

Improving Traceability in Agent Oriented Development

Rosa Candida Cavalcanti Pinto

Supervisor: Dr. Jaelson Castro

Co-supervisor: Dr. Patrícia Tedesco



Submitted in partial fulfilment of requirements of
Degree of Doctor of Software Engineering

Motivação

Rastreamento de artefatos de software tem sido reconhecido como fator de melhoria na qualidade de sistemas de software.

[Spanoudakis, Zisman 2005; Hull, Jackson, Dick 2005]

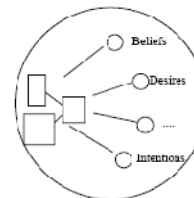
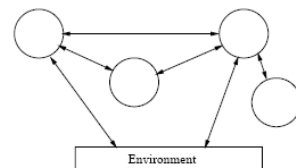
- **Entendimento** do sistema de software
- **Análise** das implicações na ocorrência de mudanças

Mecanismos de rastreamento tradicionais não abordam as necessidades relevantes a Sistemas MultiAgentes (SMA).

Motivação

Sistemas multi-agentes introduzem novas questões de projeto/desenvolvimento quando comparado com abordagens de desenvolvimento de software tradicionais.

Aspectos sociais e de autonomia devem ser considerados durante o desenvolvimento.



[Luck, et al. 2003]

[Mao, Yu 2004]

Motivação

O Problema

Como considerar características específicas de agentes durante um processo de rastreamento?

A Solução

Criação de um modelo e um processo específico para capturar essas características.

Escopo da Proposta

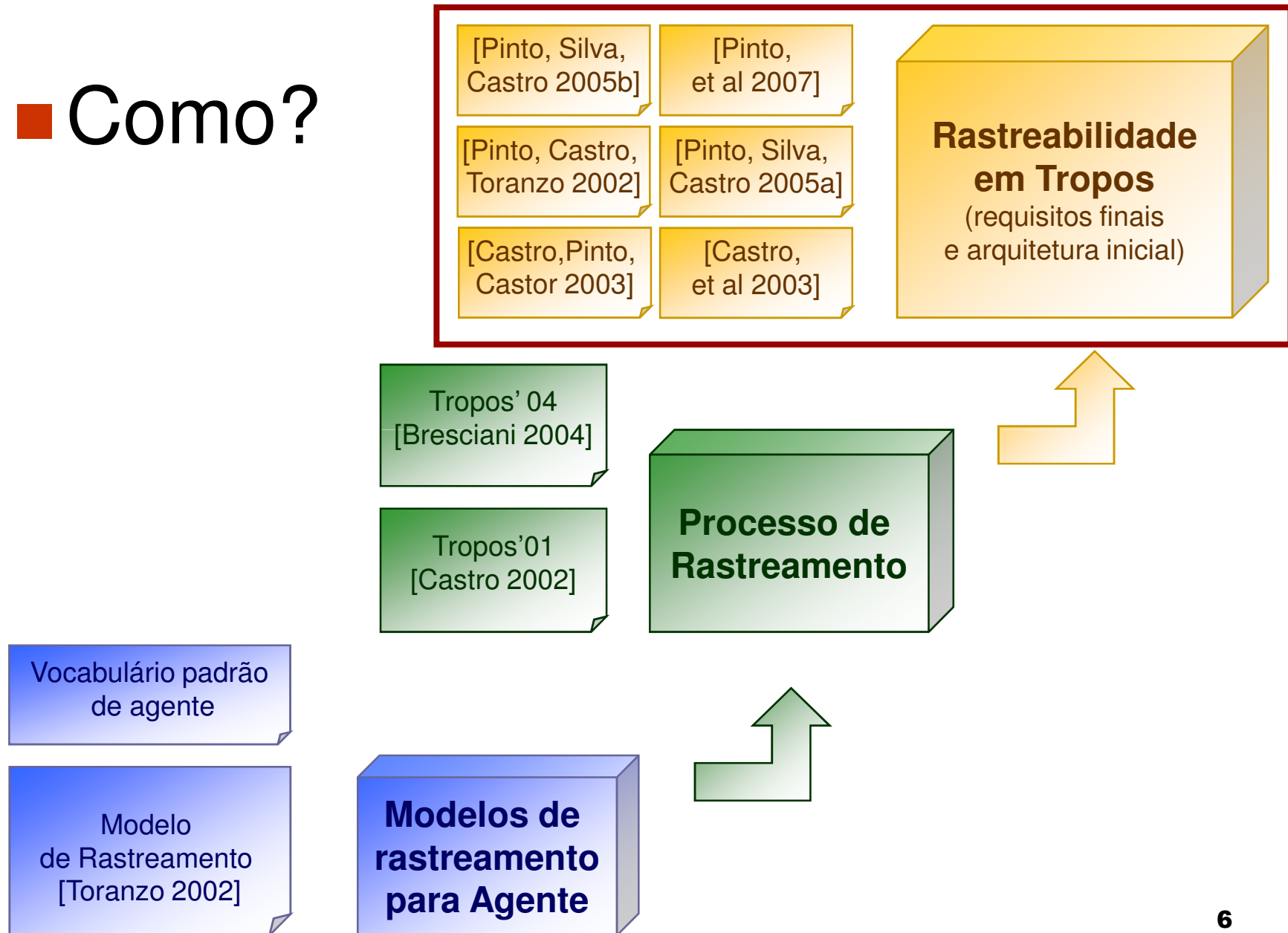
■ TROPOS

- Um **framework** usado na modelagem sistemas multiagentes e **centrado em requisitos**;
- Faz uso das abstrações e conceitos das disciplinas **organizacional** e **social** para entender, modelar, analisar e projetar;
- Área de Aplicação
 - Desenvolvimento de software centrado em requisitos
- Fases contempladas pela metodologia:



Motivação

■ Como?



Escopo da Proposta

■ Agentes

- Organizações compostas de agentes cognitivos
- Entendimento da autonomia e cooperação entre agentes
- Ambiente fechado



Abordagem Sugerida: Criação de Modelos Específicos

- Metamodelo de rastreamento
- Modelo de referência de rastreamento
 - Tabelas e Matrizes
- Processo de rastreamento (aplicado a TROPOS)

Rastreamento para SMA

■ Por que modelos específicos?

Abstração chave envolvida no paradigma orientado a agente difere do desenvolvimento tradicional de software :

- Capturar crenças, intenções, planos
- Capturar organizações, papéis
- Modelo mais rico de dependências.

Rastreamento para SMA

■ Auxilia a responder:

(para cada agente)

- Quais são os papéis que ele depende?
- Quais são os planos executados para alcançar suas metas?
- Quais são as metas ele é responsável?
- Quais são os recursos estão disponíveis para ele e quais são os de sua responsabilidade?
- Etc.

Rastreamento para SMA

■ **Metamodelo de rastreamento**

- Descrever relacionamentos encontrados comumente em desenvolvimento Orientado a Agente

■ **Modelo de referência de rastreamento**

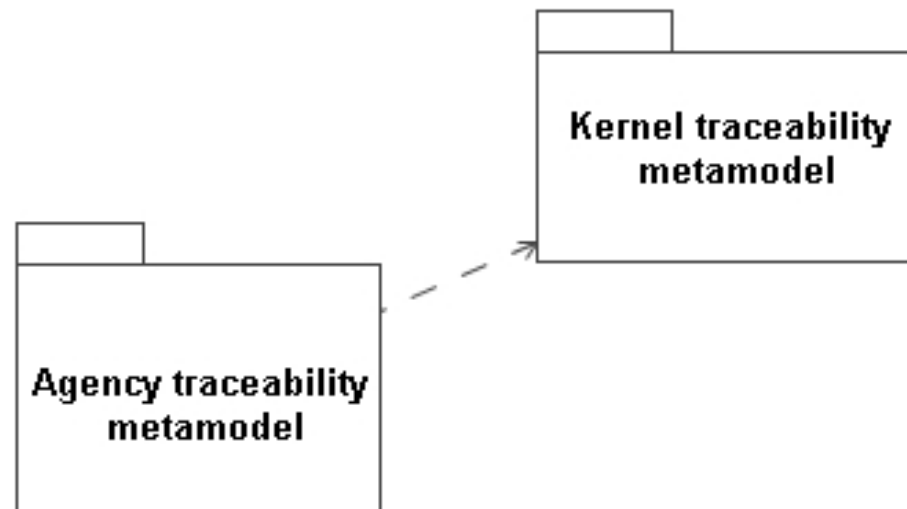
- Introduzir elementos específicos a SMA
 - Capacidades dos agentes
 - Aspectos sociais
- Auxilia desenvolvedores a encontrarem:
 - Elementos a serem rastreados
 - Relações entre esses eles

■ **Processo de rastreamento aplicado a Tropos**

- Guia a criação das matrizes de rastreamento em conexão com Tropos.

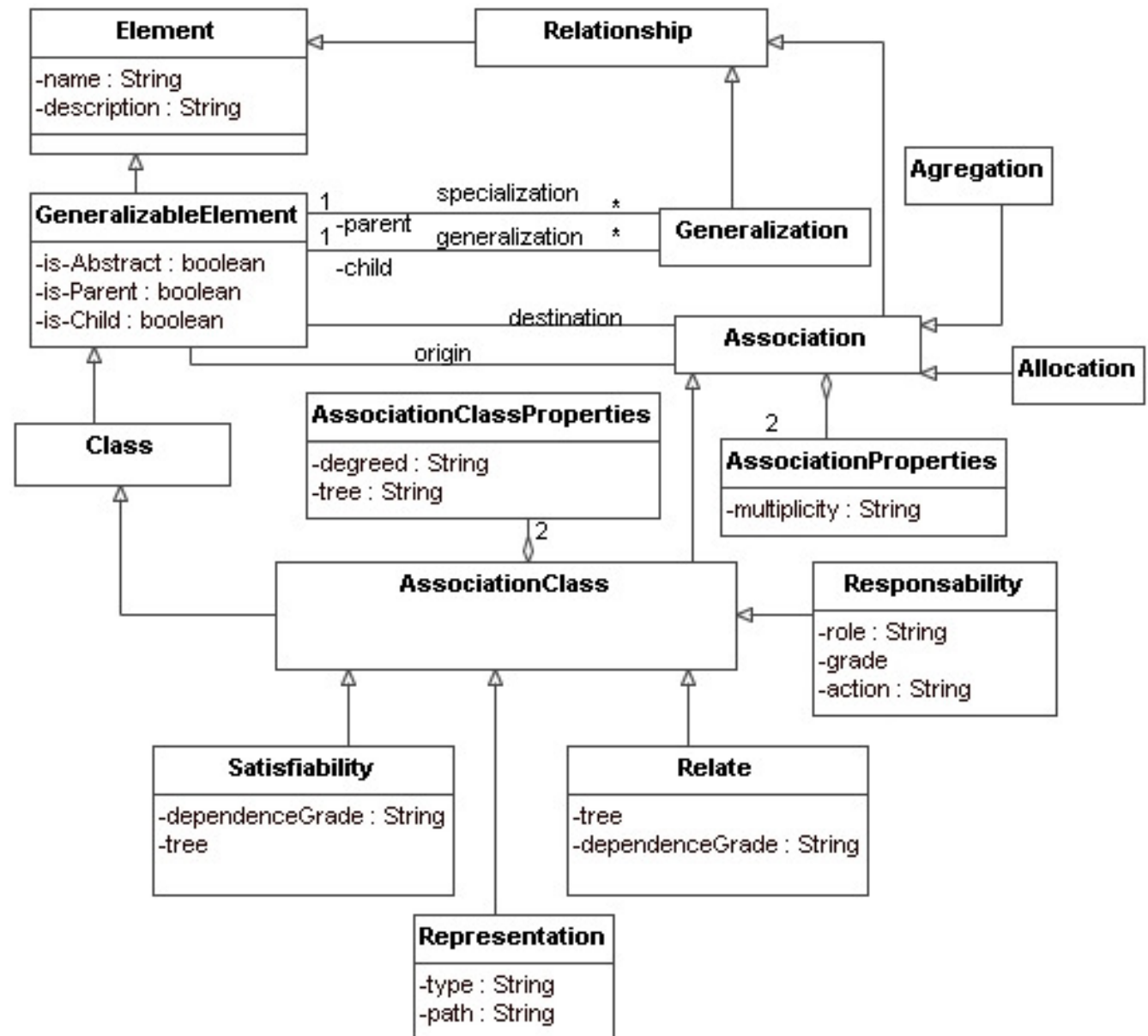
Metamodelo de Rastreamento

- Pacotes
 - Kernel traceability metamodel
 - Modelo de Toranzo
 - Agency traceability metamodel
 - Extensão a desenvolvimento orientado a agente



