Contents

Chapter 1

DYNASTAT: A Methodology for Dynamic and Static Modeling of Multi-Agent Systems

Darshan S. Dillon, Maja Hadzic, Tharam S. Dillon
Curtin University of Technology Research Lab for Digital Health Ecosystems Perth 6845, Western Australia, Australia

1.1 Abstract

Multi-agent systems are increasingly being used within various knowledge domains. The need for modeling of the multi-agent systems in a systematic and effective way is becoming more evident. In this chapter, we present the DYNASTAT methodology. This methodology involves a conceptual overview of multi-agent systems, a selection of specific agent characteristics to model, and a discussion of what has to be modeled for each of these agent characteristics. DYNASTAT is independent of any particular modeling language but provides a framework that can be used to realize a particular language in the context of a real-world example. UML 2.2 was chosen as the modeling language to implement the DYNASTAT methodology and this was illustrated using examples from the medical domain. Several UML 2.2 diagrams were selected including a use case, composite structure, sequence and activity diagram to model a multi-agent system able to assist both a medical researcher and a primary care physician. UML 2.2 provides a framework for effective modeling of agent-based systems in a standardized way which this chapter endeavors to demonstrate.

Keywords- multi-agent systems, UML 2.2, medical domain.

1.2 Introduction

Multi-agent systems are distributed software systems where each agent is represented by a process that may be either single or multi-threaded. These agents are able to cooperate and may use regular data sources such as databases and files. The collection of agents is generally designed so that they collaborate in order to fulfill a particular purpose in a particular domain. We have focused on the application of multi-agents systems in the medical domain. The use of agent-based systems enables us to propose and design intelligent and dynamic medical information systems [7].

Any multi-agent software system must have a collection of agents that have a number of characteristics. Usually, the term *intelligent* agent reflects the fact that the agent is able to perform intelligent actions within a society which consists of both computer agents and human agents. As such, agents have an environment in which they operate and objectives they are programmed to achieve, and thus, there is usually some way to measure their effectiveness and run-time success. They are able to reason in a procedural way and also use knowledge to make decisions in order to act. There are different triggers for agent decisions which can be internal or external. Agents are able to plan different courses of action to achieve their goals and make choices from these plans. Importantly, agents are social. The social interaction may include cooperation, coordination, communication or negotiation, all of which are means whereby agents interact in groups to fulfill system goals. Finally, there are a number of ways to implement agent-based systems on the various devices over which a multi-agent system may be distributed.

Existing methodologies to cover the modelling of multi-agent systems yield a number of points:

(1) Most existing methodologies to model multi-agent systems do not define an overarching model from which they choose agent characteristics to include in their scope.
(2) These methodologies cover a broad spectrum in relation to what percentages of agent characteristics are represented. Some methodologies cover a small subset in depth, while others are more complete.
(3) Some of these methodologies are abstract, and others define a concrete modelling language as a part of their process. Those that define a language, frequently have authors who create their own symbols and semantics, rather than using a standard language like UML.

The aim of this chapter is to address these issues. We will present an overarching conceptual overview of the things to be modelled in a multi-agent system, specified in DYNASTAT. We select a subset of agent characteristics from the overall conceptual framework that DYNASTAT will model. We define the phases for the conceptual modelling effort. Having defined DYNASTAT as an abstract methodology, we then use UML to work through an example of DYNASTAT in practice. Readers may choose to select a different language, but we have used UML.

In Section 2, some examples of the existing medical multi-agent systems and an overview of current progress in the modeling of multi-agent systems are provided. In Section 3, we discuss the main features of the DYNASTAT methodology. In Section 4, basic UML concepts and a number of UML diagrams that will be further used within DYNASTAT framework are discussed. In Section 5, a number of illustrative examples of the use of UML to model medical multi-agent systems are explained. Some of the advantages of medical multi-agent systems are summarized in Section 6. Finally, the chapter is concluded in Section 7.

1.3 Literature Review

1.3.1 *Multi-agent systems in the medical domain*

Multi-agent systems are increasingly being used in the medical domain. Some of these systems are designed to use information available through specific medical and health institutions, while other systems use information from the Internet.

