Sistemas Multi-agentes
Apresentação I

Apresentação: Karen da Silva Figueiredo

Instituto de computação

Introdução

- Modelagem e validação de Normas
- Objetivo da apresentação: apresentar uma linguagem de modelagem e uma metodologia de SMA que aborde normas
- Pretensão: aprimorar a linguagem em desenvolvimento NormML

Artigos

- The **Agent-Object-Relationship** Metamodel: Towards a Unified View of State and Behavior, Gerd Wagner, 2003.
- Modelling Security and Trust with **Secure Tropos**, Paolo Giorgini, Haralambos Mouratidis, Nicola Zannone, 2006

The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior

Autor: Gerd Wagner

Ano: 2003

Apresentação: Karen da Silva Figueiredo

Sistemas Multi-agentes – Apresentação I

Instituto de computação

Introdução

- Distinção entre entidades “passivas” e “ativas”
- Trabalho inspirado em Agent-Oriented Programming [Sho93]
  - O estado de um agente possui uma estrutura mental que consiste em componentes como crenças e compromissos
  - Mensagens: atos de fala, utilizam uma linguagem de comunicação independente

Introdução

- Extensão da UML
- Dois tipos de modelos AOR (Agent-Object-Relationship):
  - *External:* adota o ponto de vista de um observador externo
  - *Internal:* adota a perspectiva de um agente em particular

- Sugere um sistema de desenvolvimento: External models > Internal Models > Implementation models

Agentes e Sistemas de Informação

- Grande parte das tecnologias de sistemas de informação são baseadas em RDB e ORDB

- Estas tecnologias não dão suporte ao conceito de agentes e não fazem distinção de agentes de objetos

Princípios de ER

1. Um SI tem que representar a informação a cerca das **entidades** que fazem parte do **universo do domínio** da aplicação
2. Entidades tem **propriedades** e participam de **relacionamentos** com outras entidades
3. Entidades são classificadas por **tipos**
4. Cada entidade define uma lista de **atributos** que representam propriedades relevantes

Princípios de ER

5. Os valores de todos os atributos juntos compõem o **estado** da entidade
6. Os relacionamentos entre entidades são classificados por **tipos de relacionamentos**
7. Existem 2 tipos de relacionamento entre tipos de entidades que são independentes da aplicação do domínio: **generalização** (isSubclassOf) e **agregação** (isPartOf).

Agentes

1. Uma entidade é um agente de software se e somente se ele se **comunicar corretamente** em uma linguagem de comunicação de agentes [GK94]
2. Um agente é uma entidade cujo **estado consiste em componentes mentais** como crenças, capacidades, escolhas e compromissos [Sho93]

Agentes

- Somente agentes podem perceber eventos, executar ações, comunicar e fazer compromissos

- Objetos não se “comunicam” e “interagem” através de “mensagens”

Agentes como SI

- Características dos agentes [HR95]:
  - Percepção das condições dinâmicas do ambiente
  - Ação para afetar as condições do ambiente
  - Raciocínio para interpretar percepções, resolver problemas e determinar ações

Agentes como SI

- Características dos agentes [HR95]:
  - Percepção das condições dinâmicas do ambiente
  - Ação para afetar as condições do ambiente
  - Raciocínio para interpretar percepções, resolver problemas e determinar ações

Agentes como SI

- Um SI pode ser explicitamente designado como um agente:
  - Tratando suas informações como crenças ou conhecimento
  - Adicionando componentes mentais como percepção, memória e compromissos
  - Suportanto a comunicação agente-para-agente através de uma linguagem de comunicação de agentes

Princípios da Modelagem AOR

8. Diferentes entidades podem pertencer a diferentes **categorias**. Existem **agentes, eventos, ações, compromissos, reivindicações e objetos.**
9. Ações e eventos diferem-se entre **comunicativos** e **não-comunicativos**.
10. Ações criam eventos (chamados *action events*), mas nem todos os eventos são criados por ações.
11. Alguns dos conceitos da modelagem dependem da **perspectiva** escolhida.

Princípios da Modelagem AOR

12. Em uma perspectiva interna de um agente, um **compromisso** se refere a uma ação específica realizada em determinado tempo, enquanto uma **reivindicação** se refere a um *action event* que deveria acontecer em determinado tempo.
13. Comunicação é vista como uma transmissão de **mensagem** assíncrona. As expressões "**receber uma mensagem**" e "**enviar uma mensagem**" são sinônimas de **perceber um evento de comunicação** e **realizar um ato de comunicação.**

Princípios da Modelagem AOR

14. Existem 6 tipos de **relacionamentos** que agentes participam: **perceber** eventos de ambiente, **receber** e **enviar** mensagens, **fazer** ações não-comunicativas, ***hasCommitment*** para realizar alguma ação em determinado tempo, e ***hasClaim*** que algum event action irá acontecer em determinado tempo.
15. Existem **3** tipos de agentes: **artificial, biológico** e **institucionais.**
16. Um **agente institucional** consiste em um certo número de agentes internos (biológico, artificial ou institucional) que atuam ou pertencem a ele.

Princípios da Modelagem AOR

17. No contexto de um agente institucional cada agente interno tem certos **direitos** e **deveres**.

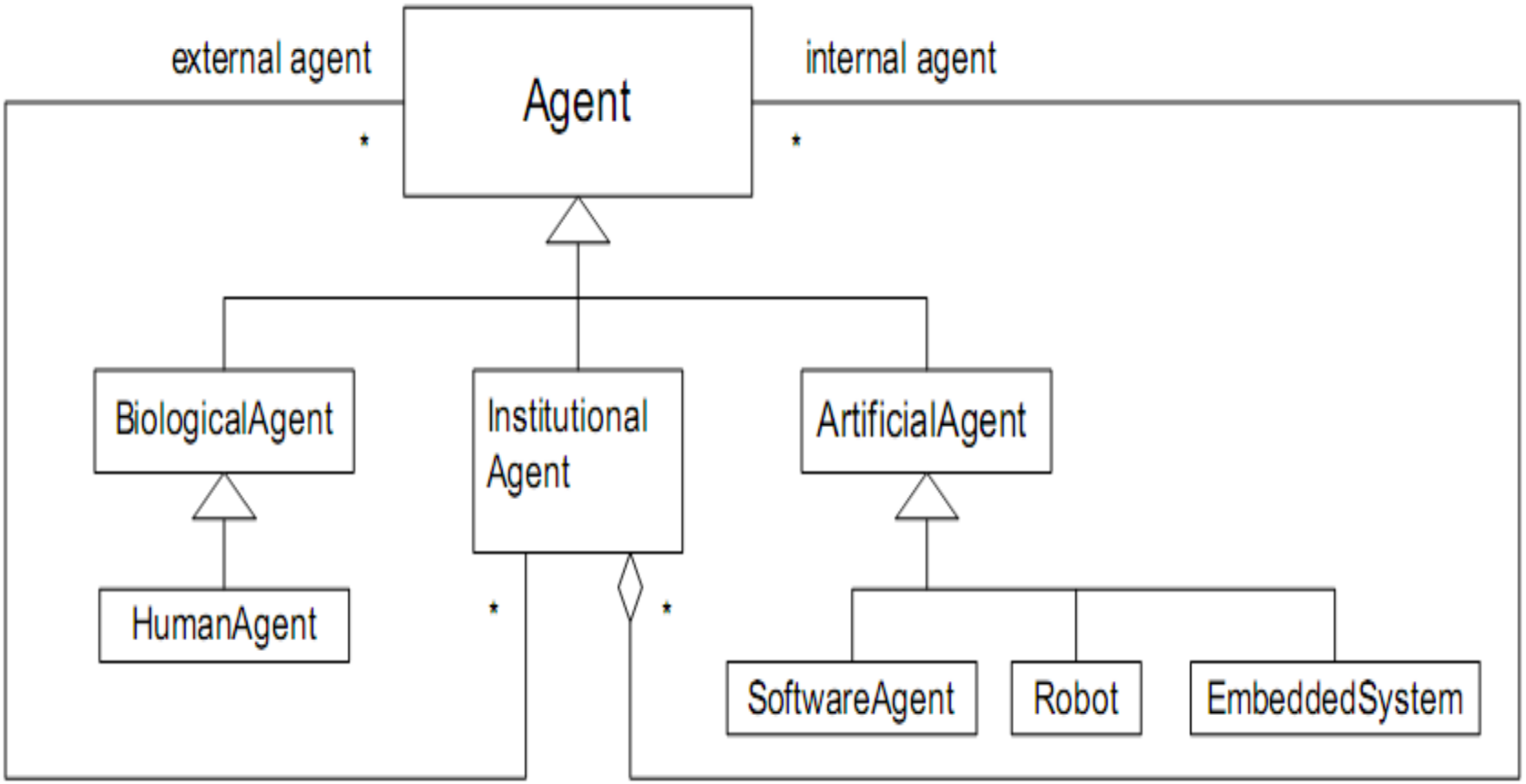
18. Existem **3** tipos de deveres: **cumprir compromisso**s, **monitorar reivindicações** e **reagir a determinados eventos** dentro da organização.

19. Um direito se refere a um determinado tipo de ação que um agente interno é **permitido** a realizar dentro da organização.

A Modelagem AOR

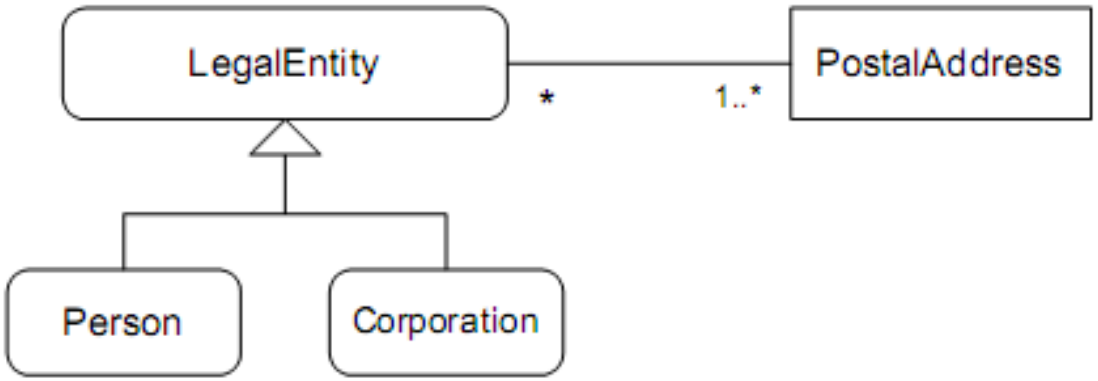
- Desenvolvida para suportar a modelagem de SI orientados a agentes de alto nível
- Um **objeto** é representado da mesma forma em que é representado em ER e **pode participar dos mesmos relacionamentos (associação, generalização, agregação)** com outros objetos e agentes (exceto por generalização)
- Utiliza as invariantes de **multiplicidade de UML**
- Utiliza a mesma representação gráfica de sublinhado para **instâncias de entidades** de UML

Tipos de Agentes



Tipos de Agentes

- Um agente institucional consiste em um número de agentes internos que percebem eventos e executam ações dentro dele através da realização de determinado **papel**
- Existem agentes de tipos **genéricos**



Compromissos e Reivindicações

- Compromisso: ***(a1, a2, α(c1, . . . , cn), TimeSpec)***

- Todo compromisso é espelhado em uma **reivindicação** no sentido contrário.

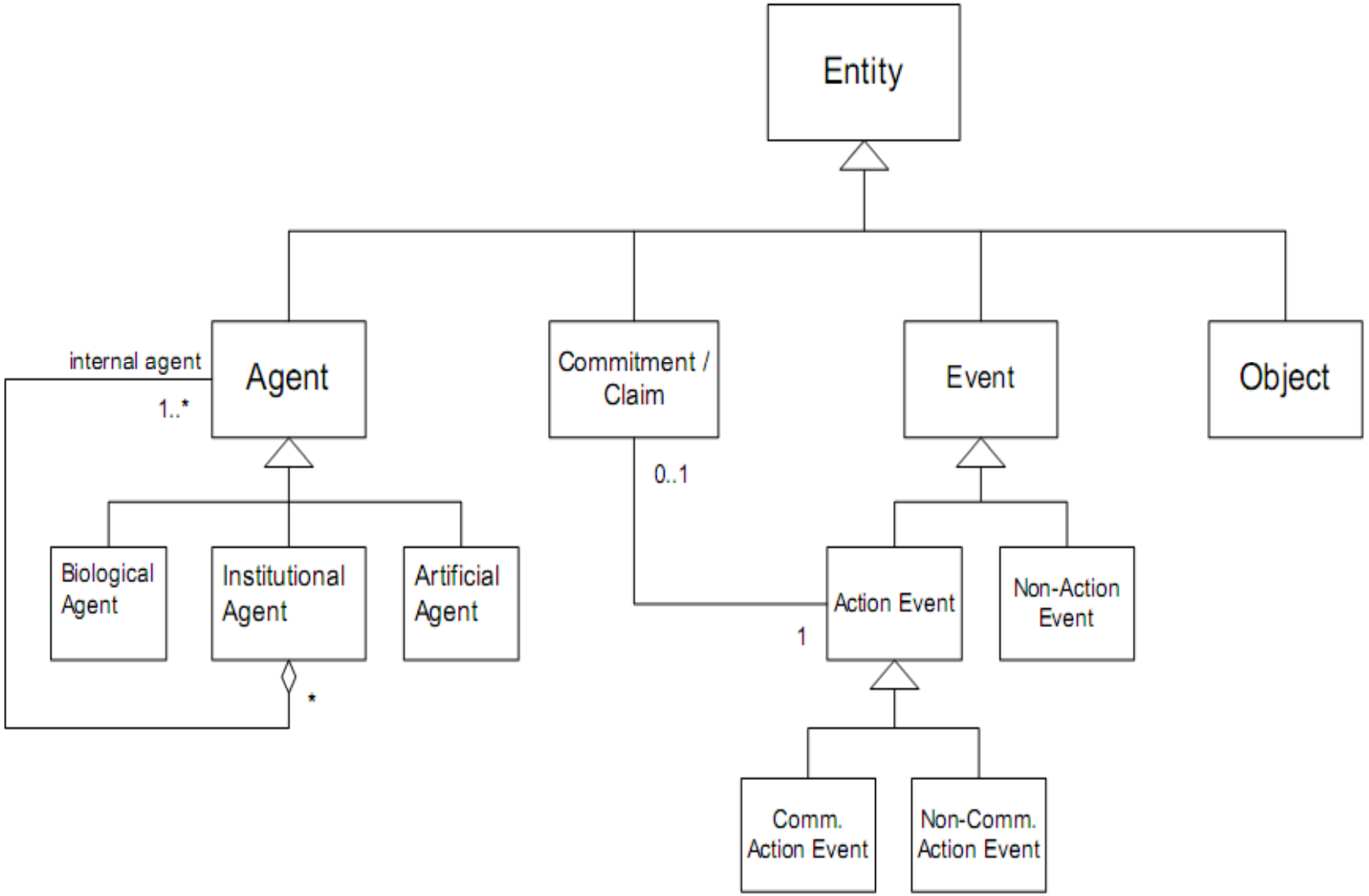
- **Operações:** criação de um compromisso/reivindicação, cancelamento de um compromisso, liberação de um compromisso, delegação de um compromisso, atribuição de uma reivindicação e cumprimento de um compromisso.