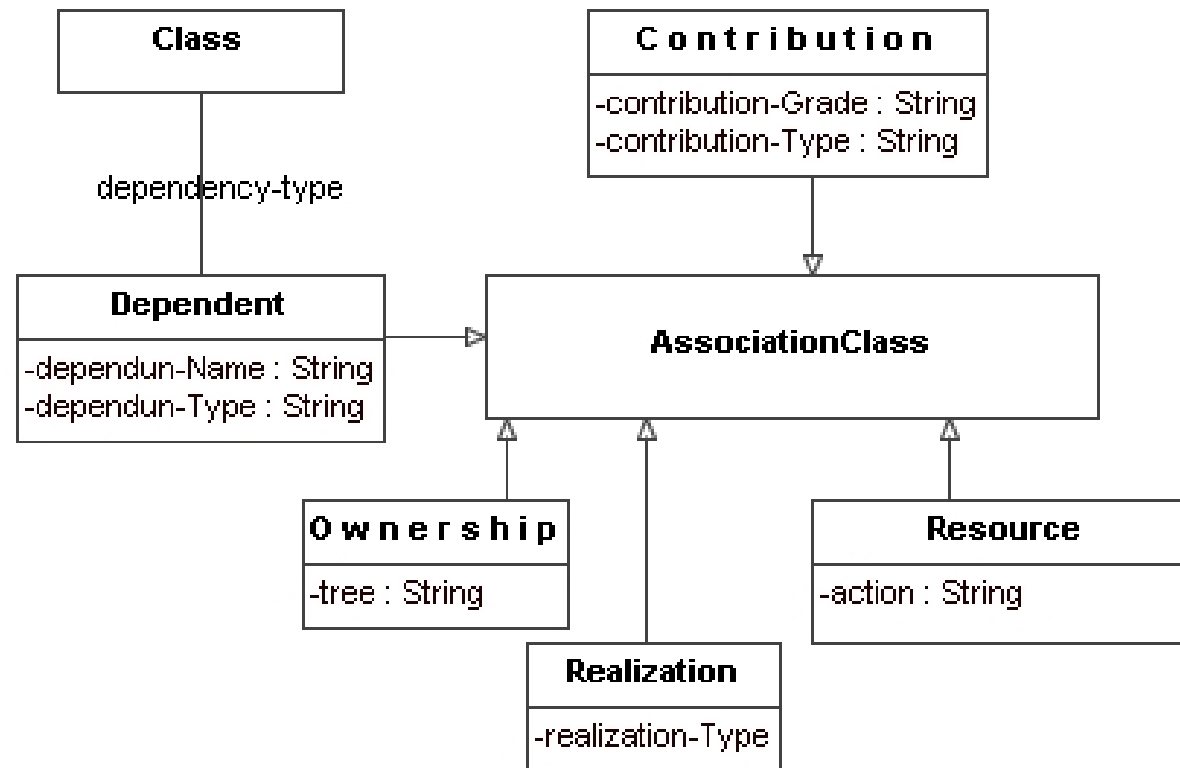
Metamodelo de Rastreamento

- Kernel traceability metamodel



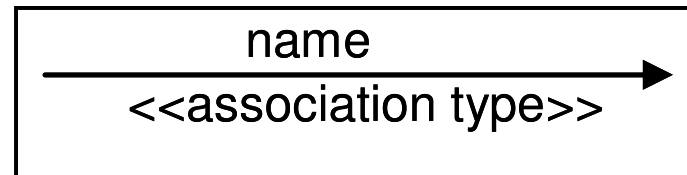
Metamodelo de Rastreamento

■ Agency traceability metamodel

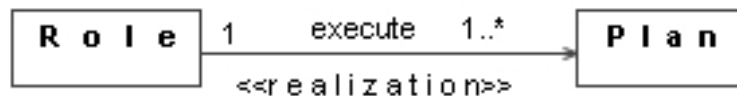


Agency traceability metamodel

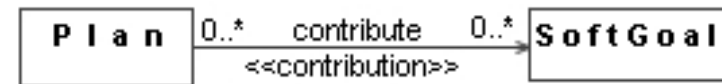
■ Relações Binárias



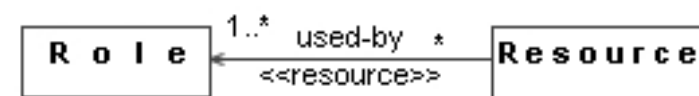
Ownership relation



Realization relation



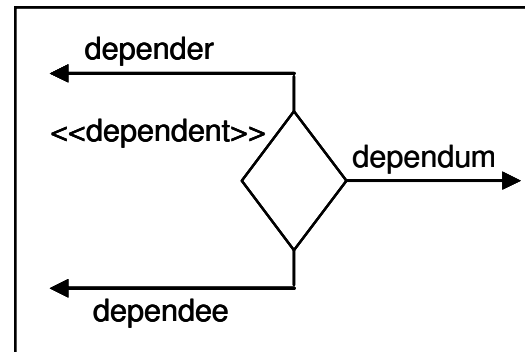
Contribution relation



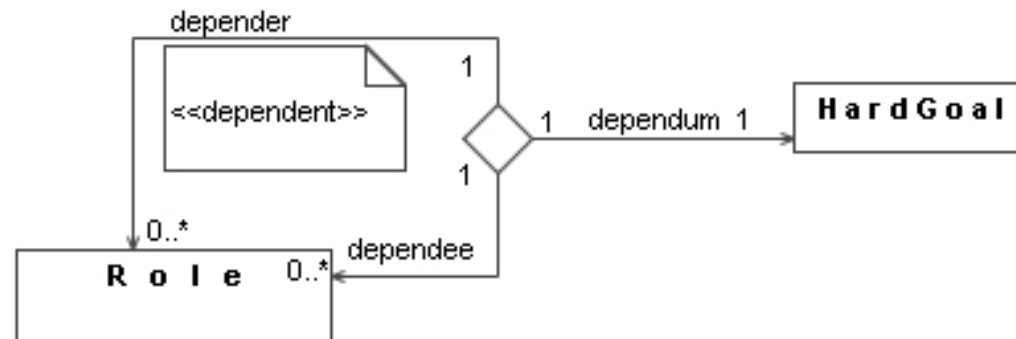
Resource relation

Agency traceability metamodel

■ Relação Ternária



Concrete syntax for dependent association

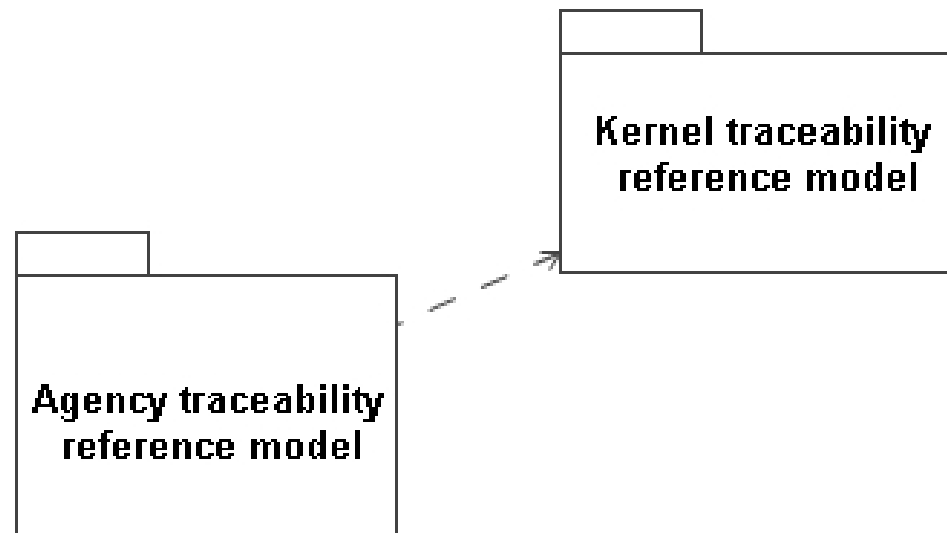


Example dependent association type.

Modelo de Referência de Rastreamento

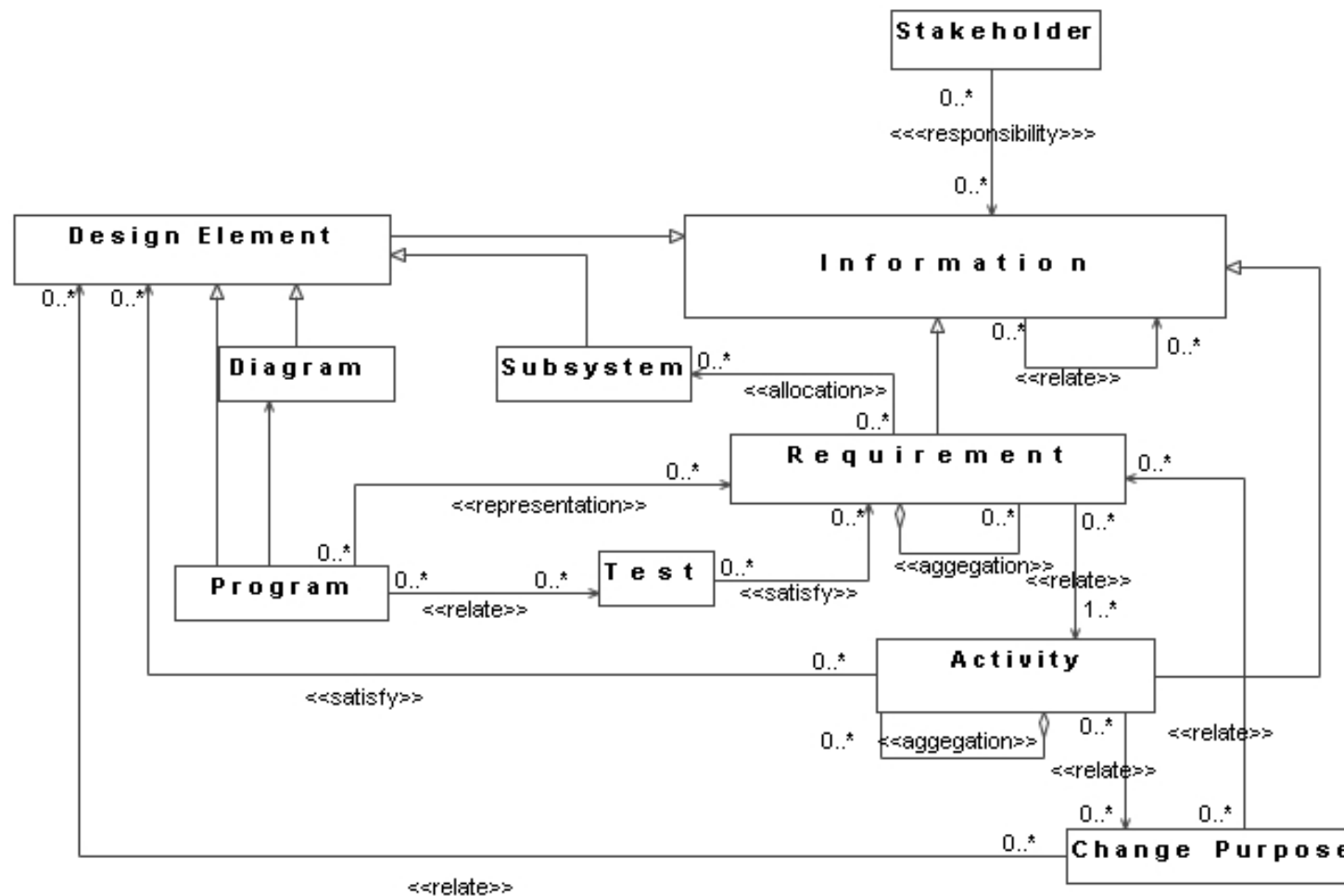
■ Pacotes

- Kernel traceability reference model
 - Modelo de Toranzo
- Agency traceability reference model
 - Extensão necessária ao desenvolvimento orientado a agente



