

**Meta-Model Sources****Gaia**

Version: March, 15, 2004

Document Authors: A. Garro, P. Turci, M.P. Huget

**Index**

<i>Note to the reader</i> .....	2
1 Introduction.....	3
2 Gaia disciplines.....	4
2.1 Requirements Capture.....	5
2.2 Analysis.....	6
2.2.1 Process roles involved.....	6
2.2.2 UML Models.....	7
2.2.3 Documents .....	7
2.3 Design .....	8
2.3.1 Process roles involved.....	8
2.3.2 UML Models.....	8
3 Gaia process description .....	10
3.1 Complete process .....	10
3.2 Phases/iterations of the process.....	11
3.2.1 The Analysis phase .....	11
3.2.2 The Design phase.....	15
4 Gaia work definitions.....	19
4.1 Analysis.....	19
4.2 Design .....	20
5 Work Products.....	21
5.1 Dependency diagram.....	21
5.2 Work products model.....	22
5.2.1 Analysis work products model.....	22
5.2.2 Design work products model.....	23
5.3 MAS Model.....	24
6 Method Fragments .....	25
6.1 The “Identify and document the interaction protocols” fragment .....	26
6.1.1 Notation.....	28
6.1.2 Deliverables .....	28
6.1.3 Preconditions .....	29
6.1.4 Relationships with the MAS meta-model.....	29
6.1.5 Guidelines .....	30
6.1.6 Glossary .....	30
6.1.7 Composition guidelines .....	31
6.1.8 Aspects of fragment .....	31
6.1.9 Relationships with other fragments .....	31
6.1.10 Summary.....	31
7 Glossary.....	32
8 Annexes.....	33
References.....	34

## **Note to the reader**

*The Methodology TC is devoted to the identification of a design methodology for Multi Agent Systems that could fit the greatest number of needs. The proposed approach is based on the method engineering that consists in the creation of a sort of meta-methodology that could be instantiated with a specific methodology for each problem [3]. More specifically the “development methodology” is built by the developer assembling method fragments from a repository of methods built up taking pieces from existing methodologies (ADELFE, AOR, Gaia, MESSAGE, PASSI, Tropos,...). To obtain this repository the first step consists in expressing all methodologies with the same notation. The notation chosen by the TC is SPEM (Software process Engineering Meta-Model) since it is a meta-model dedicated to the description of methodology process and component. In this document we express Gaia methodology[1] using SPEM and propose some method fragments that could be considered for the creation of the meta-methodology.*

*We would like to stress the fact that Gaia workproducts are not described using UML notation, instead they are expressed using schemata templates or informal textual descriptions. In order to move towards an identification of the method fragments, we believe it is important to express the models using a standard notation. For this reason, we have described the Gaia models using UML, trying not to modify the meaning and the semantics of the models.*

# 1 Introduction

Gaia is a Methodology specifically tailored to the analysis and design of agent-based systems. Before giving an overview of Gaia Methodology, it is worth commenting on the characteristics of domains for which Gaia is appropriate. Gaia is appropriate for the development of systems with the following main characteristics:

- Agents are coarse-grained computational systems, each making use of significant computational resources (think of each agent as having the resources of a Unix process).
- It is assumed that the goal is to obtain a system that maximises some global quality measure, but which may be sub-optimal from the point of view of the system components. Gaia is not intended for systems that admit the possibility of *true conflict*.
- Agents are heterogeneous, in that different agents may be implemented using different programming languages, architectures, and techniques. Gaia makes no assumptions about the delivery platform;
- The organisation structure of the system is static, in that inter-agent relationships do not change at run-time.
- The abilities of agents and the services they provide are static, in that they do not change at run-time.
- The overall system contains a comparatively small number of different agent types (less than 100).

Gaia is intended to allow an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. It is worth pointing out that Gaia authors view the *requirements capture phase* as being independent of the paradigm used for analysis and design. For this reason Gaia does not deal with the requirements capture phase but it considers the *requirements statement* as an input for the methodology. Analysis and design can be thought of as a process of developing increasingly detailed models of the system to be constructed moving from abstract to increasingly concrete concepts. Abstract entities are those used during analysis to conceptualise the system, but which do not necessarily have any direct realisation within the system. Concrete entities, in contrast, are used within the design process, and will typically have direct counterparts in the run-time system [Table 1.1].

Abstract concepts	Concrete concepts
Roles Permissions Responsibilities Protocols Activities Liveness properties Safety properties	Agent Types Services Acquaintances

**Table 1-1. Abstract and concrete concepts in Gaia**

## 2 Gaia disciplines

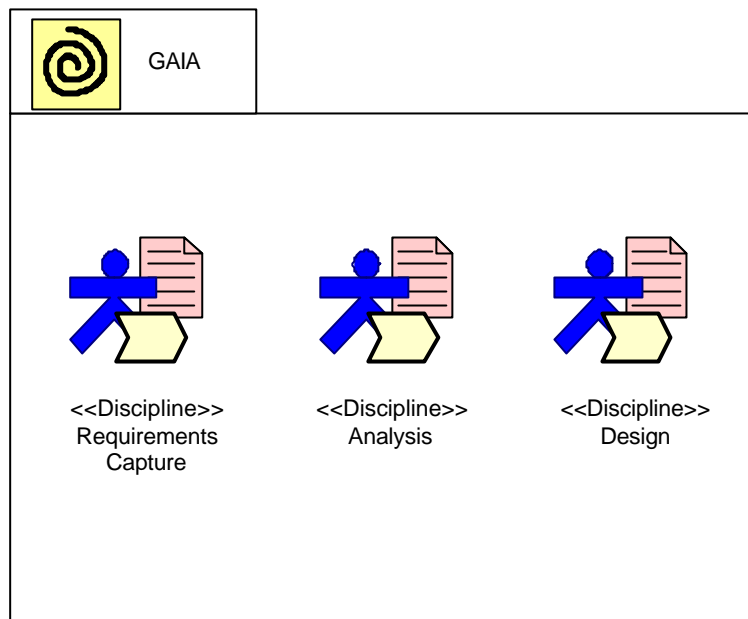


Figure 2-1. The disciplines of the Gaia process

Gaia includes three disciplines:

- Requirements Capture: Gaia authors view the requirements capture phase as being independent of the paradigm used for analysis and design. For this reason Gaia does not deal with the requirements capture phase but it considers the requirements statement as an input for the methodology.
- Analysis: The objective of the analysis stage is to develop an understanding of the system and its structure (without reference to any implementation detail). This understanding is captured in the system's organisation. An organisation can be seen as a collection of **roles**, that stand in certain relationships to one another, and that take part in systematic, institutionalised patterns of **interactions** with other roles.
- Design: The aim in the design stage is to transform the analysis models into a sufficiently low level of abstraction that traditional design techniques (including object-oriented techniques) may be applied in order to implement agents. To put it another way, the design stage is concerned with how a society of agents cooperate to realise the system-level goals, and what is required of each individual agent in order to do this. How an agent realises its services is beyond the scope of Gaia, and will depend on the particular application domain.

## 2.1 Requirements Capture

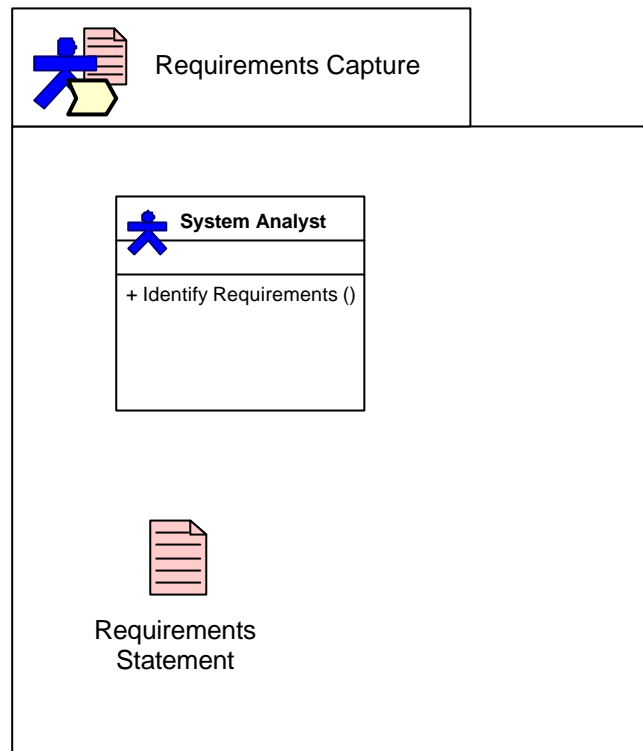


Figure 2-2. Exploitation of the Requirements Capture discipline structure

The Requirements Capture discipline involves a process roles (System Analyst) and one workproduct: the “Requirements Statement” document. In Gaia, there is no description of this document or of the process the Analyst have to follow for obtaining it. The document is just considered as an input for the methodology.

## 2.2 Analysis

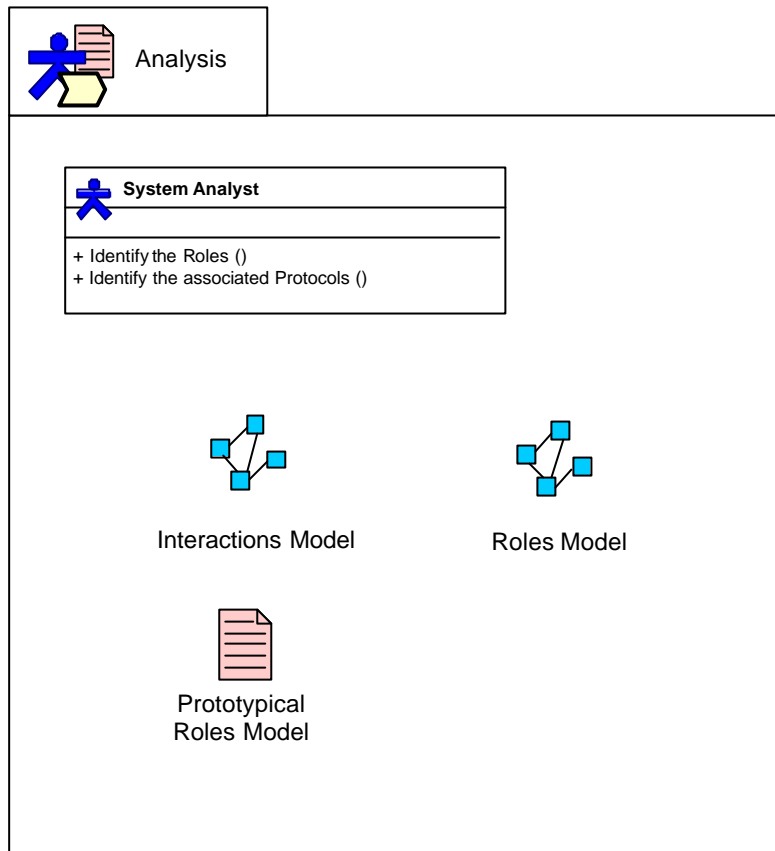


Figure 2-3. Exploitation of the Analysis discipline structure

The Analysis discipline involves one process role (System Analyst) and three work products (two UML models<sup>1</sup> and one document).

### 2.2.1 Process roles involved

#### 2.2.1.1 System Analyst

She is responsible of :

1. the roles identification;
2. the associated protocols identification.

The roles identification is a crucial activity in Gaia.

In Gaia the most abstract entity is the *system* with the meaning of society or organisation. **Roles** in a system will typically correspond to:

- individuals, either within an organisation or acting independently;
- departments within an organisation; or
- organisations themselves.

<sup>1</sup> We would like to remember the fact that Gaia workproducts are not described using UML notation, instead they are expressed using schemata templates or informal textual descriptions. In order to move towards an identification of the method fragments, we believe it is important to express the models using a standard notation. For this reason, we have described the Gaia models using UML, trying not to modify the meaning and the semantics of the models.

Roles stand in certain relationships to one another, and that take part in systematic, institutionalised patterns of **interactions** with other roles.

## 2.2.2 UML Models

The Analysis discipline includes two UML Models whose aim and notation are described in the following sub-sections.

