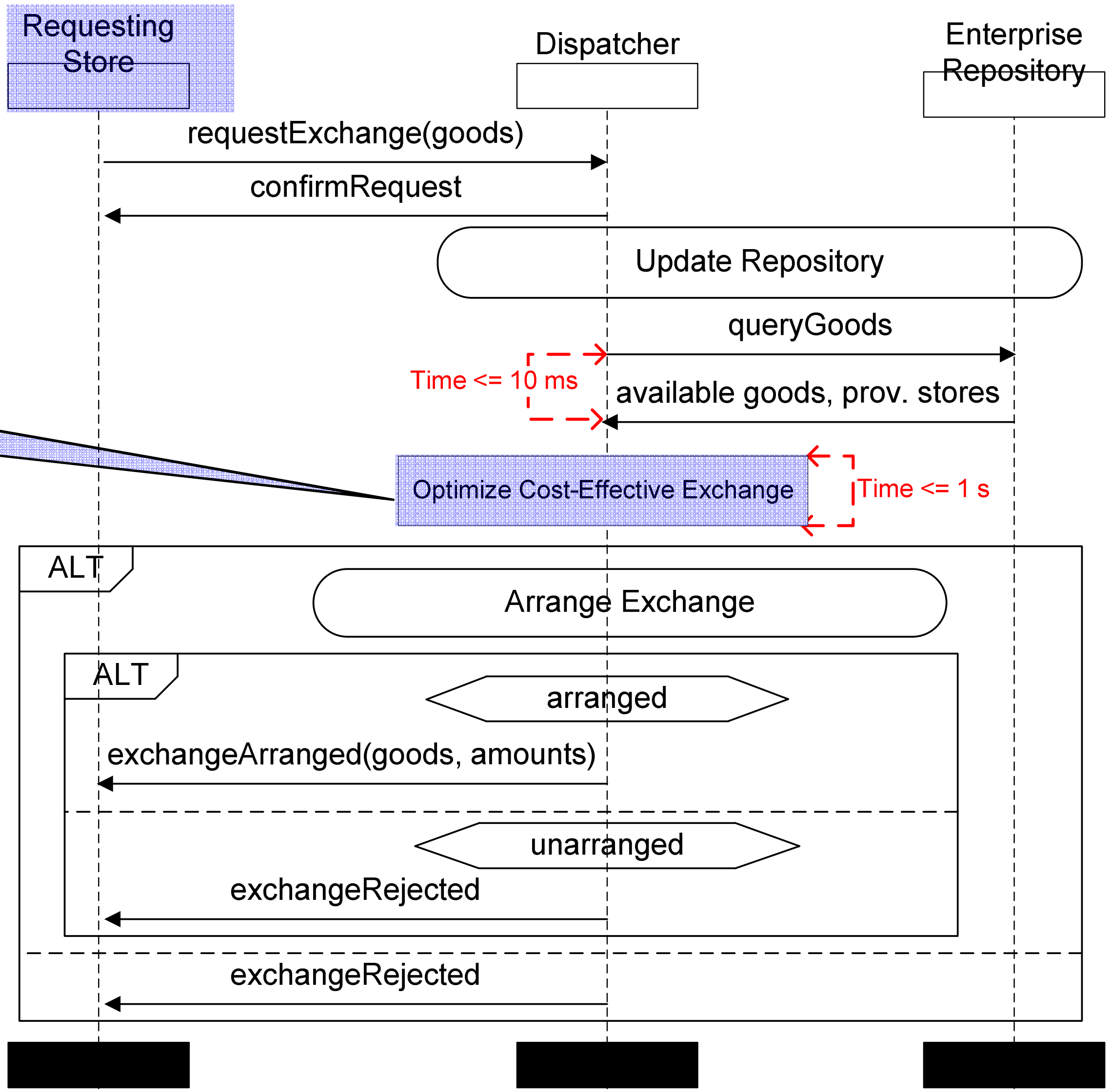
Service Elicitation | **Rich Services Architecture** | System Architecture Definition

Enterprise - Product Exchange

Product Exchange MSC

Store RS exports the roles Requesting Store, Nearby Store, and Providing Store

Internal computation

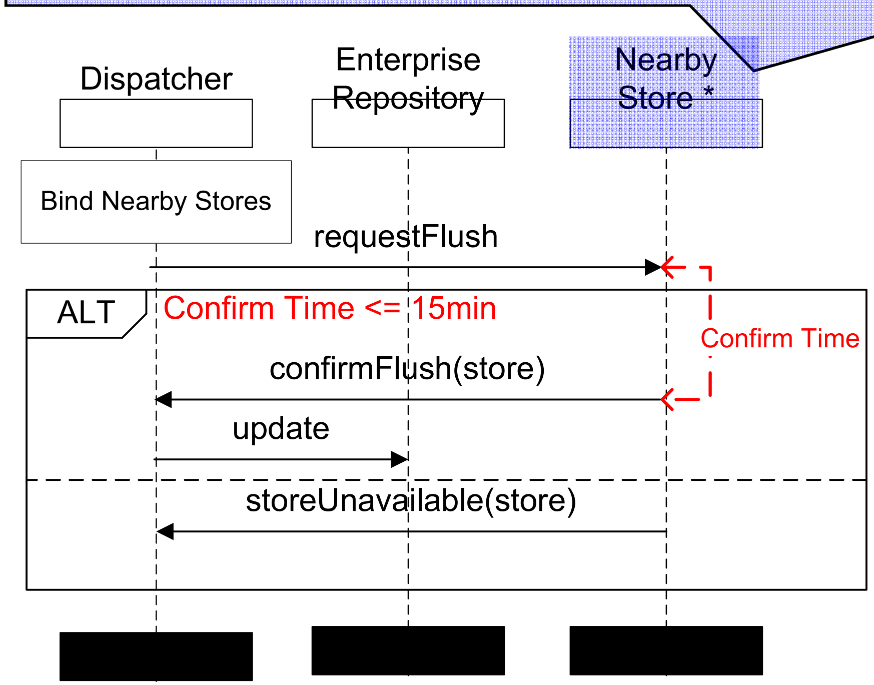


Service Elicitation | **Rich Services Architecture** | System Architecture Definition