Examples of institution-specific multi-agent systems examined included Agent Cities [12], AADCare [8] and MAMIS [5]. The Agent Cities [12] agents enable the user to access his/her medical record, to search for medical centres on the basis of a given set of requirements, or to request and make appointments. AADCare [8] is a decision support system used by physicians. It matches the patient's record against the predefined domain knowledge. This domain knowledge can contain knowledge regarding a specific disease, clinical management plans, patient records etc. MAMIS [5], Multi-agent Medical Information System, provides ubiquitous information access to physicians and health professionals.

BioAgent [11], Holonic Medical Diagnostic System [20] and Web Crawling Agents [18] are examples of internet-based multi-agent systems. BioAgent [11] is a mobile agent specifically designed to retrieve information

about genome analysis. It travels to multiple locations, collects information from each location and integrates information before despatching an answer to the user. Holonic Medical Diagnostic System [20] matches the comprehensive computer readable patient record (computer readable patient pattern, CRPP) against the information available through the Internet in order to provide enough evidence for correct diagnosis of the patient. Different web crawling agents [18] use information available on mutated genes to fetch information about associated diseases.

The need for the design of a multi-agent system for the purpose of dynamic information retrieval regarding common knowledge of human diseases has been explained in [7, 6]. The BioAgent system could be used for this purpose but it needs to be significantly modified as we are interested in information about human diseases and not genome analysis. Holonic Medical Diagnostic System does not retrieve information about human disease but it uses information specified in the patient record to provide enough evidence for correct diagnosis of this patient. Web crawling agents focus only on the genetic causes of human diseases. A large number of complex diseases are also caused by a number of environmental factors such as stress, family conditions, climate etc. Additionally, numerous human diseases are multi-dimensional not only in terms of their causes but also in terms of symptoms, causes and types. The availability of a systematic overview of the different aspects of human disease will make a significant contribution to the advancement of human disease research and practice.

1.3.2 *Methodologies for modeling multi-agent systems*

We have examined a number of methodologies used to perform conceptual modeling of multi-agent systems. These include:

The methodology defined by [10] models three distinct areas of multi-agent systems including components (agent, system, environment, data and task model), processes (cooperative, competitive and neutral processes) and finally, constraints (identity, capability, class capacity and time constraints). The modeling constructs include the use of state-variables. The agent characteristics modeled for this methodology include autonomy, goals, connectivity, problem coverage and knowledge about other agents. There were no graphical diagrams in the paper and so it appears this methodology is abstract and is not tied to a particular modeling language.

The BDI methodology [15] models a system using three different types of models: object, dynamic and functional. Object models represent data

structure, relationships and operations of objects in the system. States, transitions, events, actions, activities and interactions that describe system behavior are modeled using dynamic models, while flow of data during system activity, inside and between system components, are modeled by functional models.

In BDI methodology, agents are defined from an external perspective by defining an agent model (hierarchical relationships between agent classes) and an interaction model (responsibilities, services, control relationships between agent classes). From an internal perspective, agents are defined by belief model (the environment and internal state of an agent), goal model (set of agent's goals) and plan model (plan set). The modeling using this methodology includes an extension of object oriented techniques. The modeling language defined within the methodology has UML-like constructs for some diagrams, and OO type symbols for others. Most of these symbols appear to be author-defined.

The RoMAS [21] methodology models agent roles and appears to have a single diagram type: use cases. In this respect the use cases are captured, roles are identified and agents are generated from roles. The notation for the diagram is a type of E-R notation and appears to have author-defined symbols.

We noted that most of the existing methodologies had a specific modeling language included in the abstract process defined to perform conceptual modeling. However, in most cases, this modeling language was not a standard one such as UML, which could be extended in a controlled way using stereotypes. Each language was usually author-defined, with the symbols used having no standard meaning from one paper to another.