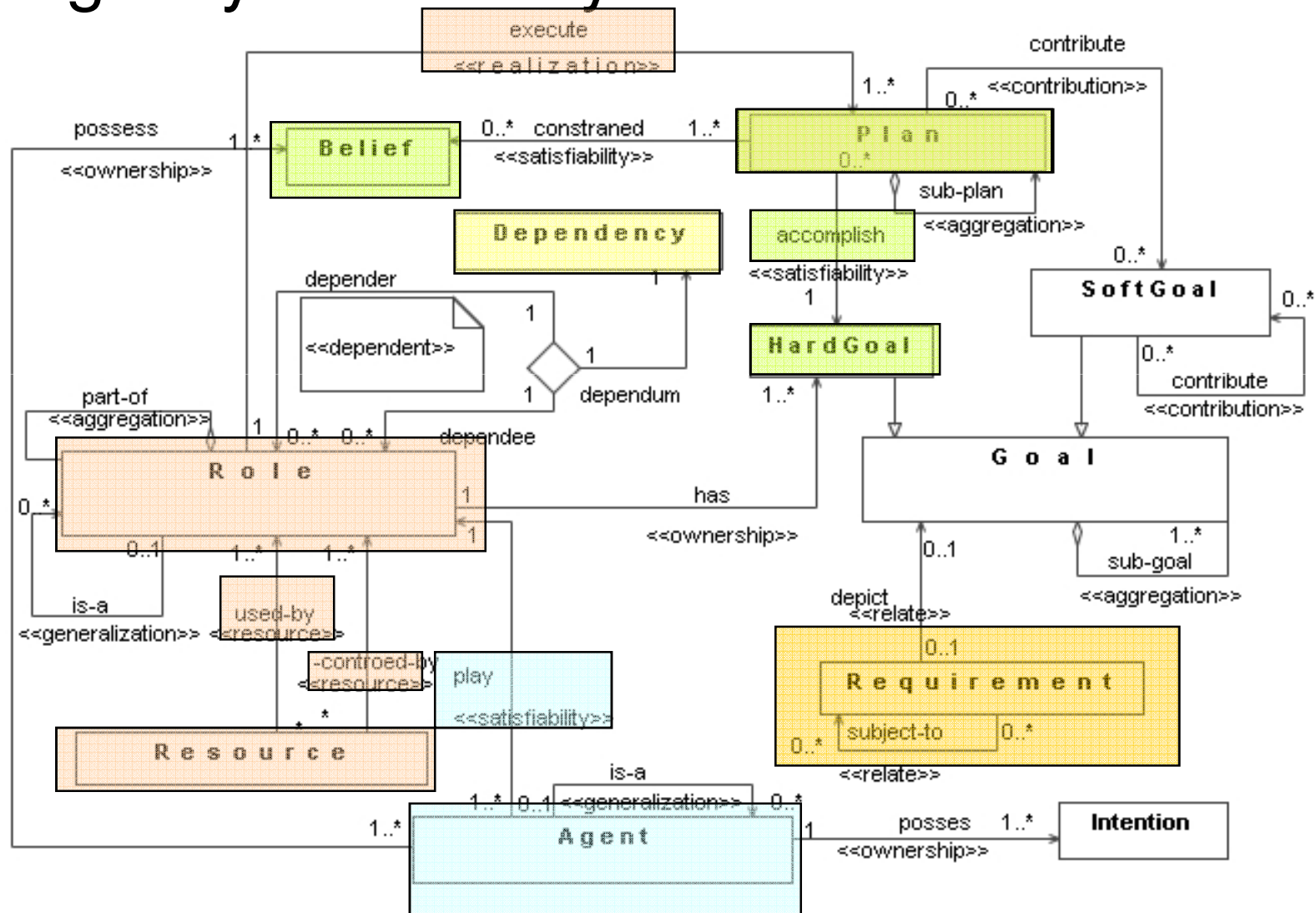
Modelo de Referência de Rastreamento

■ Kernel traceability reference model



Modelo de Referência de Rastreamento

■ Agency traceability reference model



Tabelas e Matrizes

■ Tabelas

□ Estrutura de tabela para Classe

- Usada para representar as classes envolvidas no relacionamento

ClassName: SubClassName	
ID	Instance

■ Exemplo:

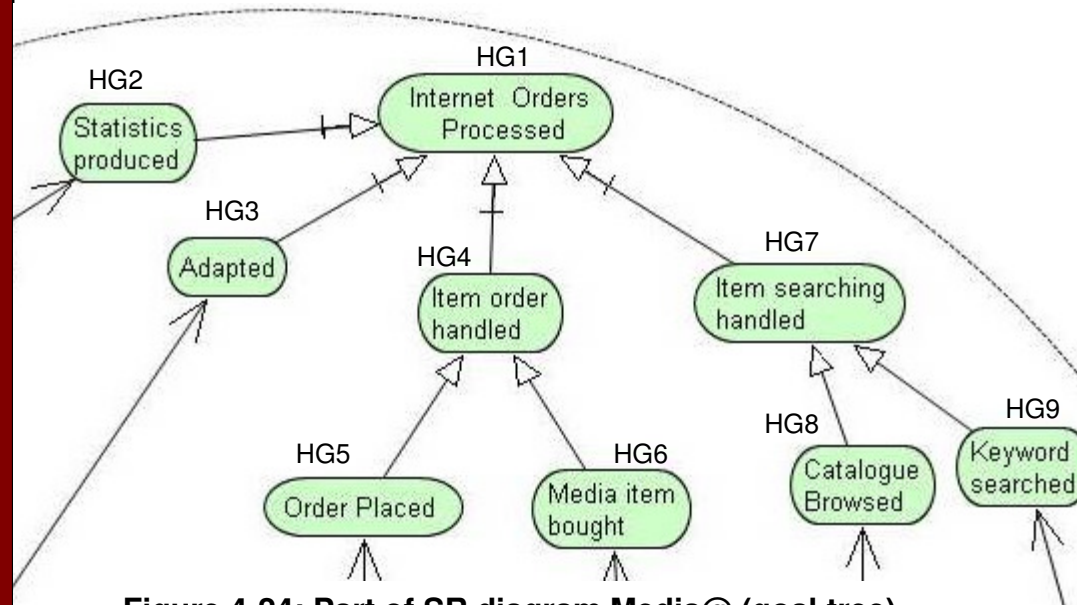


Figure 4-24: Part of SR diagram Media@ (goal tree)

GOAL: Hard-goal	
ID	Instance
HG1	Internet orders process
HG2	Statistics produces
HG3	Adapted
HG4	Item order handeled
HG5	Order placed
HG6	Media item bought
HG7	Item searching handeled
HG8	Catalogue Browsed
HG9	Keyword searched

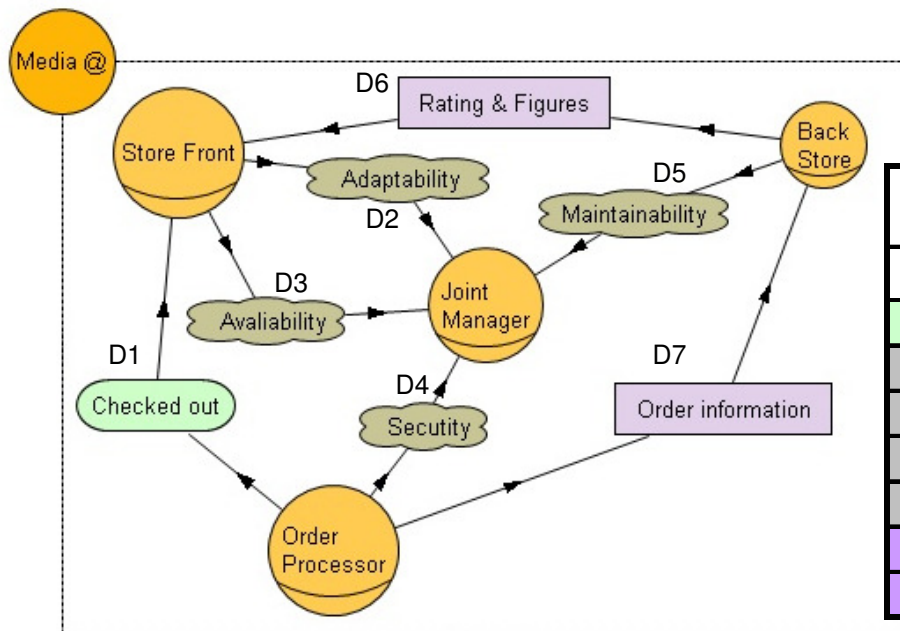
Tabelas e Matrizes

■ Tabelas

- Estrutura de tabela para Dependência
 - Usada para representar a classe dependência

Dependency		
ID	Instance	Type

■ Exemplo:



DEPENDENCY		
ID	Instance	TYPE DEPENDENCE
D1	Checked out	Goal
D2	Adaptability	Soft-goal
D3	Avaliability	Soft-goal
D4	Security	Soft-goal
D5	Maintainability	Soft-goal
D6	Rating & Figures	Resource
D7	Order information	Resource

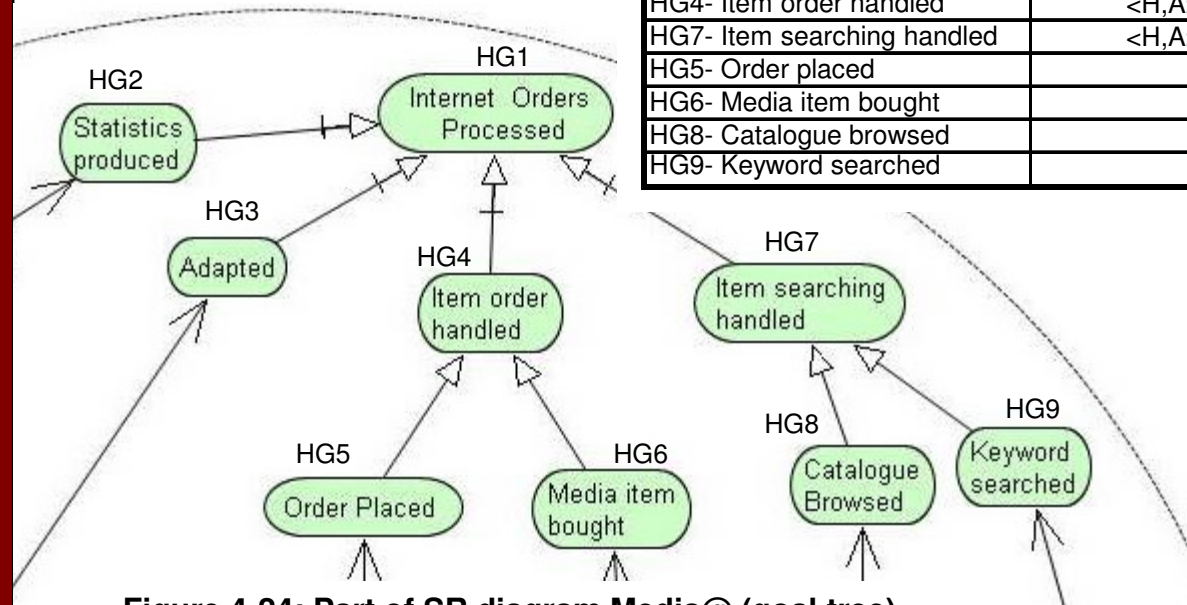
Figure 4-3: SD Diagram of the Architectural model for Medi@

Tabelas e Matrizes

- Matriz de associação
 - Usada para representar as associações binárias

Relation-Name <<association type>> →	Instance of destination class
Instance of origem class	<tuple value>

- Exemplo:



Sub-goal <<agregation>> ←	HG1-Internet orders processed	HG4- Item order handled	HG7- Item searching handled
HG2- Statistics produces	<L,A>		
HG3- Adapted	<M,A>		
HG4- Item order handled	<H,A>		
HG7- Item searching handled	<H,A>		
HG5- Order placed		<H,A>	
HG6- Media item bought		<H,A>	
HG8- Catalogue browsed			<H,A>
HG9- Keyword searched			<H,A>

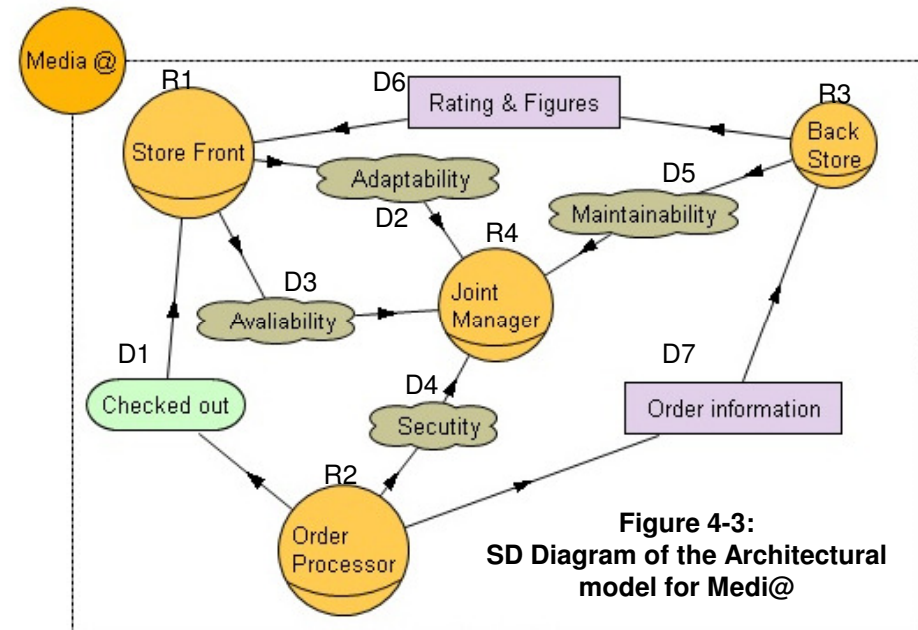
Figure 4-24: Part of SR diagram Media@ (goal tree)