### 2.2.2.1 Interactions Model

In Gaia links between roles are represented in the *Interactions Model*. This model consists of a set of *protocols*, one for each type of inter-role interaction. Here a protocol can be viewed as an institutionalised pattern of interaction. That is, a pattern of interaction that has been formally defined and abstracted away from any particular sequence of execution steps. Viewing interactions in this way means that attention is focused on the essential nature and purpose of the interaction, rather than on the precise ordering of particular message exchanges.

A protocol is characterised by the following attributes:

- *purpose*: description of the nature of the interaction (e.g., “information request”, “schedule activity” and “assign task”);
- *initiator*: the role(s) responsible for starting the interaction;
- *responder*: the role(s) with which the initiator interacts;
- *inputs*: information used by the role initiator while enacting the protocol;
- *outputs*: information supplied by/to the protocol responder during the course of the interaction;
- *processing*: description of any processing the protocol initiator performs during the course of the interaction

### 2.2.2.2 Roles Model

The Roles Model identifies the key roles in the system. Here a role can be viewed as an abstract description of an entity's expected function.

A *role* is defined by: *protocols*, *activities*, *permissions* and *responsibilities*.

*Responsibilities* determine functionality and, as such, are perhaps the key properties associated with a role. Responsibilities are divided into two types: *liveness properties* and *safety properties*. *Liveness properties* intuitively state that something good happens. They describe those states of affairs that an agent must bring about, given certain environmental conditions. In contrast, *safety properties* are invariants. Intuitively, a safety property states that nothing bad happens (i.e., that an acceptable state of affairs is maintained across all states of execution). In order to realise *responsibilities*, a role has a set of *permissions*. Permissions are the rights associated with a role. The permissions of a role thus identify the resources that are available to that role in order to realise its responsibilities. Permissions tend to be information resources. For example, a role might have associated with it the ability to read a particular item of information, or to modify another piece of information. A role can also have the ability to generate information.

The *activities* of a role are computations associated with the role that may be carried out by the agent without interacting with other agents. Finally, a role is also identified with a number of *protocols*, which define the way that it can interact with other roles.

## 2.2.3 Documents

### 2.2.3.1 Prototypical Roles Model

A Prototypical Roles Model is a list of the key roles that occur in the system, each with an informal, unelaborated description.

## 2.3 Design

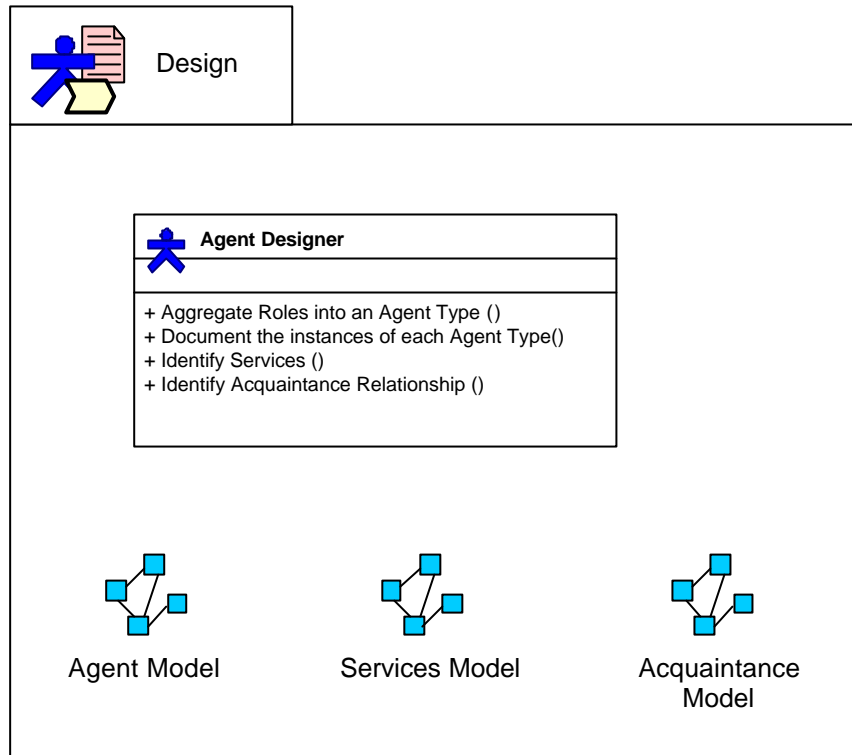


Figure 2-4. Exploitation of the Design discipline structure

The Design discipline involves one process role (Agent Designer) and three work products (three UML models).

### 2.3.1 Process roles involved

#### 2.3.1.1 Agent Designer

She is responsible of :

1. aggregate Roles into an Agent Type;
2. document the instances of each Agent type;
3. identify Services;
4. identify Acquaintance relationship.

### 2.3.2 UML Models

The Design discipline includes three UML Models whose aim and notation are described in the following sub-sections.

#### 2.3.2.1 Agent Model

The purpose of the Gaia Agent Model is to document the various agent types that will be used in the system under development, and the agent instances that will realise these agent types at run-time.

An Agent Type is best thought of as a set of agent roles. It can be visualized using a simple agent type tree, in which leaf nodes correspond to roles, (as defined in the roles model), and other nodes correspond to agent types. If an agent type t1 has children t2 and t3, then this means that t1 is composed of the roles that make up t2 and t3.

### 2.3.2.2 Services Model

As its name suggests, the aim of the Gaia Services Model is to identify the *services* associated with each agent role, and to specify the main properties of these services.

By a service, Gaia means a *function* of the agent. A service is a coherent block of activity in which an agent will engage. It should be clear that every activity identified at the analysis stage will correspond to a service, though not every service will correspond to an activity.

For each service that may be performed by an agent, it is necessary to document its properties. Specifically, we must identify the *inputs*, *outputs*, *pre-conditions*, and *post-conditions* of each service. Inputs and outputs to services will be derived in an obvious way from the *Interactions Model*. Pre- and post-conditions represent constraints on services. These are derived from the safety properties of a role. Note that by definition, each role will be associated with at least one service. The services that an agent will perform are derived from the list of protocols, activities, responsibilities and the liveness properties of a role. The Gaia services model does *not* prescribe an implementation for the services it documents. The developer is free to realise the services in any implementation framework deemed appropriate. For example, it may be decided to implement services directly as methods in an object-oriented language. Alternatively, a service may be decomposed into a number of methods.

### 2.3.2.3 Acquaintance Model

An *Acquaintance Model* simply defines the communication links that exist between agent types. It does *not* define what messages are sent or when messages are sent, it simply indicates that communication pathways exist. In particular, the purpose of an acquaintance model is to identify any potential communication bottlenecks, which may cause problems at run-time

An agent Acquaintance Model can be visualized as a *directed* graph, with nodes in the graph corresponding to agent types and arcs in the graph corresponding to communication pathways. An arc  $a \rightarrow b$  indicates that a will send messages to b, but not necessarily that b will send messages to a. An Acquaintance Model may be derived in a straightforward way from the *Roles*, *Interactions*, and *Agent* models.

### 3 Gaia process description

#### 3.1 Complete process

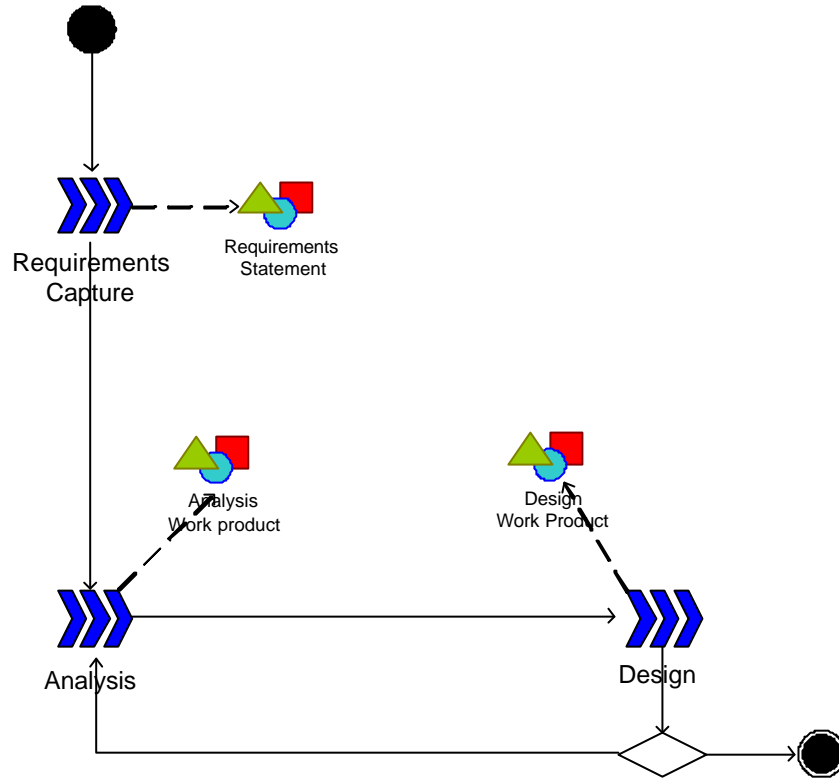


Figure 3-1. Gaia complete process

The Gaia process is composed of three different phases: Requirements Capture, Analysis, Design. Each phase produces a document that is usually composed aggregating the UML models and work products of the work definitions that are inside each phase (see section 5.2).

### 3.2 Phases/iterations of the process

#### 3.2.1 The Analysis phase

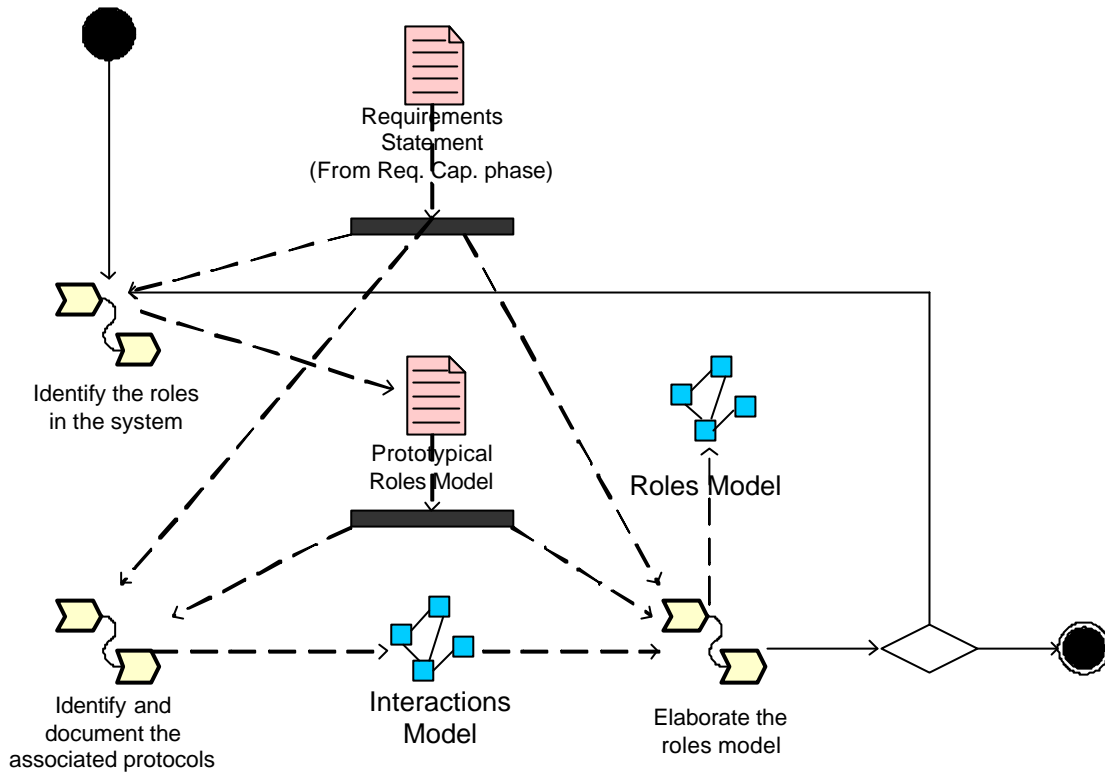


Figure 3-2. Gaia Analysis phase

The objective of the analysis stage is to develop an understanding of the system and its structure in terms of roles and patterns of interactions between the various roles.

