Karlstad University
Department of Information Systems

A New Foundation for Service – Oriented Analysis and Design

Remigijus GUSTAS
Department of Information Systems, Karlstad University
Phone: +46-54 700 17 65
E-mail: Remigijus.Gustas@kau.se
http://www.cs.kau.se/~gustas/

Two Challenges

- ✓ Foundation for Graphical Modelling of Service – Oriented Architectures (SOA)
  - SOA need to be captured, visualised and agreed across the organisational and technical system boundaries.
  - Just as the complex buildings or machines require explicit representations of design structures, so does an overall SOA
- ✓ Integration Principles of SOA

Enterprise Architecture – Framework (Zachman, 1996)

	Data (What)	**Function (How)**	**Network (Where)**	**People (Who)**	**Time (When)**	**Motivation (Why)**
Scope **Planner view**	List of concepts	List of processes	List of locations	List of organizational units	List of business events	List of business goals
Organisational system **Owner view**	Entity relationship diagram	Business Process diagram	Diagram of logistic network	Organisation decomposition chart with roles	Schedule charts	Business Strategy / Plan
Information System **Designer view**	Logical Data Architecture	Software application architecture	Distributed system architecture	Human interface architecture	Control structure	Constraints and rules
Technology **Builder view**	Physical Data architecture	Deployment architecture	Techno-logy archi-tecture	Presentation/ Layout structure	Component control structure	Rule design
Representations **Subcontractor view**	Data definition	Process design	Network architecture	Interface architecture	Timing definitions	Rule specification
Functioning System	Data	Components	Network	Organisation	Schedule	Strategy

Integration Principles

- ✓ Integrated representations of SOA are necessary to reach **consensus** partners involved.
- ✓ **Holistic** understanding of SOA is typically not available. It is necessary for planning of orderly transitional processes.
- ✓ Traditional methods are dividing system specifications into **separate** parts (they are typically devoted to data architecture, application architecture or technology architecture).
- ✓ **Interdependencies** between different views are crucial to glue the static and dynamic aspects.

Reassessment of the existing theories, concepts and practices

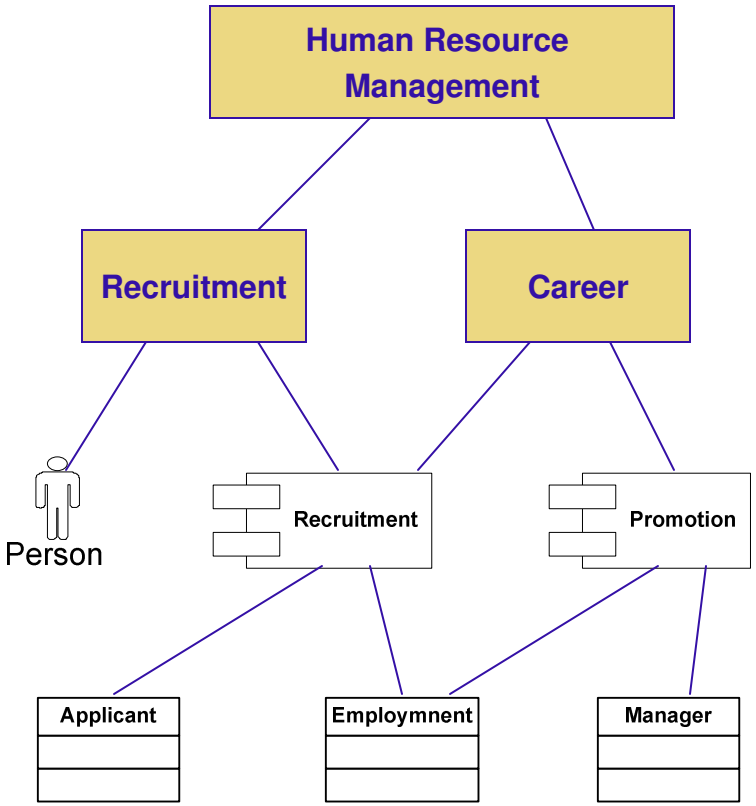
- ✓ Service Oriented Analysis and Design (SOAD) should represent **only conceptually relevant aspects** and it cannot be influenced by the possible implementation solutions. If graphical specifications follow the basic conceptualization principle (Griethuisen, 1982), they are **less complex** and, therefore, **more comprehensible** for humans.

- ✓ The fundamentals of engineering like **good abstractions**, good separation of concerns never go out of style (Booch, 2004). Nevertheless, the implementation bias of many system analysis methods is a big problem, since the same implementation - oriented foundations are applied for the system analysis stage, without **rethinking these concepts fundamentally**.

- ✓ Technology neutral descriptions of SOA should provide **integration principles** of the isolated diagrams at the lower levels of abstraction. Separate technical system representations are difficult to maintain. More reasonable is to **conceptualize SOA**, before the supporting technical system is defined. **Interdependencies** across multiple diagrams should be a critical part of business process modelling, since it is supposed to orchestrate the interoperation details into **one model**.

Design Layers

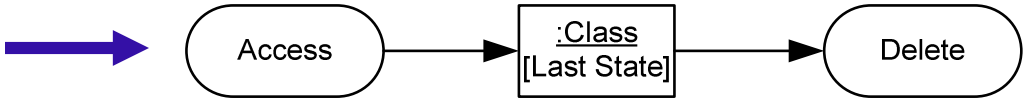
Business Layer	**Business Process modeling**
Service Layer	**Service-Oriented Design**
Component Layer	**Component-Oriented Design**
Class Layer	**Object-Oriented Design**



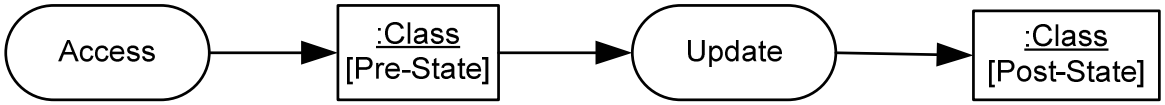
Example of design Hierarchy

Basic Events in OO Analysis

- ✓ Creation
- ✓ Termination



- ✓ Connection
- ✓ Disconnection



Examples of design structures in terms of Object Flow diagrams

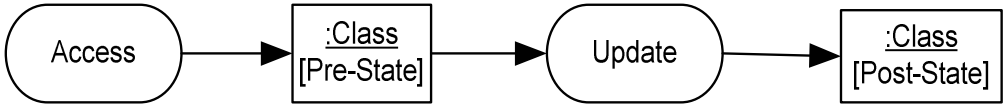
- ✓ Classification
- ✓ Declassification

Source: (Martin and Odell, 1998)

Create, **Access** (Read, Search), **Update** and **Delete** operations are used for definition of design solutions

Compound Events in OO Analysis

- ✓ Reconnection
  - May look the same as connection



- ✓ Reclassification (similar as classification, declassification)

  These events can be defined as follows:
  - Creation of the object in Class2 by copying all attribute values from the object in Class1 to the newly created object in Class2,
  - Disconnect all associations pointing to the old objects and connect them to the new one in Class2,
  - Remove the old object in Class1.