Modelos Externos

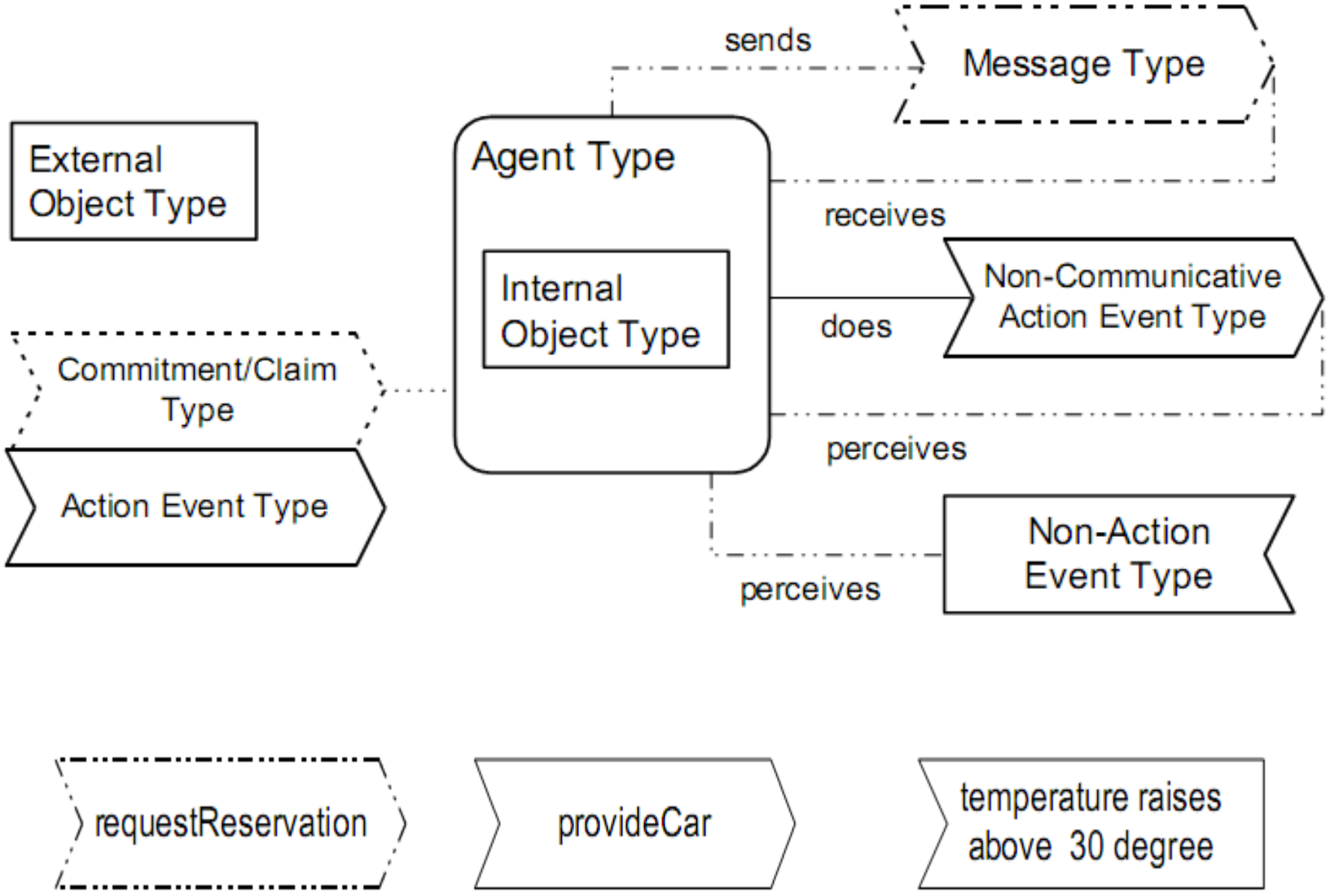
Visão do observador externo

- Agentes
- *Action events* comunicativos e não comunicativos
- *Non-action events*
- Compromissos e reivindicações
- Objetos comuns
- Relacionamentos
- Associações simples

Modelos Externos



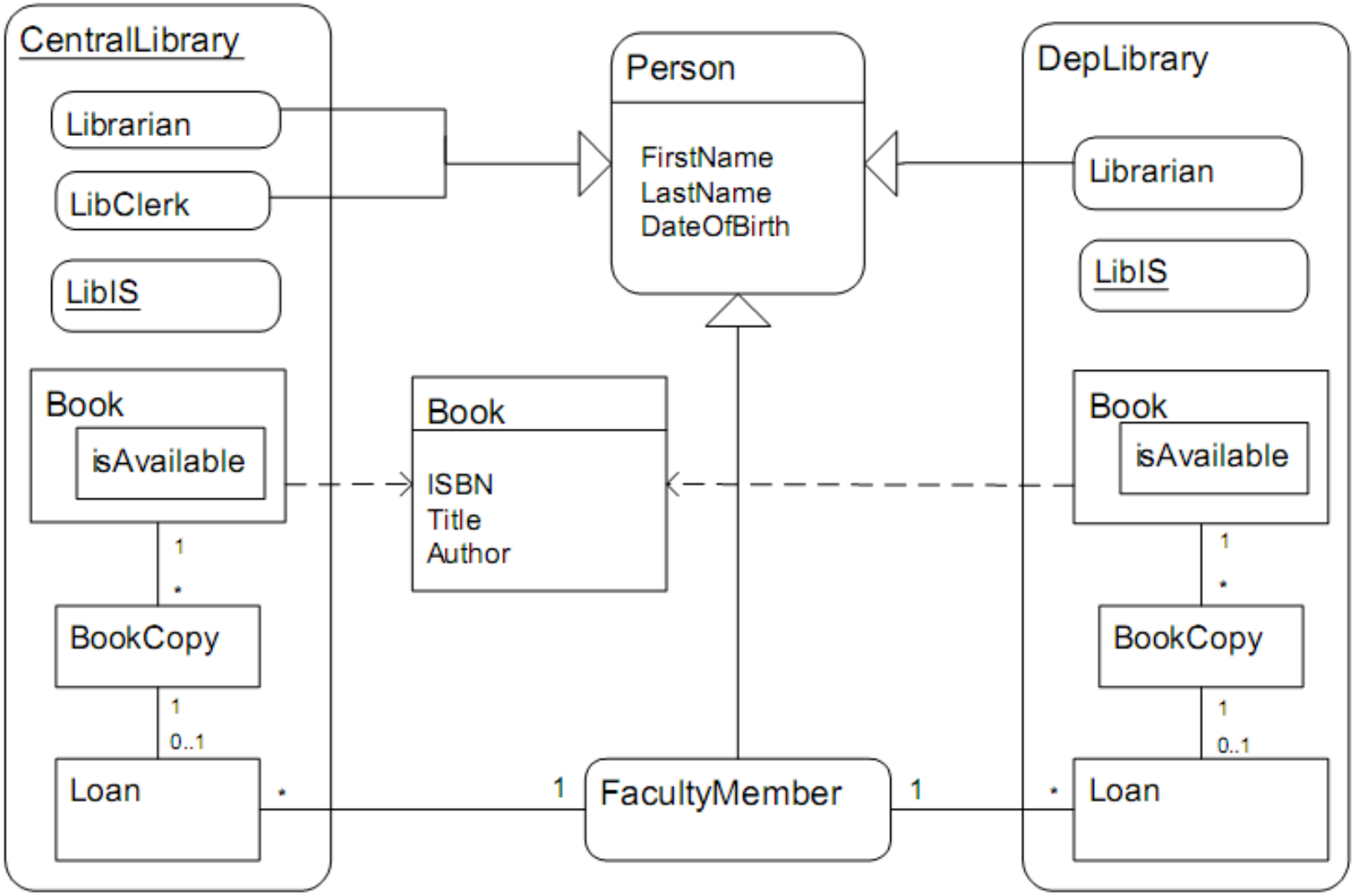
Modelos Externos



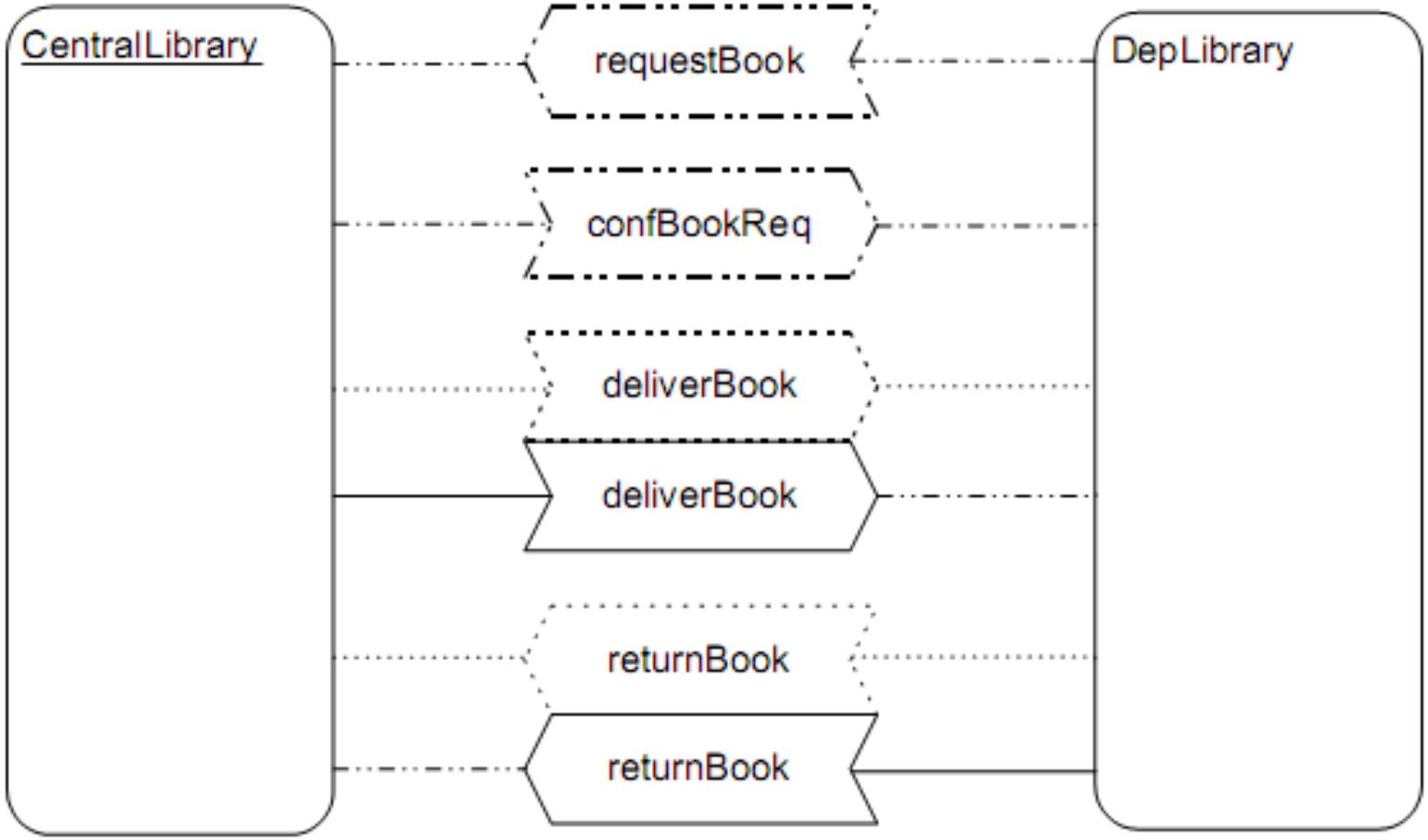
Modelos Externos

- Diagramas de Agentes
- Diagramas de quadros de interação
- Diagramas de sequência de interação
- Diagramas de padrões de interação