The first step consists in identifying the roles in the system building a *Prototypical Roles Model*. A list of the key roles that occur in the system, each with an informal, unelaborated description. Then, for each role, the Analyst has to identify and document the associated protocols (*Interaction Model*). Protocols are the patterns of interaction that occur in the system between the various roles. Finally, using the *Interaction Model* as a basis, the System Analyst elaborates the *Roles Model*, which documents the key roles occurring in the system, their permissions and responsibilities, together with the protocols and activities in which they participate.

### 3.2.1.1 Work definitions and Activities of this phase/iteration

#### 3.2.1.1.1 The “Identify the roles in the system” work definition

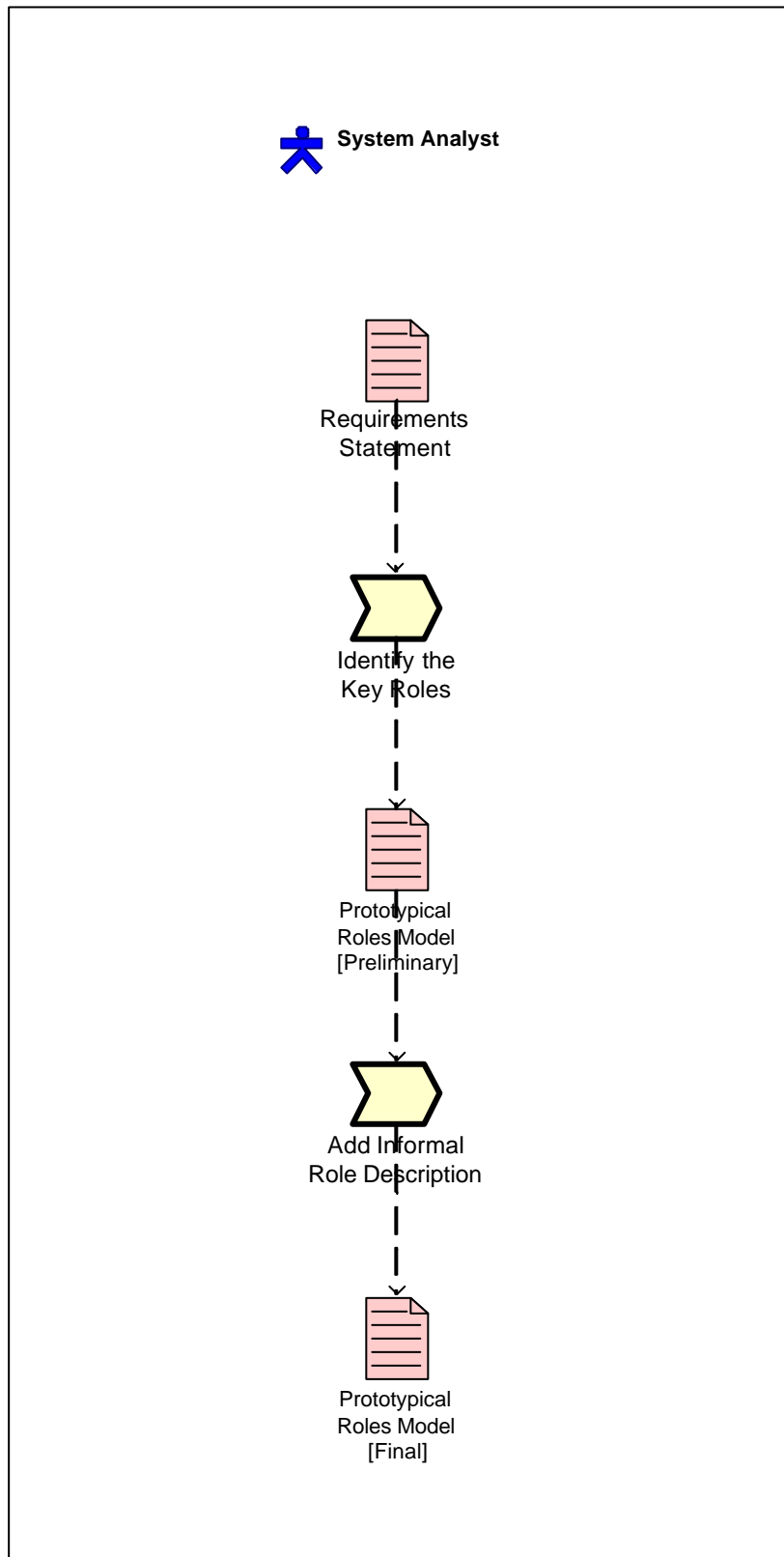


Figure 3-3. The “Identify the roles in the system” work definition

3.2.1.1.2 The “Identify and document the associated protocols” work definition

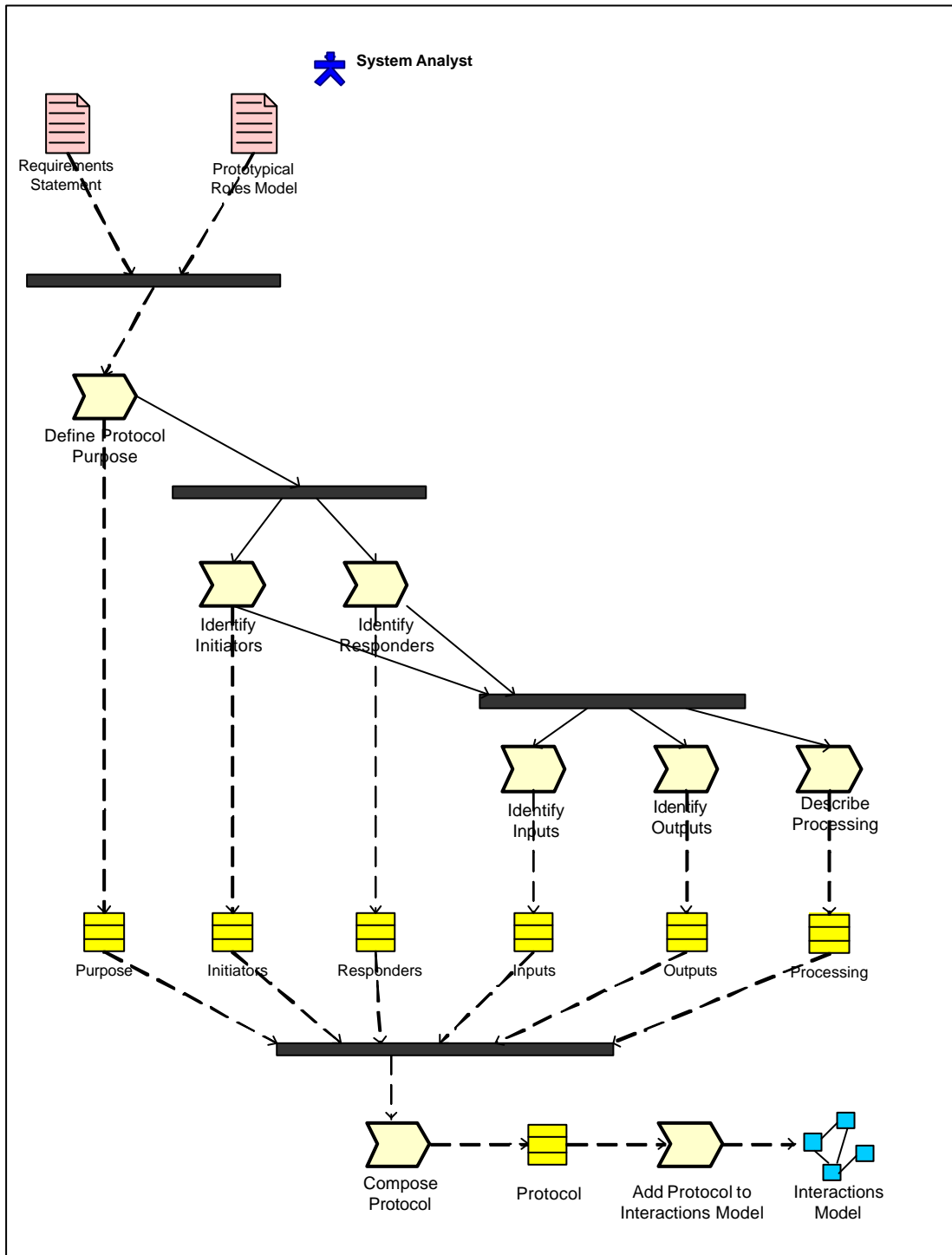


Figure 3-4. The “Identify and document the associated protocols” work definition



Protocol

Note to the reader: This is the symbol of an element of the MAS Model (see section 5.3 below)