- ✓ **Basic events should be treated as one action in SOAD**

Characteristics of Service Oriented Representations

- ✓ **Conceptual relevance** (only conceptually relevant aspects are included)
- ✓ **Implementation Independent Notation** (technical aspects are not represented)
- ✓ **Analyzability** (whether SOA is Incomplete, Inconsistent, Redundant, Incoherent, Unambiguous)
- ✓ **Traceability** (must be traceable through all levels)
- ✓ **Formality** (well defined models)

Basic Semantic and Pragmatic Dependencies

Dynamic (Semantic) Dependencies

Communication

Transition

Static (Semantic) Dependencies

Inheritance

Association

Composition

Classification

Pragmatic Dependencies

Goal Dependency g

Problem Dependency p

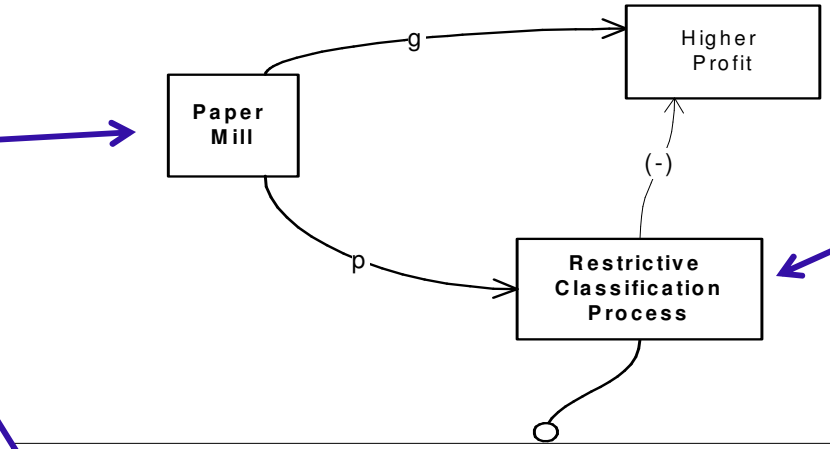
Opportunity Dependency o

Negative Influence (-)

Positive Influence (+)

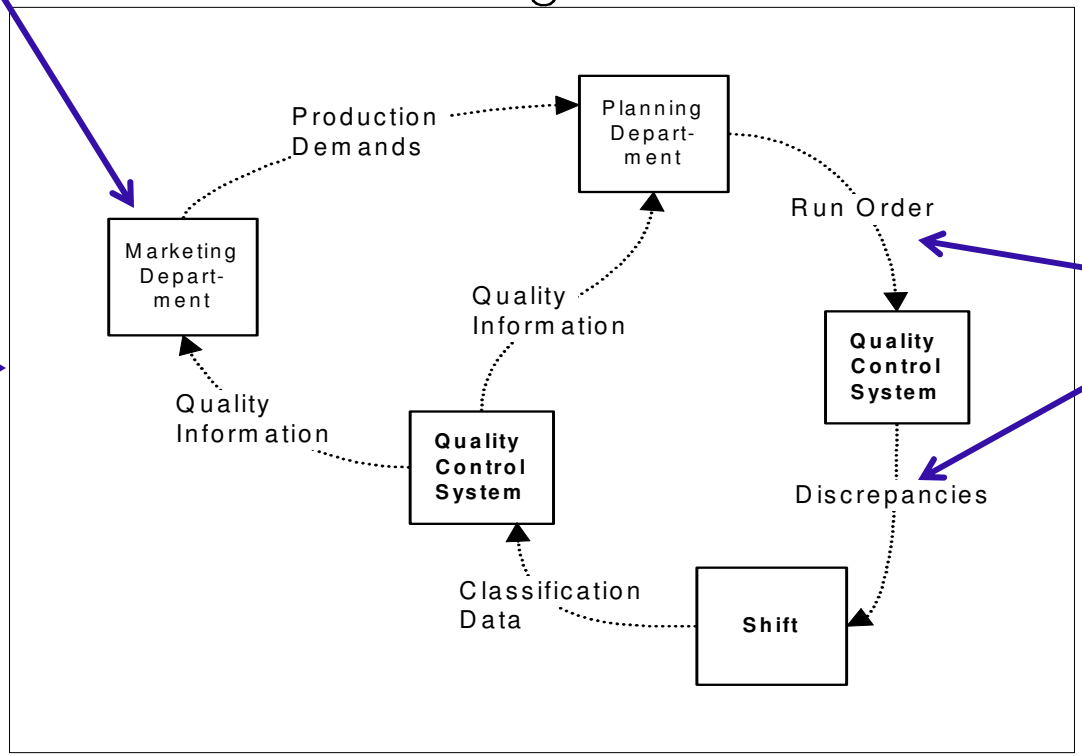
Refinement

Pragmatic Dependencies (examples)



Actor

Description of a situation or process, which Actors may view as Goal, Problem or Opportunity

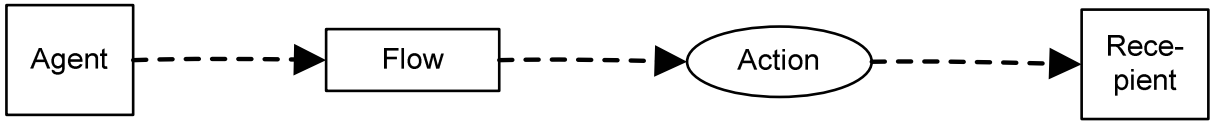


Context of Business Process

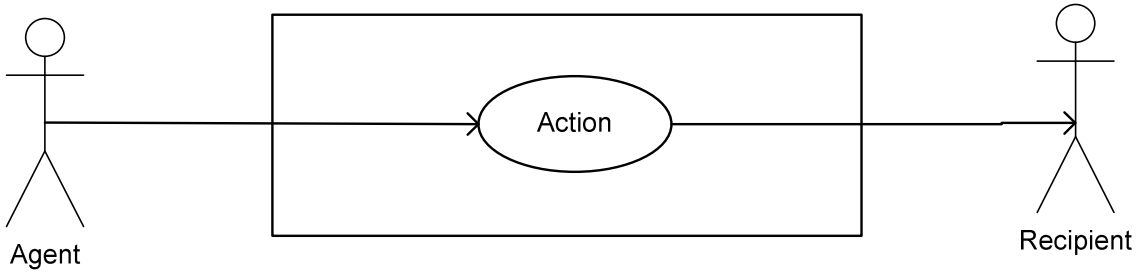
Flow

Basic Events in SOAD

- ✓ Creation,
- ✓ Termination,
- ✓ Reclassification
  (consists of termination and creation events).

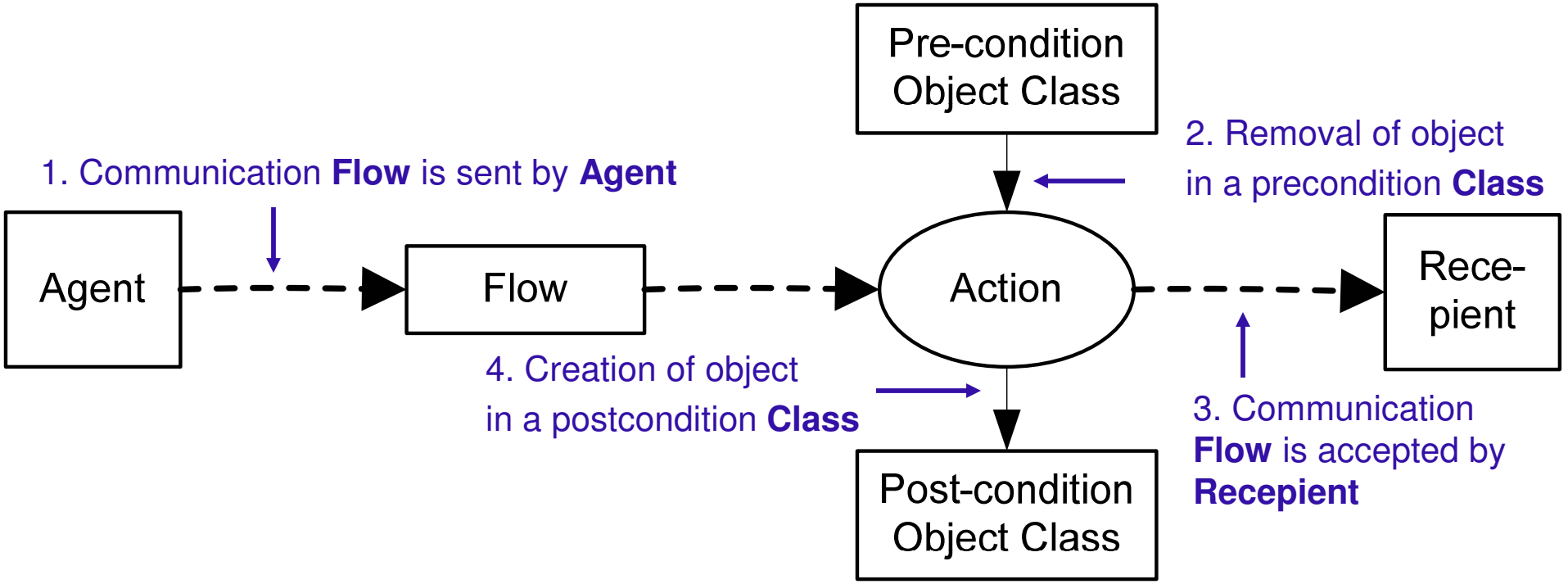


Events are trigerred by communication **Action**.