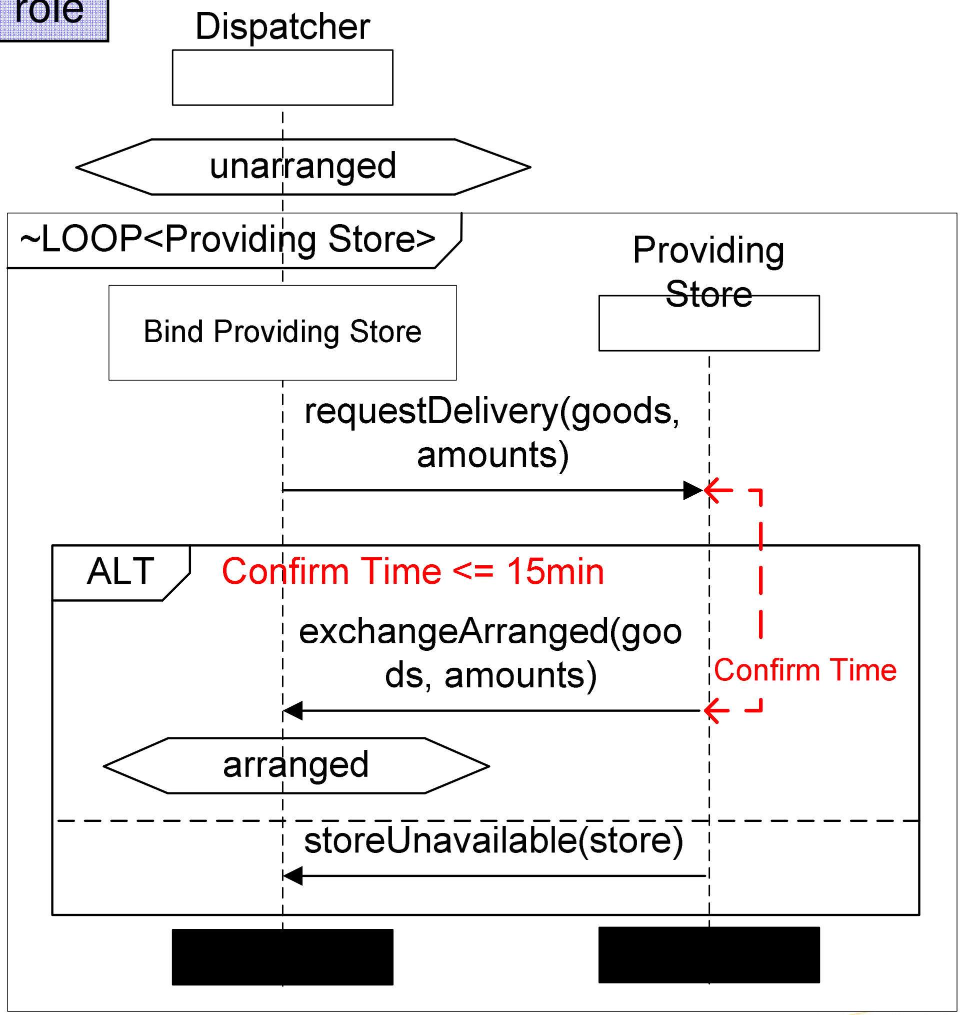
Enterprise - Product Exchange

Update Repository MSC

The asterisk (‘*’) notation =
The axis represents all components that implement this role



Arrange Exchange MSC



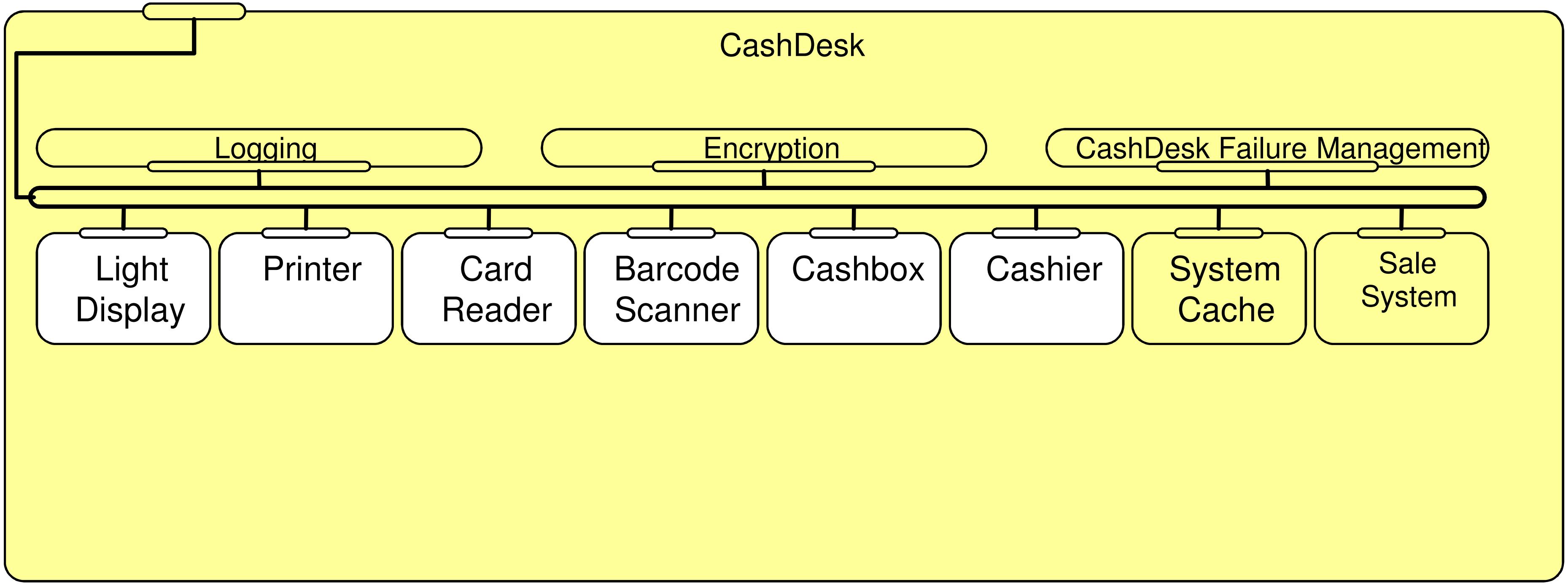
Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Hierarchical Deployment of Trading System