1.3.3 *UML modeling of multi-agent systems*

A number of researchers have attempted to use UML in modeling multi-agent systems [2, 9, 3, 4, 13]. This research effort has contributed significantly to the development of a standardized methodology for modeling multi-agent systems. However, a number of issues have emerged from these research efforts that need to be addressed. These issues include poorly detailed presentations, modified semantics of UML diagrams without using stereotypes and limited use of stereotypes. Lack of detailed presentation is evident in [4] where VisualAgent presents some initial ideas, but does not use many existing UML diagrams or stereotypes. The semantics of UML diagrams has been modified by many researchers. For instance, Kavi [9]

uses smiley faces, thought clouds, and the like, which have not been defined by the UML semantics. Da Silva [3] changes the semantics of rectangles without the use of a stereotype. Odell [13] defines a rectangle to be an Agent/Role combination in sequence diagrams which means that a single Agent can be represented by multiple rectangles, each rectangle representing a single role. Again, this was done without the use of a stereotype. We also noted a limited use of stereotypes. For example, Odell [13] did not use stereotypes to define an agent.

This chapter makes a number of contributions to the existing modeling approaches.

We show how different types of UML diagrams can be used to model different aspects of multi-agent systems. For instance, a Sequence Diagram is used to model the social nature of agents, Composite Structure Diagrams to model agent internals and an Activity Diagram to model the processes implemented by agents.

We do not change the semantics of existing symbols defined in the UML standard. No symbols are introduced into UML diagrams in this chapter that are not defined in the standard. The messages in the Sequence Diagram are an exception to this.

We use stereotypes to extend and modify the meaning of existing symbols in UML diagrams. Most often, this is done in order to have a symbol that usually represents a class or object, represent an agent instead.

1.4 DYNASTAT methodology

The authors propose a methodology that is termed DYNASTAT. This stands for: Dynamic and static methodology for modeling multi-agent systems.

The methodology is a process that is independent of any particular conceptual modeling language and includes:

- An overview of what a multi-agent system is, and what individual agents are, in conceptual terms.
- A selection of specific agent characteristics that are to be modeled.
- A discussion of what is required in modeling terms, for each of these agent characteristics.
- Modeling phases.

1.4.1 *Conceptual Overview of Multi Agent Systems*

In conceptual terms, multi-agent systems have a number of characteristics [14, 19, 16].

Any multi-agent system usually has an environment in which agents operate. The agent may perform one task at a time, or it may multiplex many tasks. Each agent usually has the ability to observe the environment. Agents are usually able to observe part of the environment, but not all of it. The environment is *centralized* when it has a single controlling agent and *non-centralized* when many agents control the system. If changes in the environment are caused only by the agent itself, it is termed *static*, but if other things also contribute to the changes, the environment is called *dynamic*. If the action of an agent is unrelated to the past and future acts of the agent, we are speaking of an *episodic* environment. In a *sequential* environment, there is a relationship between current and past and/or future actions of the agent. In *deterministic* environments, the finish state of the environment can be uniquely predicted from the start state and the agent actions. Conversely, in *stochastic* environments, the finish state cannot be uniquely predicted based on the information above. The environment has either a finite number of states (*discrete*) or a very high number of states (*continuous*).

Every agent has a purpose, and so there is usually a need to measure agent effectiveness. *Performance measure* is used to measure how successful an agent is in carrying out its tasks. A rational agent is one that makes decisions to maximize its future performance. A performance measure must define the criteria of success. *Omniscience* is a characteristic that indicates an agent can predict the outcome of all its decisions and actions, under every circumstance.

Each agent has certain *beliefs* and plays different *roles*. It also has *goals*, particular outcomes the agent is programmed to achieve. The agent follows precisely defined *procedures* and also performs calculations based on formulas. The agent can *reason* and use knowledge to decide on a particular course of action in a particular environment. This usually means choosing a particular plan or path to a goal under certain conditions. An agent can *plan* a particular path to a goal. An agent makes use of *sensors* to observe the environment and effectors to act on the environment.

Every agent performs various actions. Usually, some event *triggers* an agent's actions. The trigger can be a request, a specific occurrence in the environment or an internal process. An agent usually uses *plans*, a collection

of acts, ordered in a workflow with decision points. At each *decision point*, the agent will be able to *select* a particular plan. The agent will also *schedule* and *prioritize* its actions.

In addition to what an agent is, and what it does, each agent is capable of change. An agent is able to *learn* from the results of action and modify internal intelligence in order to make better decisions. Additionally, an agent can *adapt* and reconfigure itself in response to changes in the environment.