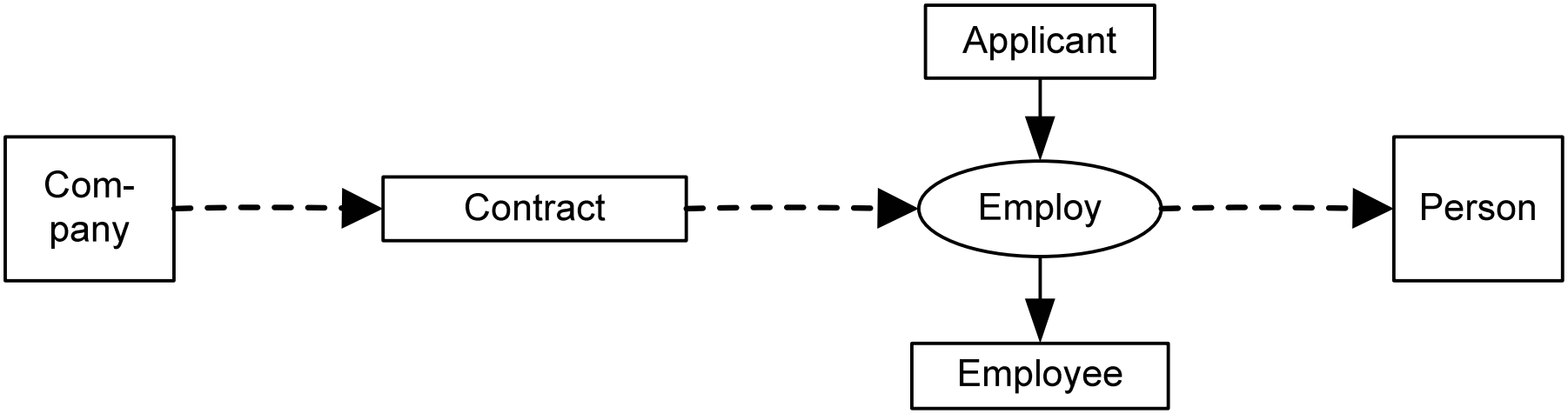
Use Cases (UML) can be viewed as Actions

Reclassification (Compound) Event



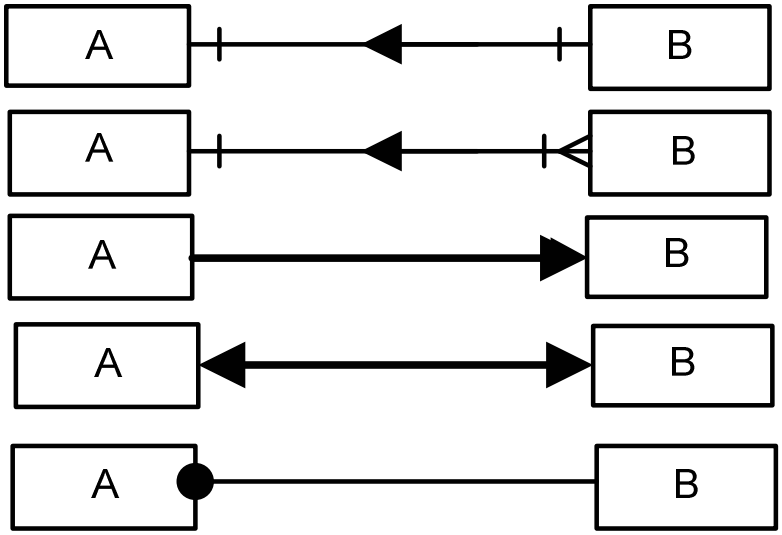
Note: **Flows** can be **Material**, **Information** or **Control**
(control flow is represented just by communication dependency link)

Example of Reclassification



Note: Reclassification of an object from one concept (could be a class) to another without relating it to any other concepts is not useful. Analysts must to identify a noteworthy semantic difference between two concepts (Applicant and Employee). Otherwise, the action is not useful.

Semantic Difference can be specified by using the Static Dependencies



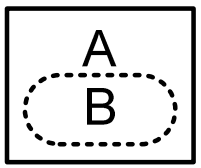
Every A is composition of exactly one instance of B