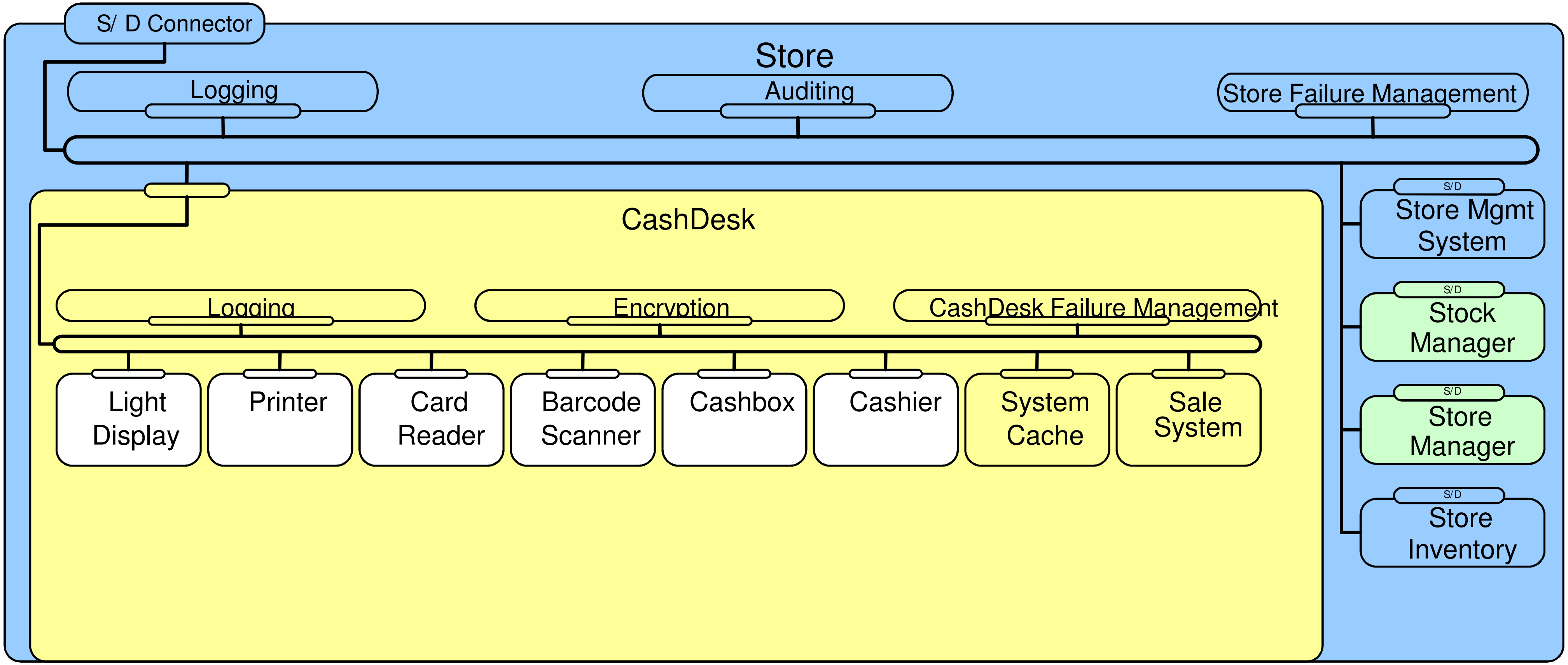
🟨 Cash desk PC

□ Other entities



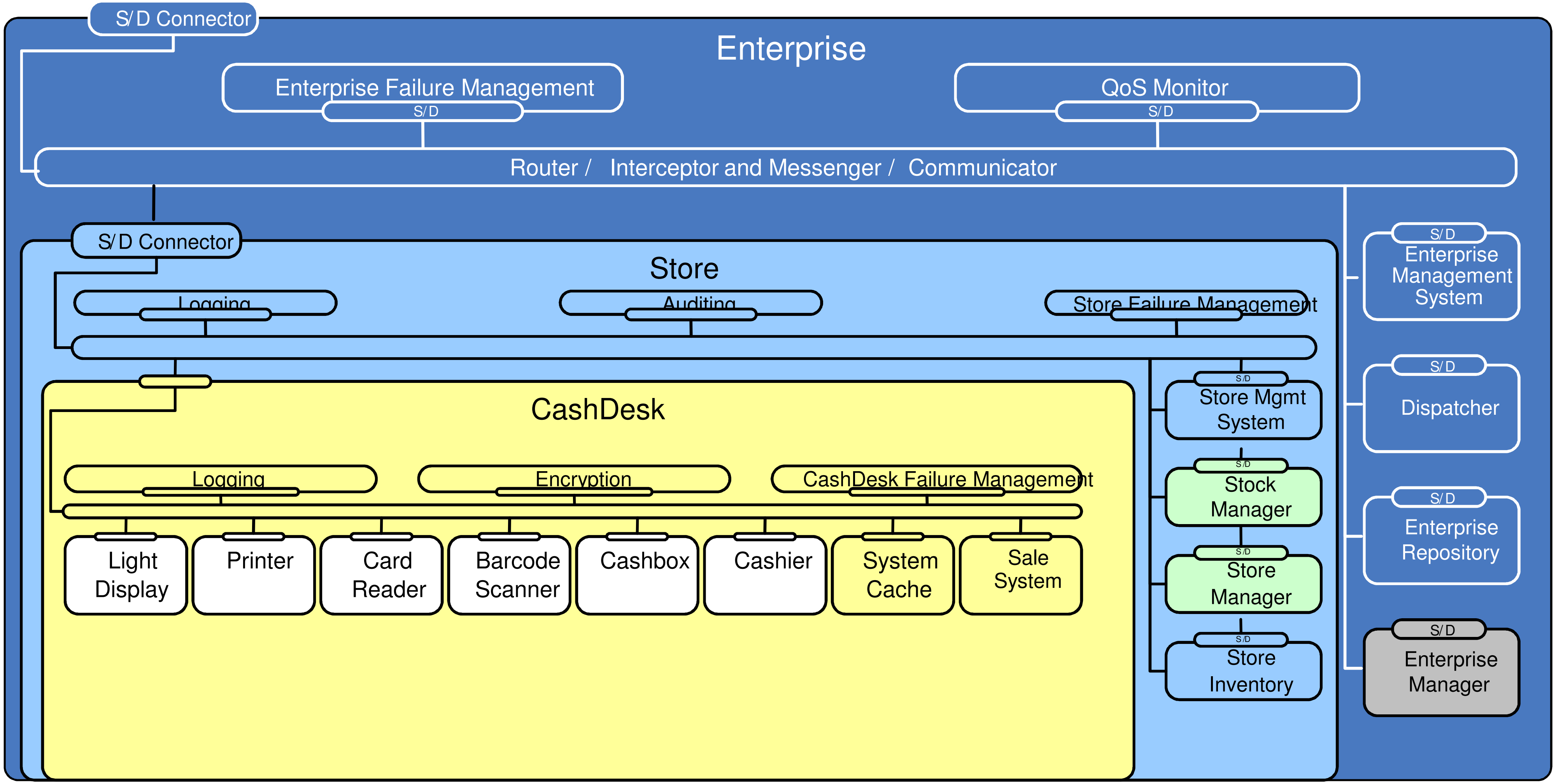
Hierarchical Deployment of Trading System