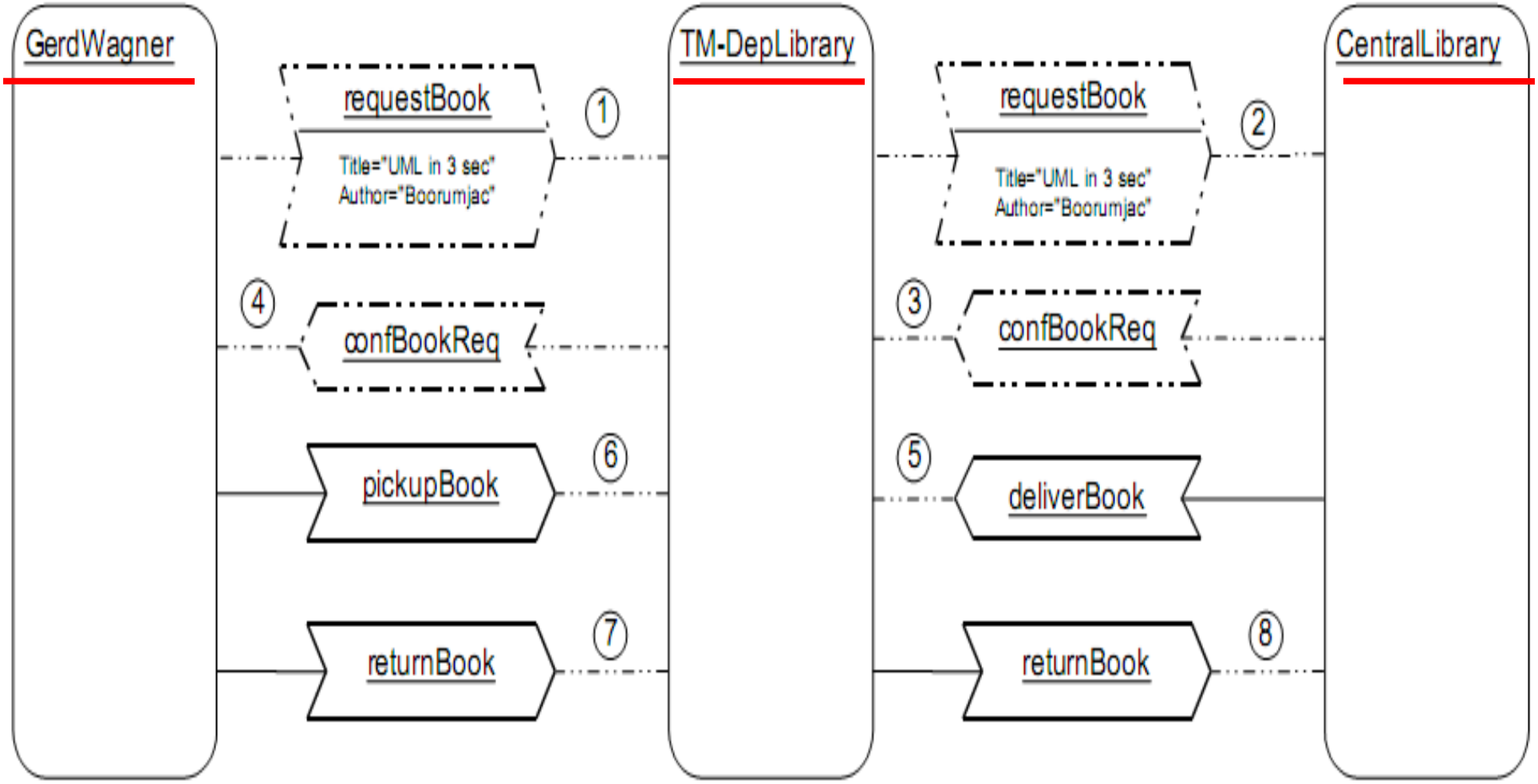
Diagramas de agentes



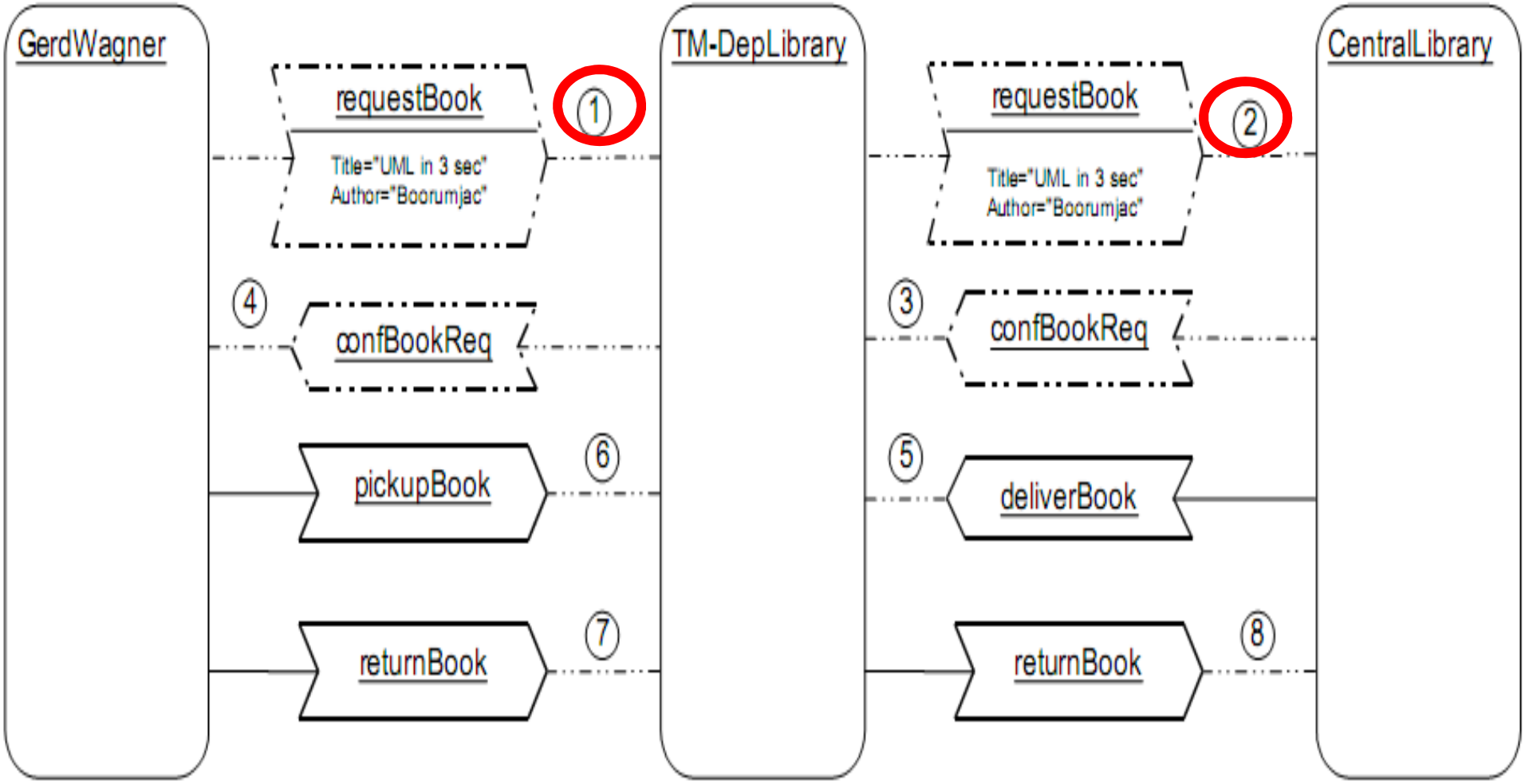
Diagramas de quadros de interação



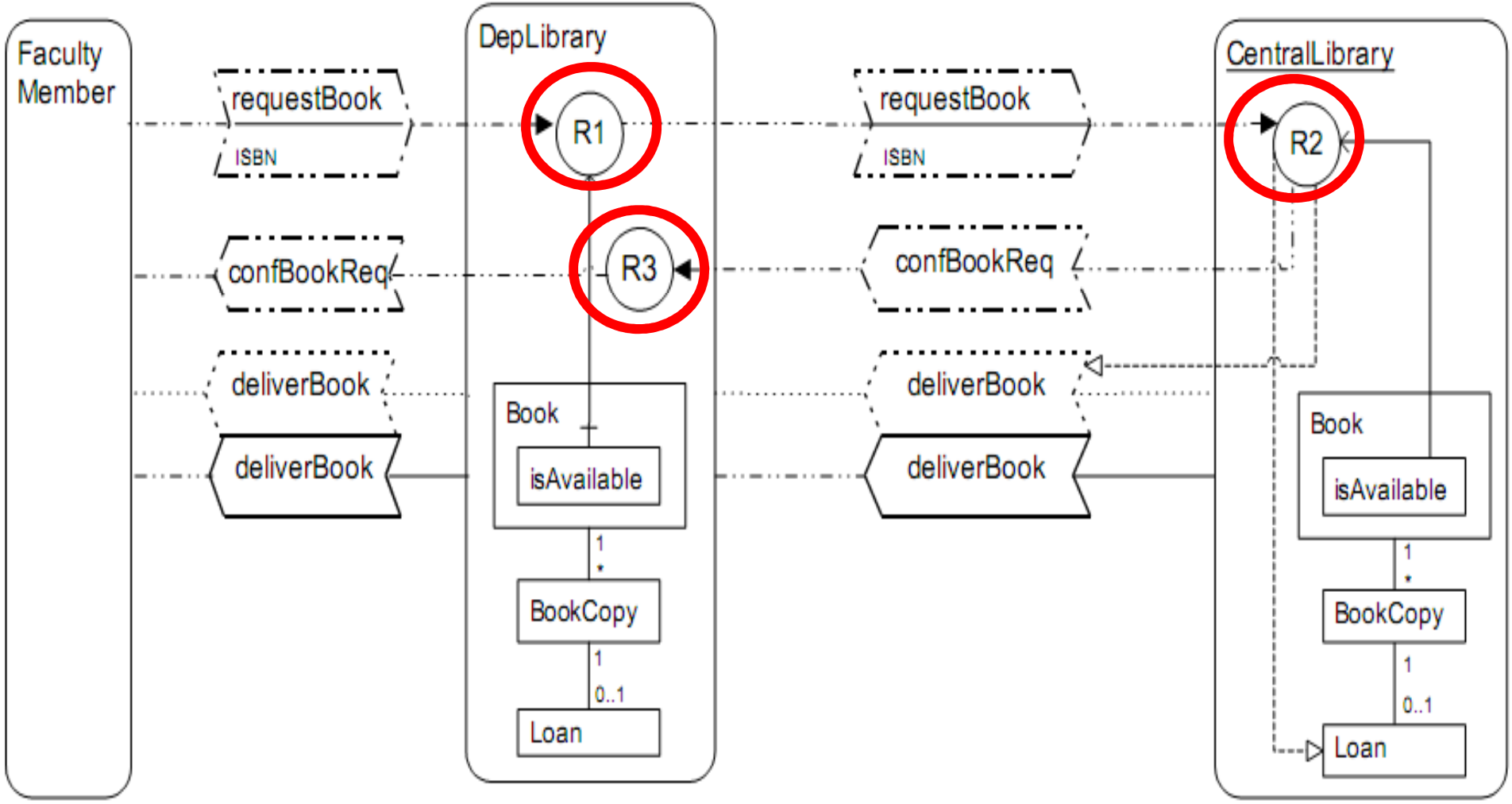
Diagramas de sequência de interação



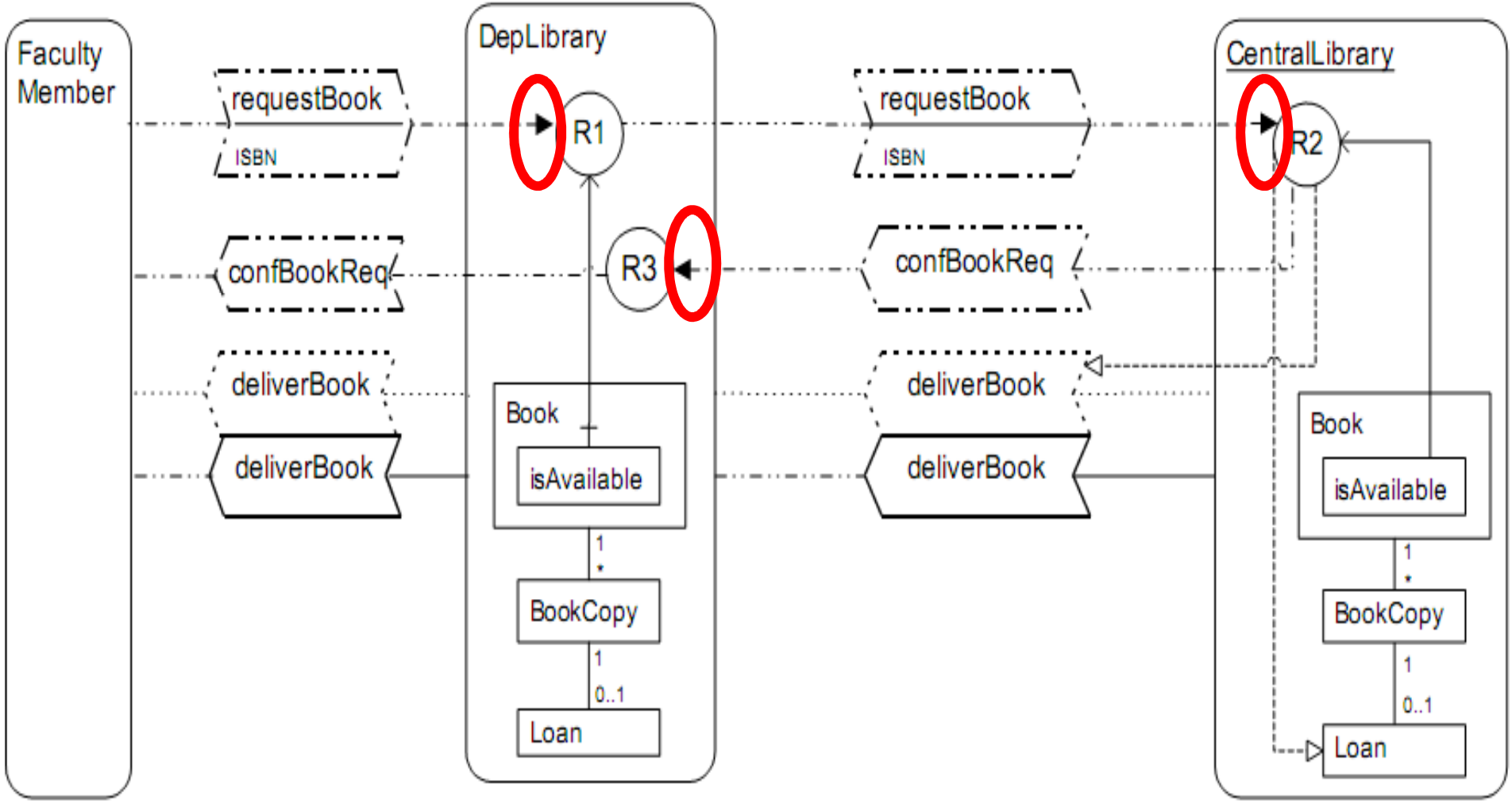
Diagramas de sequência de interação



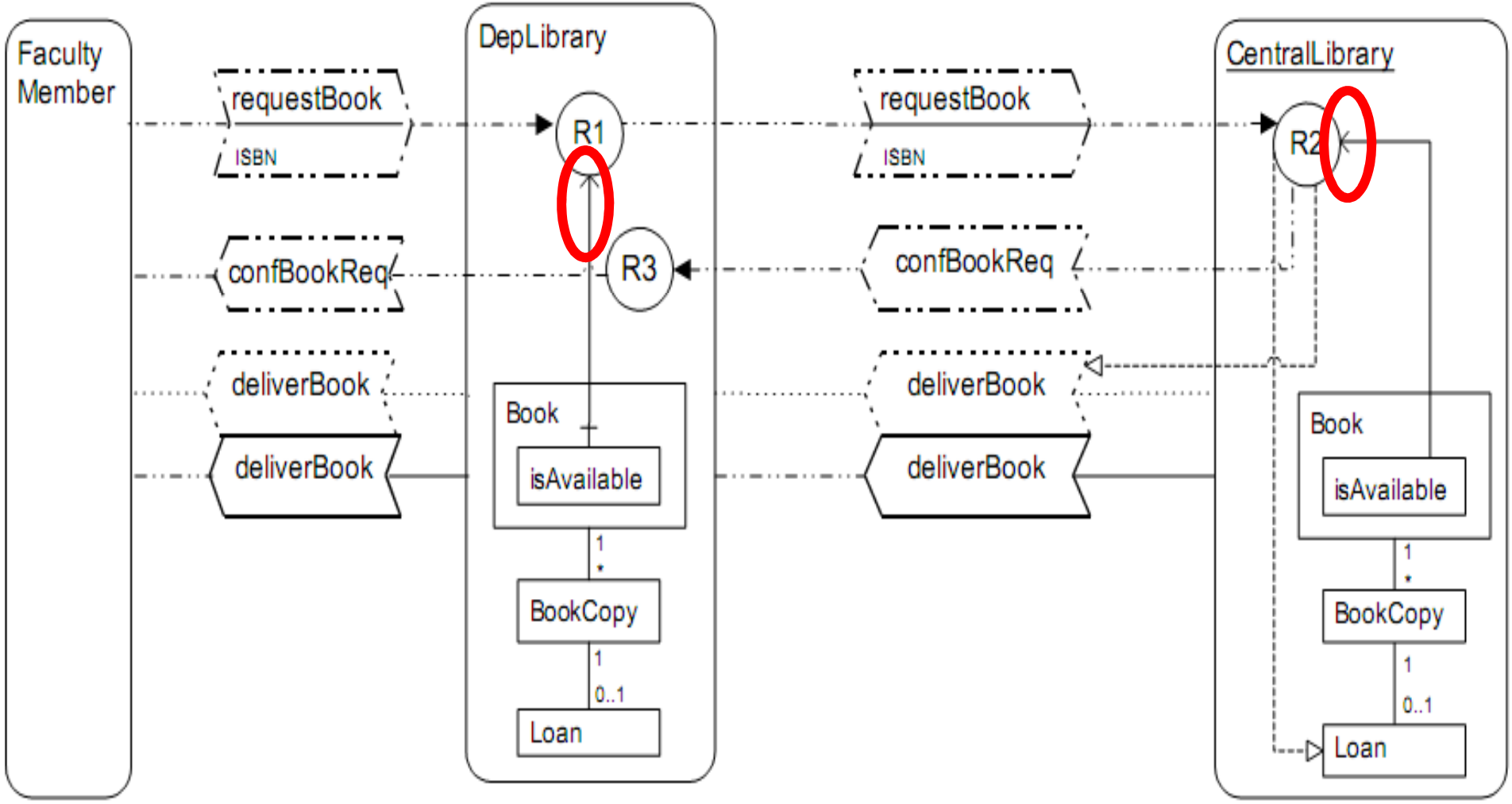
Diagramas de padrões de interação e *Reaction rules*



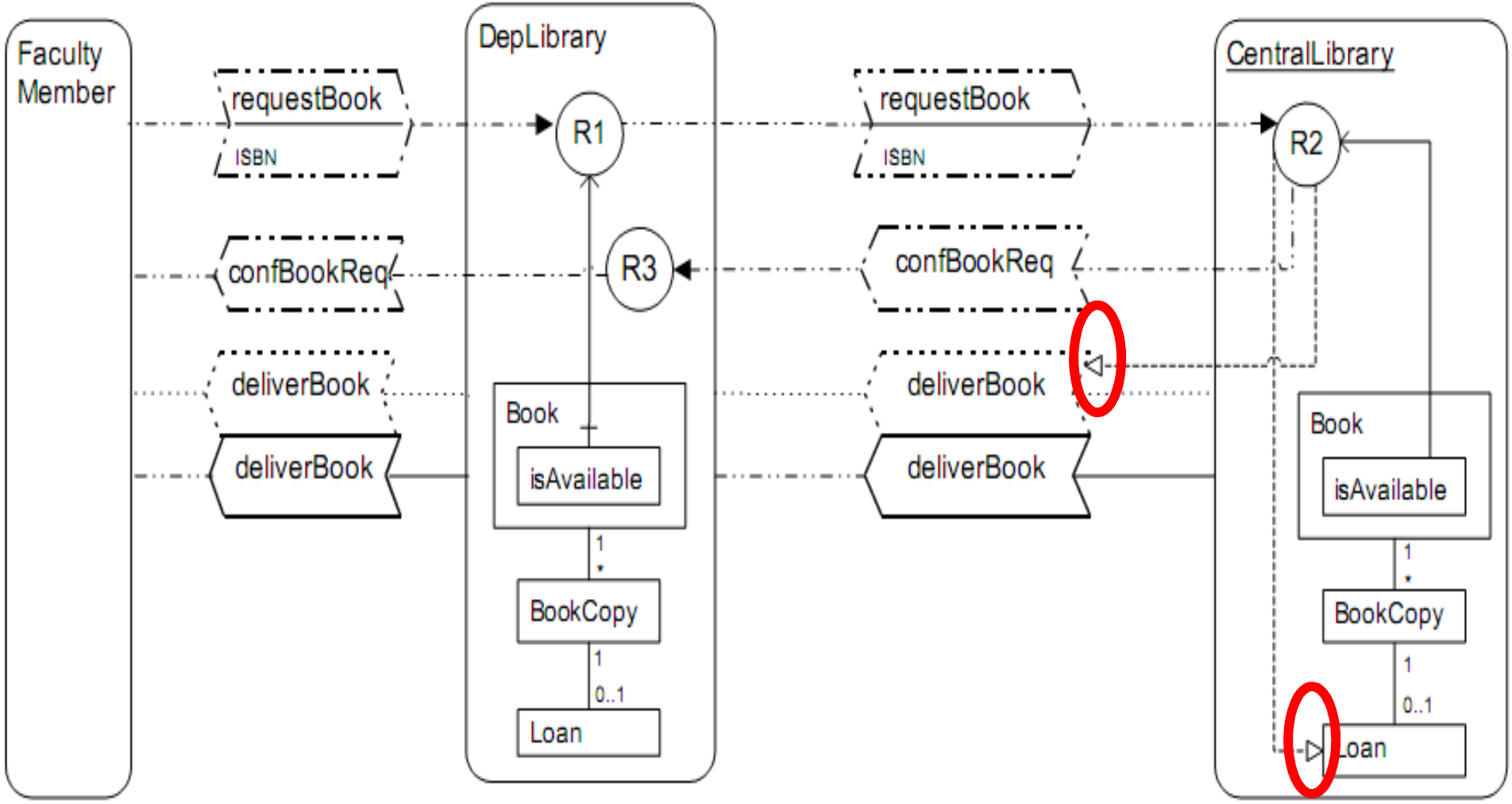
Diagramas de padrões de interação e *Reaction rules*