Every A is composition of one or more instances of B

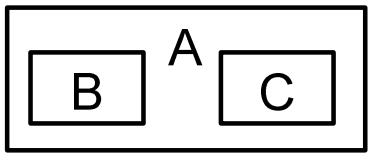
A is specialisation of B

A and B are synonyms

A is an instance of B

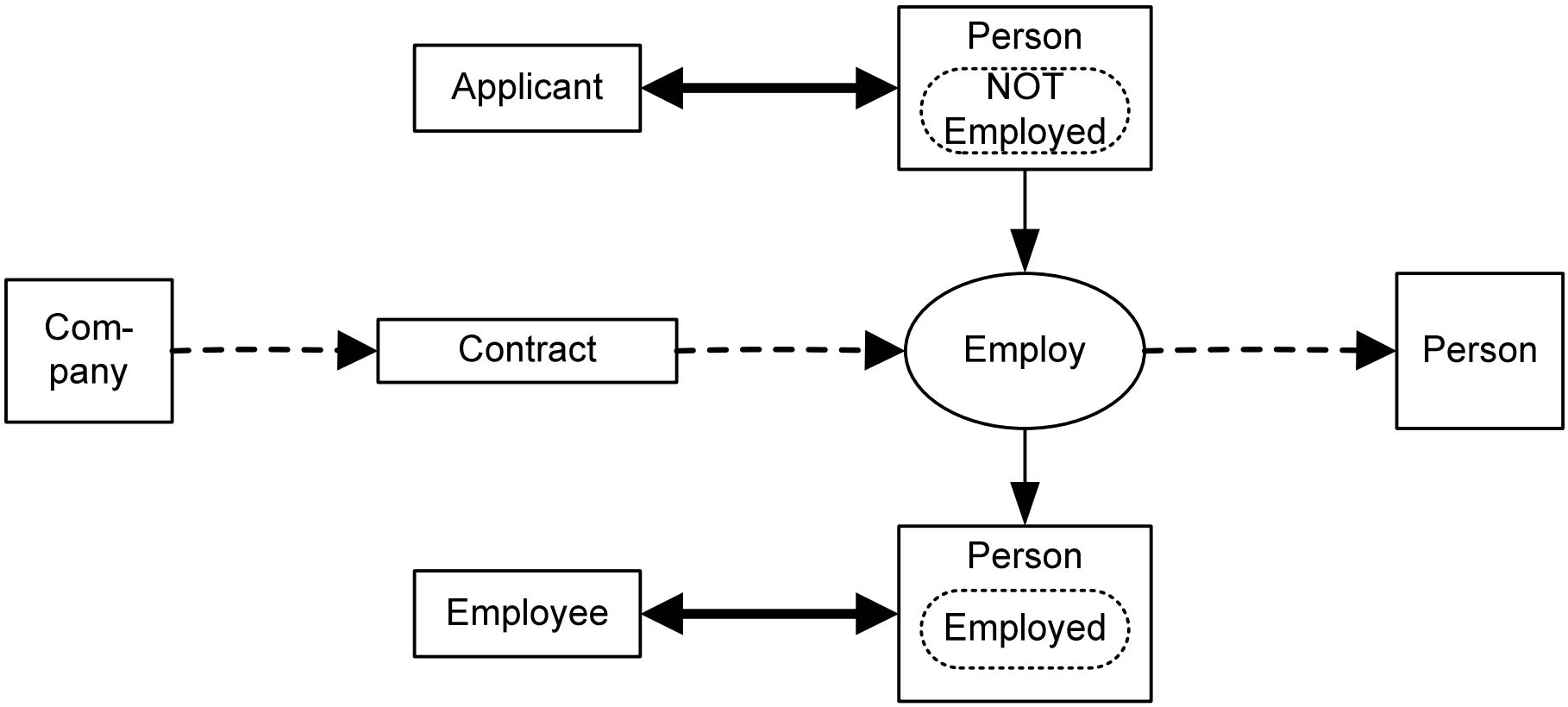


B is a state of A

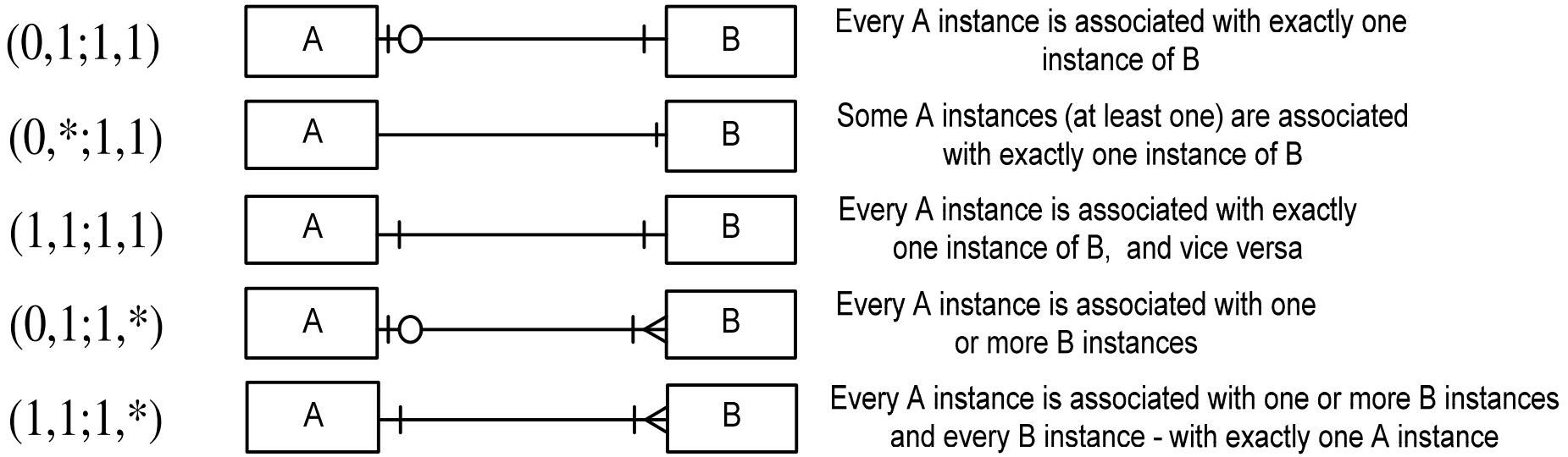


B and C is an exclusive specialisation of A

Example of Reclassification by identifying two states of Person

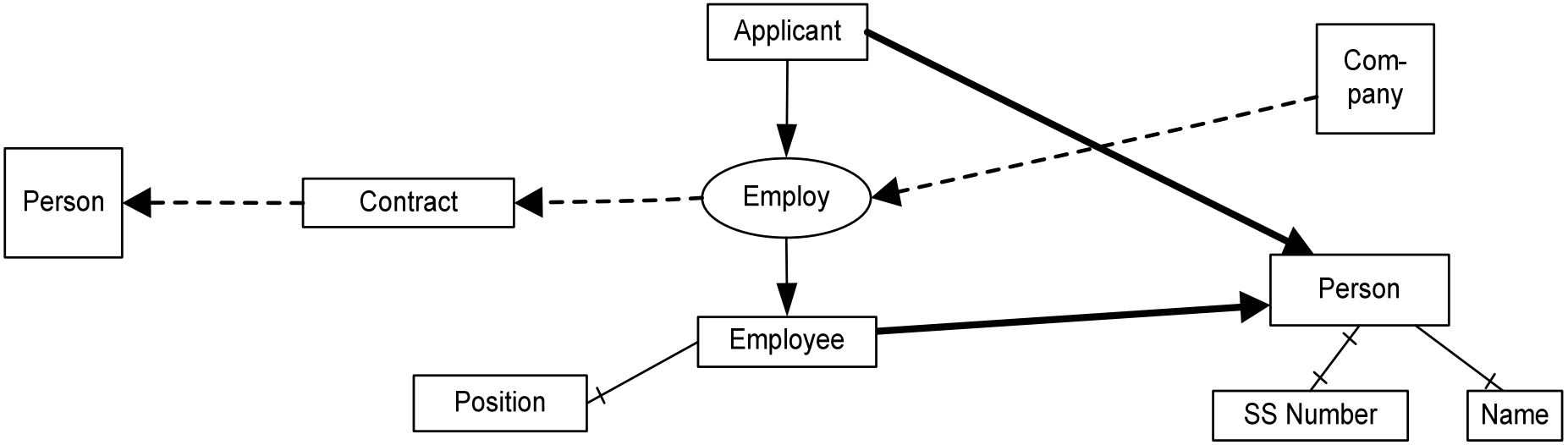


Basic Associations can be used to define the semantic difference



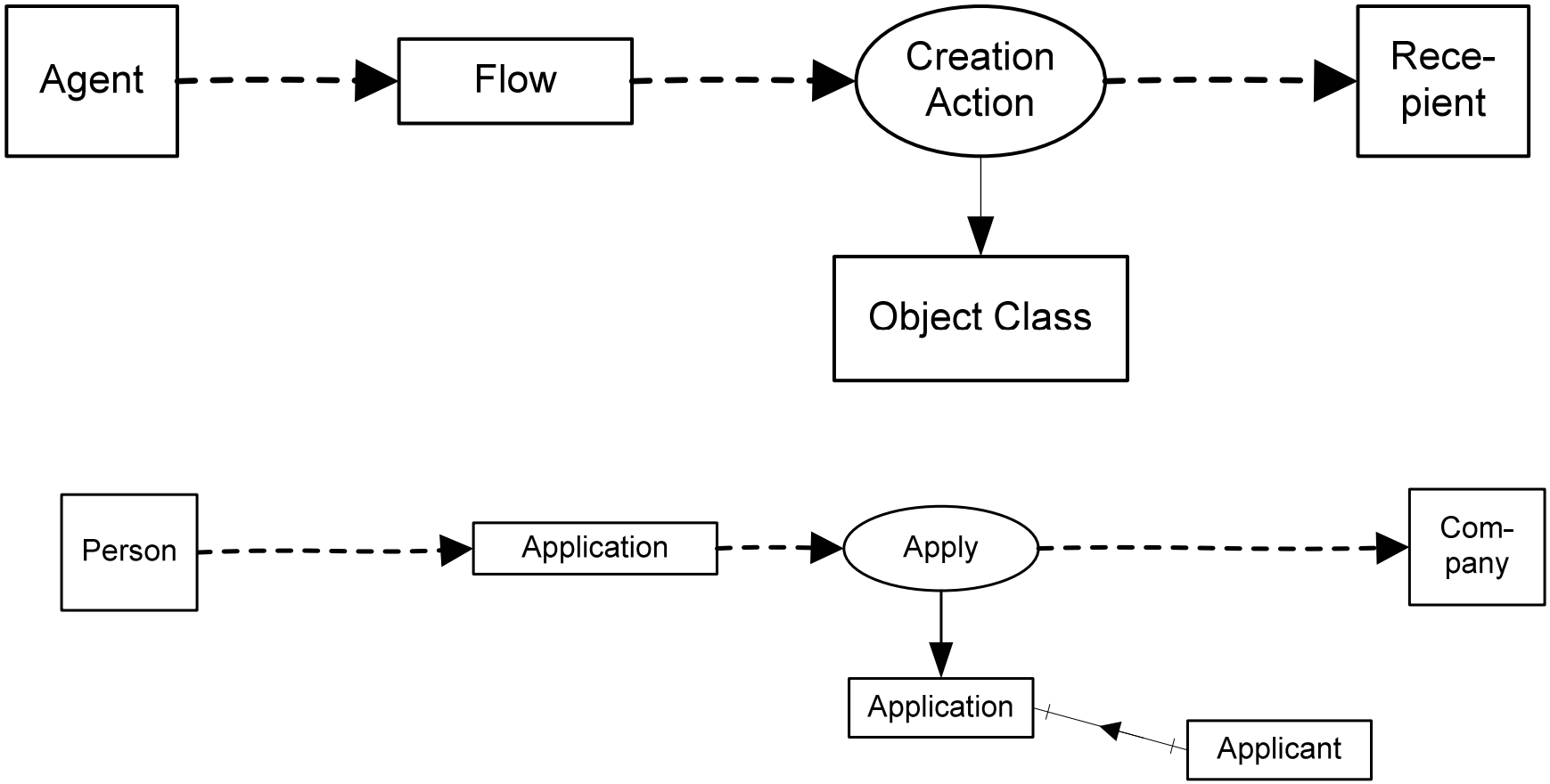
Not Basic Associations (the remaining 5 types of static relations) are not recommended for the final SOAD phase. They give rise to the