Agents are sociable and are often involved in *coordinations*, *cooperations*, *communications* and *negotiations*. Agents work together, cooperate to achieve particular goals, exchange information and come to agreements.

In terms of the implementation of agent-based systems as software, the *mobility* (ability of agents to move from machine to machine), *fault tolerance* (reliability of an agent), *distributed* (execution of a single agent over many machines with different threads of execution) and *persistent* (agent runs continuously) nature of agents need to be modeled.

The authors have chosen a subset of the agent characteristics to model in this chapter. They include agent's beliefs, roles, procedures, reasoning and decision making. We also model the sociable nature of agents including coordination, cooperation, communication and negotiation.

1.4.2 *Modeling Phases*

Regardless of the modeling language chosen for modeling, a number of steps have to be taken.

The specific diagrams to be used have to be identified. A mapping between particular characteristics and particular diagrams has to be established. The specific symbols in each of the diagrams to be used need to be identified and the meaning of each must be understood by the modeler.

The problem domain has to be understood in general terms. A specific application that is the subject for modeling needs to be identified. A number of scenarios would have to be analyzed in order to decide on what will be modeled. These scenarios are then worked through in the context of specific diagrams in order to complete the models for each case.

This completes our discussion of the DYNASTAT methodology. It is a methodology that applies to modeling multi-agent systems irrespective of the particular modeling language used. In this chapter, UML 2.2 is used to realize this methodology for the purpose of example. The next section discusses this modeling language as a prelude to a working example.

1.5 Use of UML 2.2 in the framework of DYNASTAT methodology

1.5.1 *Overview of UML 2.2*

The UML standard defines 13 different diagram types [17, 1]. Each of these diagrams can contain a distinct set of symbols, which can be combined in a standard set of ways to model the system being examined. Each of these symbols has a specific meaning defined in the UML standard.

There are many versions of UML that have been released over time. The current version in use is UML 2.2 and this version is in beta. This is the version that has been used for diagrams in this chapter.

One important fact about UML is that it can be extended in a controlled way through the use of stereotypes. This essentially means that an existing UML symbol can be qualified to change the standard meaning it has in the standard. The symbol is qualified by a <<stereotype>> name text block somewhere in the symbol, or by defining a completely different icon. Either way, the idea is to allow modelers to effectively add an additional symbol to the pre-defined set in the UML standard and include that symbol in UML diagrams used in modeling.

1.5.2 *Usage of UML for modeling multi-agent systems*

In some UML diagrams, there is at least one symbol that represents a class or object. Stereotypes are defined for some of these symbols to have them represent agents instead. Some diagrams also model messages between classes and objects, and these messages now represent inter-agent communication.

The Sequence Diagram and Composite Structure Diagram are two diagrams used in this chapter that fall into this category.

Other UML diagrams have a symbol that represents something independent of a class. For instance, a Use Case Diagram has a symbol to represent system functionality and an Activity Diagram has a symbol to represent an Activity in a process. In this case, stereotype symbols are not required to represent agents.

UML also models events that cause state transitions between a finite number of states. Although this chapter does not explore UML diagrams that use state machines, future work could further extend the paradigm shift above by having state machines model the internal state of an agent,

the state of the agent environment, or the system as a whole. As above, this would require stereotyping some of the symbols used in state diagrams.

1.5.3 *Selected UML diagrams*

This section discusses the basic diagram types in this chapter and defines the symbols used in the UML 2.2 diagrams drawn to model the multi-agent system used as an example. The symbols here are a subset of what is included in the UML standard, but provide a context so that the semantics of the drawn diagrams are clear.

Four basic UML diagram types have been included: use case, composite structure, sequence and activity diagram. They are discussed below.

Use Case Diagram

[17, 1]

This is a diagram that models the functionality of the system used by external end-users of that system.

The symbols most commonly used in this diagram type are shown in Figure 1 and include: (i) Actor, which represents an external person or system using the system being modeled; (ii) Use Case, which represents a unit of functionality; (iii) Association Relationship, which indicates the usage of a unit of functionality by an external actor; and (iv) Generalization Relationship, which indicates the inclusion of a specialization of the functionality in the parent.

In order to model a multi-agent system, stereotyping of these symbols was not required since the functionality of the system and its actors are not particularly centric to either classes or agents.