### 3.2.1.1.3 The “Elaborate the roles model” work definition

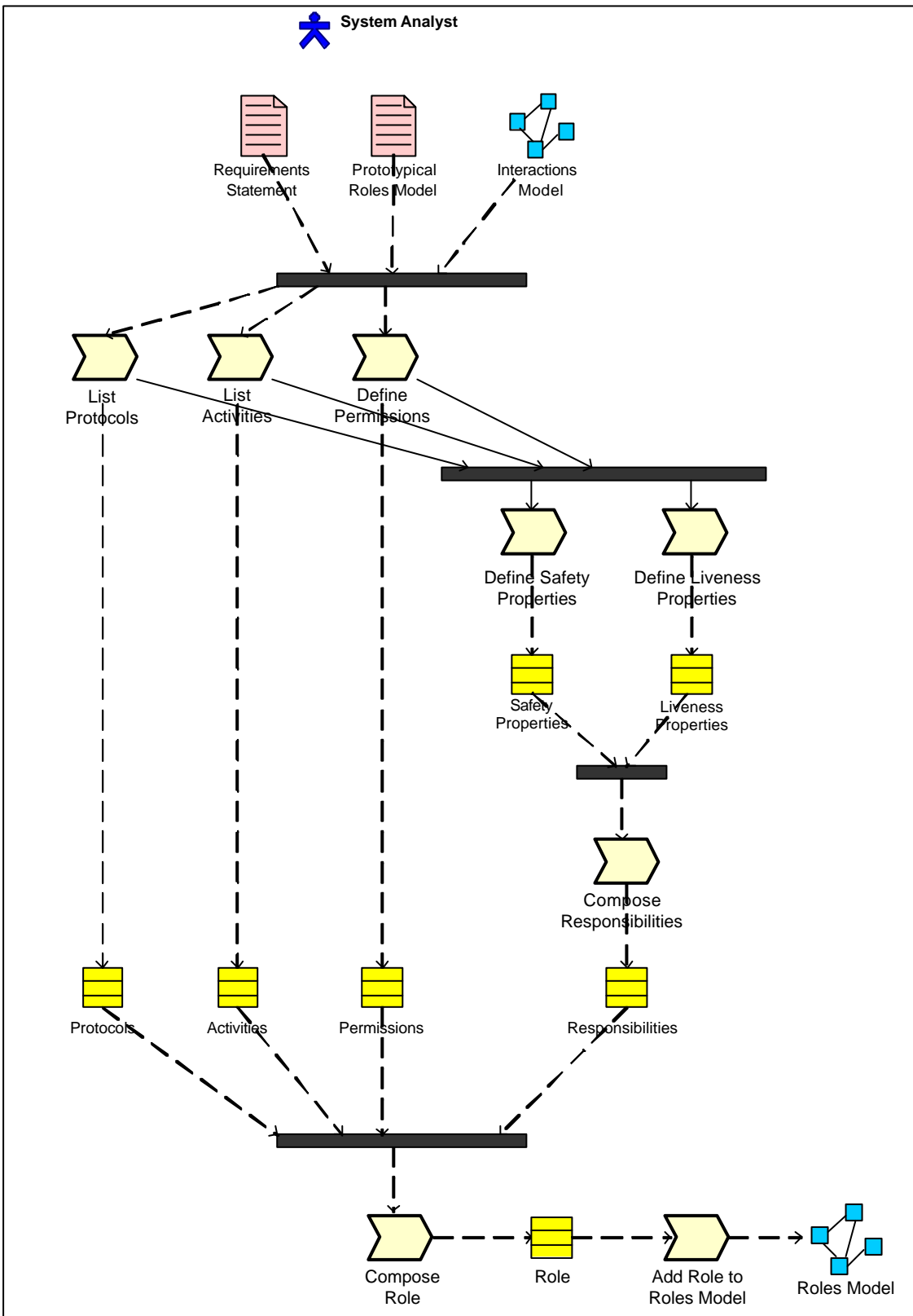


Figure 3-5. The “Elaborate the roles model” work definition

### 3.2.2 The Design phase

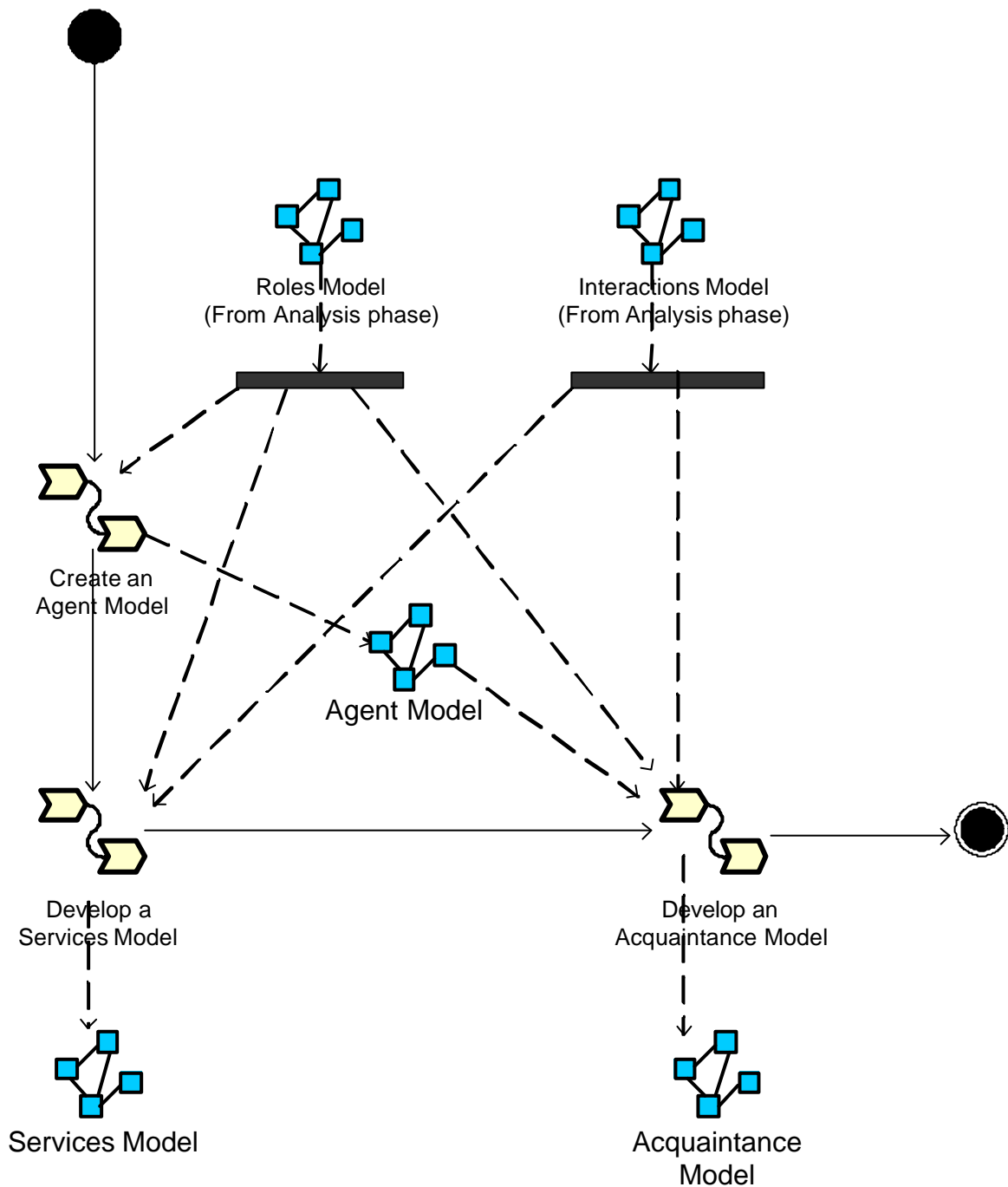


Figure 3-6. Gaia Design phase

The Gaia design process involves generating three models: Agent Model, Services Model, Acquaintance Model. The **Agent Model** identifies the agent types that will make up the system, and the agent instances that will be instantiated from these types. The **Services Model** identifies the main services that are required to realise the agent's role. Finally, the **Acquaintance Model** documents the lines of communication between the different agents.