semantic holes. Since other association types are used in OO analysis and design, they are not forbidden in early SOAD phases.

Semantic Difference is defined by new association for Employee

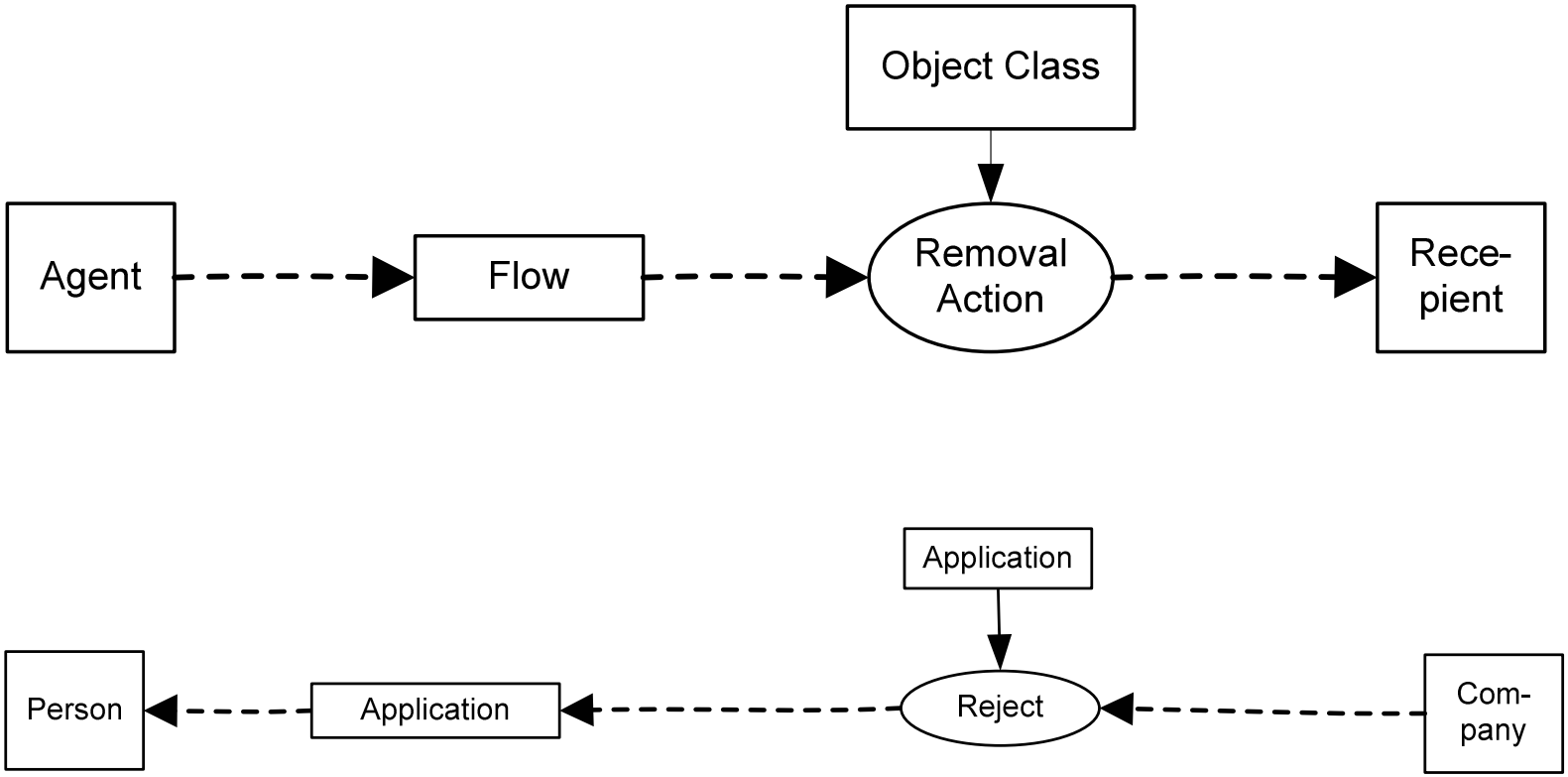


Termination of Applicant and creation of Employee is performed by the same action of Employ. Postcondition object requires creation of a Position object (it should be sent by Company as a part of flow that is entitled as Contract). Precondition object of Applicant must have two associations (see SS Number and Name). They are preserved after the action is executed, because of the inheritance dependency from Employee to Person.