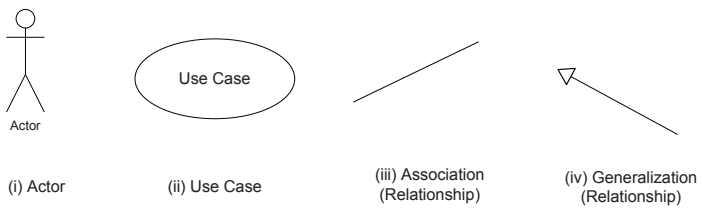

Fig. 1.1 Use Case Diagram (Symbols)

Composite Structure Diagram

[17, 1]

This is a diagram defined in the UML standard to represent the internals of a class.

The symbols commonly used in this diagram type are shown in Figure 2 and these include: (i) Classifier, a rectangle to represent the class as a whole; (ii) Parts, rectangles inside the outer rectangle to represent different areas of processing of the class; and (iii) Ports, small squares at the borders of the outer rectangle and represent the connections between the class and the interfaces that are external to it.

In this case, the outer rectangle was stereotyped to represent an agent rather than a class. The inner rectangles were stereotyped to represent the parts of an agent, rather than a class. Finally, the ports were stereotyped to represent agent roles, rather than connectors to the external environment of the class.

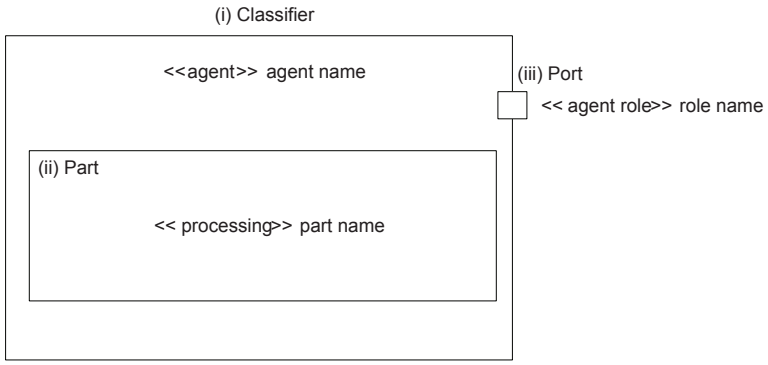

Fig. 1.2 Composite Structure Diagram (Symbols)

Sequence Diagram

[17, 1]

This is a diagram defined in the UML standard that represents inter-object or inter-class interaction by way of messages.

Generally, the commonly used symbols are shown in Figure 3 and these include: (i) Heads, rectangles that represent the classes that are communicating and include a dotted line that goes down the page to represent

time passing; (ii) Messages, arrows that go between the dotted lines and represent messages being passed between the classes that are communicating; and (iii) Activation - long rectangle that goes over the dotted line to represent the fact that the class has a run time instance that is operating.

In this case, the heads were stereotyped to represent agents, rather than classes. Ports were used at the edge of these rectangles to indicate roles played by the agents. This is a significant extension of the existing UML standard in semantic terms, and stereotypes were used to do it. Individual messages were not stereotyped, although they now represent inter-agent, rather than inter-class communication.

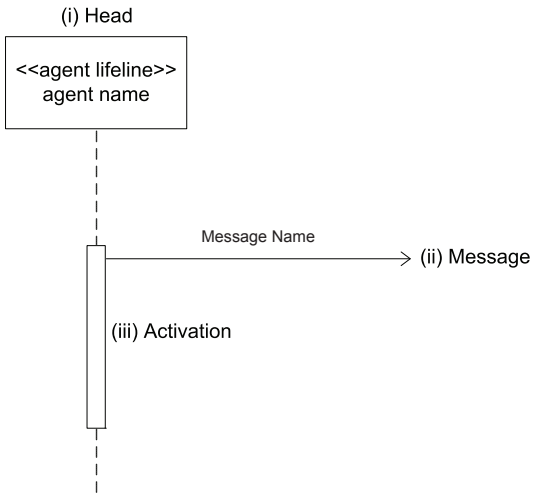

Fig. 1.3 Sequence Diagram (Symbols)

Activity Diagram

[17, 1]

The UML Activity Diagram models any process in a system. A process may be implemented in a class, an agent, or any other type of construct.