🟦	Store server	🟩	Store client
🟨	Cash desk PC	□	Other entities



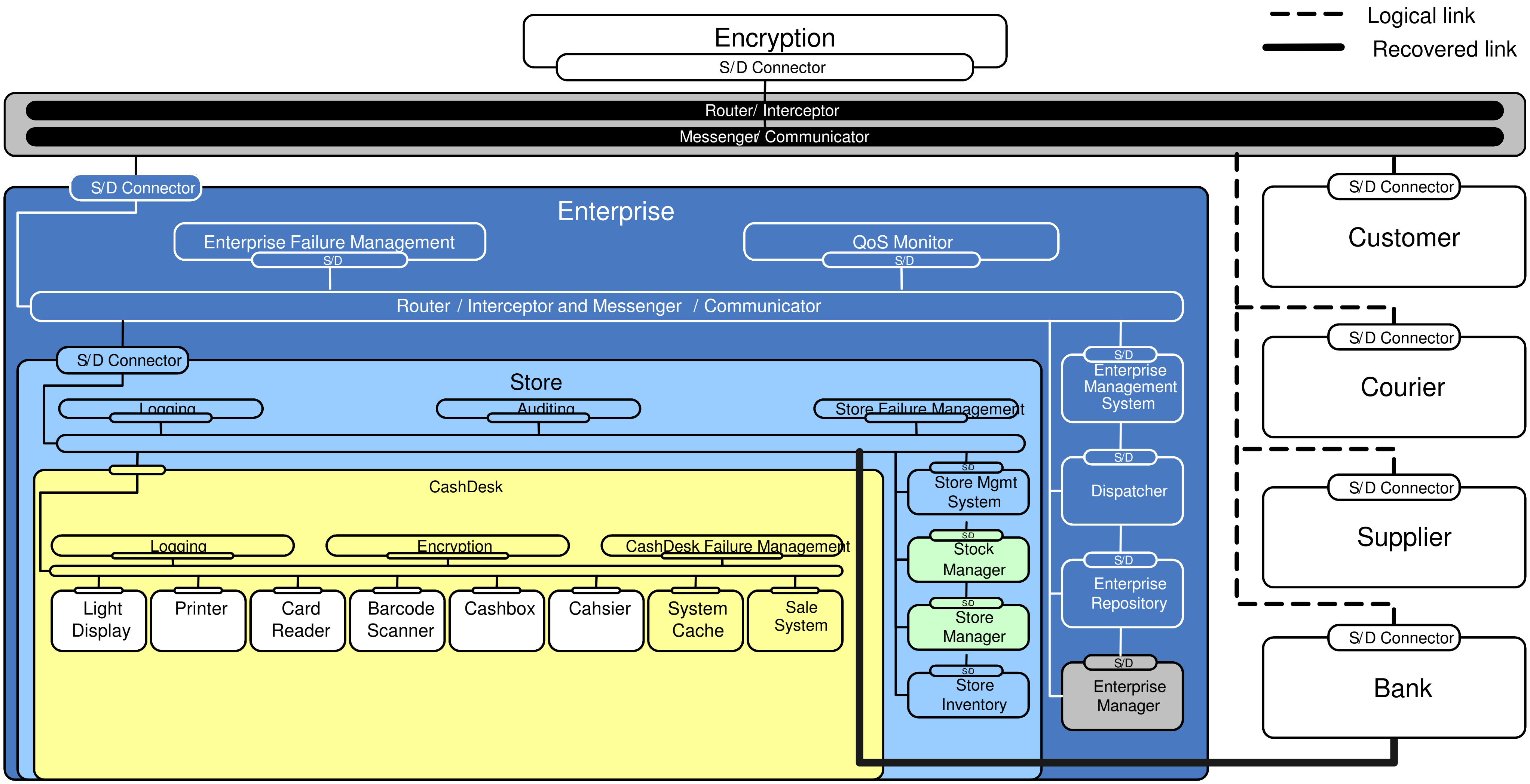
Hierarchical Deployment of Trading System

🟦	Enterprise server	◼	Enterprise client
🟦	Store server	🟩	Store client
🟨	Cash desk PC	□	Other entities



Service Elicitation | Rich Services Architecture | **System Architecture Definition**

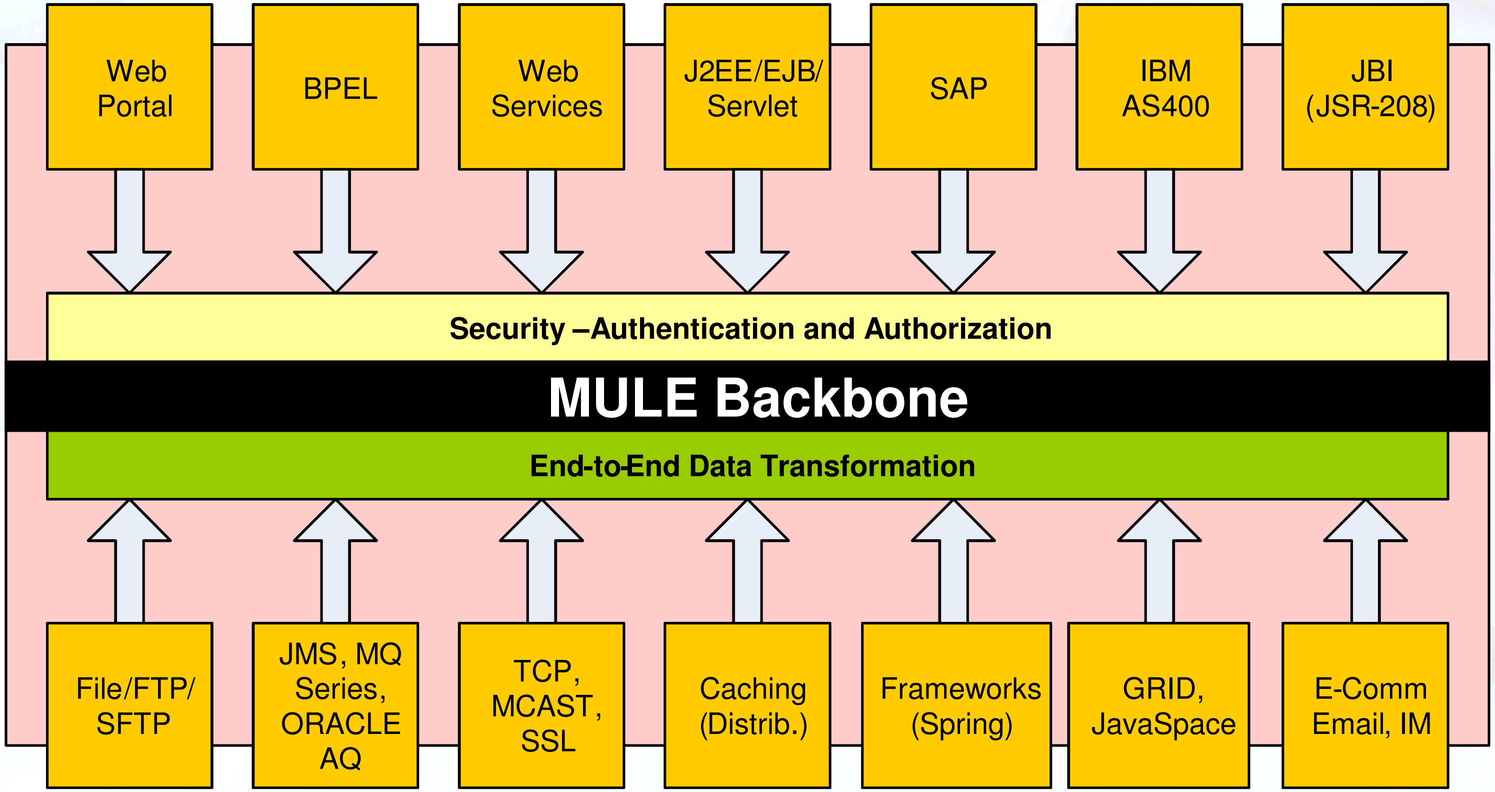
Hierarchical Deployment of Trading System