### 3.2.2.1 Work definitions and Activities of this phase/iteration

#### 3.2.2.1.1 The “Create an Agent Model” work definition

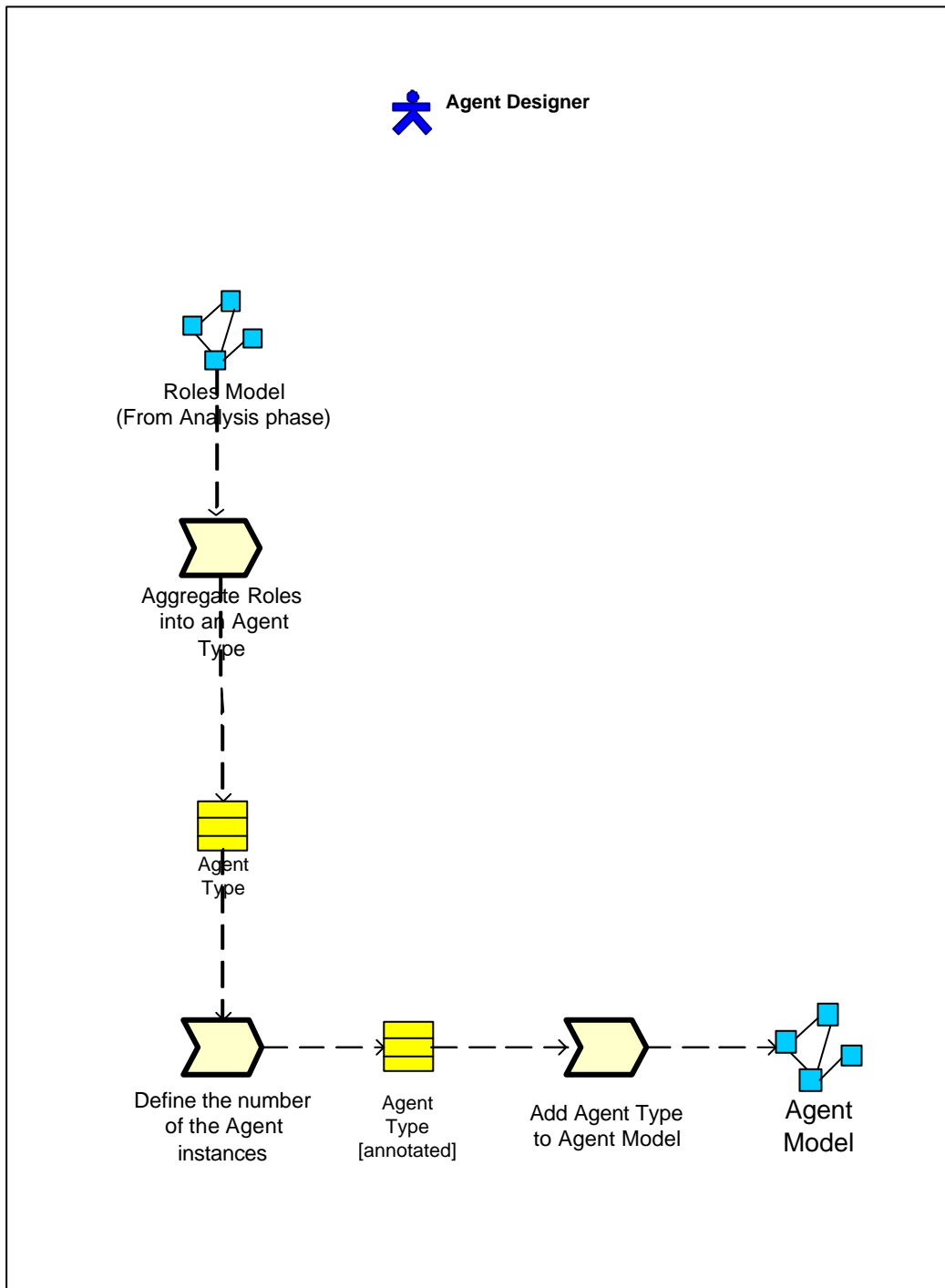


Figure 3-7. The “Create an Agent Model” work definition

### 3.2.2.1.2 The “Develop a Services Model” work definition

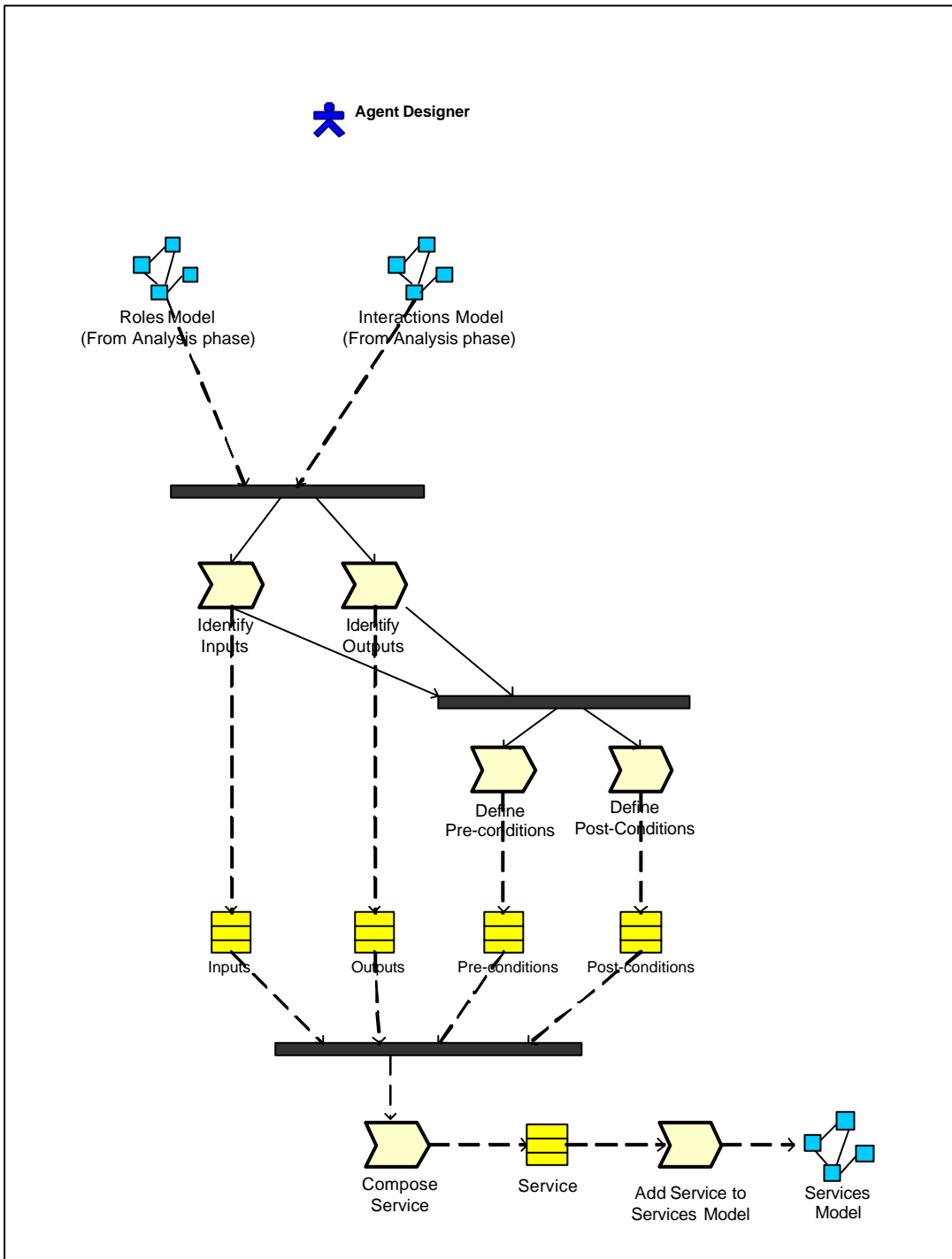


Figure 3-8. The “Develop a Services Model” work definition

### 3.2.2.1.3 The “Develop an Acquaintance Model” work definition

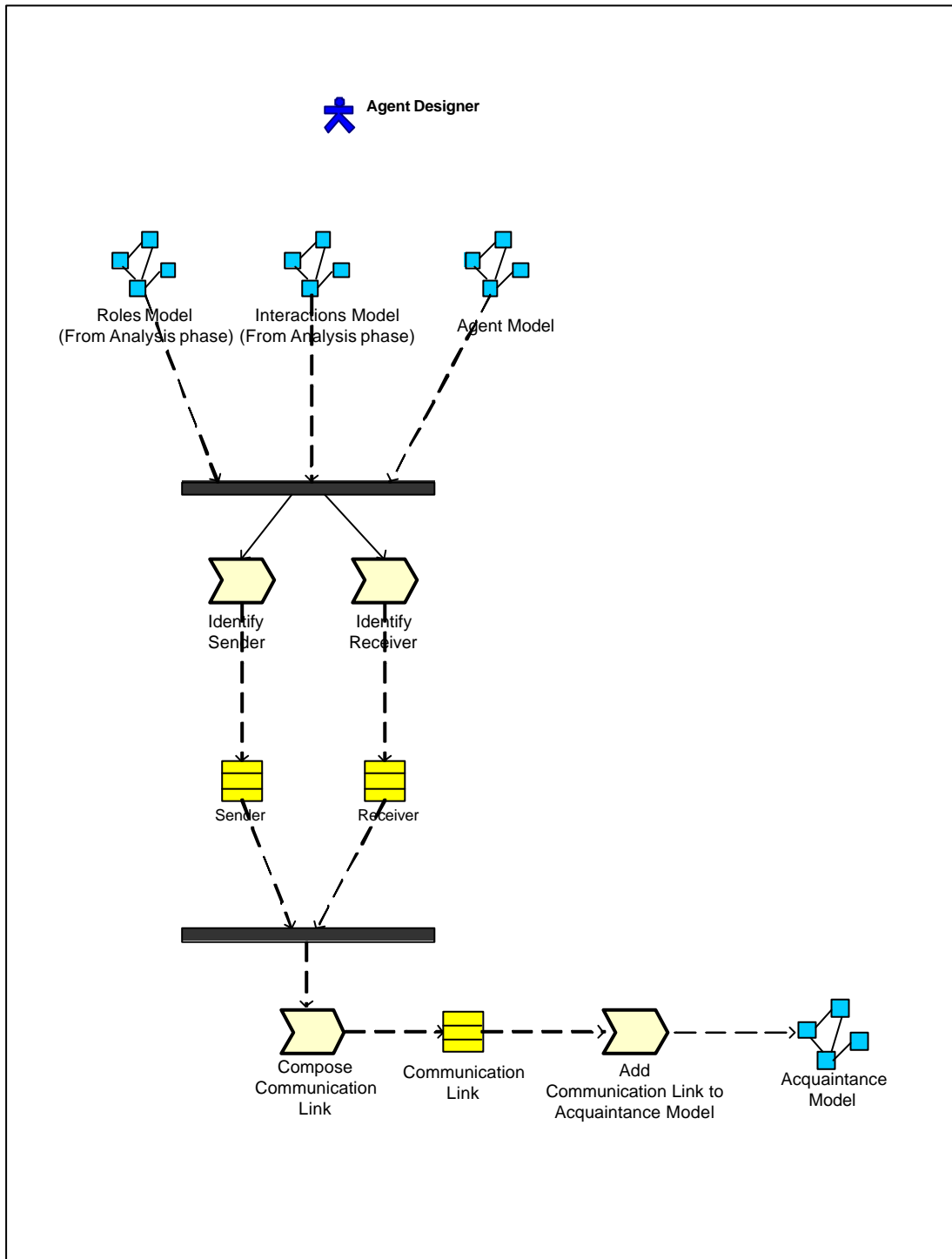


Figure 3-9. The “Develop an Acquaintance Model” work definition

## 4 Gaia work definitions

### 4.1 Analysis

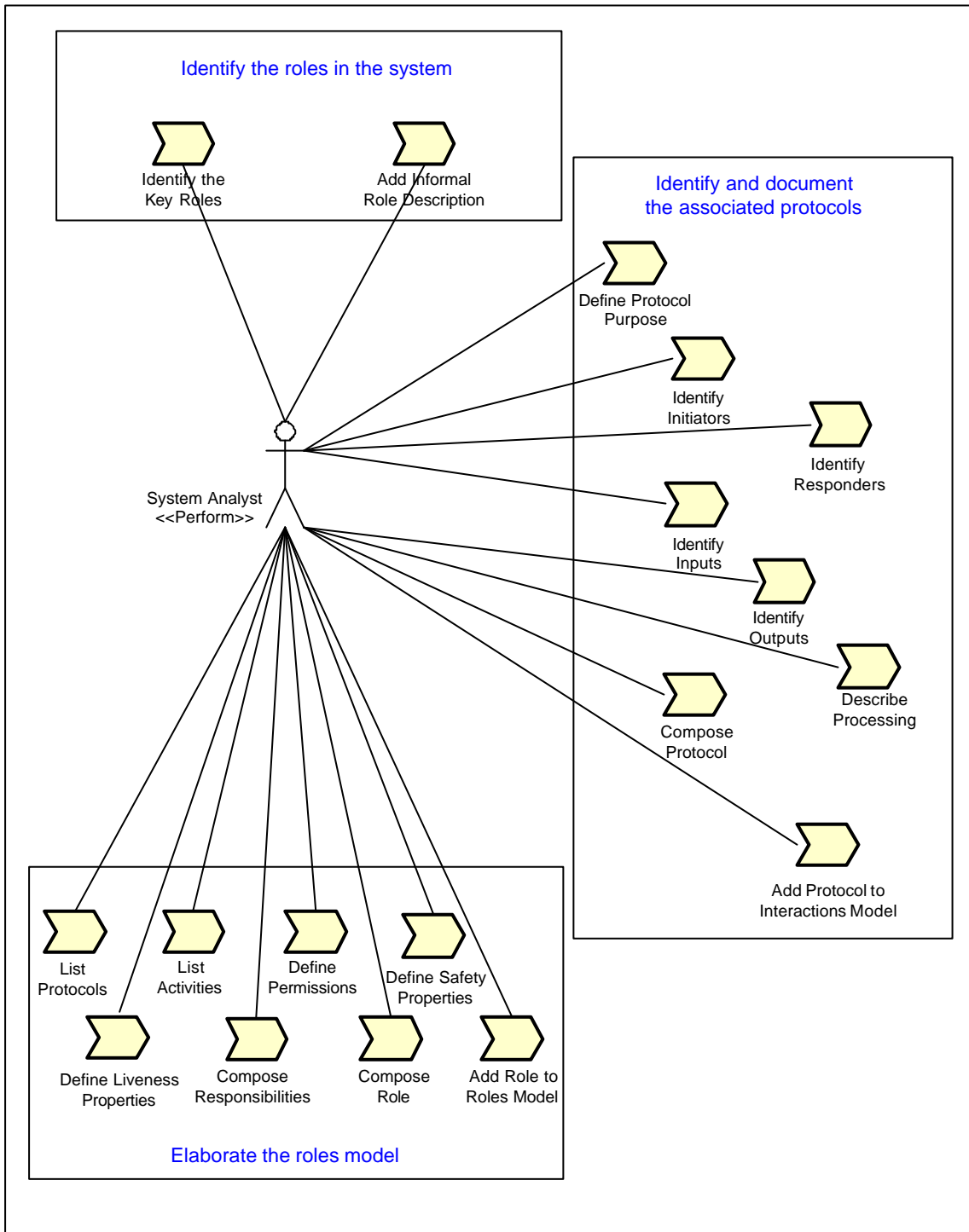


Figure 4-1. The “Analysis” work definition

## 4.2 Design

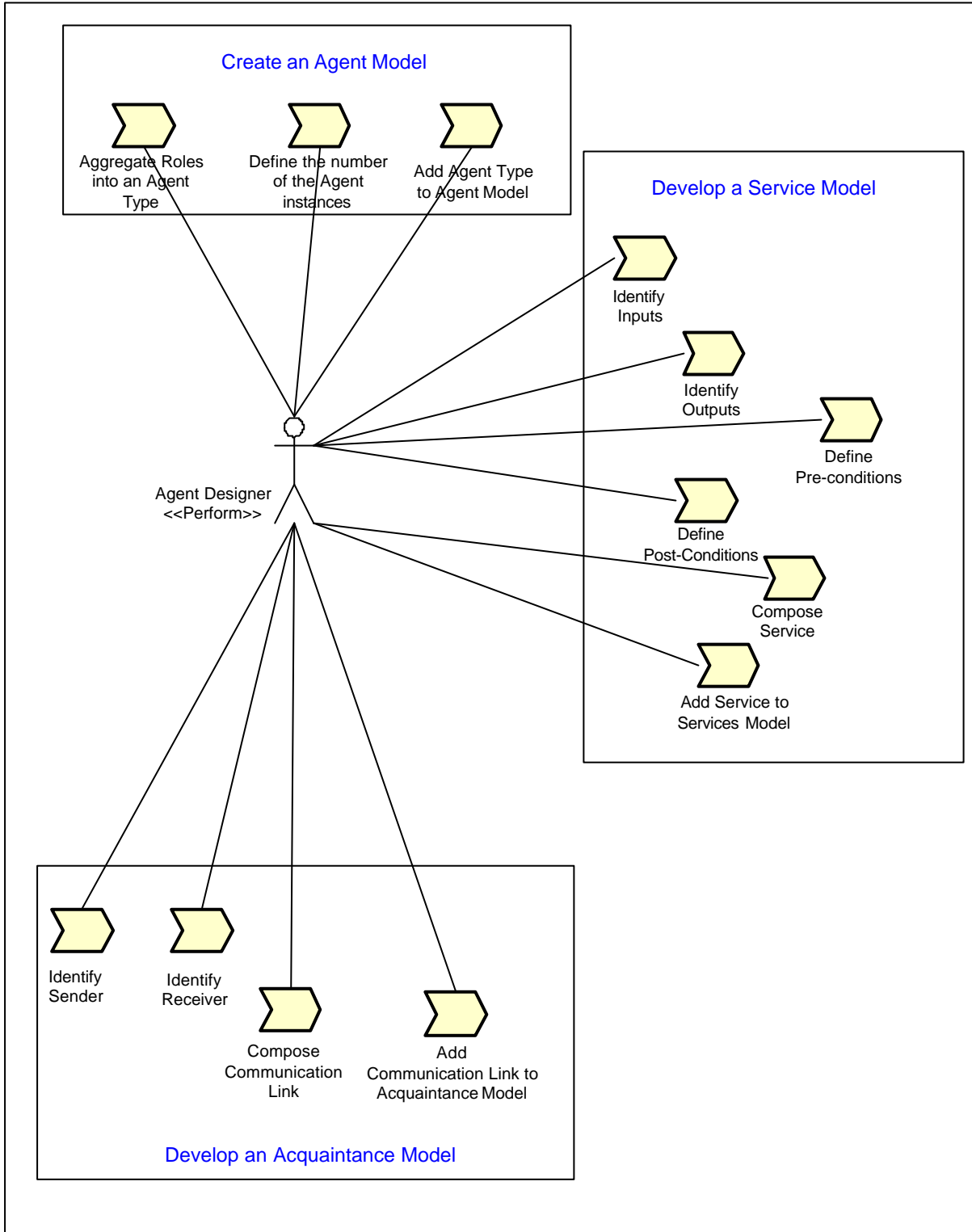


Figure 4-2. The "Design" work definition

## 5 Work Products

### 5.1 Dependency diagram

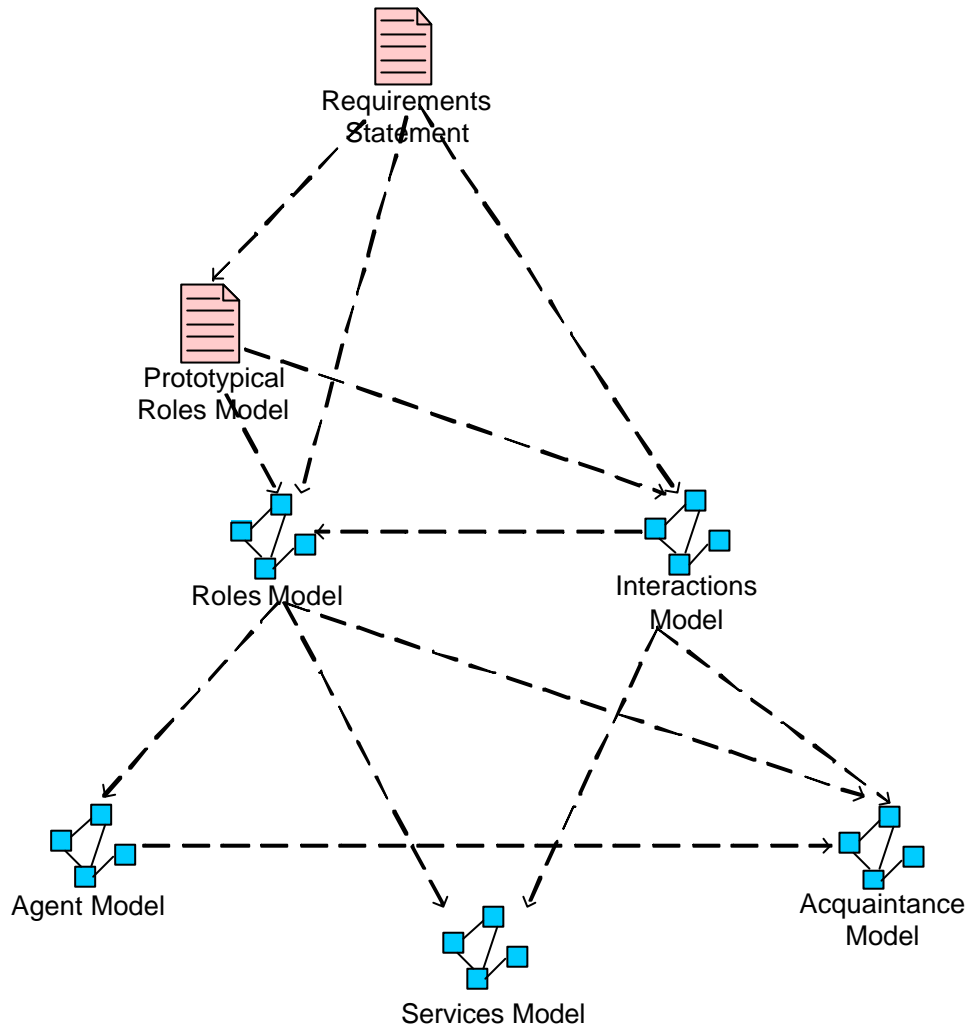


Figure 5-1. Work products dependency diagram

## 5.2 Work products model

### 5.2.1 Analysis work products model

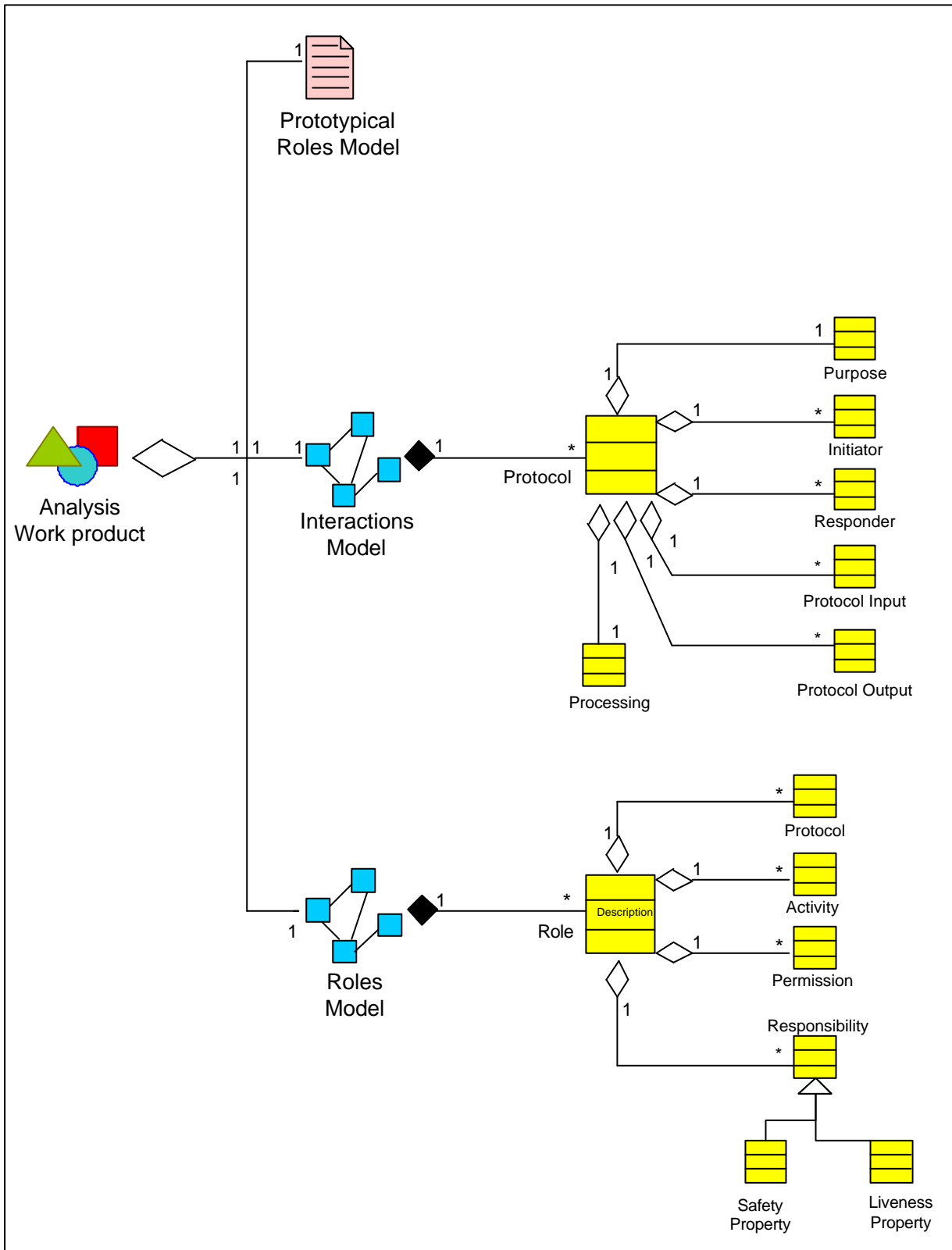


Figure 5-2. Analysis work products model

### 5.2.2 Design work products model

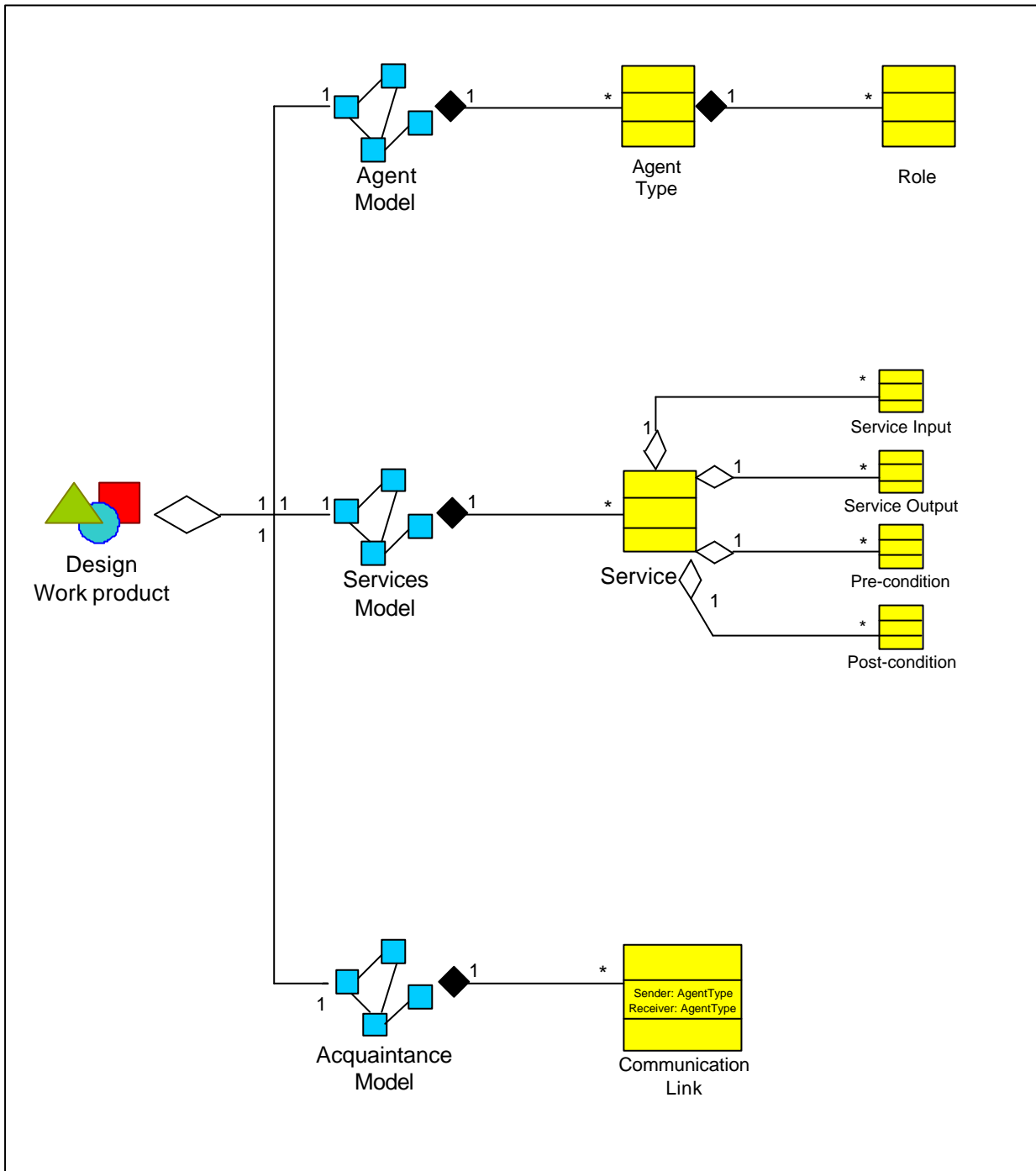


Figure 5-3. Design work products model

### 5.3 MAS Model

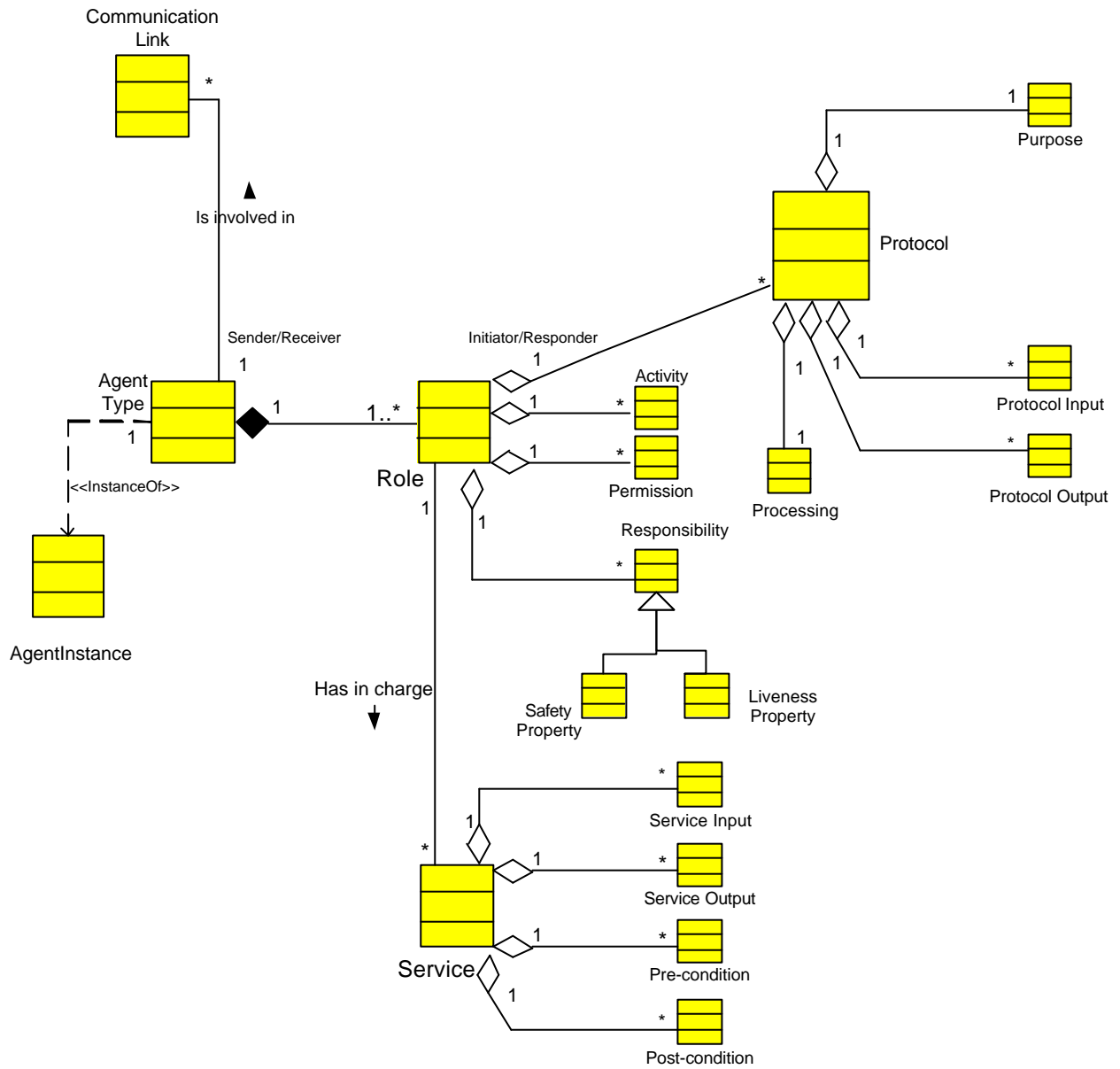


Figure 5-4. The Gaia Multi Agent System Model

## 6 Method Fragments

A Method fragment is a portion of the development process, composed as follows:

1. A portion of process (what is to be done, in what order), defined with a SPEM diagram
2. One or more deliverables (artifacts like (A)UML/UML diagrams, text documents and so on).  
The result of the work could also be some kind of product/artifact that is not be delivered to anyone outside the development process. It includes a reference to a recommended notation/language/structure to be used to represent AUML/UML/other diagrams if they are part of the deliverables (as it is common)
3. Some preconditions (they are a kind of constraint because it is not possible to start the process specified in the fragment without the required input data or without verifying the required guard condition)
4. A list of concepts (related to the MAS meta-model) to be defined (designed) or refined during the specified process fragment.
5. Guideline(s) that illustrates how to apply the fragment and best practices related to that
6. A glossary of terms used in the fragment (in order to avoid misunderstandings if the fragment is reused in a context that is different from the original one)
7. Composition guidelines – A description of the context/ problem that is behind the methodology from which the specific fragment is extracted.
8. Aspects of fragment. Textual description of specific issues like for example: platform to be used, application area,...
9. Dependency relationships useful to assemble fragments

It should be noted that not all of these elements are mandatory. Some of them (for instance notation, guideline or inputs) could be not applicable or not necessary for some specific fragment.

The fragment refers to a MAS meta-model and its aim is to contribute in increasing the definition of the instance of this meta-model that will solve the problem the designer is facing

In the following, two method fragments, extracted from the Gaia methodology, will be considered and discussed.

### 6.1 The “Identify and document the interaction protocols” fragment

Consider the Gaia process represented in Fig. 6-1. and particularly the Analysis phase.

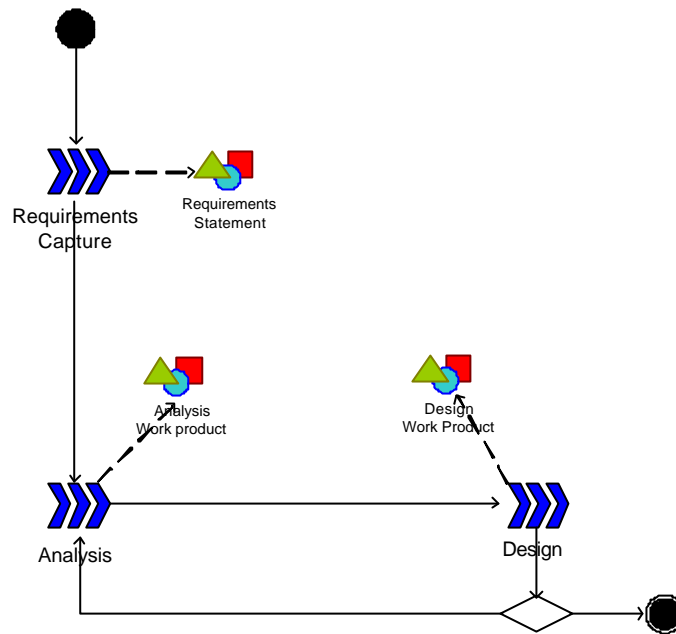


Figure 6-1. Gaia complete process

Within this phase we perform the activities specified in Fig. 6-2. In this context we can identify our fragment (red oval).

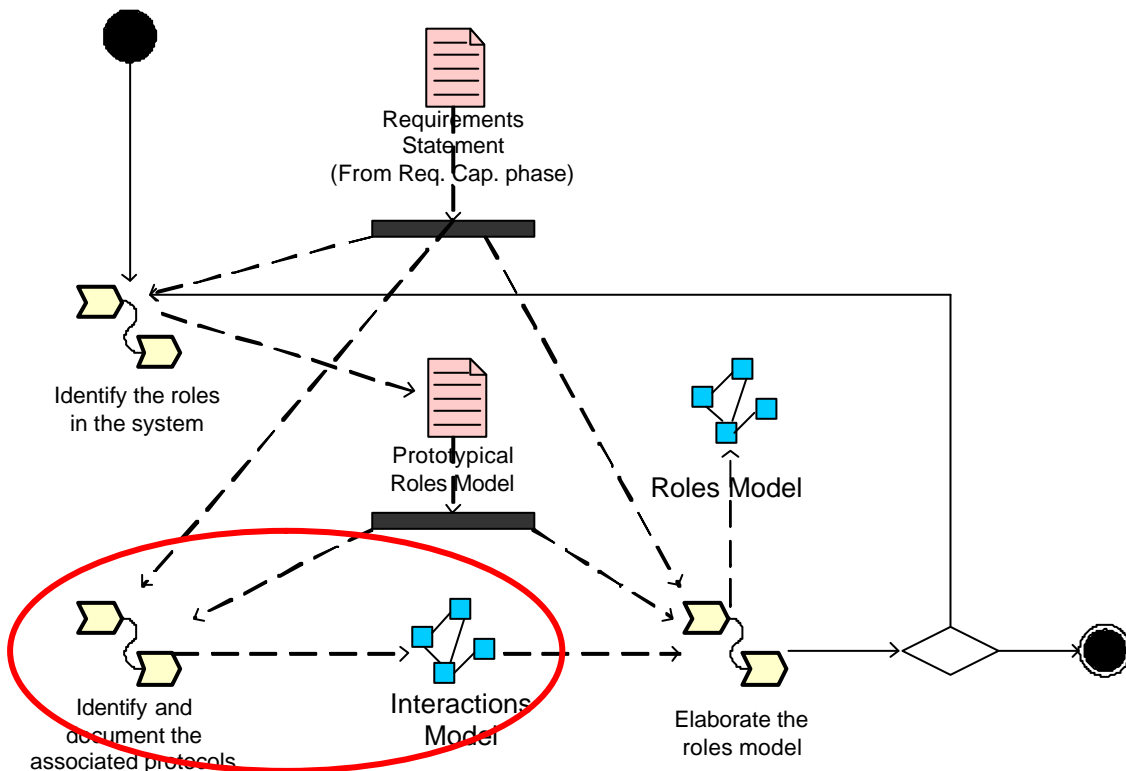


Figure 6-2. Gaia Analysis phase

Consider the work definition “Identify and document the associated protocols” and the consequent outcome (UML model “Interactions Model”). This is a fragment whose aim is to identify and document the patterns of interaction that occur in the system between the various *roles*.

The process that is to be performed in order to obtain the result is represented in Fig. 6-3 as a SPEM activity diagram.

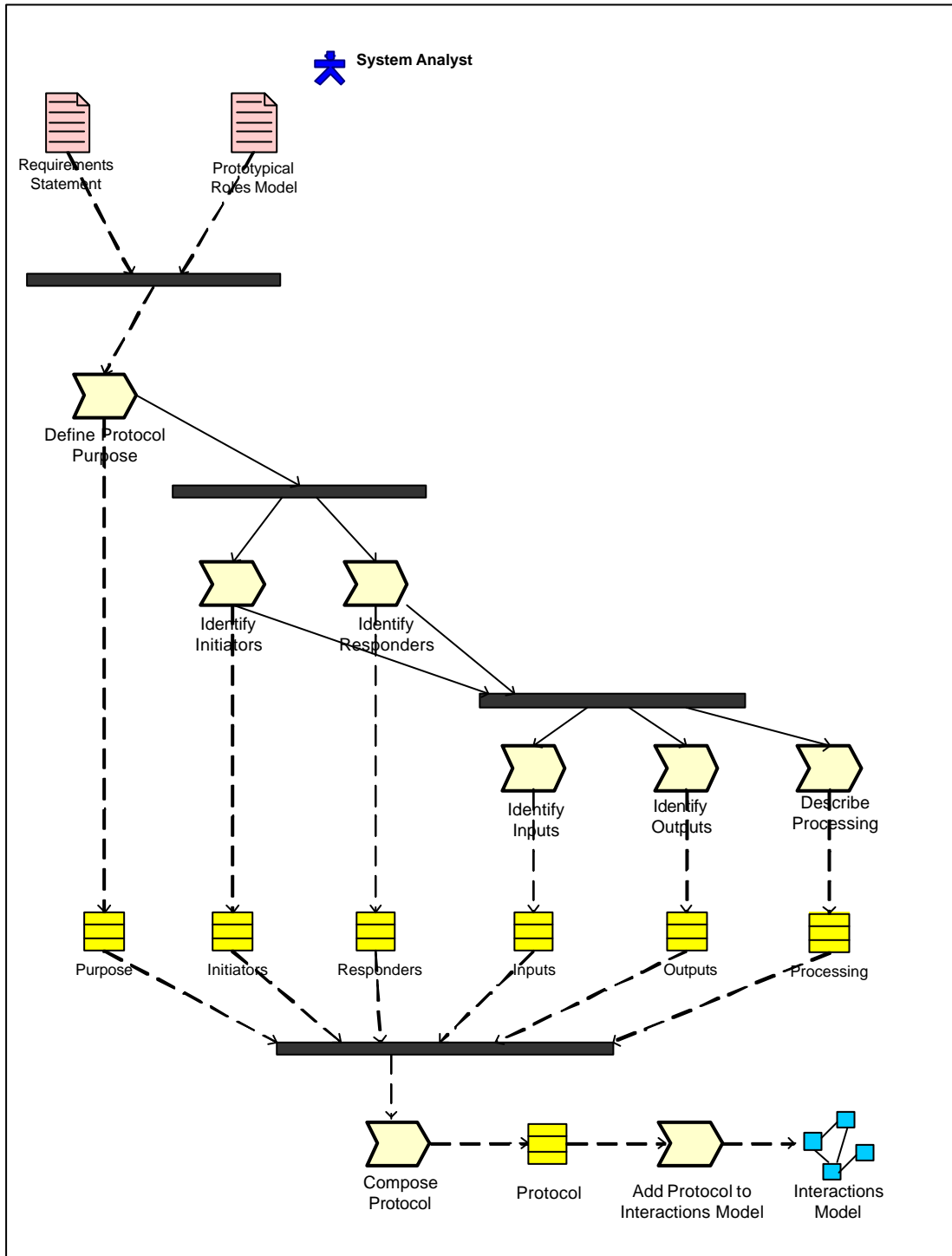
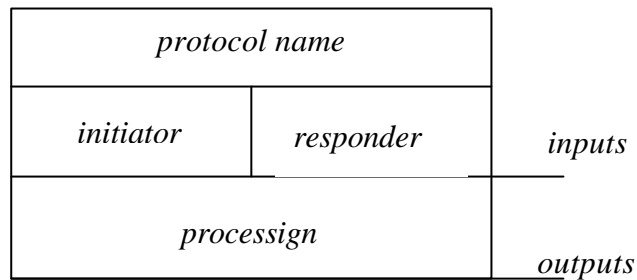