Diagramas de padrões de interação e *Reaction rules*



Diagramas de padrões de interação e *Reaction rules*



Diagramas de padrões de interação e *Reaction rules*

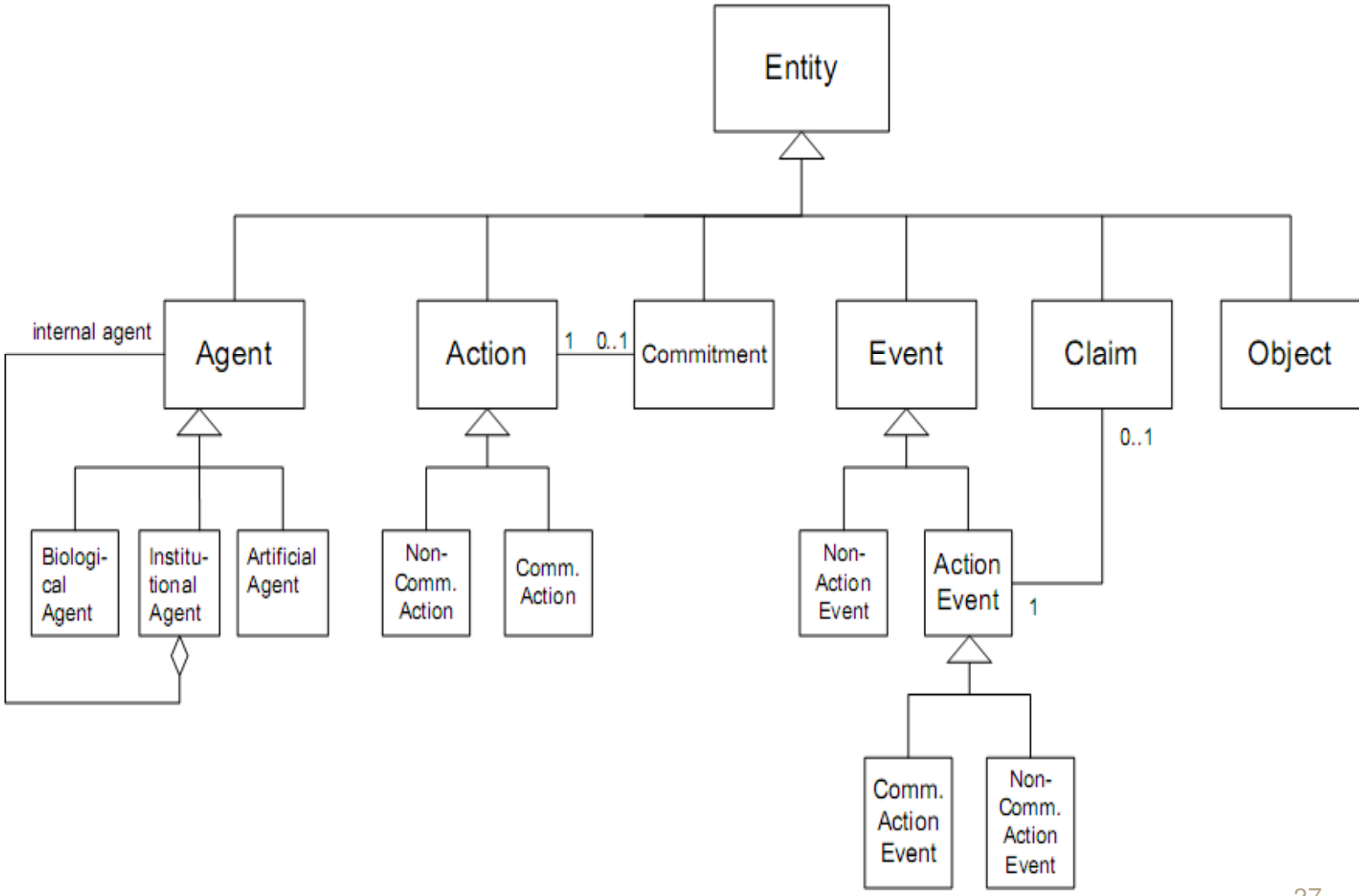
ON	Event	RECEIVE requestBook(?ISBN) FROM ?DepLib
IF	Condition	BookCopy.isAvailable(?ISBN, ?InvNo)
THEN	Action	SEND confBookReq(?ISBN) TO ?DepLib
	Effect	CREATE COMMITMENT TOWARDS ?DepLib TO deliverBook(?ISBN) BY tomorrow(); CREATE BELIEF Loan(?DepLib, ?ISBN, ?InvNo, today())

Modelos Internos

Visão de um agente em particular

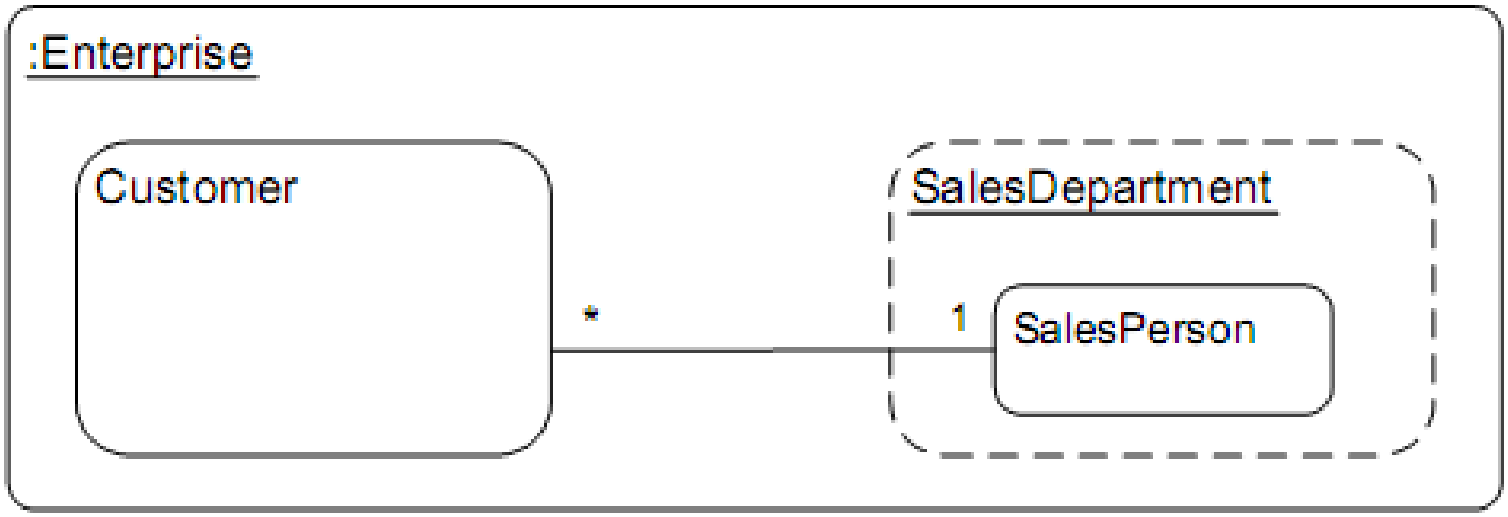
- Outros Agentes
- Ações
- Eventos
- Compromissos e reivindicações
- Objetos comuns
- Relacionamentos
- Associações

Modelos Internos

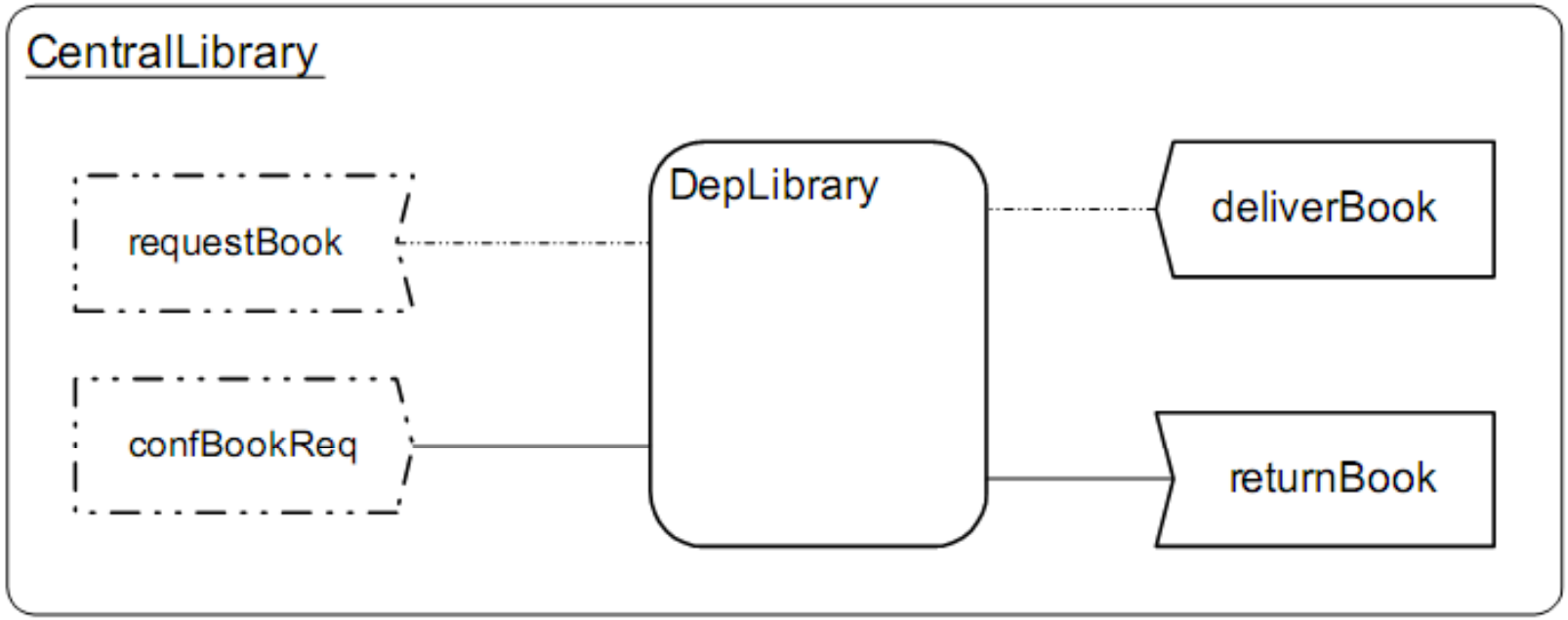


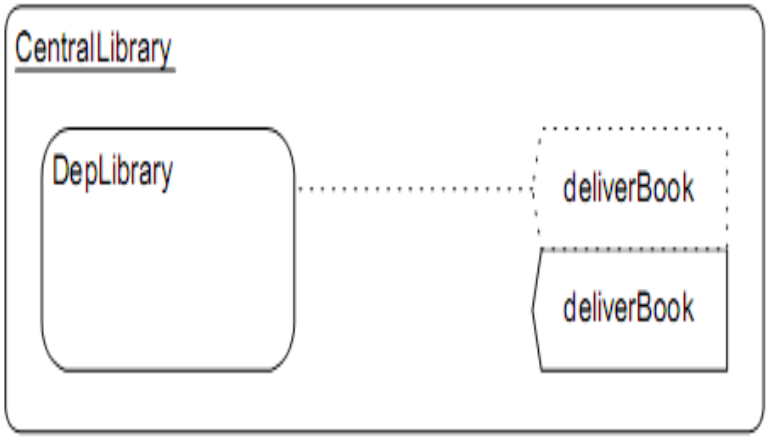
Modelos Internos

- Diagramas de quadro de reação
- Diagramas de sequência de reação
- Diagramas de padrões de reação



Modelos Internos





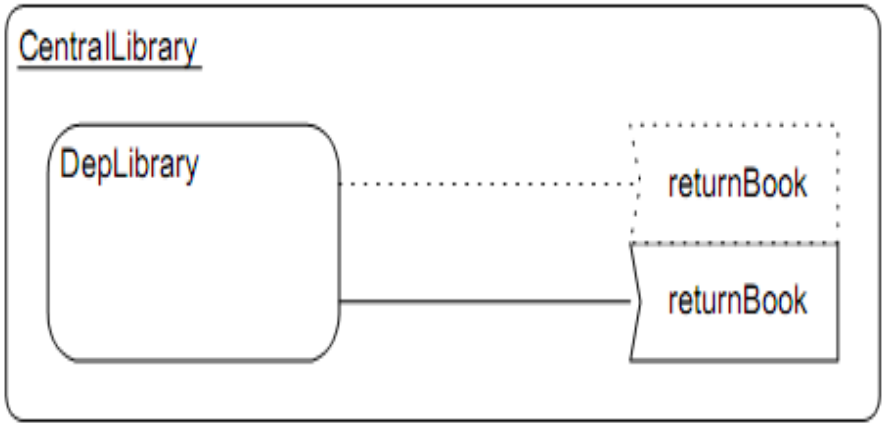


Diagrama de quadro de reação

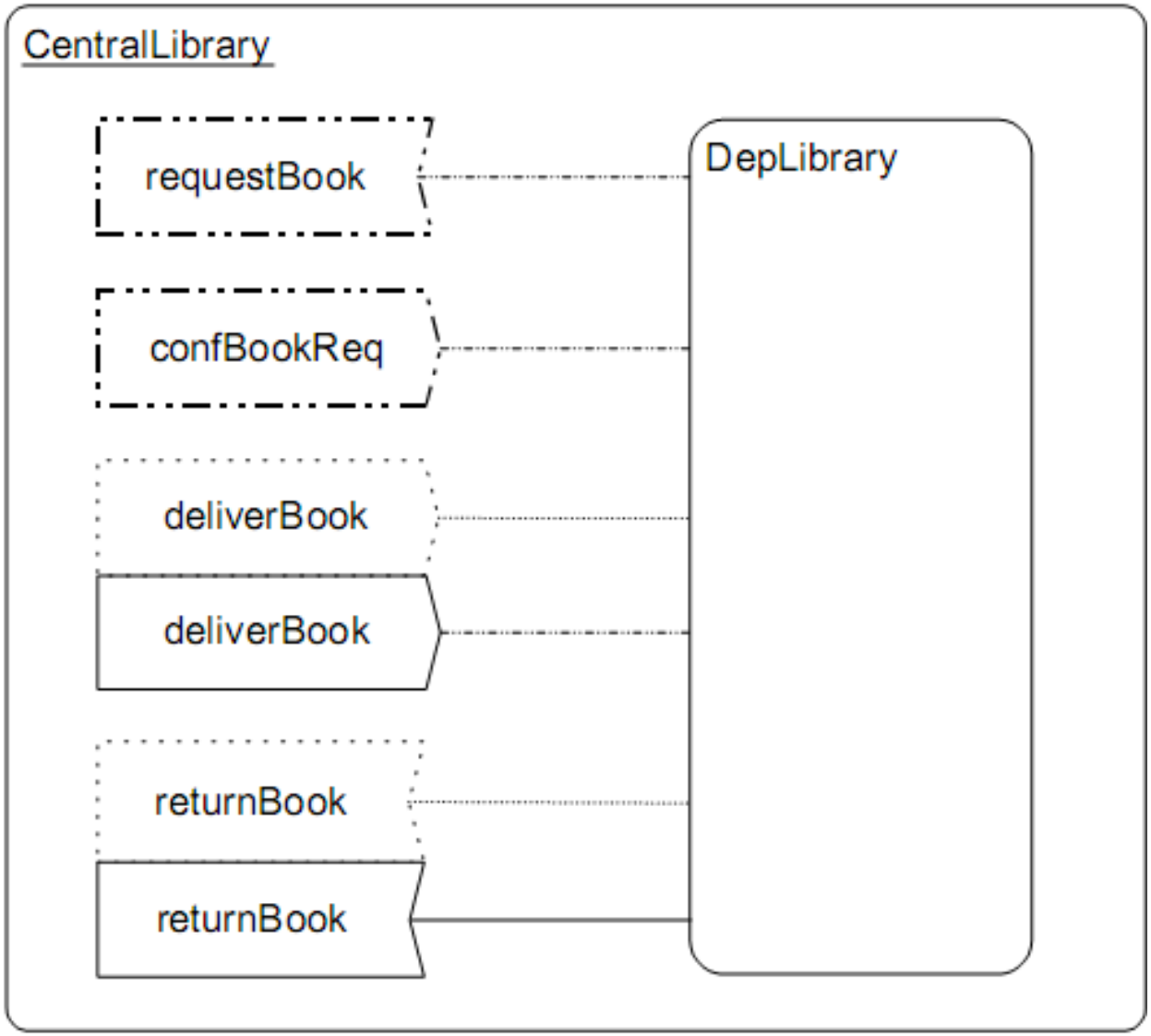
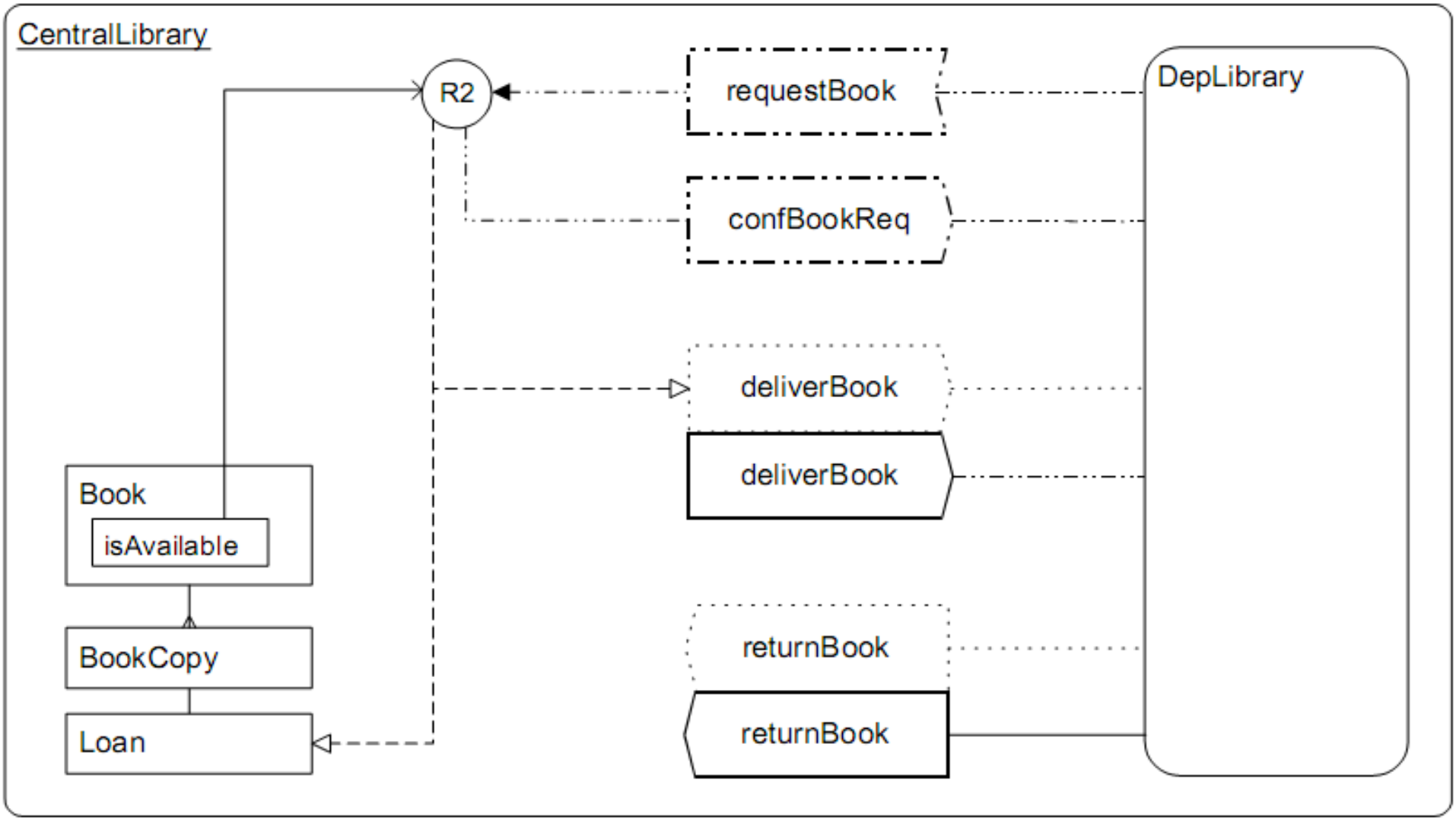