Generally, the commonly used symbols are shown in Figure 4 and include: (i) Initial node which is represented as a black dot to indicate the start of the process; (ii) Activity node which is represented as a rectangle with rounded edges to indicate a unit of processing; (iii) Decision node

which represents a decision or choice of an agent or class down different paths; (iv) Fork which is a heavy black bar with one arrow entering and many arrows exiting to represent one path of action diverging into many; (v) Join which is a heavy black bar with many arrows entering and one exiting to represent many paths of action converging into one; (vi) Termination node which is represented as a cross within a circle to signify the termination of a course of action, but not the completion of the entire process; and (vii) Final node which is represented as a black dot within a circle to indicate the end of the entire process.

In this case, it was not necessary to stereotype since the Activity Diagram that models a process independently of the process is packaged into a class, component or agent. Since this is the case, it is not necessary to change the semantics of the symbols using stereotypes.

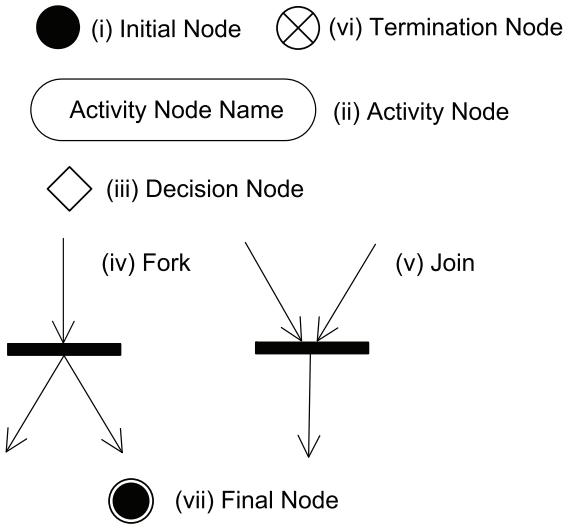

Fig. 1.4 Activity Diagram (Symbols)

1.6 UML to model medical multi-agent systems

In building the system, the following two scenarios from the medical domain were used:

(1) medical researchers retrieving information from various information re-

sources, and collecting evidence to design and conduct medical experiments, and

(2) general practitioners who have standard consultations with patients (primary care).

1.6.1 *Medical Researcher (UML Examples)*

The process of medical research is quite different from the day-to-day operation of the health care system. Most scientific medical research involves the use of certain rules to set up and conduct experiments and also the use of certain principles when analyzing the data collected to reach experimental conclusions. The system modeling for a medical researcher is focused on the intelligent retrieval of information from various information sources in order to collect evidence for designing and conducting medical experiments effectively. The retrieved information can also be used to formulate a hypothesis which is later tested through data collection and analysis. The system modeled in this chapter will automate the integration of existing sources of data and will assist the researcher to formulate an initial hypothesis. The researcher will analyze the experimental data and test the hypothesis with the help of data mining techniques.

1.6.1.1 *Modeling Information Retrieval*

The proposed technological solution is an ontology framework of the Generic Human Disease Ontology (GHDO) which contains generic information regarding human diseases [6]. The ontology is designed as a superset of four ontologies capturing Disease Types, Phenotype (Symptoms), Causes and Treatments. The GHDO ontology is used to support the information retrieval process within the multi-agent structure. It is used to derive the Specific Human Disease Ontology template (SHDO template). The SHDO template specifies the information the user has requested in relation to a particular disease. For example, if a user is interested only in the causes of a specific disease, the SHDO template will contain only the causes sub ontology of the GHDO. When agents feed instances into this SHDO template after information retrieval, the SHDO template is converted into a Specific Human Disease Ontology (SHDO) which is presented to the user as the answer to his/her query.