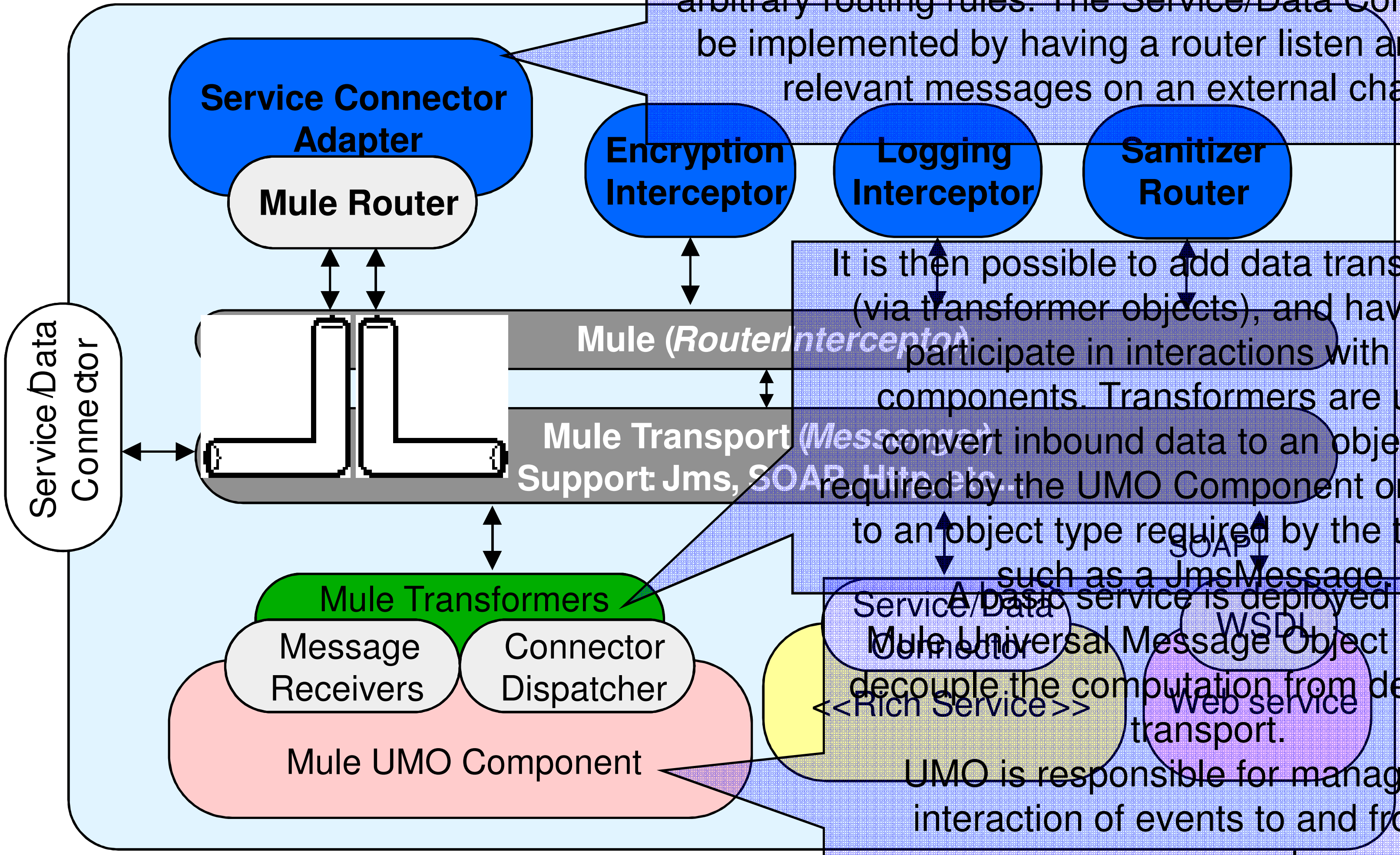
Service Elicitation | Rich Services Architecture | **System Architecture Definition**

MULE as deployment system

- **MULE Enterprise Service Bus**
  - –Relatively new technology with great potential
  - –Ad-Hoc development process, needs new SOA perspective
  - –Rich Services are a perfect match



Deployment of Rich Services using MULE



Routers allow the Mule framework to receive messages from one channel and re-publish them according to arbitrary routing rules. The Service/Data Connector can be implemented by having a router listen and publish relevant messages on an external channel

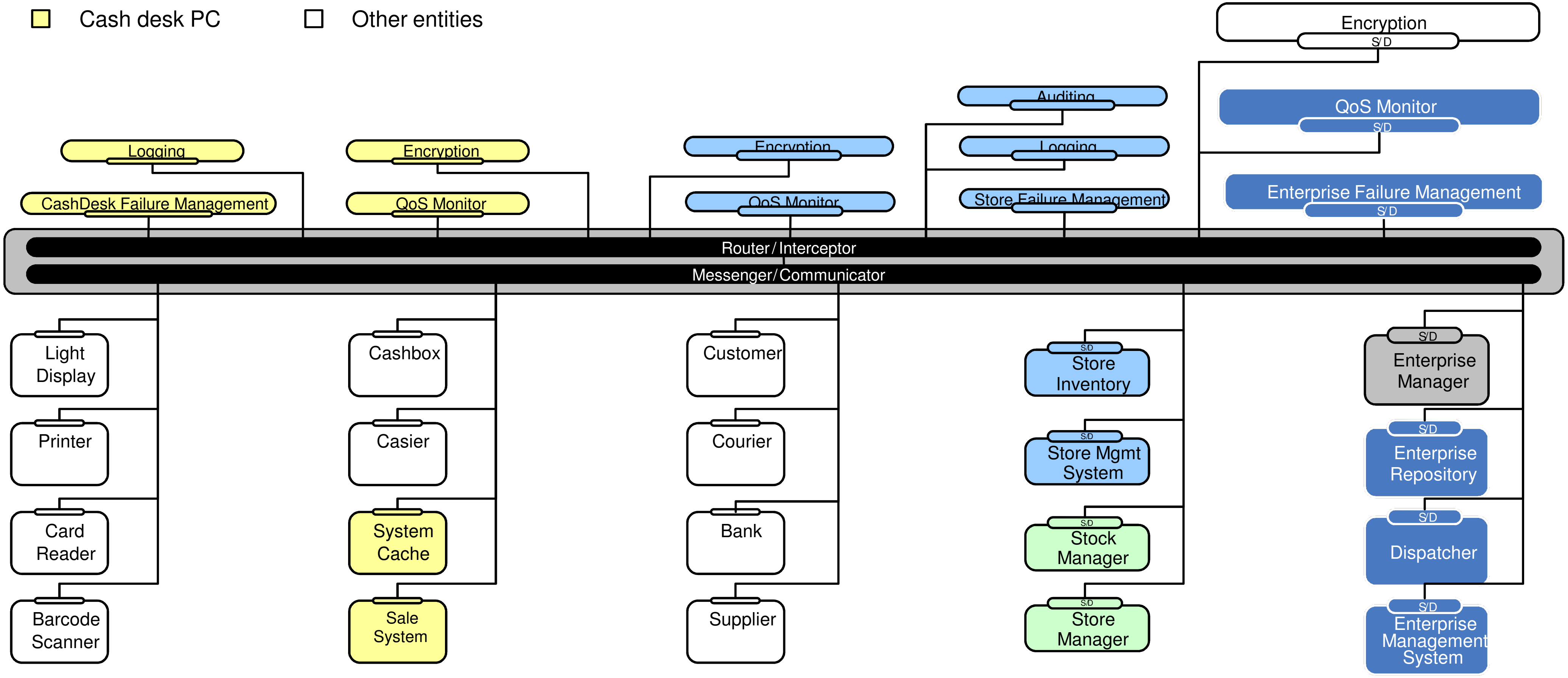
It is then possible to add data transformation (via transformer objects), and have UMOs participate in interactions with other components. Transformers are used to convert inbound data to an object type required by the UMO Component or outbound to an object type required by the transport such as a JmsMessage.