Diagrama de padrões de reação



Internalization

1. **Omitir o agente em foco** cuja perspectiva será modelada;

2. Transformar todos os retângulos de ***action events*** direcionados **para o agente em foco** para retângulos de **eventos**;

3. Transformar todos os retângulos de ***action events*** direcionados **para os parceiros de interação do agente em foco** para retângulos de **ações**;

Internalization

4. Transformar todos os retângulos de **compromissos/reivindicações** direcionados para **o agente em foco** para retângulos de **reivindicações**;
5. Transformar todos os retângulos de **compromissos/reivindicações** direcionados para os **parceiros de interação do agente em foco** para retângulos de **compromissos**;
6. **Pontilhando as linhas** dos retângulos para os agentes internos de nível mais alto;
7. **Fundindo** as representações internas e externas dos **objetos**.

Modelando direitos e deveres de agentes interno

- Ter o dever de reagir a certo evento

  DR(i, ε)

- Ter o dever de cumprir um compromisso

  DfC(i, α)

- Ter o dever de monitorar certa reivindicação

  DmC(i, ε)

- Ter o direito de realizar certa ação

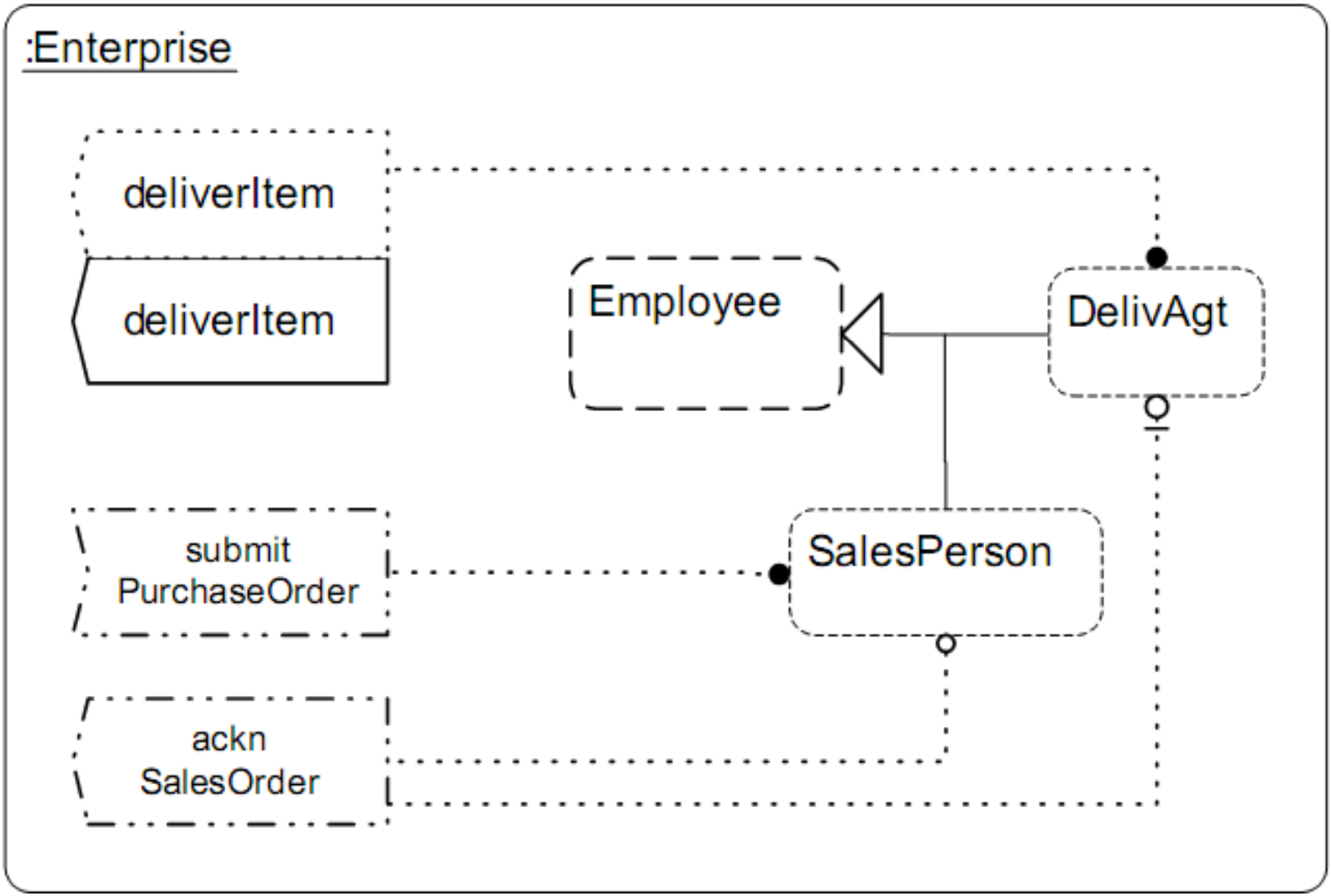
  Right(i, α)

  DfC(i, α) ⊃ Right(i, α)

- Não ter direito de realizar certa ação

  Proh(i, α)

Modelando direitos e deveres de agentes interno



Modelando direitos e deveres de agentes interno

- Princípio da indeterminação normativa

$$\neg \forall i \forall \alpha \, (\mathbf{Right}(i, \alpha) \vee \mathbf{Proh}(i, \alpha))$$

$$\exists i \exists \alpha \, (\neg \mathbf{Right}(i, \alpha) \wedge \neg \mathbf{Proh}(i, \alpha))$$

- Princípio da inconsistência normativa

$$\exists i \exists \alpha \, (\mathbf{Right}(i, \alpha) \wedge \mathbf{Proh}(i, \alpha))$$

Outras técnicas para modelagem de comportamento

- Propõe utilizar diagramas de **atividades da UML** e invariantes de comportamento para complementar a modelagem
- Usar as raias de natação de UML para definir *action events* dos agentes
- Definir **invariantes de progresso** e **invariantes de segurança** para expressar propriedades importantes do sistema e para supervisão automatizada futura.

Trabalhos Relacionados

(Extended) ER	UML	External AORML	Internal AORML
entity type	class active class sent signal received signal	object type agent type message type commitment/claim type	object type agent type outgoing message type incoming message type commitment type claim type
relationship type	association	association sends receives does perceives hasCommitment hasClaim	association isSentTo isReceivedFrom isPerceivedBy isCreatedBy hasCommitment Towards hasClaimAgainst
–	–	reaction rule	reaction rule

Trabalhos Relacionados

(Extended) ER	UML	External AORML	Internal AORML
ER diagram	class diagram	agent diagram interaction frame diagram	reaction frame diagram
–	sequence diagram	interaction sequence diagram	reaction sequence diagram
–	activity diagram state machine d.	activity diagram	activity diagram state machine d.
–	–	interaction pattern diagram	reaction pattern diagram

Trabalhos Relacionados

- AUML: também utiliza diagramas de sequência e atividade. **Não há distinção (gráfica) entre agentes e objetos.**
- Modelagem de Negócios:

AORML	*CIMOSA*	*Enterprise Ontology*	*PfBM*
agent	functional entity	actor	worker
action	functional operation	activity, action	–
event	event	–	–
object	enterprise object	entity	entity
reaction rule	behavioral rule	–	–

Trabalhos Relacionados

- The Resource-Event-Agent (REA): **distingue categorias de entidades como recursos, eventos e agentes** mas não possui linguagem visual.

- [KGR96]: **metodologia para análise de sistemas multi-agentes, utiliza BDI**, mas não possui linguagem visual.

- [EL99, WJK00]: **considera papéis de agentes, direitos e deveres, contratos, protocolos de comunicação**, mas não possui linguagem visual.

Pontos fortes da AORML

- AORML tem conceitos ontológicos ricos, permitindo **capturar maior semântica do domínio** se comparado a ER, UML e AUML
- AORML **inclui conceitos** fundamentais encontrado em modelagem de empresas **como CIMOSA e the Eriksson-Penker**
- Permite **combinar estado e comportamento** em 1 um único diagrama
- Inclui **modelagem de direitos e deveres**
- Primeira pesquisa que **distingue modelos internos e externos**, e inclui ***reaction rules*** na modelagem

Pontos fracos da AORML

- O processo completo da análise para a implementação não foi completamente definido
- AORML não inclui o conceito de atividades
- AORML não inclui o conceito de objetivos que é fundamental para enumeras pesquisas
- AORML não permite modelagem de comportamento proativo dos agentes.

Modelling Security and Trust with Secure Tropos

Autores: Paolo Giorgini, Haralambos Mouratidis, Nicola Zannone

Ano: 2006

Apresentação: Karen da Silva Figueiredo

Sistemas Multi-agentes – Apresentação I

Instituto de computação

Introdução

- Segurança de Sistemas de Informação não se trata apenas de prover uma série de mecanismos de segurança como autenticação e confidenciabilidade.
- Capacidade de **raciocinar** sobre segurança
- Confiança (Trust)

Introdução

- Propõe **a integração de duas linhas de pesquisa**, resultando em uma metodologia que considera segurança e confiança partes do processo de desenvolvimento

  - Security-oriented process (Mouratidis, 2004; Mouratidis, 2005)

  - Trust management process (Giorgini, 2004; Giorgini, 2005a; Giorgini, 2005b; Giorgini, 2005d)

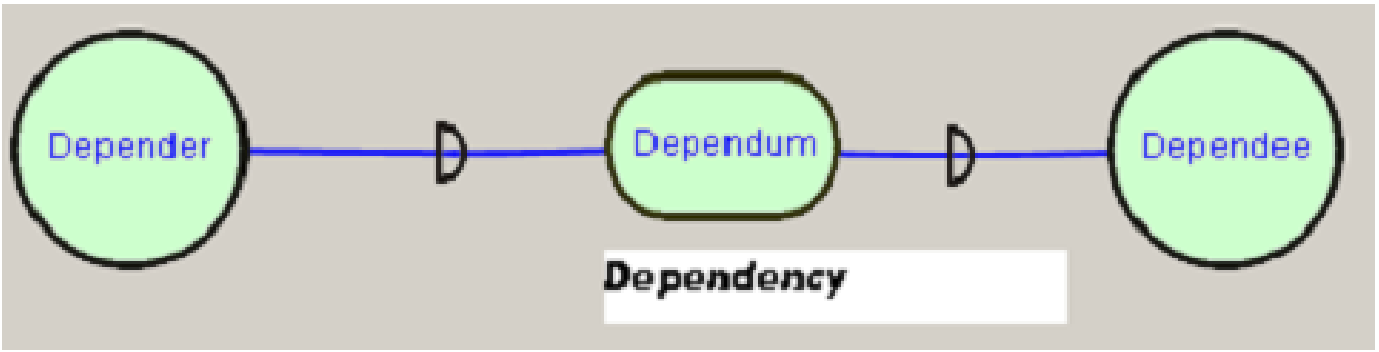
Tropos

- Metodologia de desenvolvimento de software para descrever **ambientes organizacionais** de um sistema e o **próprio sistema**

- **Framework i*** (Yu, 1996): atores, objetivos, planos, recursos e dependências sociais para definir as obrigações dos atores

Tropos

- Ator
- Objetivo (*Hard goals x Soft goals*)
- Plano
- Recurso
- Dependência



Tropos

- 4 Fases de desenvolvimento:
  - Early Requirements
  - Late requirements
  - Architectural design
  - Detailed design

Tropos

- Modelagem de atores
- Modelagem de dependências
- Diagramas de atores
- Diagramas de objetivos

Secure Tropos

- Security-oriented process (Mouratidis, 2004; Mouratidis, 2005)
  - Invariantes de segurança e capacidades de segurança
- Trust management process (Giorgini, 2004; Giorgini, 2005a; Giorgini, 2005b; Giorgini, 2005d)
  - Ownership, confiança, delegação
  - Diferenciar a idéia de possuir um recurso e oferecer um recurso

Secure Tropos

- Extende a metodologia Tropos
- Conceitos:
  - Invariantes de Segurança
  - Entidades Seguras
  - *Ownership*
  - Provisionamento
  - Confiança de Permissão
  - Confiança de Execução

Secure Tropos

Conceitos (cont):

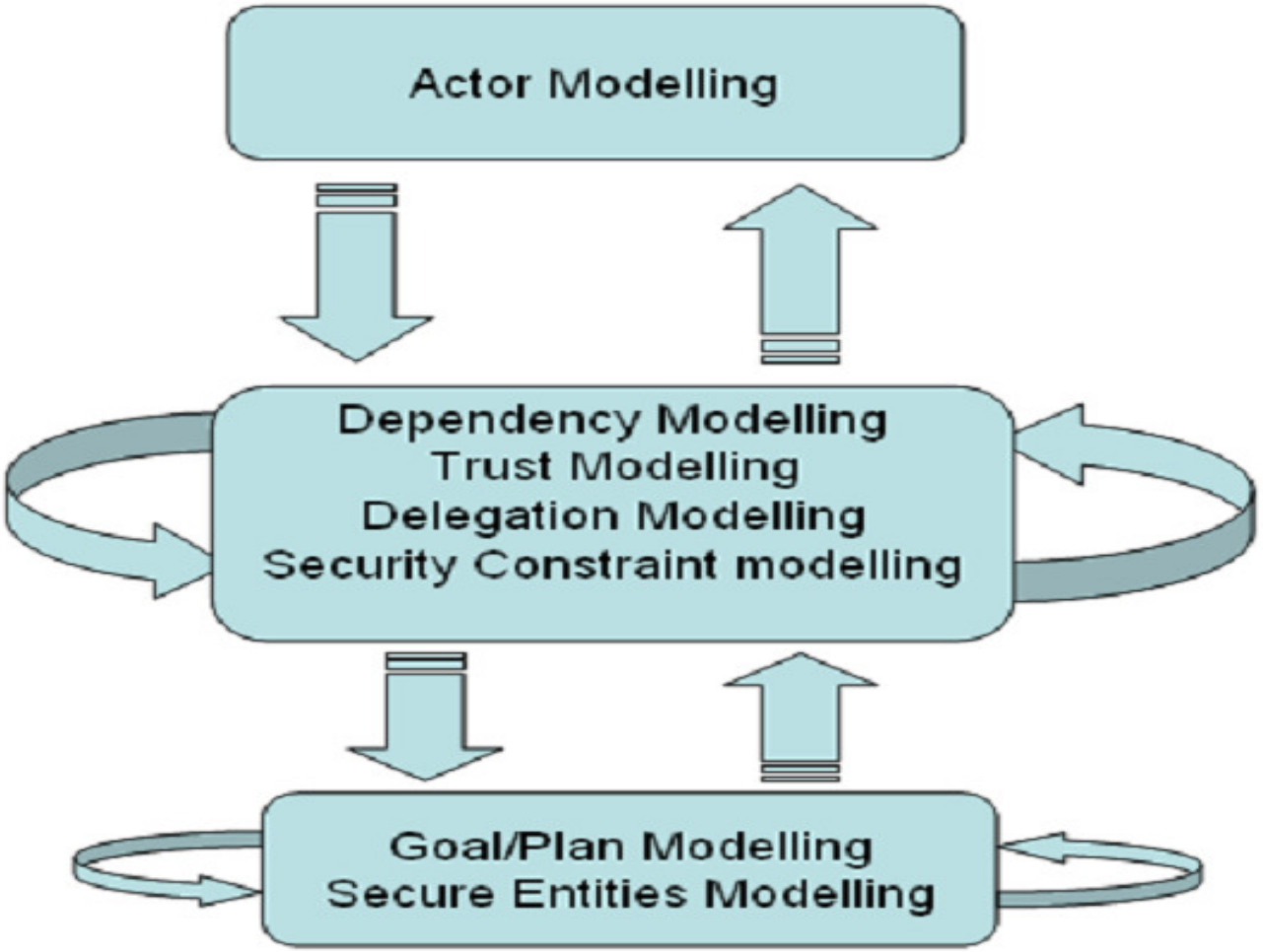
- Delegação de Permissão
- Delegação de Execução
- Confiança de Permissão Segura
- Delegação de Permissão Segura