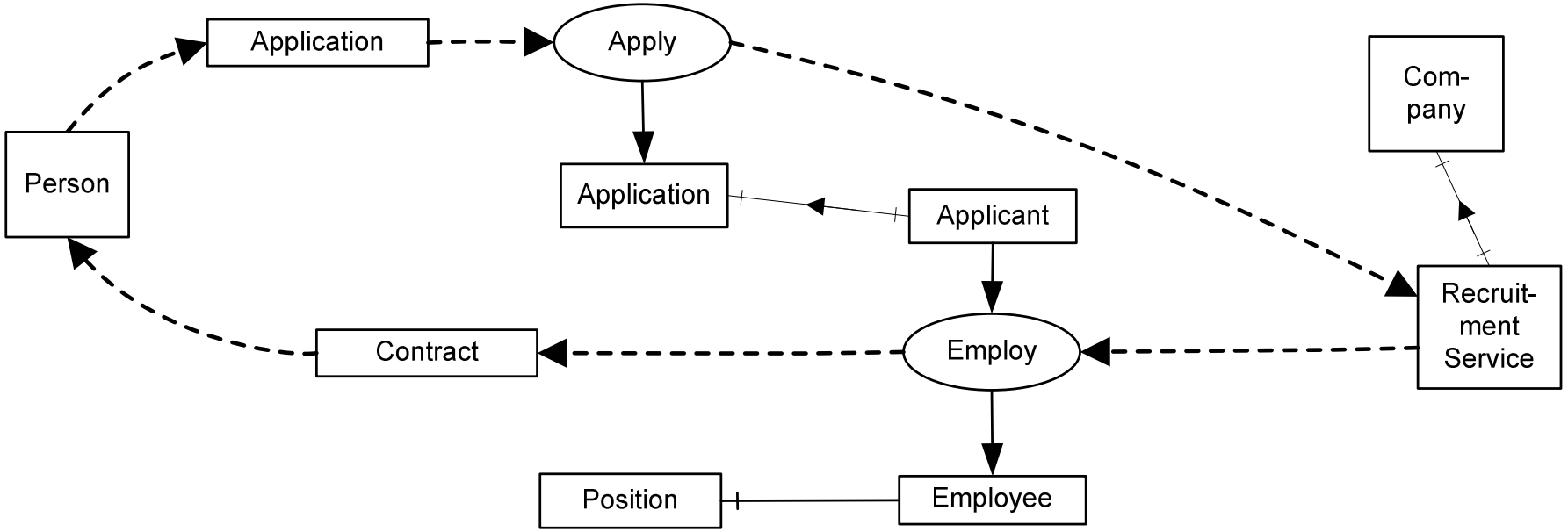
Creation Event Notation



Termination Event Notation

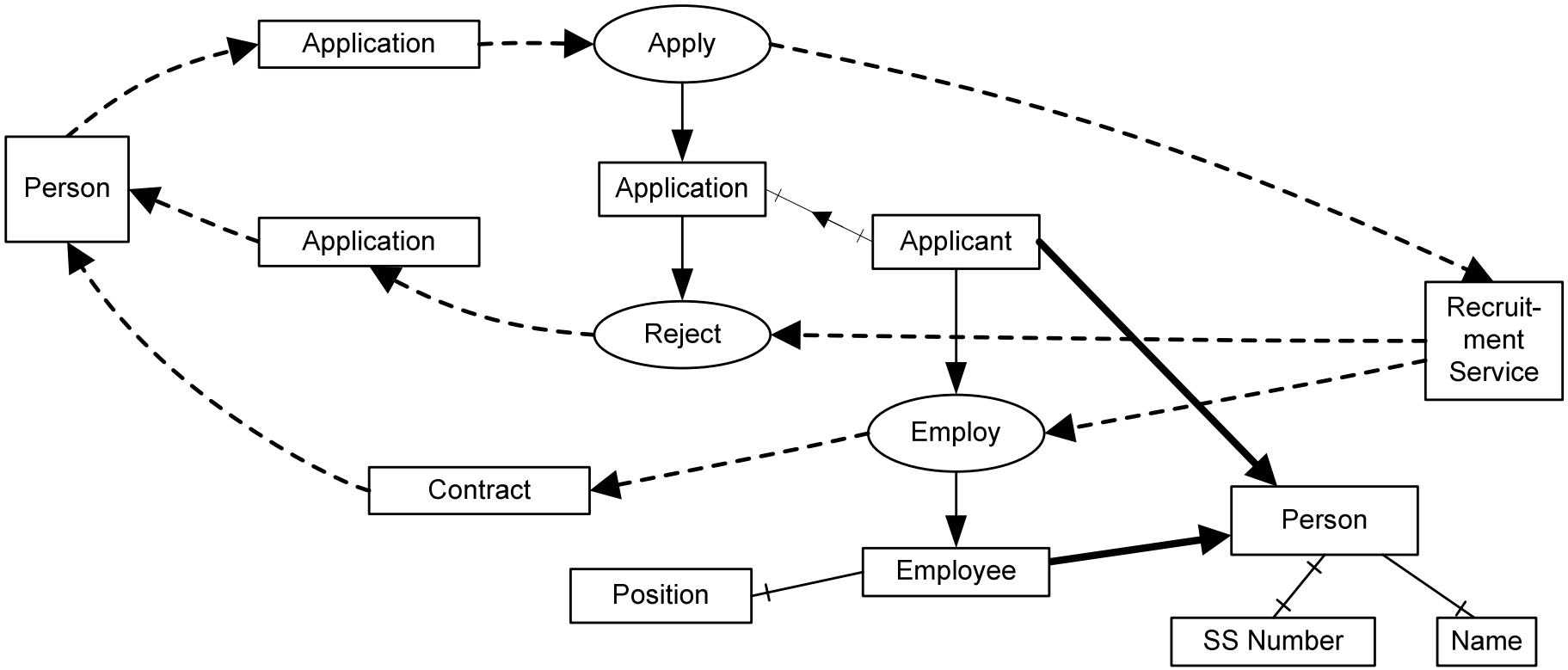


Communication Loops

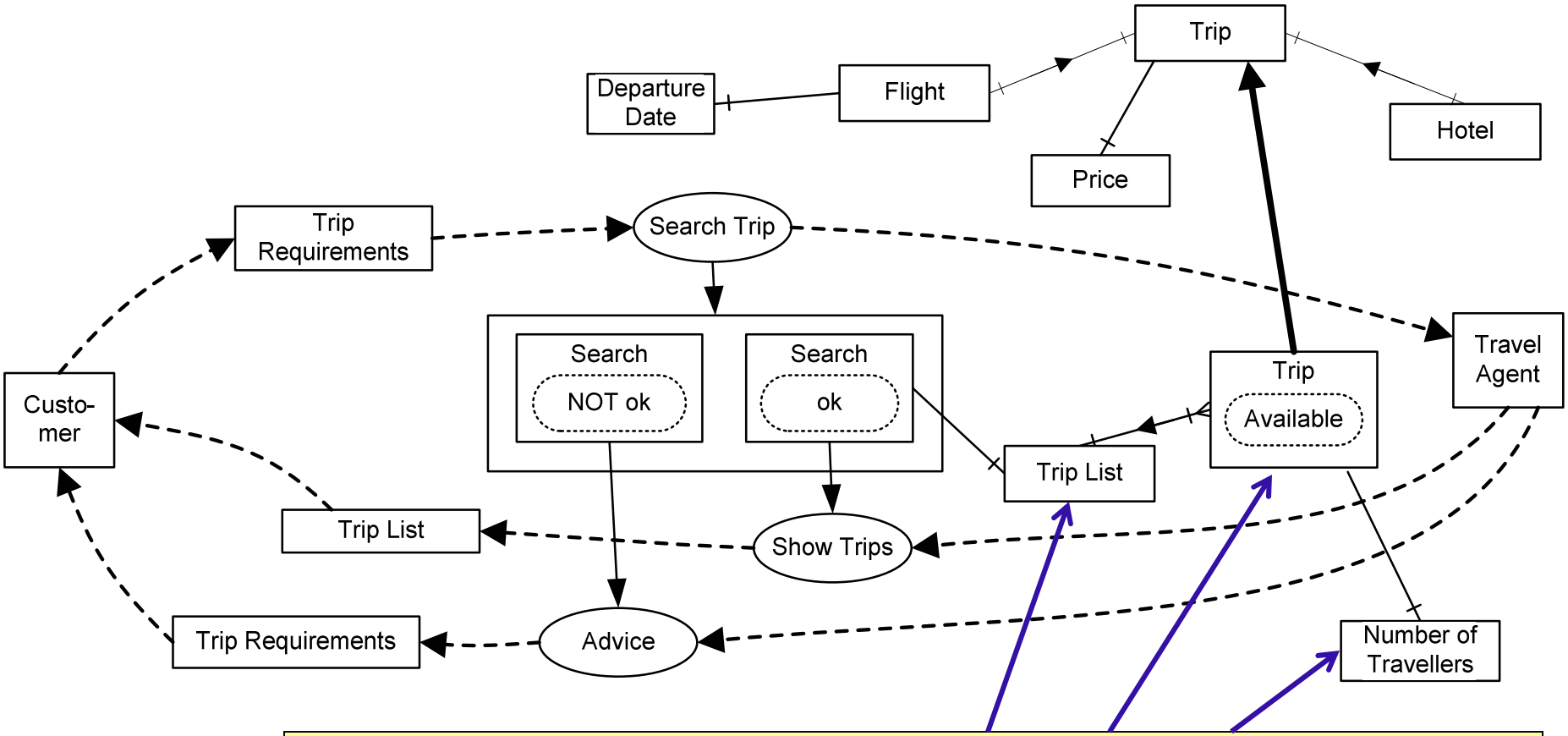


Company may be decomposed into services as its parts with clearly defined communication protocols for delegated functions.