A basic service is deployed as a Mule Universal Message Object (UMO) to decouple the computation from delivery and transport.

UMO is responsible for managing the interaction of events to and from the component

Flat Deployment of Trading System

🟦	Enterprise server	▩	Enterprise client
🟦	Store server	🟩	Store client
🟨	Cash desk PC	◻	Other entities



Service Elicitation | Rich Services Architecture | **System Architecture Definition**

Outline

- Team Introduction
- State of the Art and Challenges of SOA Integration
- Our methodology
  - Rich Services
  - Development process
  - Message Sequence Charts
- Modeling the CoCoME
  - Modeling of the static view
  - Modeling of the behavioral view
  - Deployment Strategies for Rich Services using ESB Technology
- Summary, Experiences, and Outlook

Summary

- Rich services
  - THE integration piece of the SOA puzzle
  - Flexible handling of horizontal & vertical integration concerns
  - Address cross-cutting concerns, including failure management
  - Useful as logical *and* deployment architecture model
  - Immediate mapping to wide variety of deployment architectures, including ESB

- Service-orientation & workflows go well together
  - Workflows become service choreographies
  - Resource location independence

Rich Services provide a flexible and comprehensive architectural framework that reconcile the notion of hierarchy with the service notion!

Experiences and Lessons Learned from CoCoME

- **Decomposition**
  - Decomposing a system based on a concern influences the whole system architecture, and has inevitable consequences.
    - tyranny of the dominant decomposition
  - One should study such impacts before making the decision.

- **Deployment**
  - Flattened versus hierarchical deployments
  - Advantages of using an ESB based deployment
    - Built in services and intercepting mechanisms
    - One-to-one mapping between the architecture and deployment
    - Greatly increases code reuse

Experiences and Lessons Learned from CoCoME

- **Crosscutting Concerns**
  - Rich Services allow addressing the crosscutting concerns in a centralized way.
    - we can easily deal with policies imposed at different business tiers.
  - Intercepting capability of Rich Infrastructure Services
  - Routing capabilities
    - enables the definition of routing policies and interceptors without changing existing services

Experiences and Lessons Learned from CoCoME

- CoCoME was a very good case study for Rich Services
  - A good match to Rich Services hierarchy and support for policies
  - It validated our ideas on crosscutting concerns
  - We improved our development process for the transition from services to Rich Services
  - We improved our notations for Service/Data Connectors,
    - as we had many imported/exported roles in CoCoME
  - We found out that Application Services can also define routing schemes
    - In the Product Exchange Use Case, the Enterprise's Dispatcher dynamically binds Providing Stores to actual stores

We thank the organizers for the selection of the case study!

Outlook for Rich Services

- We support deadline constraints for interactions
  - Supporting more QoS properties and error-recovery behaviors when constraints are not met

- We have built a tool chain to support a traditional service-oriented software development process
  - Leveraging these tools and support ESBs in the context of Rich Services
  - the integration of dedicated refactoring techniques to simplify the integration of legacy systems into the Rich Services framework
  - UML Profile for Rich Services (ADL)

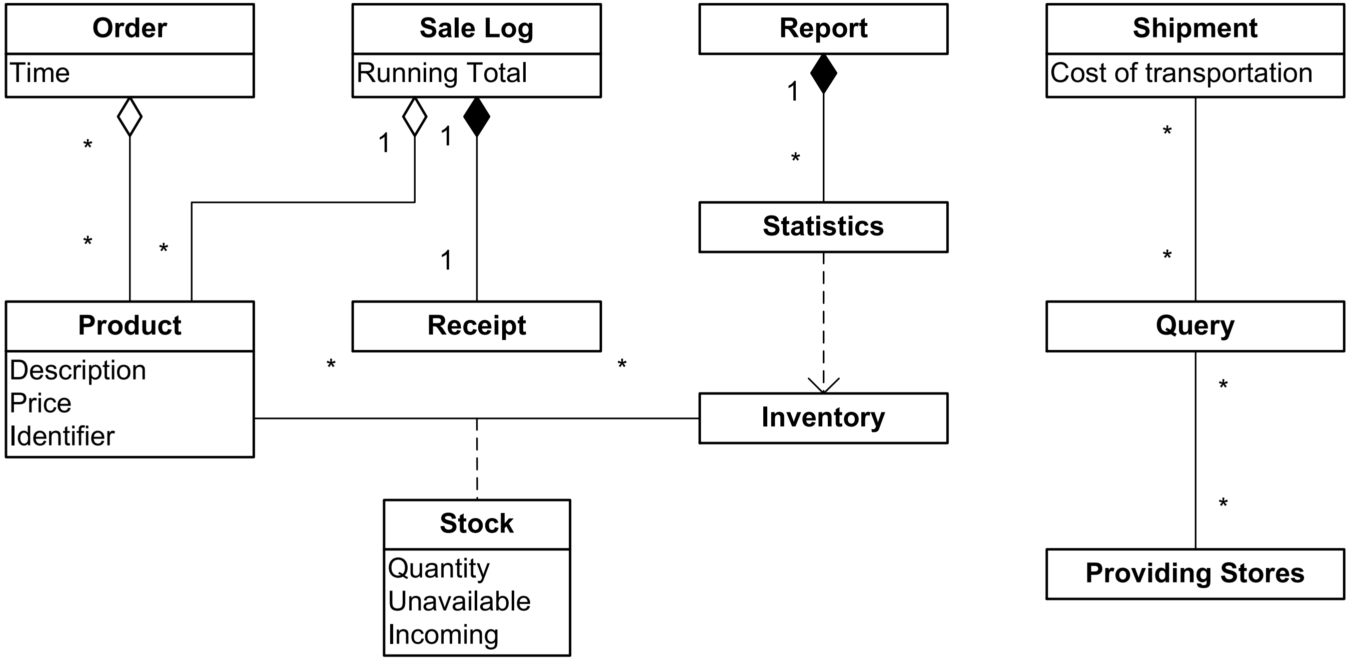
The End

Thank you !