Secure Tropos

- Etapas de modelagem:
  - Modelagem de Confiança de Permissão
  - Modelagem de Delegação de Permissão
  - Modelagem de Invariantes de Segurança
  - Modelagem de Entidades Seguras

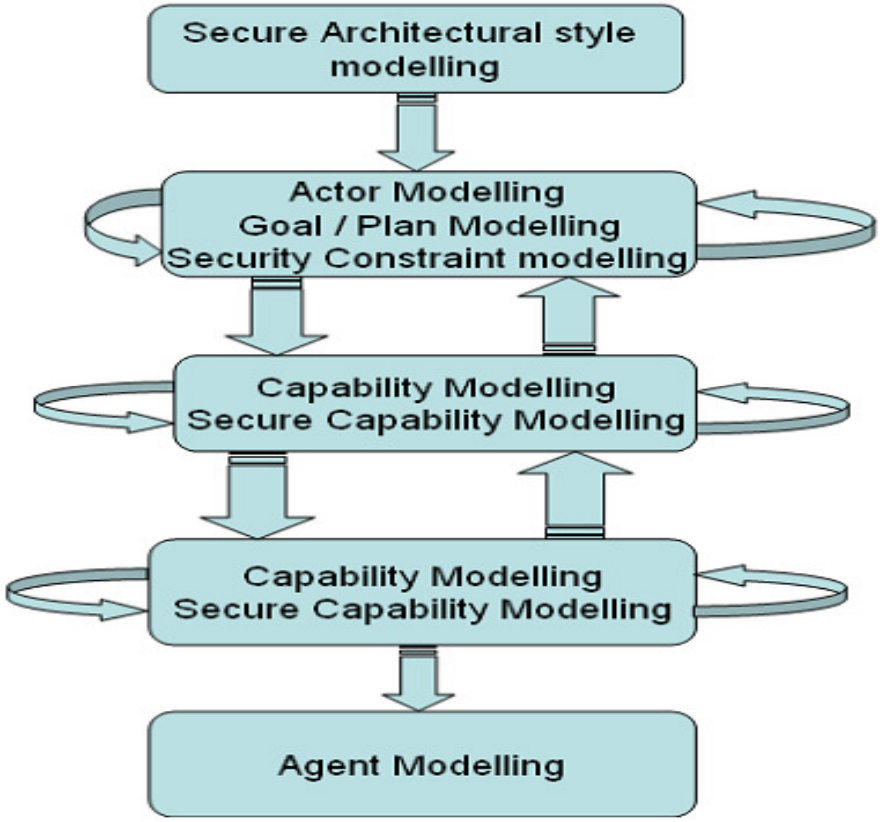
O processo da Secure Tropos

- Early / Late Requirements Phase



O processo da Secure Tropos

- Architectural Design



- Detailed Design Stage

Secure Tropos – A formalização

- Com objetivo de automatizar a verificação da corretude e consistência
- Baseada em Datalog (Abiteboul, 1995)
- t ∈ {exec,perm} / t ∈ {satisfy,exec,owner}.

General predicates
delegate(Type : t. Actor : a. Actor : b. Service : s)
delegateChain(Type : t. Actor : a. Actor : b. Service : s)
trust(Type : t. Actor : a. Actor : b. Service : s)
trustChain(Type : t. Actor : a. Actor : b. Service : s)
confident(Type : t. Actor : a. Service : s)

Secure Tropos – A formalização

Specific for execution
requests(Actor : a. Service : s)
provides(Actor : a. Service : s)
should_do(Actor : a, Service : s)
Specific for permission
Owns(Actor : a. Service : s)
has_per(Actor : a, Service : s)
Goal refinement
goal(Service : s)
subgoal(Service : s1 . Service : s2)
OR_subgoal(Service : s1 , Service : s2)
AND_decomp(Service : s1 , Service : s2 , Service : s3)

Secure Tropos – A formalização

Axioms for delegation and trust	
Ax1	*delegateChain(T,A,B,S) :- delegate(T,A,B,S)*
Ax2	*delegateChain(T,A,C,S) :- delegate(T,A, B,S), delegateChain(T,B,C,S)*
Ax3	*trustChain(T,A,B,S) :- trust(T,A,B,S)*
Ax4	*trustChain(T,A,C,S) :- trust(T, A,B,S), trustChain(T,B,C,S)*
Ax5	*trustChain(exec,A,B,S') :- subgoal(S,S'), trustChain(exec,A,B,S)*
Ax6	*trustChain(perm,A,B,S') :- subgoal(S',S), trustChain(exec,A,B,S)*
Axioms for execution	
Ax7	*should_do(A,S) :- delegateChain(exec,B,A,S), provides(A,S)*
Ax8	*should_do(A,S) :- requests(A,S),provides(A,S)*
Ax9	*confident(satisfy,A,S) :- should_do(A,S)*
Ax10	*confident(satisfy,A,S) :- delegateChain(exec,A,B,S), trustChain(exec,A,B,S), confident(satisfy,B,S)*
Ax11	*confident(satisfy,A,S) :- OR subgoal(S',S), confident(satisfy,A, S')*
Ax12	*confident(satisfy,A,S) :- AND_decomp(S, S',S''), confident(satisfy,A,S'), confident(satisfy,A,S'')*

L :- L1,...,Ln

Secure Tropos – A formalização

Axioms for permission	
Ax13	*has_per(A,S) :- owns(A,S)*
Ax14	*has_per(A,S) :- delegateChain(perm,B,A,S), has per(B,S)*
Ax15	*has_per(A,S') :- subgoal(S',S), has_per(A,S)*
Ax16	*confident(owner.A.S) :- owns(A.S). not diffident(A.S)*
Ax17	*diffident(A.S) :- delegateChain(perm.A.B.S). diffident(B.S)*
Ax18	*diffident(A.S) :- delegateChain(perm.A.B.S). not trustChain(perm.A.B.S)*
Ax19	*diffident(A.S) :- subgoal(S'.S). diffident(A.S')*
Axioms for execution and permission	
Ax20	*confident(exec,A,S) :- should_do(A, S), has_per(A,S)*
Ax21	*confident(exec.A.S) :- delegateChain(exec.A.B.S). trustChain(exec.A.B.S). confident(exec.B.S)*
Ax22	*confident(exec,A,S) :- OR_subgoal(S',S), confident(exec,A,S')*
Ax23	*confident(exec,A,S) :- AND_decomp(S,S',S''), confident(exec,A,S'), confident(exec,A,S'')*

L :- L1,...,Ln

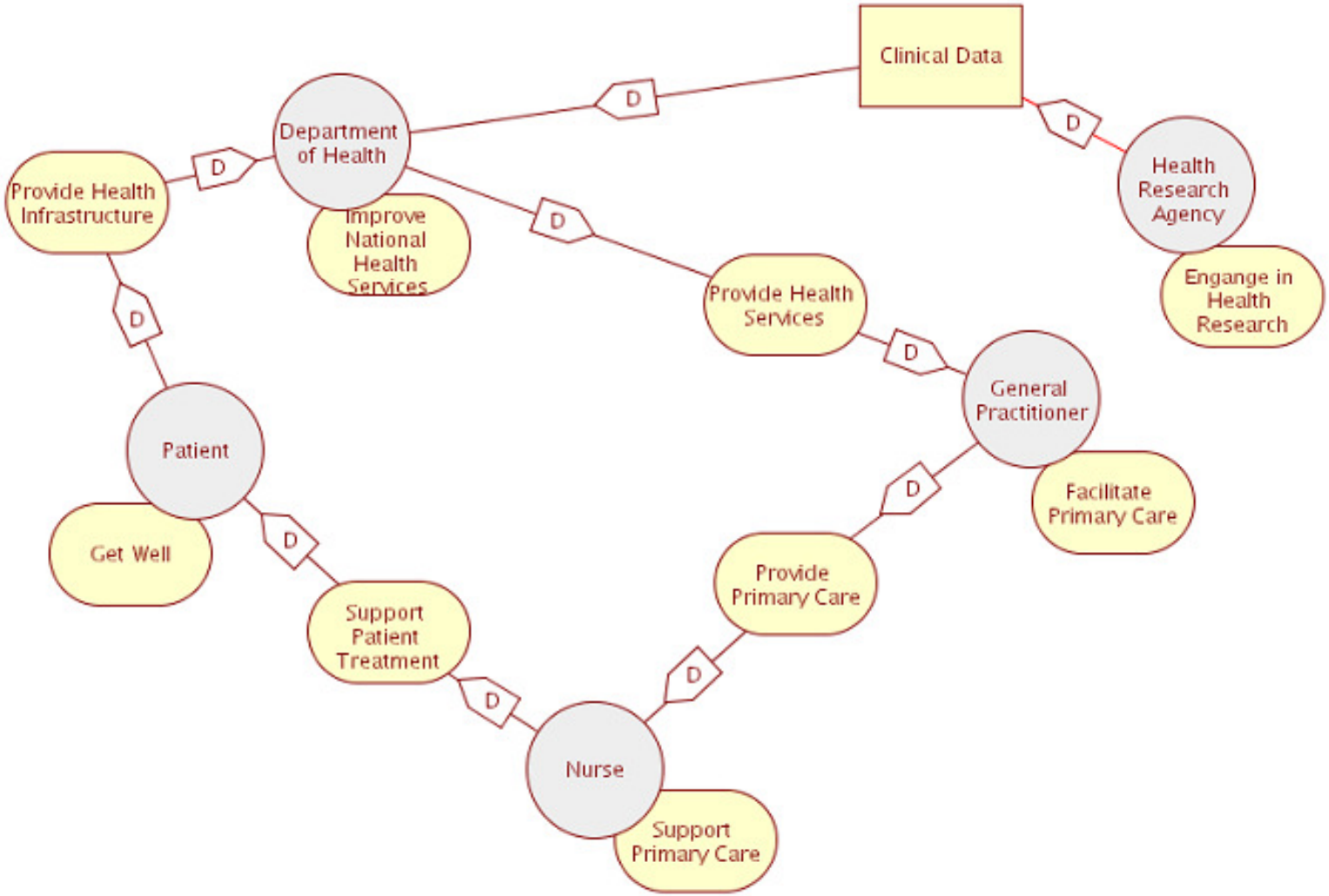
Secure Tropos – A formalização

Pro1	*delegateChain(T.A.B.S) ⇒ ? trustChain(T.A.B.S)*
Pro2	*requests(A.S) ⇒ ? confident(satisfy.A.S)*
Pro3	*owns(A.S) ⇒ ? confident(owner.A.S)*
Pro4	*requests(A.S) ⇒ ? confident(exec.A.S)*
Pro5	*delegateChain(perm,A,B,S) ⇒ ? has_per(A,S)*
Pro6	*should_do(A,S) ⇒ ? not delegateChain(exec,A,B,S)*
Pro7	*owns(A.S) ⇒ ? not delegateChain(perm.B.A.S). A ≠ B*

A ⇒ ? B
:- A, not B.