Communication Loops with Alternative Actions



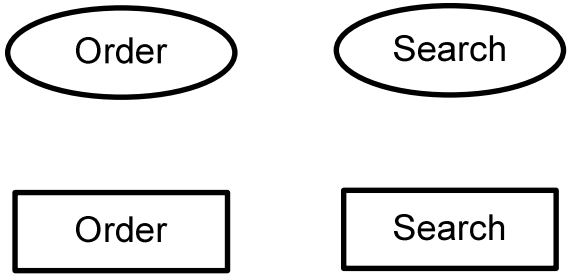
Communication Loops as Queries



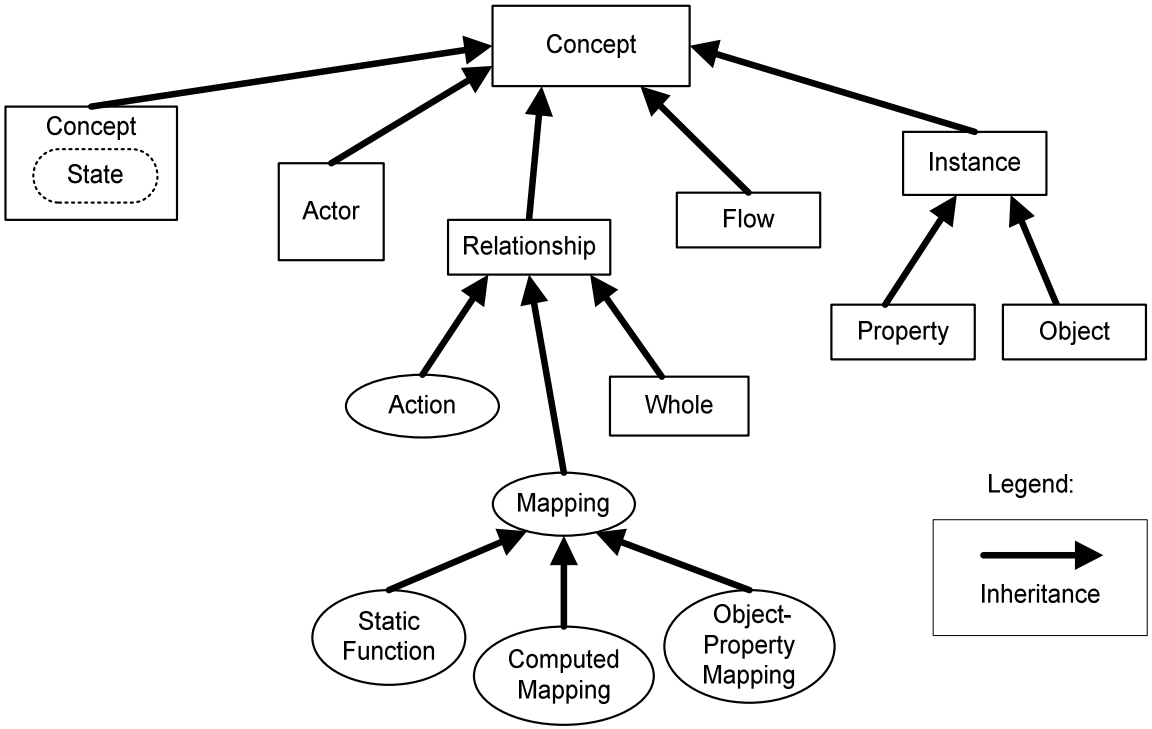
Not removed objects even when Search object is terminated.
They are necessary in the next communication loop.

Relative Interpretation of Concepts

Verbs (to Order, to Search)

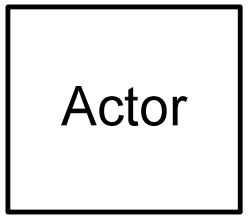


Nouns (an Order, a Search)



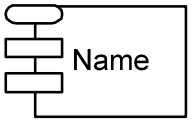
The same concept
can be interpreted in a number of ways (see the classification hierarchy).
Semantic dependecy types have no given names.
They define how a concept is classified.

Component Level: Examples of Business and Technical Components



Actor can be instantiated as Business or Technical Components

Organisational component: Division, Department



Software component, Application

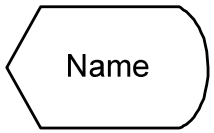
Organisational component: Role, Job Position

Database, File

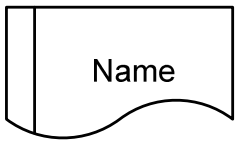
Hardware component: Computer

Hardware component: Server

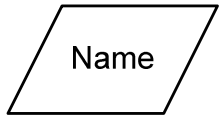
Interfaces have to be defined between Technical and Business components