Figure 6-3. The Identify and Document the Interaction Protocols Fragment - Procedural aspects

### 6.1.1 Notation

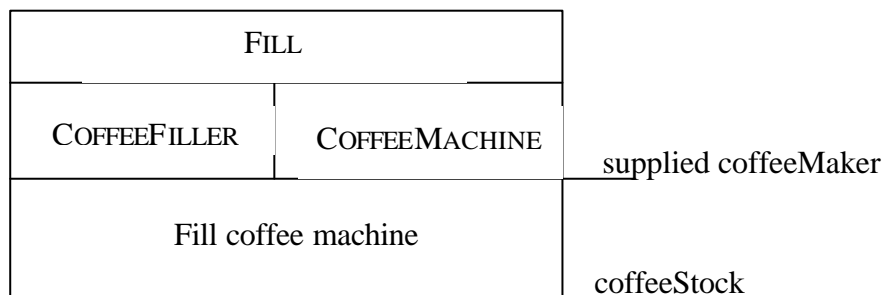
The “Interactions Model” consists of a set of protocol definitions, one for each type of inter-role interaction. According to the notation presented in [1] a protocol can be defined using the diagram showed in Fig. 6-4 where:

- initiator indicates the *role(s)* responsible for starting the interaction;
- responder indicates the *role(s)* with which the initiator interacts;
- inputs detailed the information used by the role initiator while enacting the protocol;
- outputs detailed the information supplied by/to the protocol responder during the course of the interaction;
- processing describes any processing the protocol initiator performs during the course of the interaction.



**Figure 6-4. The Protocol Definition diagram**

As an illustration consider the FILL protocol presented in [1] and showed in Fig. 6-5. This states that the protocol FILL is initiated by the role COFFEEFILLER and involves the role COFFEEMACHINE. The protocol involves COFFEEFILLER putting coffee in the machine named coffeeMaker, and results in COFFEEMACHINE being informed about the value of *coffeeStock*.




**Figure 6-5. The Fill Protocol Definition**

### 6.1.2 Deliverables

This fragment produces a set of protocol definitions expressing using the diagram showed in Fig. 6-4.

**The overall output of this fragment is the interactions model** (considered as a set of protocol definitions). The following figure illustrates the relationship between this fragment outcome and the MAS meta-model.

(The symbol:  represents an element of the MAS model)

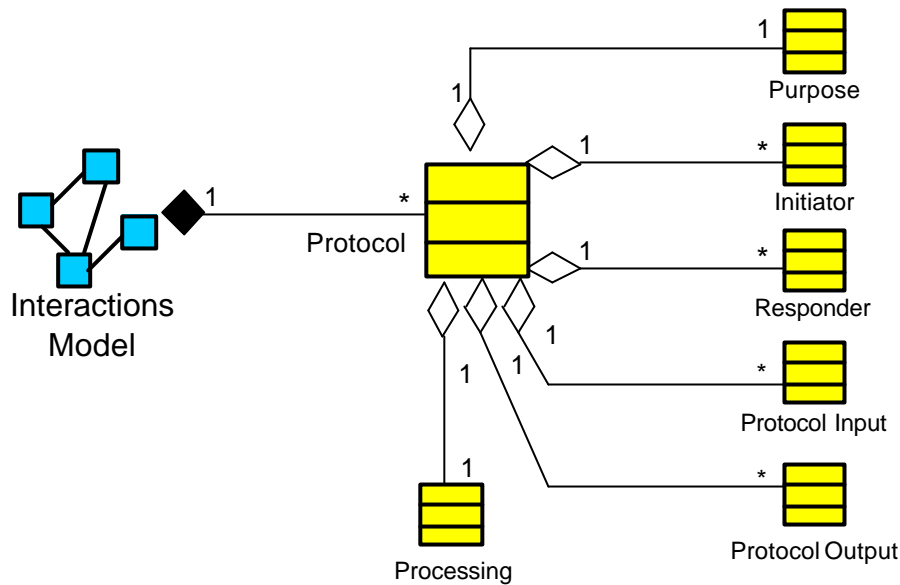


Figure 6-6. Structure of the Identify and document the interaction protocols fragment work-product with respect to the MAS meta-model

### 6.1.3 Preconditions

Inputs, output and elements to be designed in the fragment are detailed in the following table.

Input	To Be Designed	Output
Requirements Statement	Protocol Definitions	Protocols (MAS Meta-model component)
Prototypical Roles Model		

In order to design protocols we need to capture the system's organisation. An organisation can be seen as a collection of roles, that stand in certain relationships to one another, and that take part in systematic, institutionalised patterns of interactions with other roles. To identify this patterns of interactions (protocol) we need the System Requirements and a Prototypical Roles Model describing the key roles that occur in the system, each with an informal, unelaborated description.

### 6.1.4 Relationships with the MAS meta-model

The final result is the protocol that is an element of the MAS metamodel (see Fig. 6-6.)

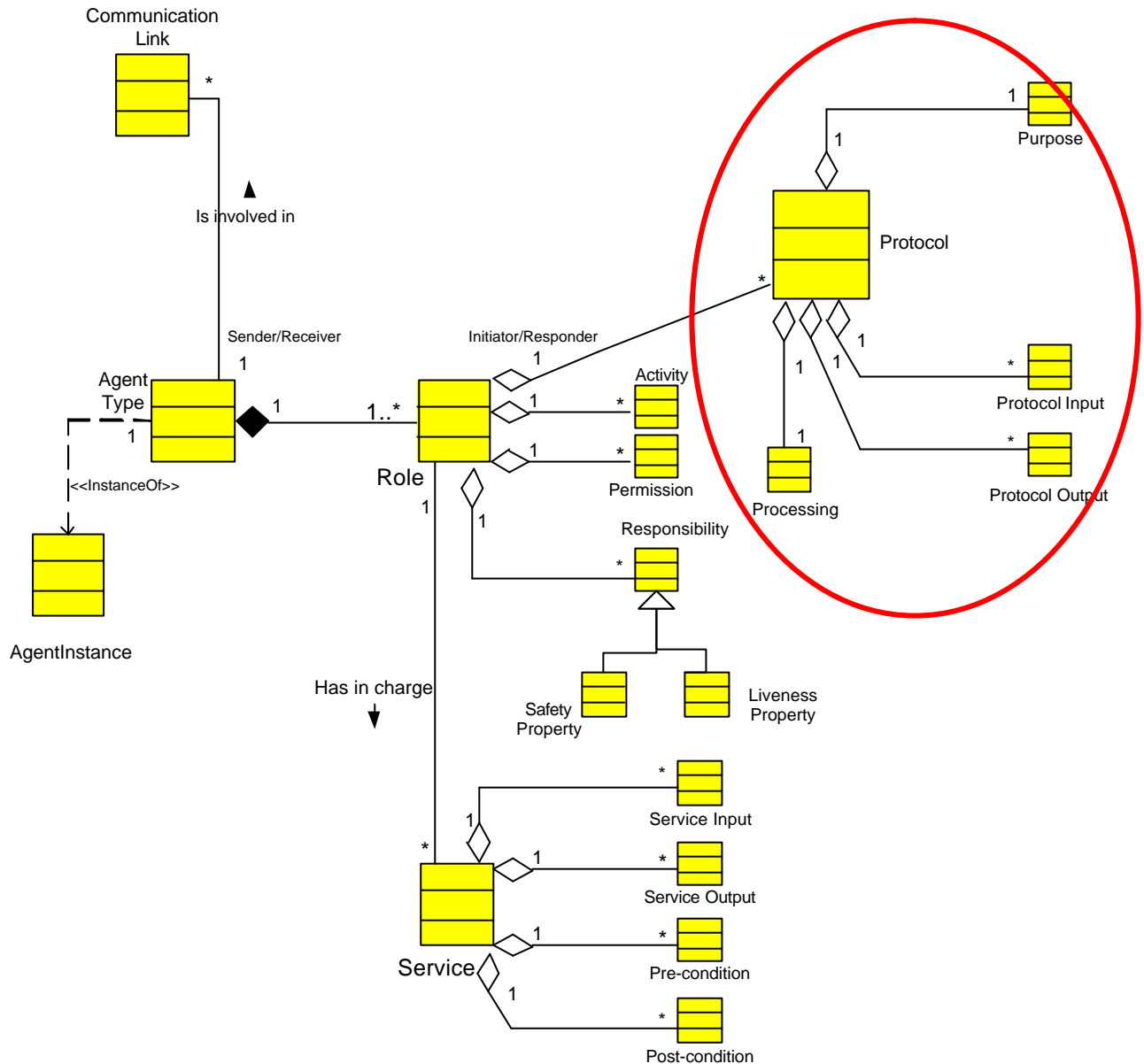


Figure 6-7. The MAS meta-model adopted in Gaia

### 6.1.5 Guidelines

- 1) The strategy of identification of the protocols is out of the scope of this fragment and it is left to the analyst;
- 2) The implementation details, such as the sequence of execution steps and messages exchanged, are out of the scope of this fragment. They are referred to the following stages.

### 6.1.6 Glossary

“Identify and document the interaction protocols” fragment uses this list of model element:

**Role** – A portion of the social behavior of an agent that is characterized by some specificity such as a goal, a set of attributes (for example responsibilities, permissions, activities, and protocols) or providing a functionality/service.

**Protocol** – patterns of interaction that occur in the system between the various *roles*.

### 6.1.7 Composition guidelines

The only inputs for this fragment are the system requirements descriptions (text document) and the prototypical roles model (text document). The roles model and interactions model are closely related with one another. This fragment, belonging to the analysis phase of the Gaia methodology, can be reused as part of the analysis phase, probably with predilection for the final stages of the analysis process, or possibly in the early stages of the design process. In any case it can be reused when the organizational structure of the system is defined.

### 6.1.8 Aspects of fragment

The aim of this fragment is mainly to identify and describe the interactions required to accomplish the roles. In it more attention is paid to the nature and purpose of the interaction than to the implementation details. Interactions designed with this fragment are supposed to be static. As a matter of fact the organisation structure of the system is considered static, in that inter-agent relationships do not change at run-time.

### 6.1.9 Relationships with other fragments

This fragment is related with the “identify the roles in the system” fragment and with the fragment devoted to the requirements capture phase (Gaia does not deal with this phase; it considers the requirements statement as an input for the methodology).

### 6.1.10 Summary

The “Identify and document the associated protocols” fragment aim to identify and document the patterns of interaction that occur in the system between the various *roles*.

It is important to highlight the fact that Gaia encourages a developer to think of building agent-based systems as a process of *organisational design*. As a consequence the objective of the analysis stage and therefore of the present fragment is to develop an understanding of the system and its structure. This understanding is captured in the system's *organisation*. An organisation is viewed as a collection of roles, that stand in certain relationships to one another, and that take part in systematic, institutionalised *patterns of interactions* with other roles.

This fragment can be used in the definition of a customized methodology to formalize the patterns of interactions, involving roles, known at analysis time. It deals with the description of the *interaction model*. This model consists of a set of *protocol definitions*, one for each type of inter-role interaction; where a protocol can be viewed as an institutionalised pattern of interaction. The protocol definition is a simple table detailing the purpose of the protocol, the role initiating the protocol (initiator), the role in charge of responding to it (responder), the input and output information processed in the protocol, as well as a brief textual description of the type of information processing taking place during the execution of this protocol.

## 7 Glossary

TBD

## **8 Annexes**

## References

- [1] M. Wooldrige, N. R. Jennings and D. Kinny, The Gaia Methodology for Agent-Oriented Analysis and Design, *Autonomous Agents and Multi-Agent Systems*, volume 3, pp 285-312, Kluwer Academic Publishers, The Netherlands, 2000.
- [2] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, and P. Jeremaes, *Object-Oriented Development: The fusion Method*. Prentice Hall International: Hemel Hempstead, England, 1994.
- [3] M. Cossentino, G. Hopmans, J. Odell. FIPA Standardization Activities in the Software Engineering Area.