Services – Workflow Management

- Service-orientation and workflows
  - Good match: every service can become an action/step within a workflow
  - Resource location no longer matters
  - Web service base technologies (publishing, lookup, binding, transport, …) well understood
- Areas of opportunity
  - System-of-systems integration
  - Handling of cross-cutting concerns, especially: policies, governance, QoS, failure management
  - Service- and workflow hierarchies
  - How to elicit and define workflows?

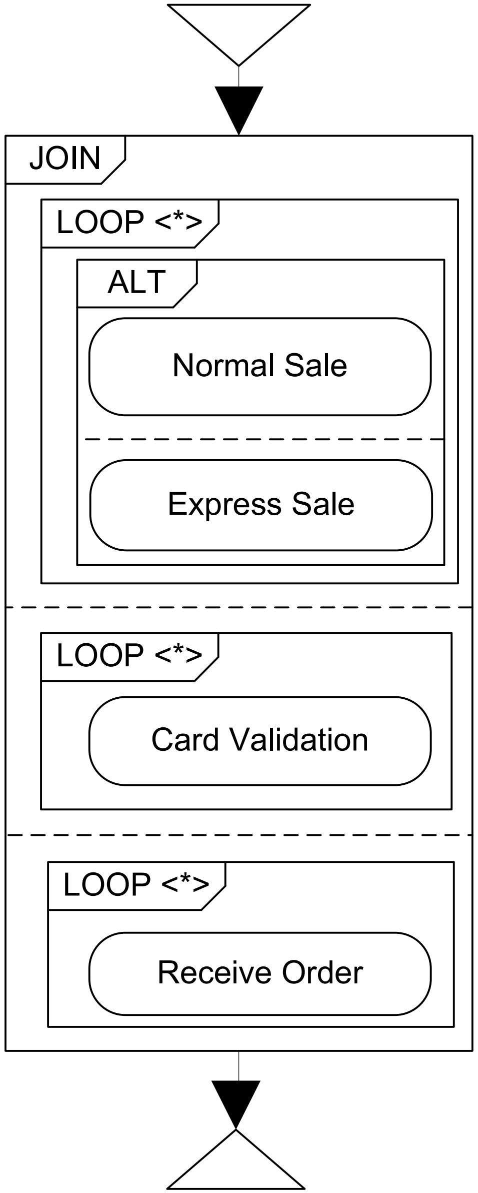
Trading System – Data Domain Model



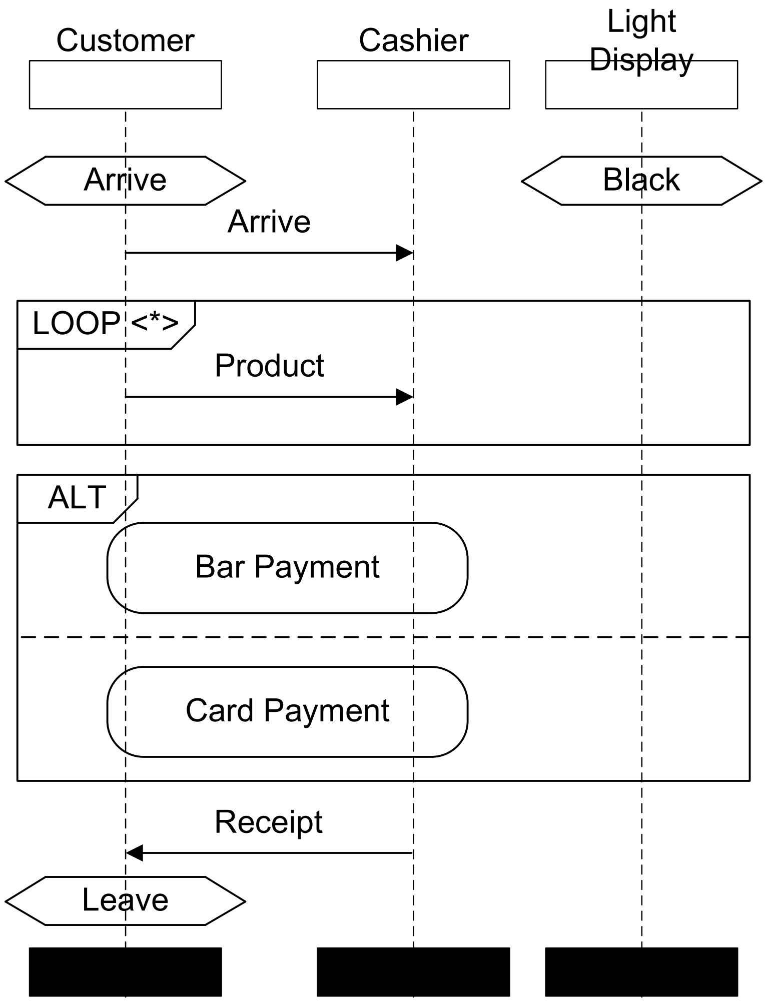
UCSD
UCIrvine

Trading system MSCs

Main MSC

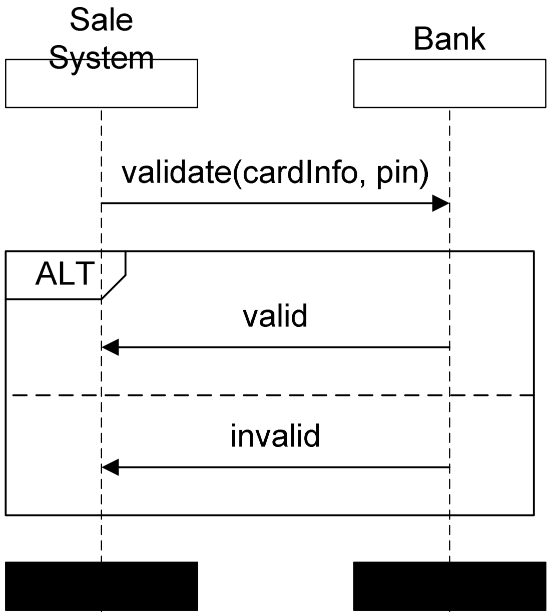


Normal Sale MSC

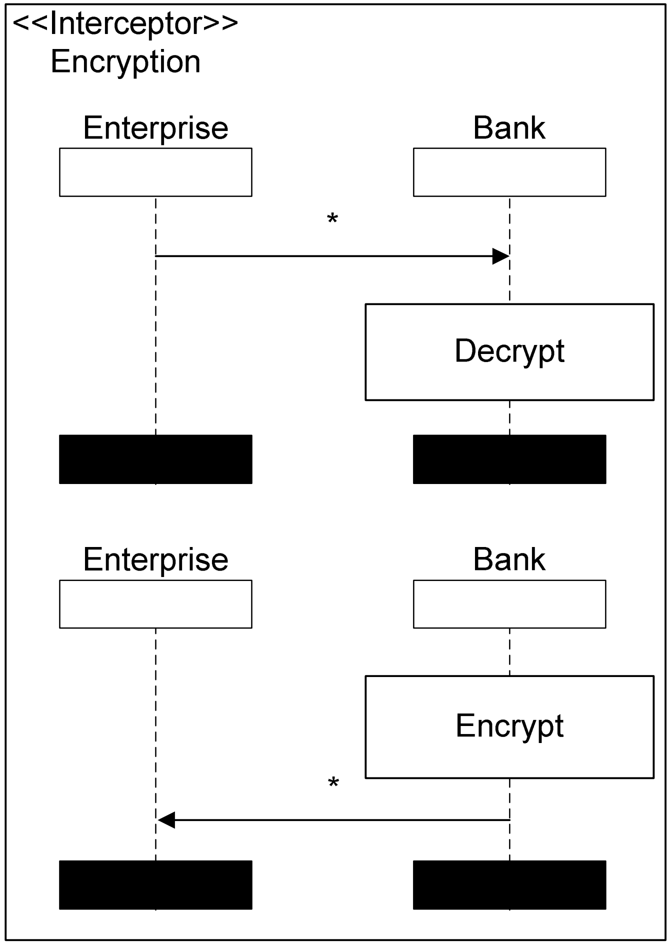


Trading system

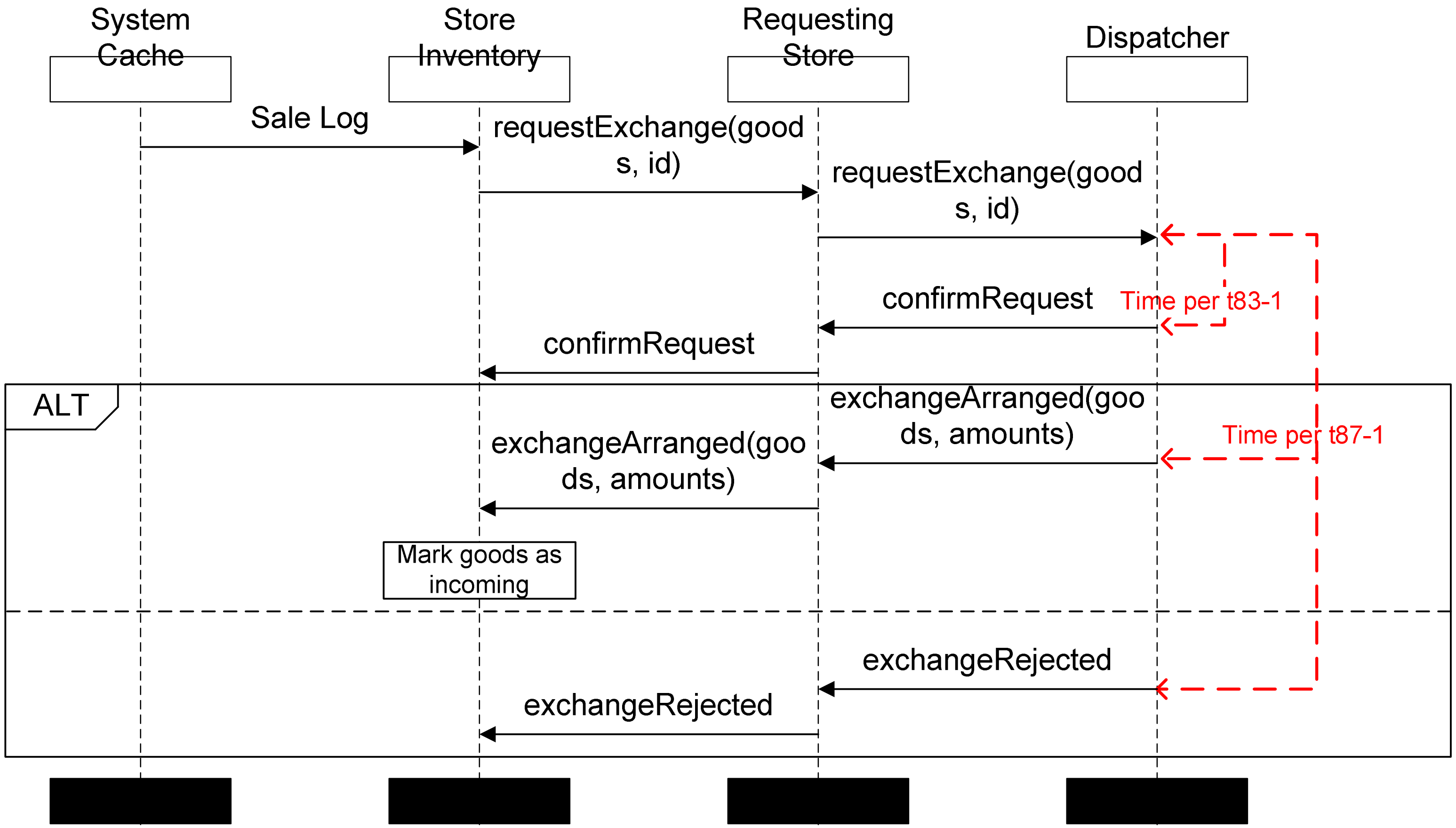
Card Validation MSC



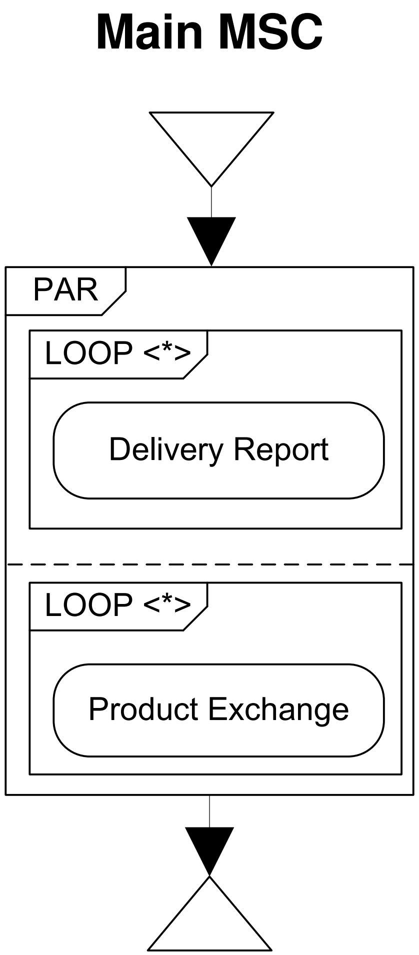
Encryption MSCs



Store – Product Exchange



Enterprise - Main MSC


Main MSC
PAR
LOOP <*>
Delivery Report
LOOP <*>
Product Exchange