Tabelas e Matrizes

- Matriz de dependência
 - Usada para representar a associação ternária (<<dependent>>)

Collaboration <<dependent>> →	Instance of dependee class
Instance of dependent class	{<dependum nome , dependum type>}

- Exemplo:



Collaboration <<dependent>> →	R1- Store Front	R2- Order Processor	R3- Back Store	R4- Joint manager
R1- Store Front				<D2- Adaptability, soft-goal> <D3- Avaliability, soft-goal>
R2- Order Processor	<D1- Checked out, hard-goal>		<D7- Order information, resource>	<D4- Security, soft-goal>
R3- Back Store	<D6- Rating & Figures, resource>			<D5- Maintainability, soft-goal>
R4- Joint manager				

O Processo de Rastreamento

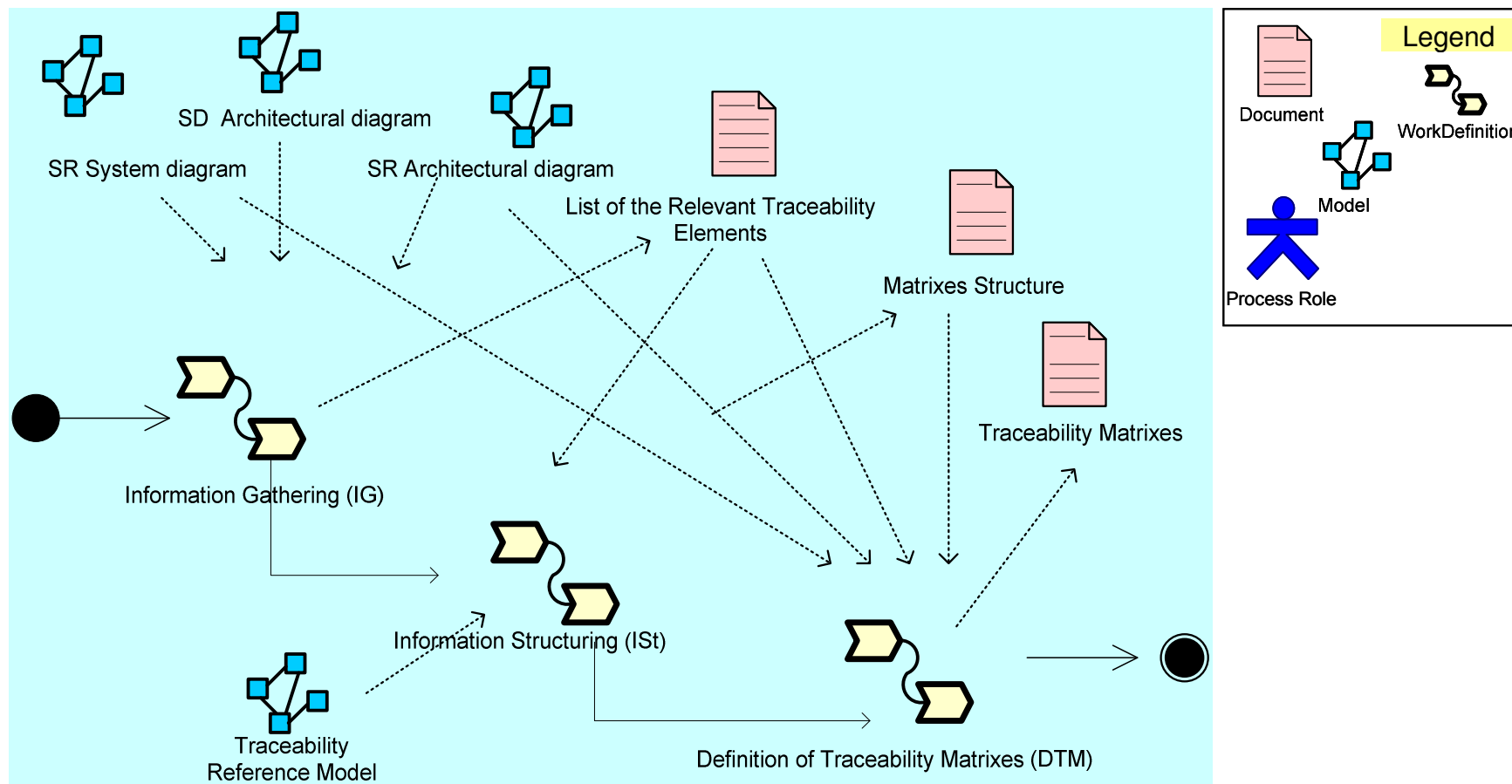


Figure 4-29: WorkDefinitions Flow for traceability process

O Processo de Rastreamento

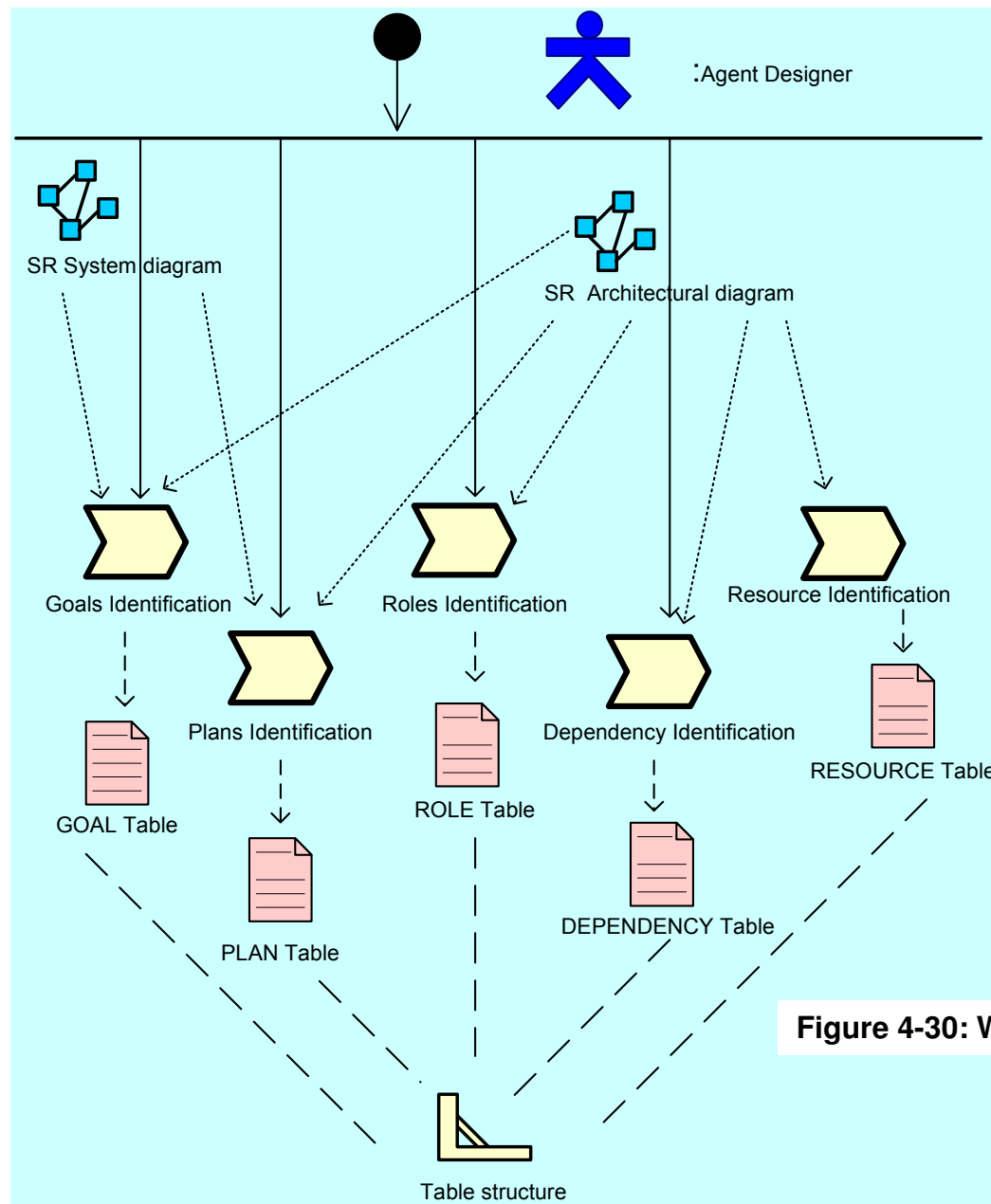


Figure 4-30: Workflow for Information Gathering (IG)

O Processo de Rastreamento

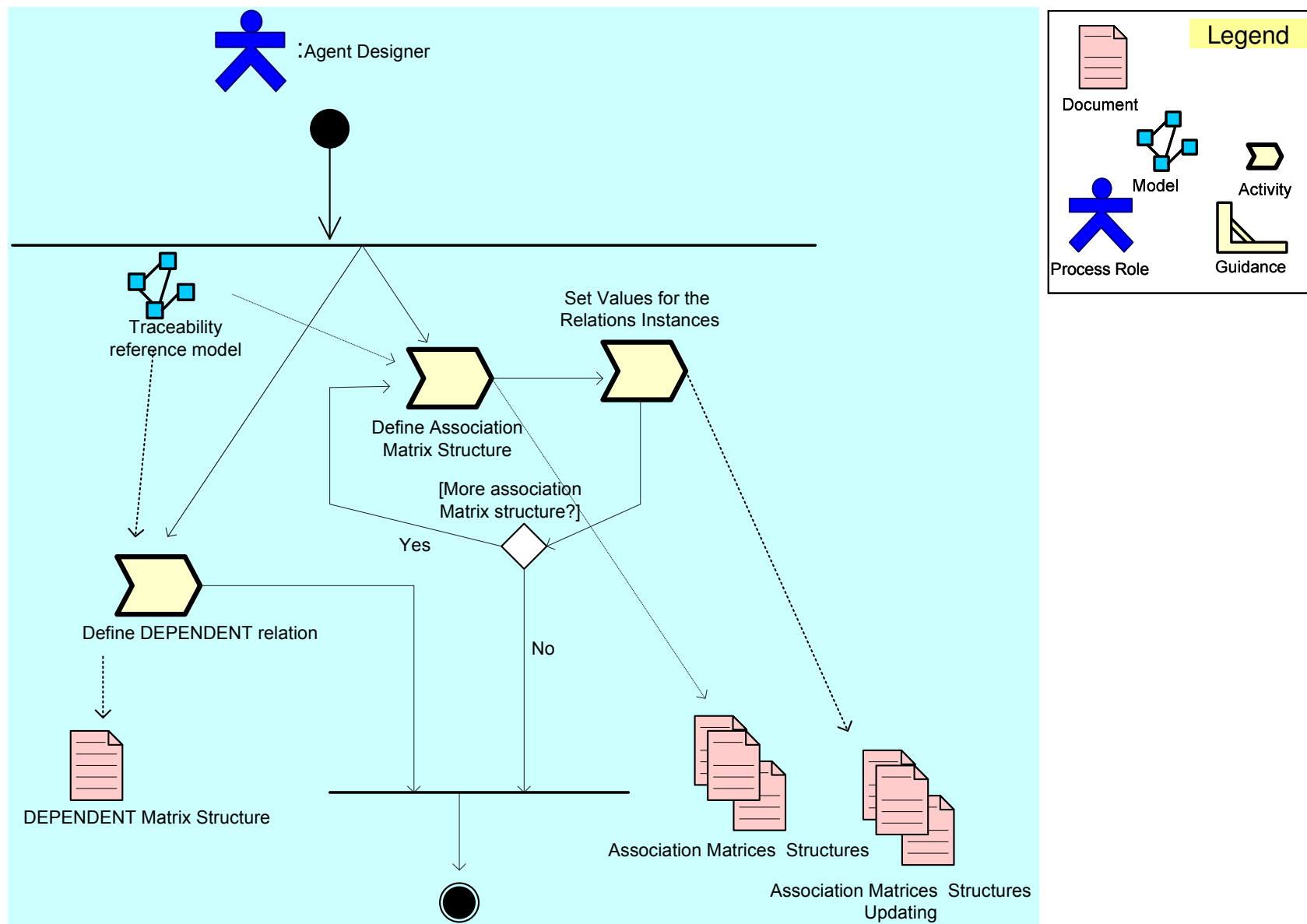


Figure 4-32: Workflow for Information Structuring (Ist)