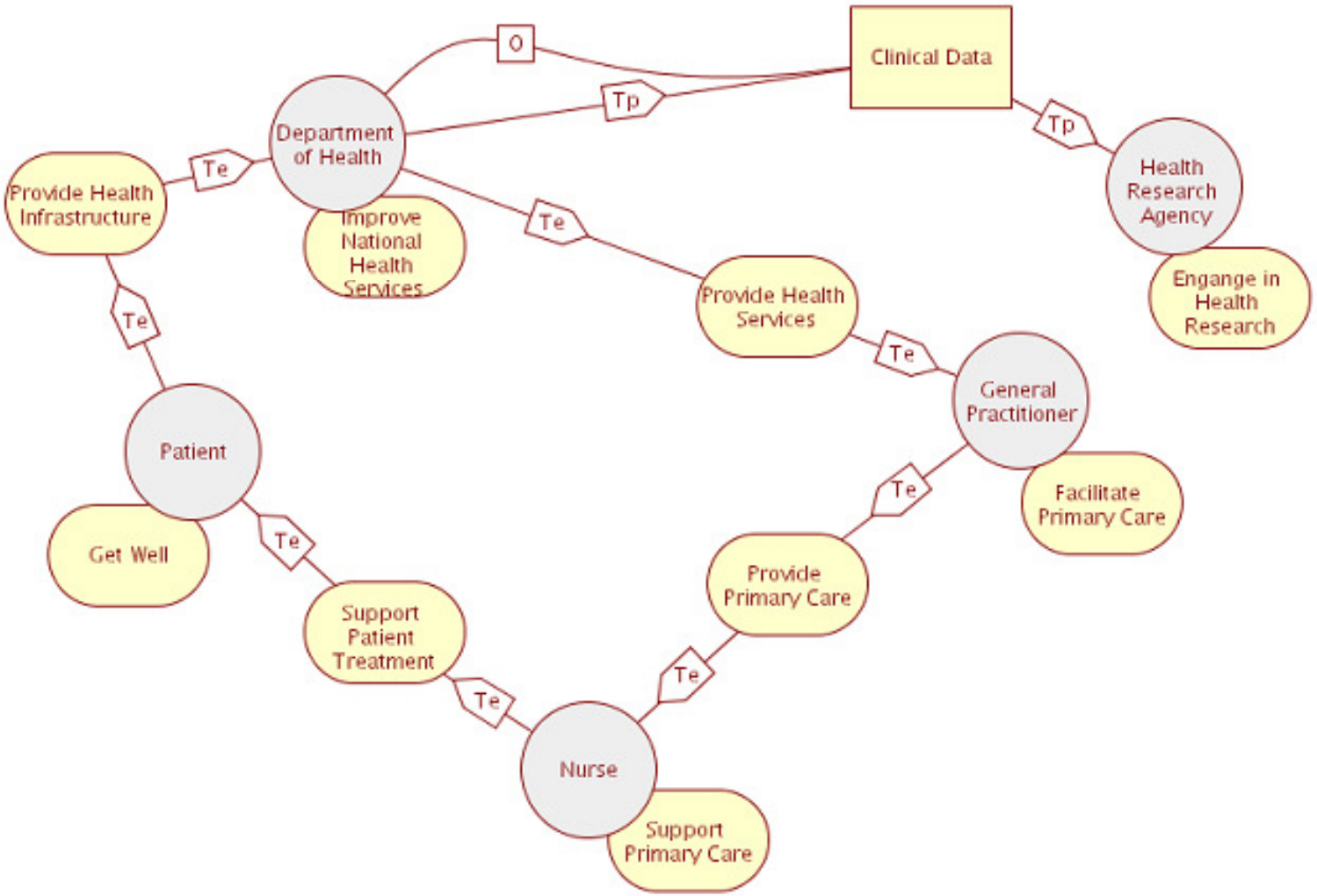
Estudo de Caso - ERP

Initial Dependency Model



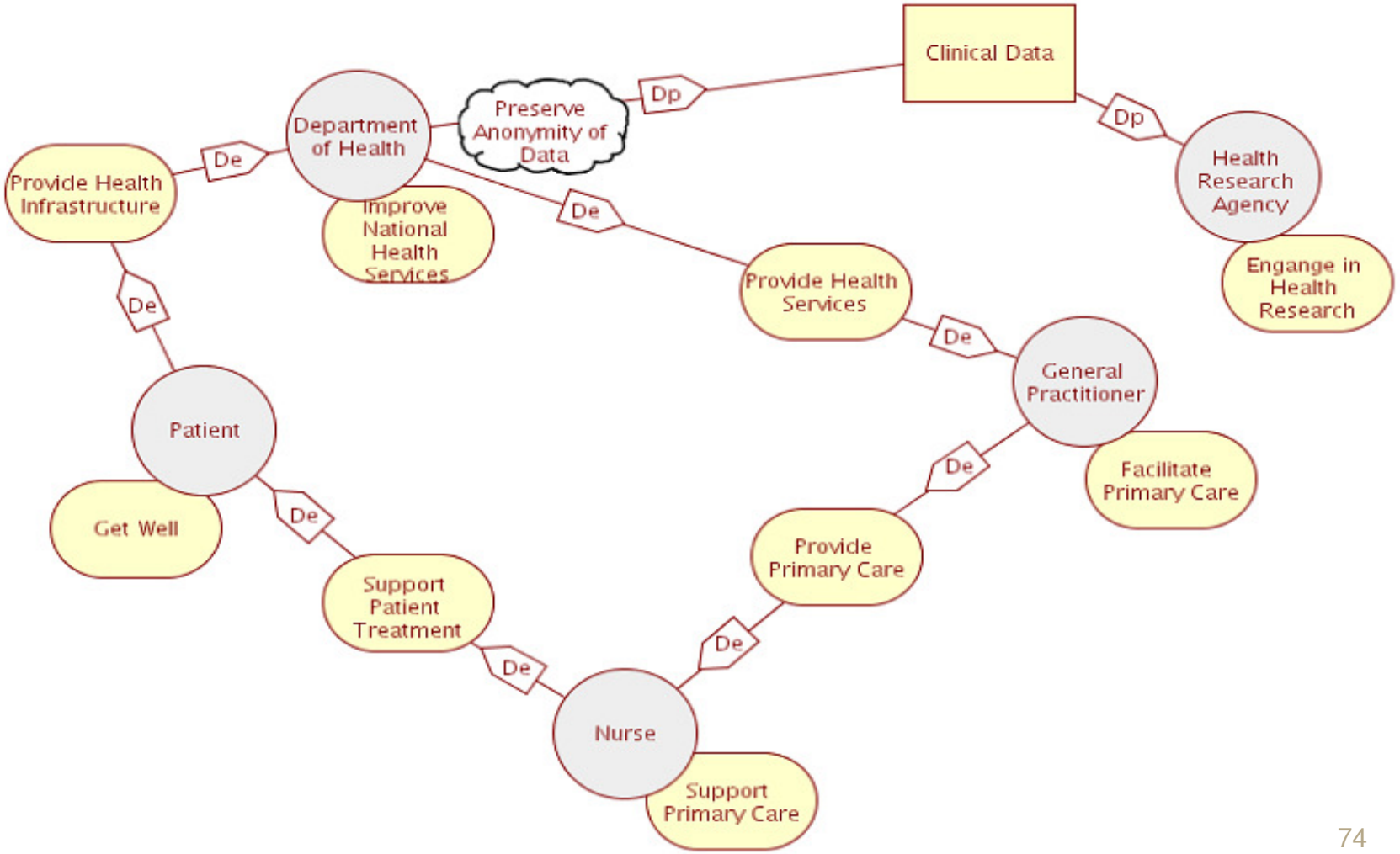
Estudo de Caso - ERP

Trust Analysis



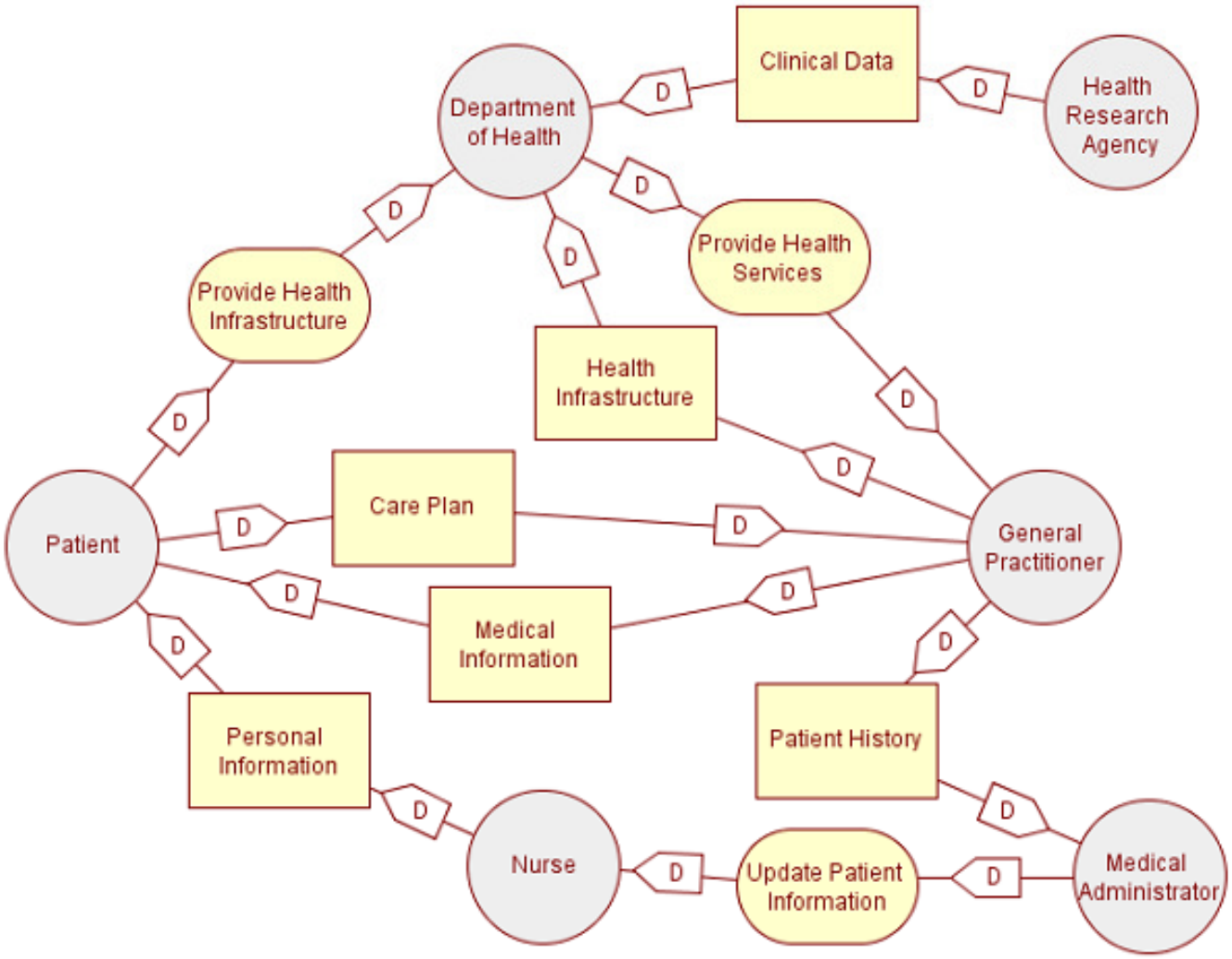
Estudo de Caso - ERP

Delegation and Security constraint analysis



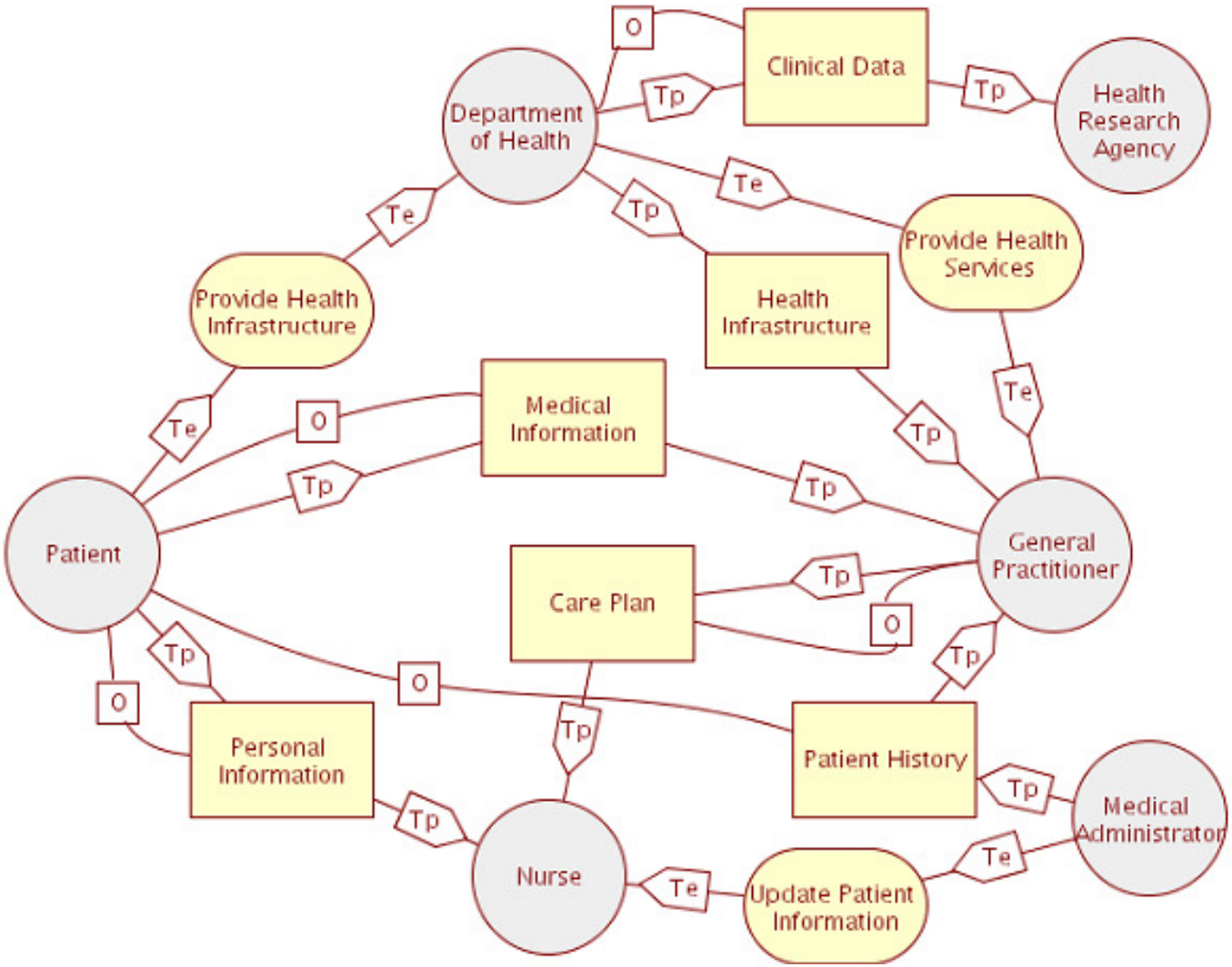
Estudo de Caso - ERP

Refined Dependency Diagram



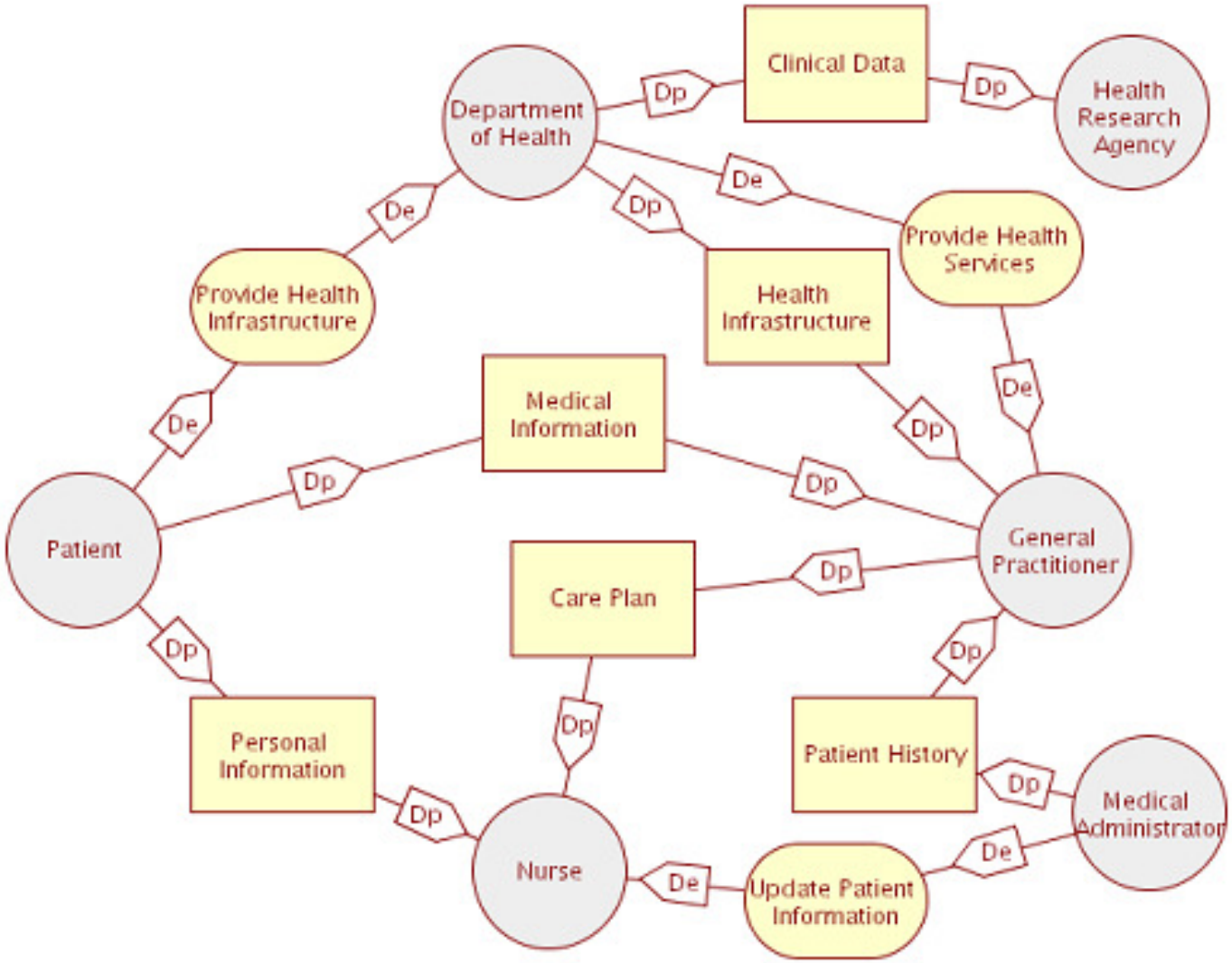
Estudo de Caso - ERP

Refined Trust relationship



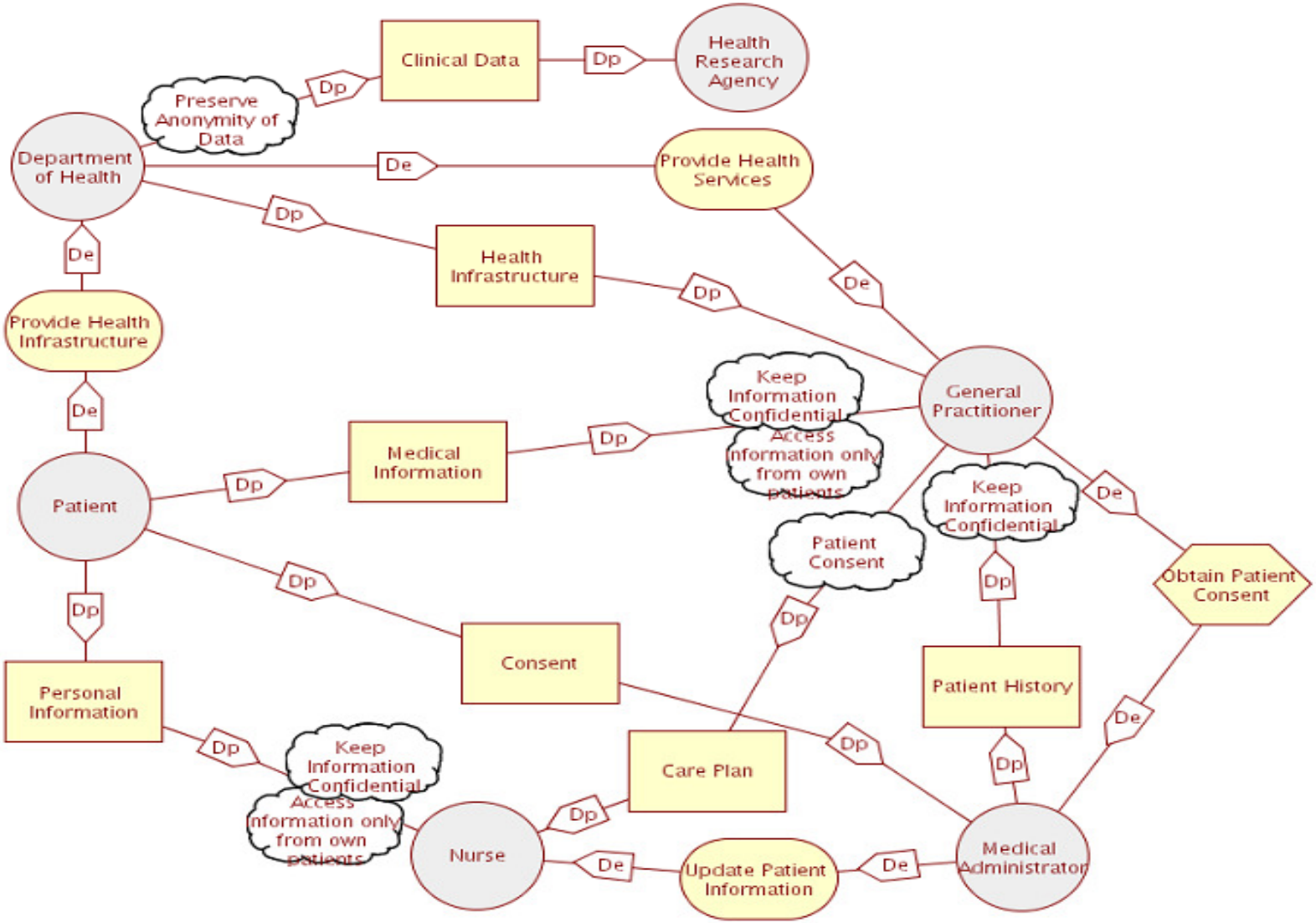
Estudo de Caso - ERP

Refined Delegation relationships



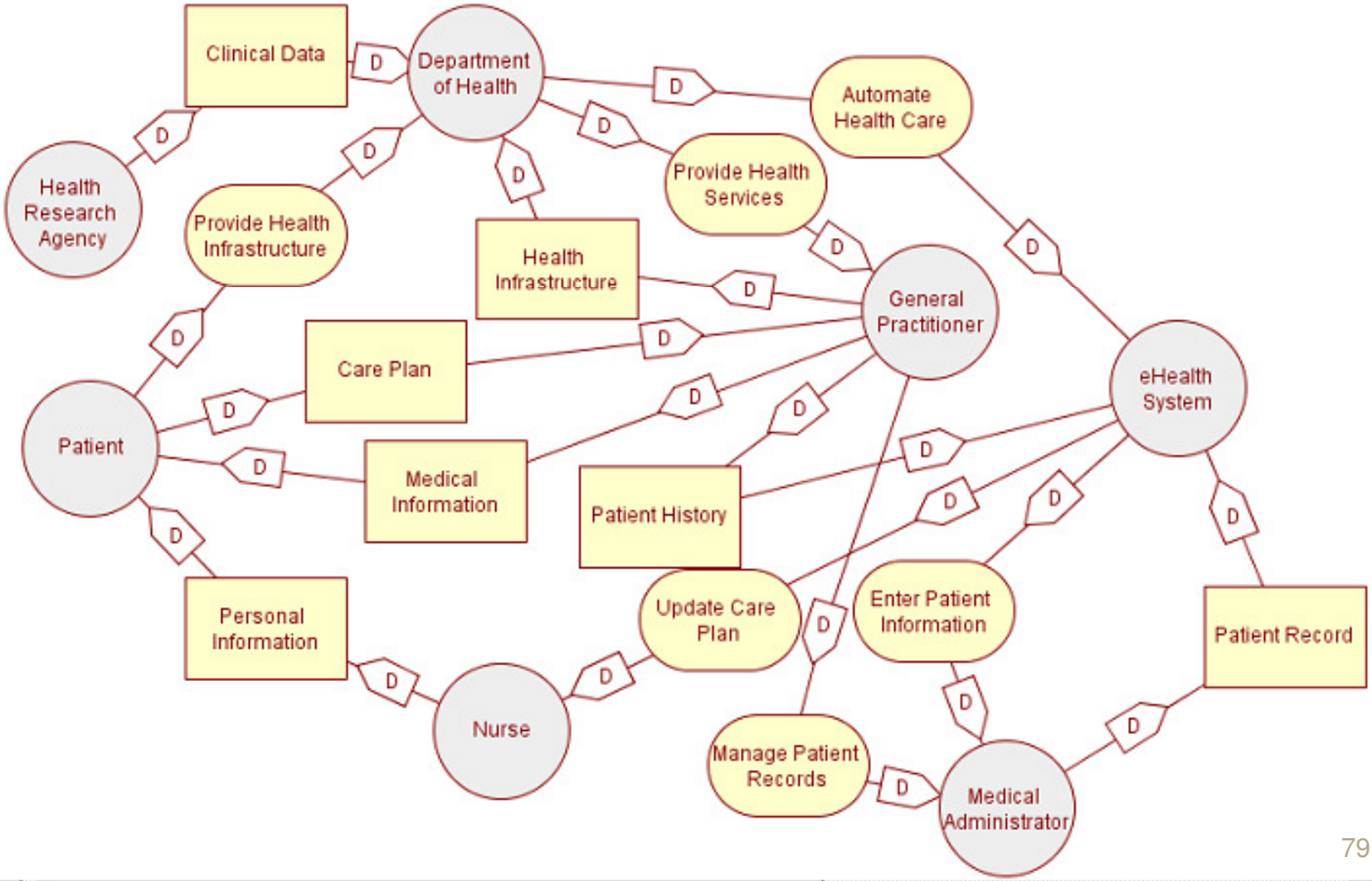
Estudo de Caso - ERP

Refined Security Constraints



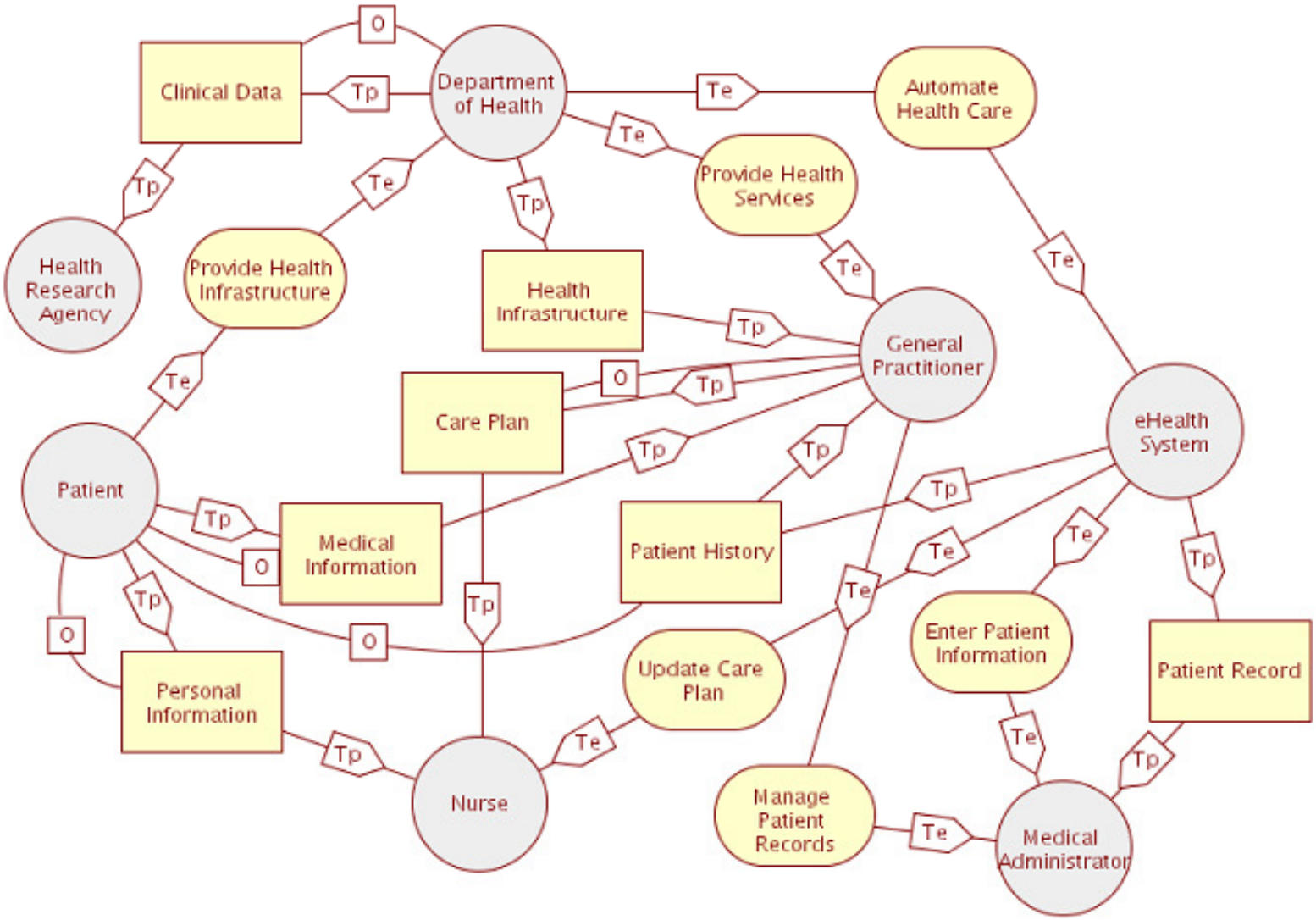
Estudo de Caso - LRP

The revised dependency diagram including the system



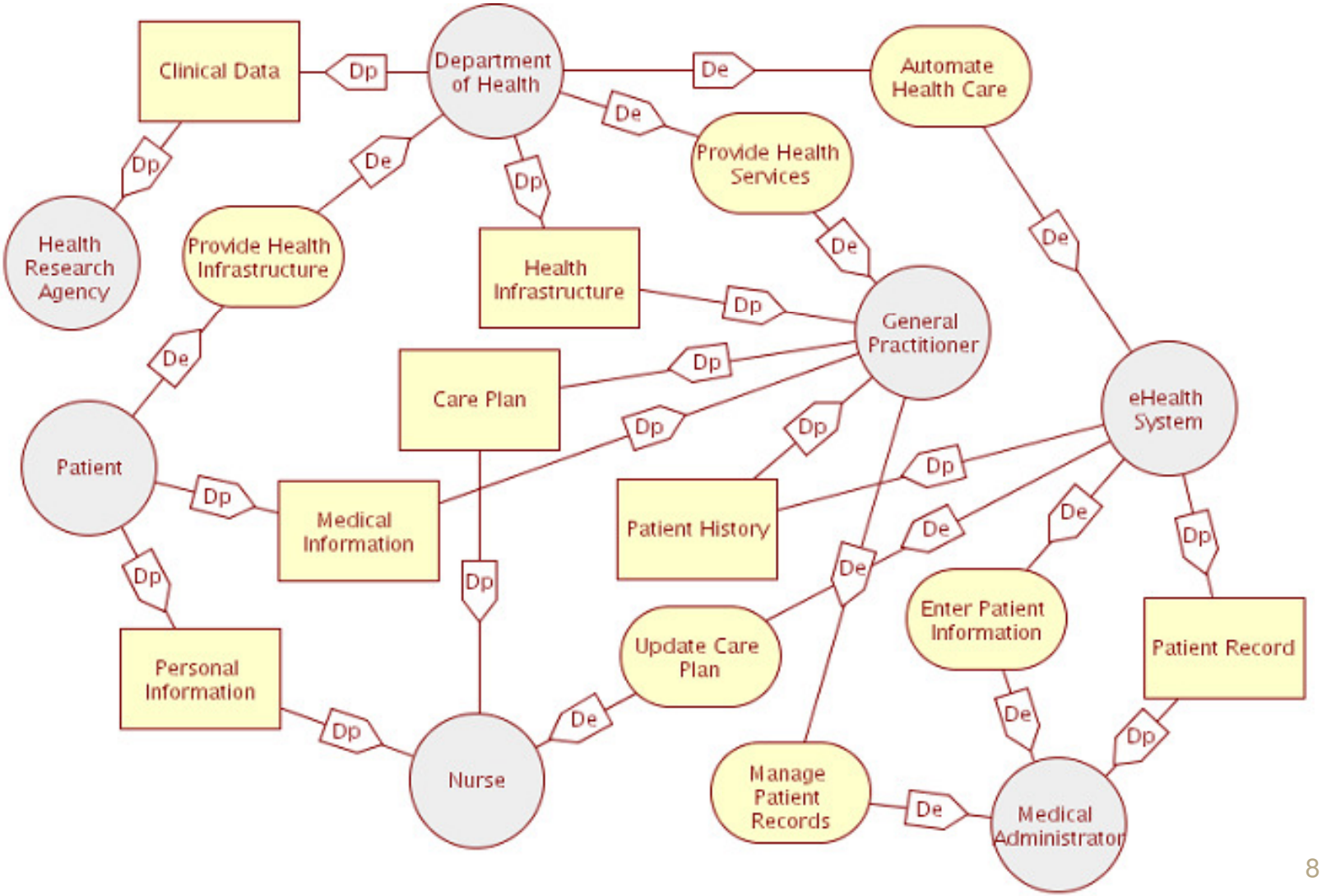
Estudo de Caso - LRP

The trust relationships including the system



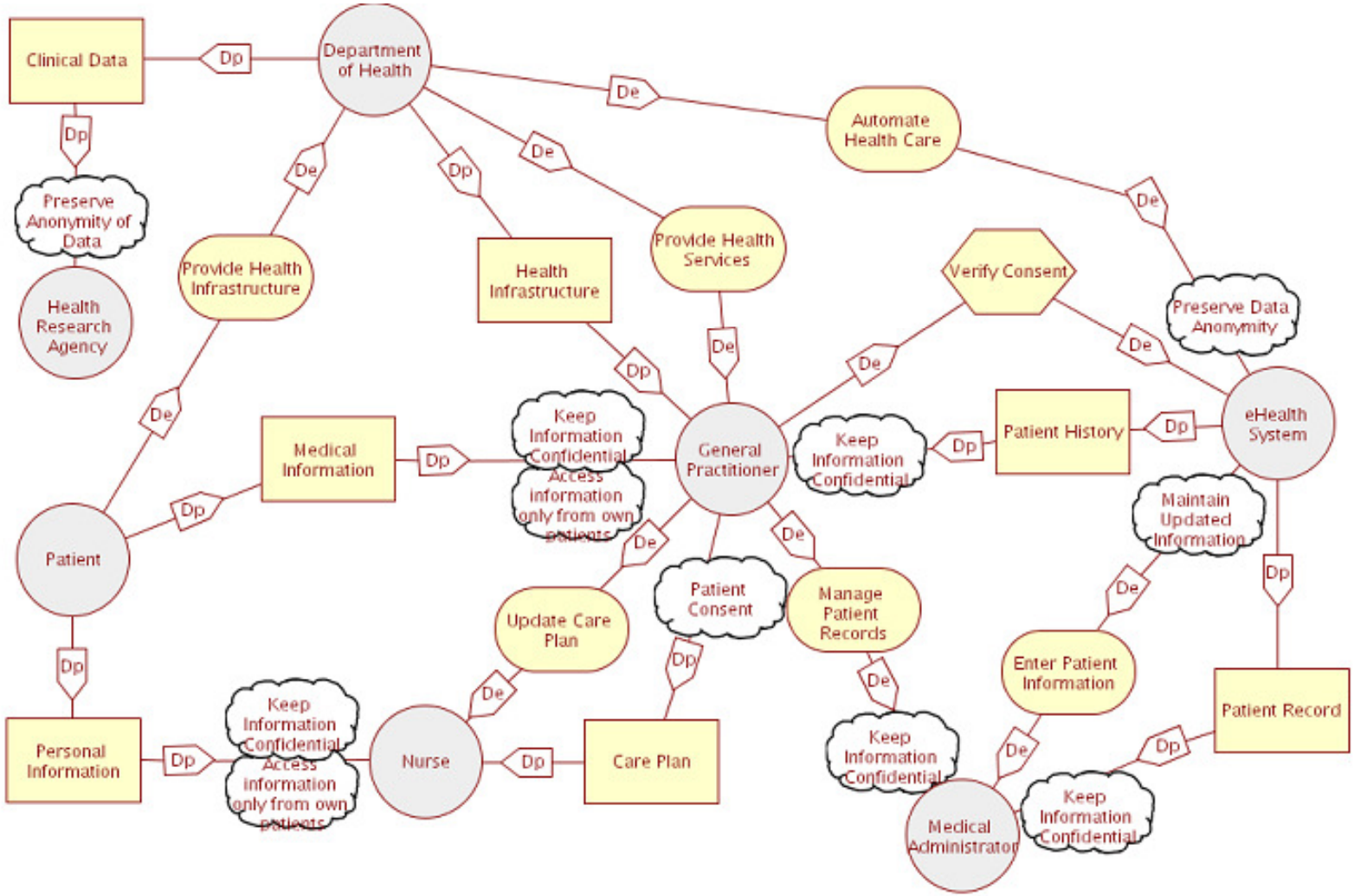
Estudo de Caso - LRP

The delegation relationships including the system



Estudo de Caso - LRP

Security Constraints including the system



Estudo de Caso

- Após modeladas as representações dos modelos, inconsistências podem ser verificadas:
  - 3 inconsistências com relação a propriedade 5