Screen Layout

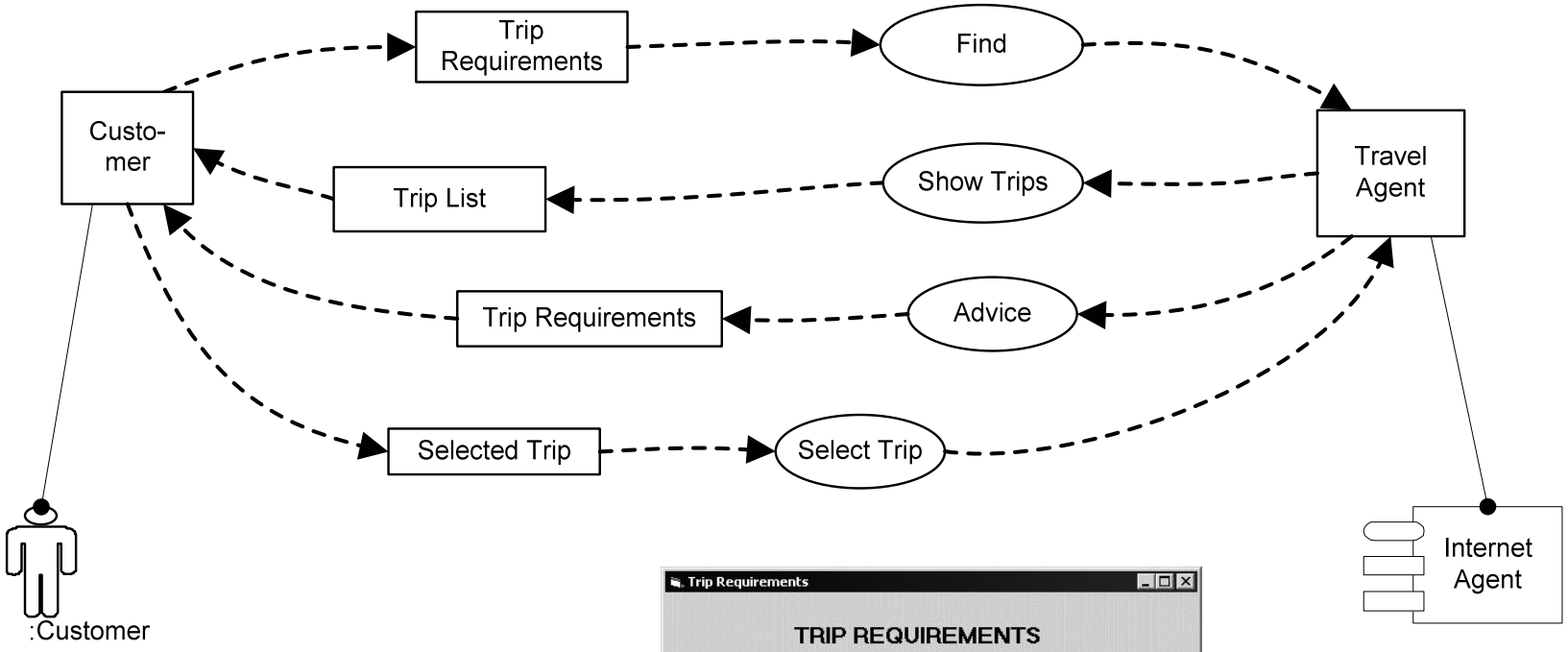


Printout Layout

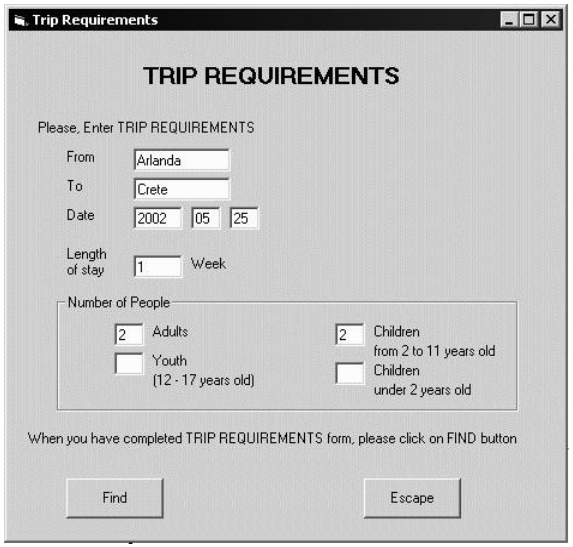


Interface between Technical Components: Message Layout

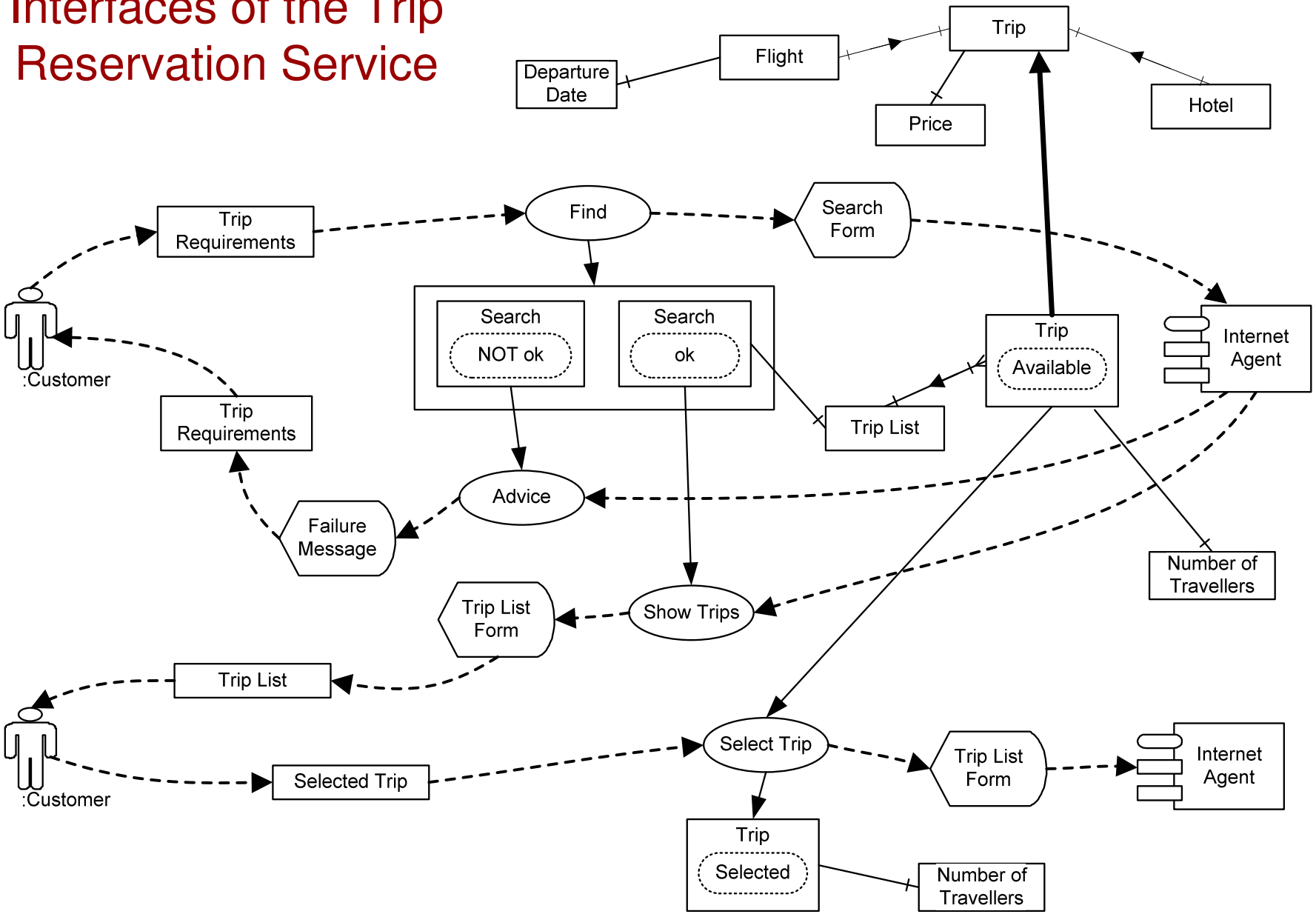
Components and Communication Flows of the Trip Reservation Service



Example of **Search Form** (see component diagram) Layout filled with **Trip Requirements** data



Components and Interfaces of the Trip Reservation Service



Qualities of SOAD

Some discussion on qualities can be found in (Zimmermann et al.,2004)

- ✓ Service orientation facilitate reuse and ease of reconfiguration through loose coupling.
- ✓ Dependencies between service components are minimised and explicitly defined.
- ✓ Independent services are must be stateless.
- ✓ SOA should be understandable for domain experts without deep technical expertise.
- ✓ SOAD is defined by using interaction (communication) patterns among components (business and technical: hardware, software). It is based on the same systematic way of thinking.

SOAD facilitates solution of difficult problems:

- ✓ **Ambiguity problem**. Business processes and services are spanning across organizational and technical system boundaries. Interpretation of the same concept in various contexts may be different. SOA insists on clear definitions and motivation of every concept.
- ✓ **Integrity problem**. SOA provide a comprehensible foundation for interplay between various syntactic (implementation dependent), semantic and pragmatic dependencies.
- ✓ **Consistency problem**. The same reality can be perceived in a number of ways and therefore it can be represented on various levels of abstraction. Inconsistencies can be identified by using SOA (inference rules).
- ✓ **Completeness problem**. Semantic and pragmatic dependencies suggest a new way for the semantic incompleteness and overspecification control.
- ✓ **Change evolution problem**. Every new solution can be considered to be a symptom of a new problem. Business architecture and SOA evolution problems can be tackled in a systematic way.

Summary

- ✓ Graphical Models that are used for SOAD follow the basic conceptualization principle. Since models are **implementation agnostic,** their complexity is lower and, therefore, comprehensibility by humans is higher
- ✓ Represent interoperation of organisational and technical components in **one model** (on different levels of abstraction). Separate perspectives (many models) in traditional architectures are difficult to maintain
- ✓ Integrated representations are necessary to reach **consensus** and to develop **holistic** understanding for planning of orderly transitional processes
- ✓ Services can be represented as autonomous descriptions that are defined and **published on the Internet** by using machine readable formats
- ✓ Significant **competitive advantages**, since models become a subject for search, composition, evolution and integration