O Processo de Rastreamento

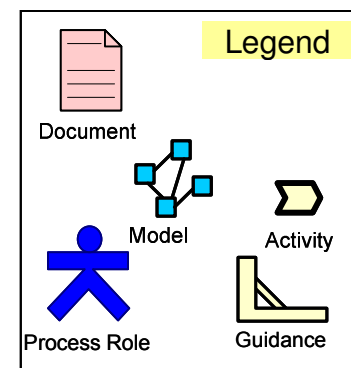
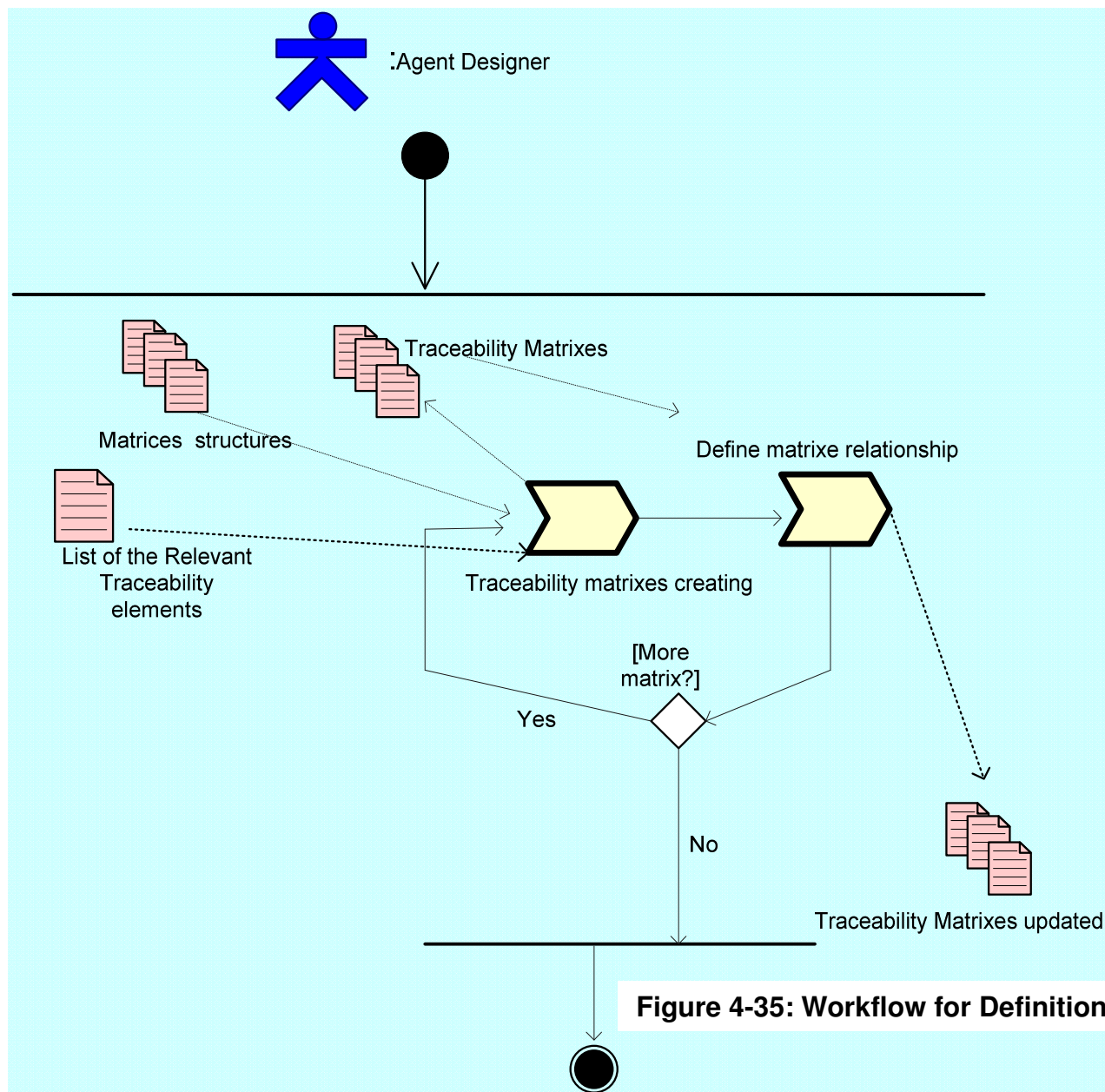


Figure 4-35: Workflow for Definition of Traceability Matrixes (DTM)



Exemplo DBSitter-AS

- DBSitter-AS
- Projeto abstrato
- Aplicação do processo de rastreamento

DBSitter-AS

■ O que é?

É um SMA cujo objetivo é fornecer autonomia ao tradicional SGBD.

- **DBSitter-As** é apresentado como um framework arquitetural para componentes de resolução de falhas autônomos
- Atua como um administrador de banco de dados eletrônico
- Modelado usando a metáfora organizacional
 - **Detecting and solving faults**
 - Notifications and records
 - Knowledge and suggestion

DBSitter-AS: Projeto Abstrato

- Objetivo: descrever como o projetista pode identificar:
 1. Quais são os papéis que o coordenador DBSitter depende para alcançar suas metas
 2. Quais são os planos que o coordenador DBSitter executa na resolução de falhas

DBSitter-AS: Projeto Abstrato

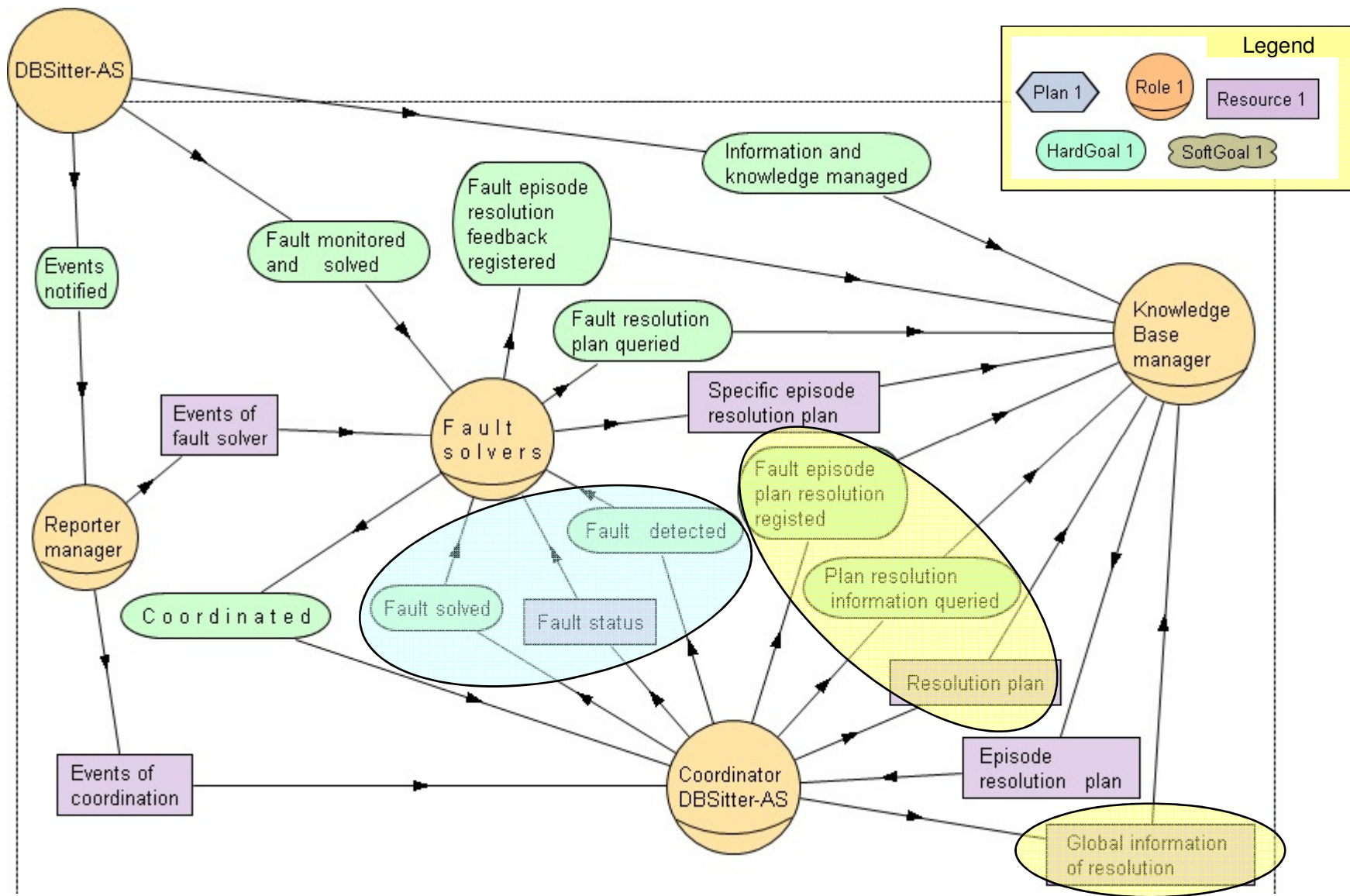


Figure 5-4: Architectural diagram for DBSitter-AS

DBSitter-AS: Aplicando o Processo

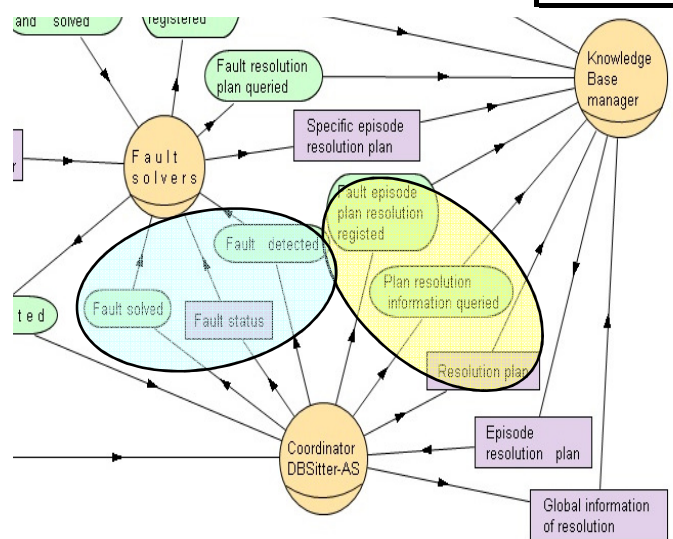
Role		Diagram
ID	Instance	
R1	Coordinator DBSitter-AS	
R2	Reporter manager	
R3	Knowledge base manager	
R4	Fault solvers	

Figure 5-4

DEPENDENCY		
ID	Instance	Type Dependency
D1	Events of fault solver	Resource
D2	Events of coordination	Resource
D3	Coordinate	Goal
D33	Fault monitored and solved	
D4	Fault solved	
D5	Fault status	
D6	Fault detected	
D7	Fault episode resolution feedback	
D8	Fault resolution plan queried	
D9	Specific episode resolution plan	

Collaboration <<dependent>> →	Instance of dependee class
Instance of dependent class	{<dependum nome , dependum type>}

Collaboration <<dependent>> →	R4- Fault solvers	R3- Knowledge base manager
R1- Coordinator DBSitter-AS	<D4-Fault solved, goal> <D5-Fault status,resource> <D6-Fault detected,goal>	<D10-Fault episode resolution plan registered, goal> <D11-Plan resolution information queried, goal> <D12- Resolution plan, resource> <D14- Global information of resolution,resource>



1. Quais são os papéis que o coordenador DBSitter depende para alcançar suas metas



Conclusões

- Considerações
- Análise comparativa
- Contribuições
- Trabalhos futuros

Considerações

- Rastreabilidade habilita desenvolvedores seguir rastros entre requisitos e os respectivos artefatos, entendendo melhor SMA
- Definimos um vocabulário de agente para se usado no contexto de desenvolvimento de software
- DBSitter foi usado como um exemplo para demonstrar a utilidade da nossa proposta
- Concluimos que ela captura elementos que auxilia o entendimento de autonomia e cooperação dos agentes

Análise comparativa

	Nossa proposta	Proposta de Toranzo
Requisitos são classificados explicitamente	√	x
Rastreamento da análise means-end de softgoal	√	x
Complexidade de papel pode ser abstraída	√	x
Dependência entre papéis expressa explicitamente	√	x

Contribuição

- Definição de relações
 - **Metamodelo** de rastreabilidade de agente
 - **Modelo de referência** de rastreabilidade de agente
- Introduzir rastreabilidade nas fases de requisito final e arquitetura inicial do Tropos
 - Um **processo** para construção das matrizes de rastreamento durante o desenvolvimento de software.

Trabalhos Futuros

- Aplicar o processo de rastreabilidade as demais fases de Tropos
- Conduzir mais estudos de casos para melhor avaliar a nossa proposta
- Aumentar a “granularidade” de outros aspectos de rastreamento em agente. Por exemplo, qual ação específica do plano é dependente do outro agente.
- Aplicar a nossa proposta a outras metodologias orientadas a agentes (Ex. Gaia)
- Desenvolver uma ferramenta CASE para suportar a construção das matrizes e o processo de rastreamento

Além dos trabalhos futuros, planejamos estender a nossa proposta com a intenção de:

- Considerar outros tipos de agente (ex. Agentes de aprendizagem)
- Permitir ser aplicada no contexto de sistemas abertos.