Pro5 $delegateChain(perm,A,B,S) \Rightarrow ? has\_per(A,S)$

  - O Departamento de saúde delega permissão da infra-estrutura de saúde para o clínico.
  - O sistema dá a permissão para o clínico a respeito do histórico registro e histórico do paciente.

Trabalhos Relacionados

- Liu et al. (2002), Haley et al. (chapter 2), Yu et al. (chapter 4) focam na *Early phase*
- Jürgens (2004), by Houmb et al. (chapter 9), Koch et al. (chapter 10) usam um nível mais baixo de abstração (operacional)
- Falcone (1998) destaca a importância da confiança em SMA como um estado mental ou atitude social

Trabalhos Relacionados

- Jøsang (1997) propõe um modelo formal para raciocínio confiança em segurança da informação.
- Frameworks: PolicyMaker (Blaze, 1996), KeyNote (Blaze, 1999), DeTreville (2002)
- **RT framework** (Li, 2002) baseado em programação lógica, RBAC (Sandhu, 1996)
- Entretanto há uma falha entre estes frameworks e o processo de análise de requisitos.

Conclusão

- Análise dos relacionamentos de confiança, delegação e *ownership*
- Identificação das invariantes de segurança
- Permite a modelagem de confiança, delegação e segurança simultaneamente

- Estão trabalhando em novos estudos de caso