Improving Traceability in Agent Oriented Development

Rosa Candida Cavalcanti Pinto

Supervisor: Dr. Jaelson Castro

Co-supervisor: Dr. Patrícia Tedesco



Submitted in partial fulfilment of requirements of
Degree of Doctor of Software Engineering

DBSitter-AS: Projeto Abstrato

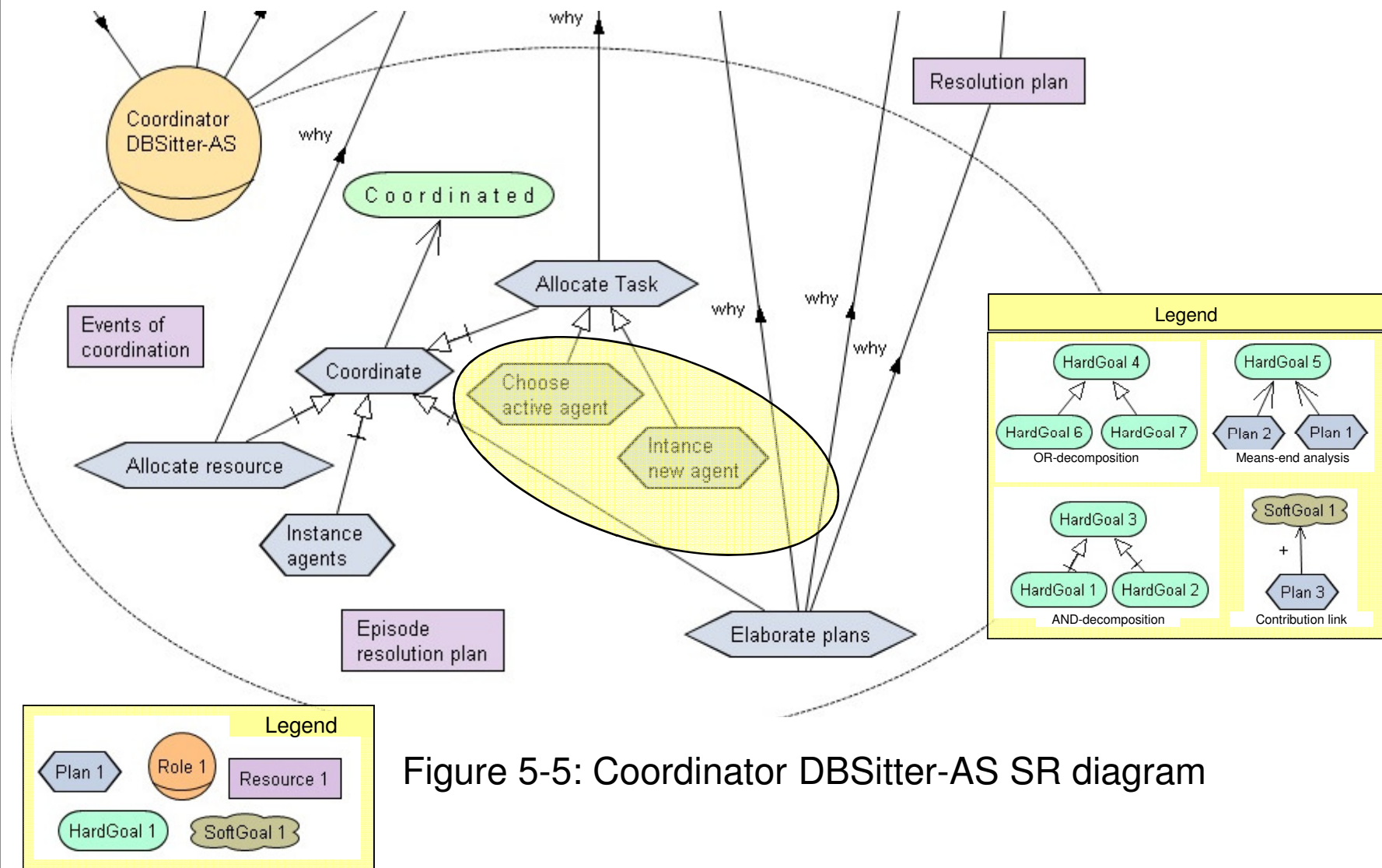


Figure 5-5: Coordinator DBSitter-AS SR diagram

DBSitter-AS: Aplicando o Processo

■ Definition of Traceability Matrixes (DTM)

Has <<ownership>> →	HG5-Cordinated
R1-Coordinator DBSitter-AS	<A>

Figure 5-18: Has matrix

Acomplish <<satisfy>> →	HG5- Cordinated
PL1- Coordinate	<H,O>

Figure 5-19: Accomplish matrix

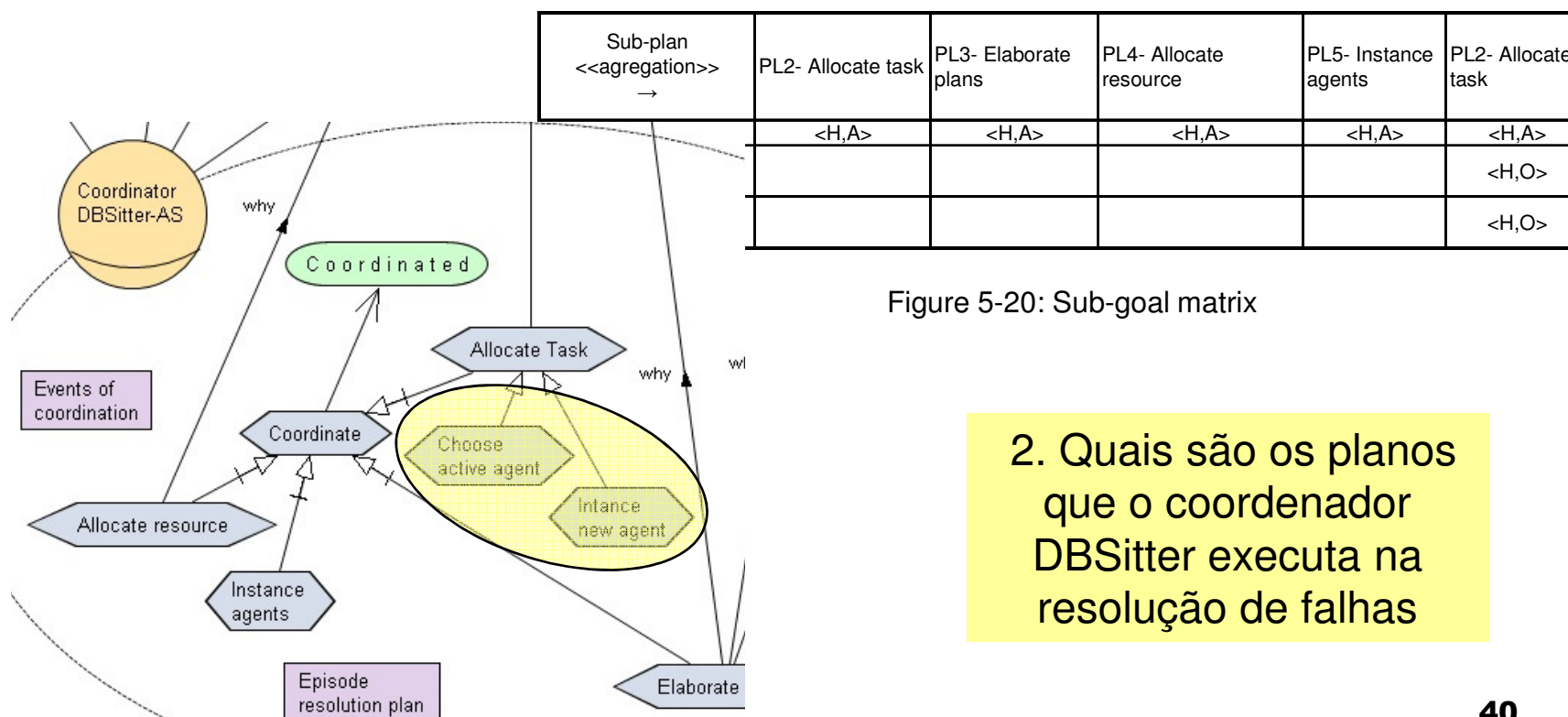


Figure 5-5: Coordinator DBSitter-AS SR diagram

Figure 5-20: Sub-goal matrix

2. Quais são os planos que o coordenador DBSitter executa na resolução de falhas

Escopo da Proposta

■ Rastreabilidade

□ Relações relacionadas a produtos

- Relacionamento entre os artefatos independente de como foram criados

□ Relações relacionadas a processo

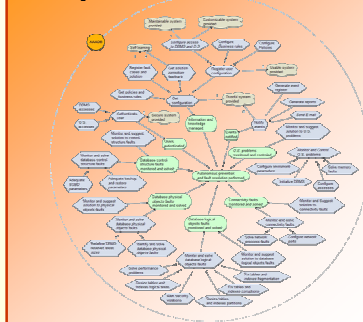
- Só são capturados quando se tem a história das ações que a construíram pegadas no processo de desenvolvimento
 - Quem criou?
 - Quem modificou?
 - Por que?

Escopo da Proposta

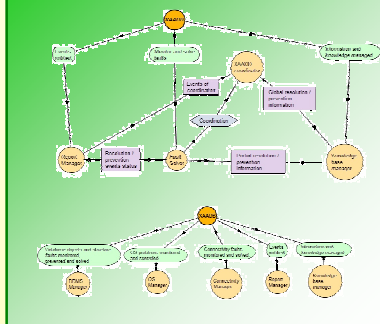
Requisitos Iniciais

TROPOS

Requisitos Finais



Projeto de arquitetura

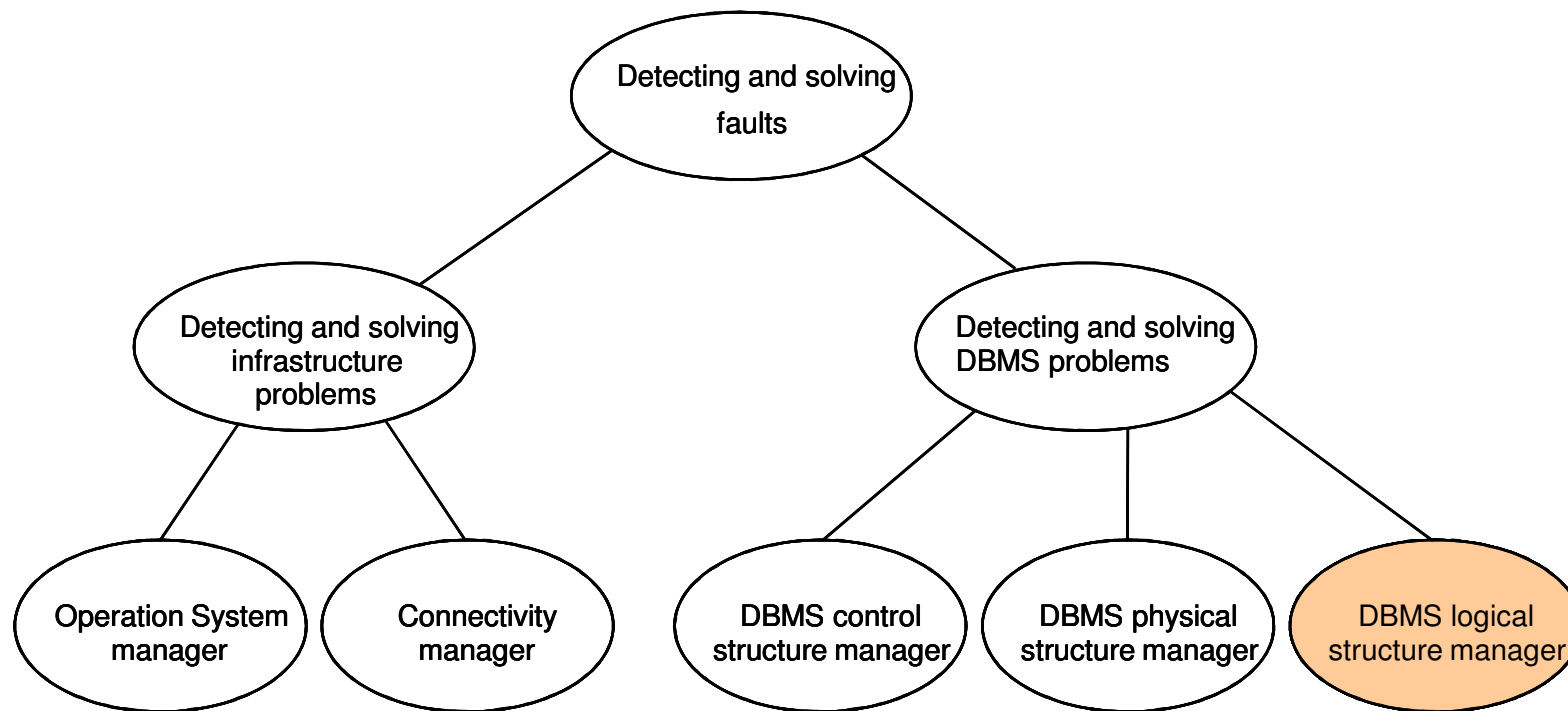


Projeto Detalhado

Implementação

DBSitter-AS

■ Estrutura Organizacional “Detecting and solving faults”



DBSitter-AS

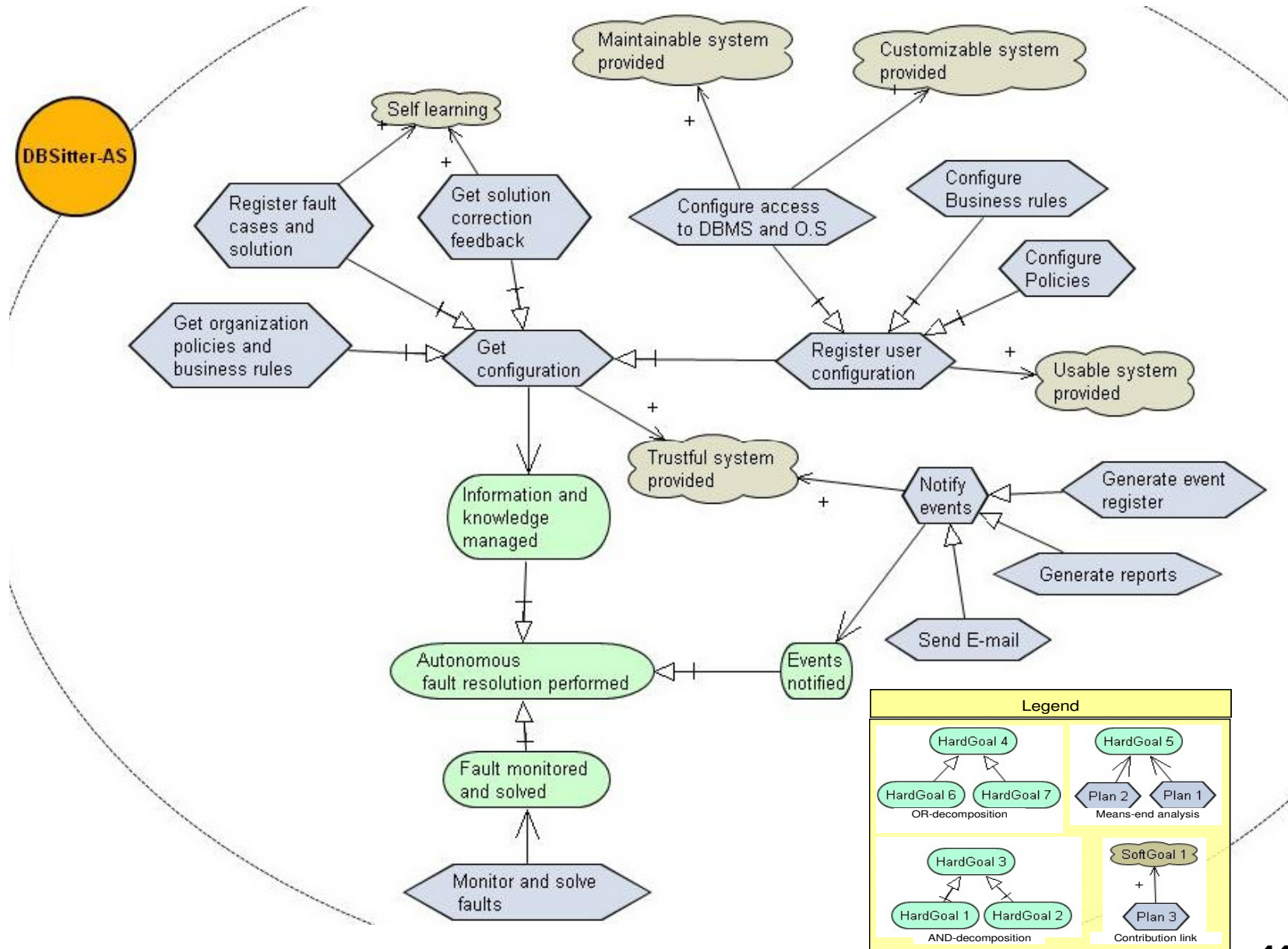


Figure 5-3: Strategic rationale model (SR) for the DBSitter-AS

DBSitter-AS: Projeto Abstrato

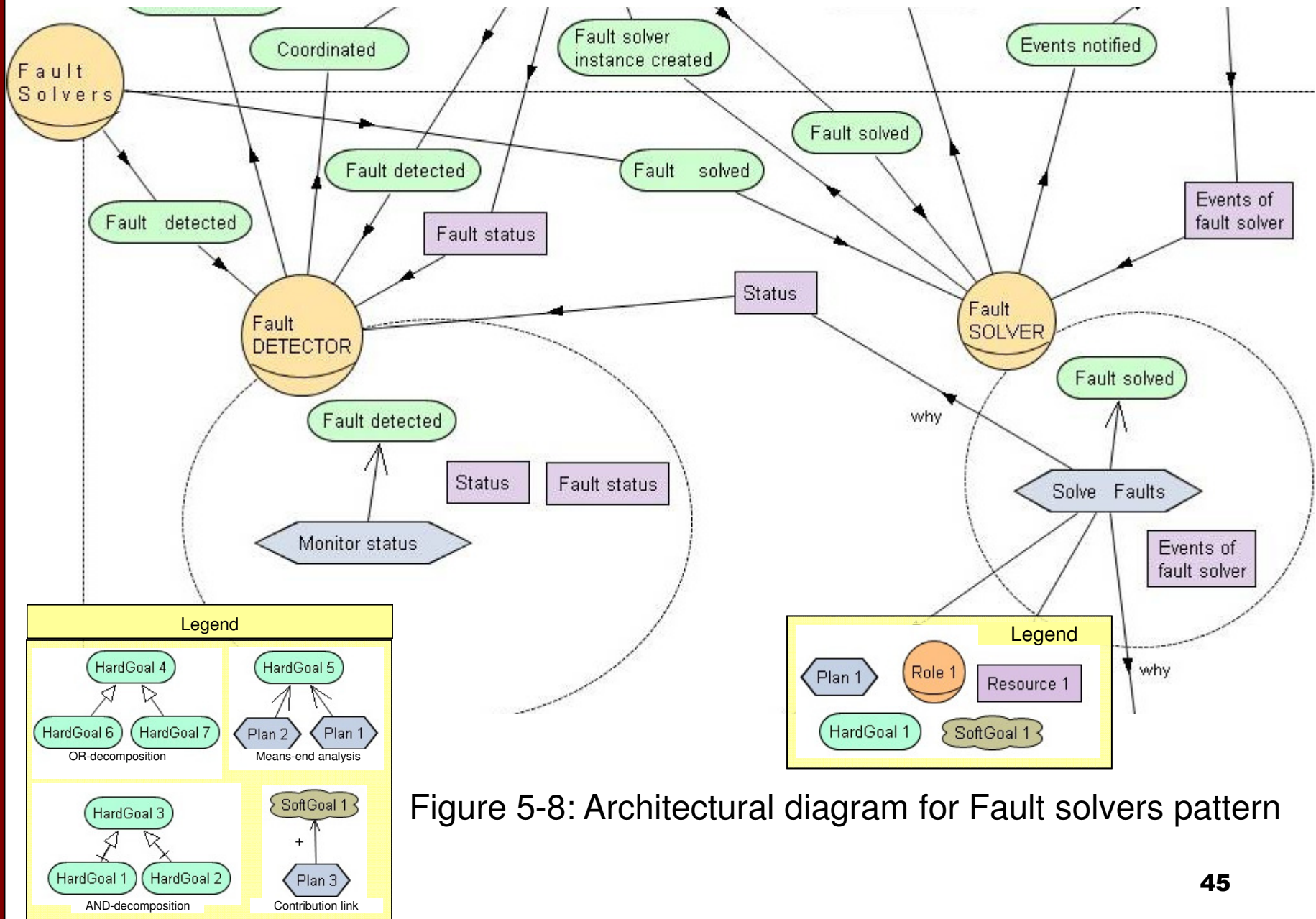


Figure 5-8: Architectural diagram for Fault solvers pattern

DBSitter-AS: Projeto Abstrato

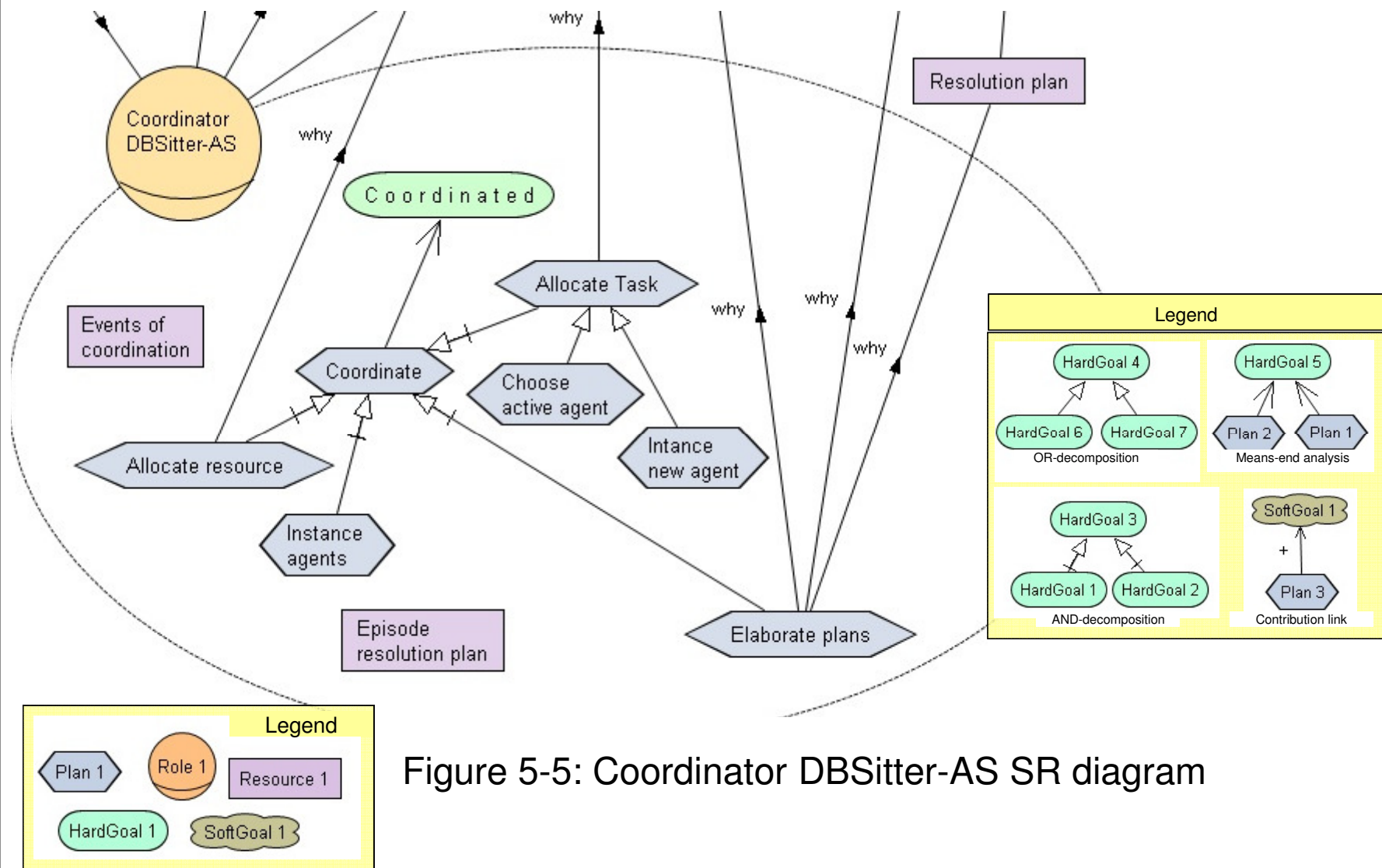


Figure 5-5: Coordinator DBSitter-AS SR diagram

DBSitter-AS: Aplicando o Processo

■ Etapas do Processo:

□ Information Gathering (IG)

- Identificação do elementos de agente
- Armazenada em estrutura de tabelas

□ Information Structuring (ISt)

- Identificação das relações entre os elementos de agente
- Apresentado em forma de matriz

□ Definition of Traceability Matrixes (DTM)

- Instancia as matrizes (ISt) com os elementos de agentes (IG)

DBSitter-AS: Aplicando o Processo

■ Information Gathering (IG)

□ IG1. Goal Identification

■ Diagrama SR do Sistema, e arquitetural

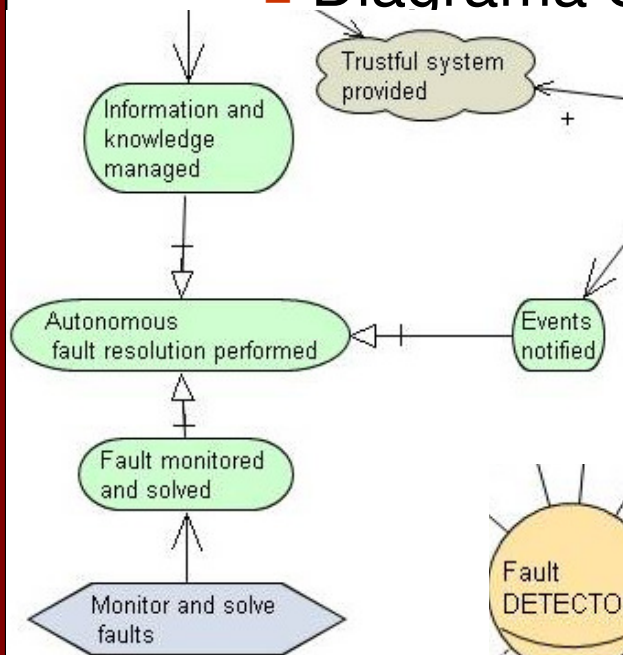


Figure 5-3: SR model (SR) for the DBSitter-AS

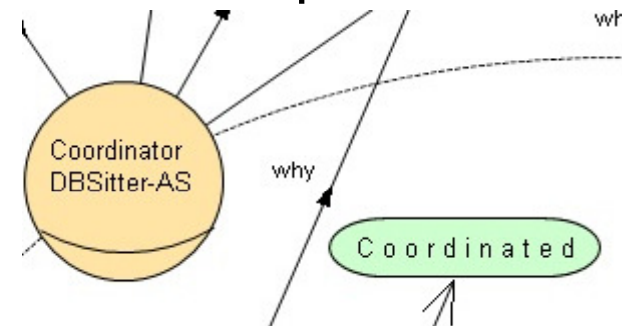


Figure 5-5: Coordinator DBSitter-AS SR diagram

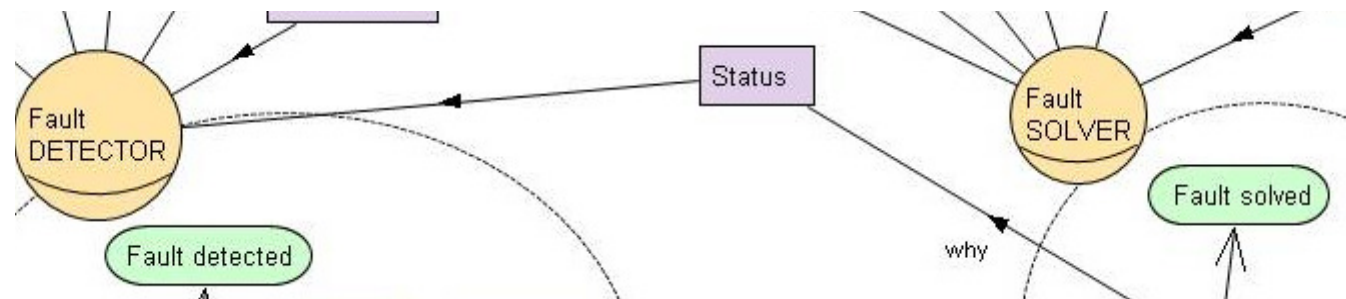


Figure 5-8: Architectural diagram for Fault solvers pattern

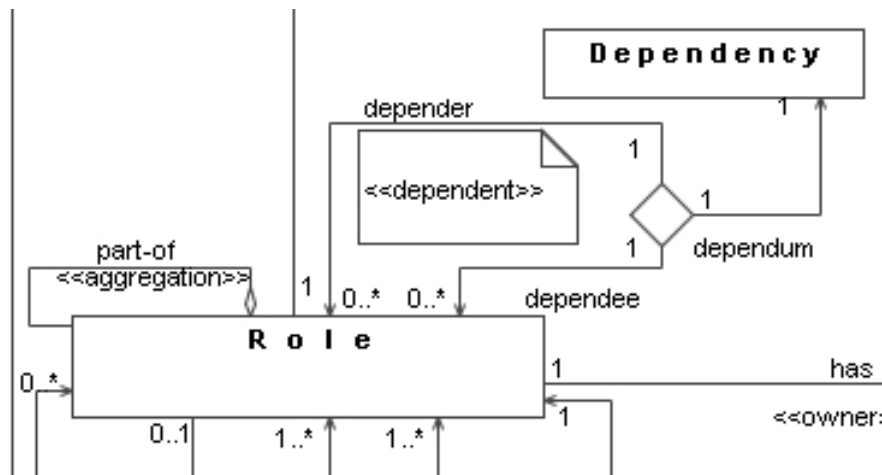
DBSitter-AS: Aplicando o Processo

- Information Gathering (IG)
 - IG1. Goal Identification
 - Armazenamento

GOAL: Hard-goal		Diagram
ID	Instance	
HG1	Autonomous fault resolution performed	Figure 5-3
HG2	Fault monitored and solved	
HG3	Events notified	
HG4	Information and knowledge managed	
HG5	Cordinated	Figure 5-5
HG6	Fault solved	Figure 5-8
HG7	Fault detected	

DBSitter-AS: Aplicando o Processo

- Information Structuring (ISt)
 - ISt1. Define Dependent Relation



Relação <<dependent>>
Parte do Modelo de referência

Collaboration <<dependent>> →	Instance of dependee class
Instance of dependent class	{<dependum nome , dependum type>}

Estrutura da Matriz Colaboração

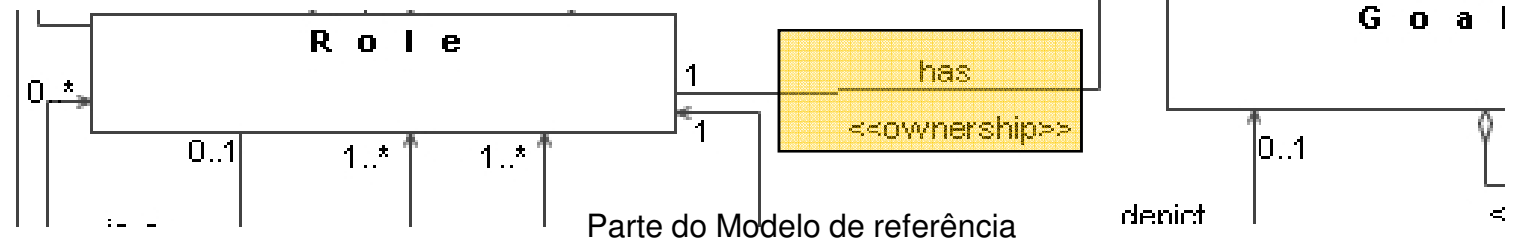
DBSitter-AS: Aplicando o Processo

- Information Structuring (ISt)
 - ISt2. Define Association Matrix Structure
 - IST3. Set Values for the Relations Instances

Has <<ownership>> →	HARD-GOAL
ROLE	<A/O>

Accomplish <<satisfiability>> →	HARD-GOAL
PLAN	<H/M/L,A/O>

Sub-Plan <<aggregation>> →	PLAN
PLAN	<H/M/L, A/O>



DBSitter-AS: Aplicando o Processo

- Definition of Traceability Matrixes (DTM)
 - DTM1. Create Traceability Matrixes
 - DTM2. Define Matrix Relationship

DBSitter-AS: Aplicando o Processo

■ Definition of Traceability Matrixes (DTM)

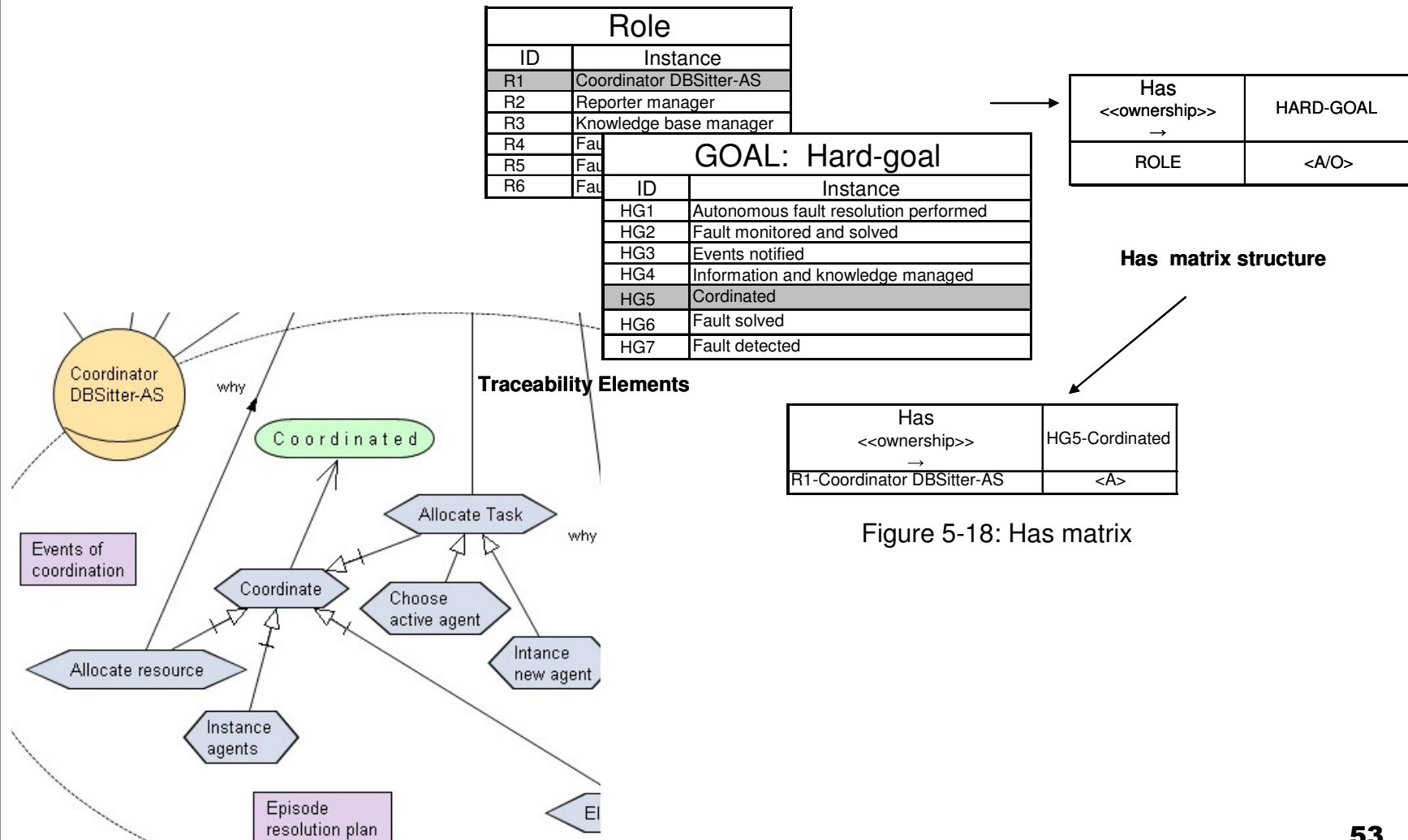


Figure 5-5: Coordinator DBSitter-AS SR diagram

Figure 5-18: Has matrix

Análise comparativa

■ Nossa proposta X Toranzo

- Requisitos são classificados explicitamente:
 - Requisitos do funcionais do usuário são descritos por hard-goal;
 - Não-funcional (soft-goal)
 - Requisitos de sistema (planos)
- Usando classificação estamos aplicando a técnica de hierarquia que auxilia a desenvolver software com qualidade
- O rastreamento de de uma hierarquia means-end de soft-goal até serem tratadas por links com planos são capturadas pela relação *contribuição*. O que facilita o entendimento da realização dos softgoals.

Análise comparativa

- Nossa proposta X Toranzo
 - A complexidade de papéis pode ser abstraída usando a relação de generalização.
 - A dependência entre papéis é expressa explicitamente podendo-se assim capturar cooperação entre agentes
 - Toranzo trata agentes como sub-sistemas
 - A estrutura representacional dos papéis para expressar o raciocínio por trás das decisões de processo pode ser rastreada, o que auxilia o entendimento de autonomia.