Four different agent types within the GHDO-based multi-agent system are defined here:

(1) The Interface Agent constructs Specific Human Disease Ontology (SHDO) templates from GHDO. The SHDO template is sent to the Manager Agent once it has been generated.
(2) The Manager Agent assigns tasks to various Information Agents. The Manager Agent must have the appropriate expertise to do this; namely, it must have knowledge of the task structure. This is specified by the SHDO template.
(3) The Information Agents retrieve the requested information from a wide range of biomedical databases. Depending on the requested information, an agent may be required to retrieve information about disease types, symptoms, causes and/or treatments. The Information Agents send the retrieved information to the Smart Agent.
(4) The Smart Agent collects and analyzes the information received from Information Agents, filters it for the relevant information and then uses the data to populate the SHDO template. The SHDO is sent to the Interface Agent to be presented to the user as the answer to his/her query.

Interaction between different agents is presented in the Sequence Diagram in Figure 5. The Sequence Diagram is used to model the inter-agent communication between all four agent types in the system. The diagram describes the process of a user querying the system about a particular disease. The generated SHDO template will be passed from Interface to Manager Agent, who will then message the Information agents to request data on the disease, symptoms, causes and treatments. This information will be gathered by Information Agents, who will then pass it to a Smart Agent that is responsible for integrating it into the SHDO template. Finally, that template will be returned to the Interface Agent, who will return it to the user. The agents themselves are modeled by head symbols that are stereotyped as <<agent lifeline>>. The Information and Smart Agents play more than one role and this is modeled by a port stereotyped as <<agent role>>.

In Figure 6, a Composite Structure Diagram is used to model the internals of the Smart Agent. It has a classifier stereotyped as <<agent>> to name the Smart Agent. It has four distinct areas of processing including: a controller to coordinate the actions of the agent, a part to analyze the data given to it by Information Agents, a part to filter that information for what is relevant and finally, a processing unit to assemble what is relevant into the SHDO template, before that populated template is returned to the

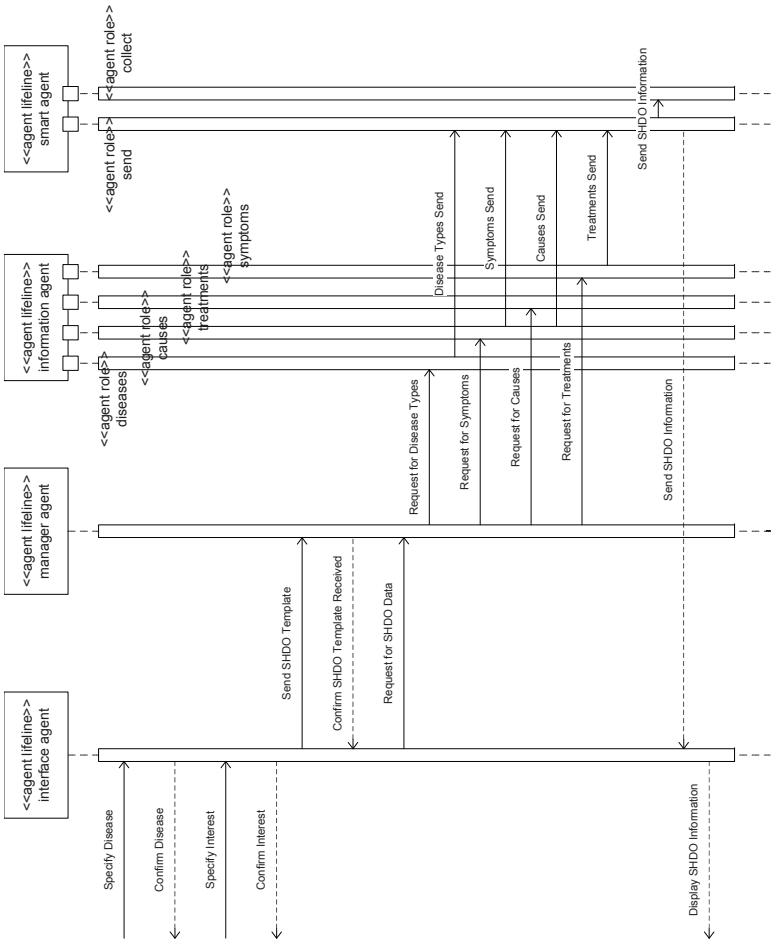

Fig. 1.5 Sequence Diagram representing interactions between different agent types in the GHDO-based multi agent system. The different agents are collaboratively working in this intelligent information retrieval system.

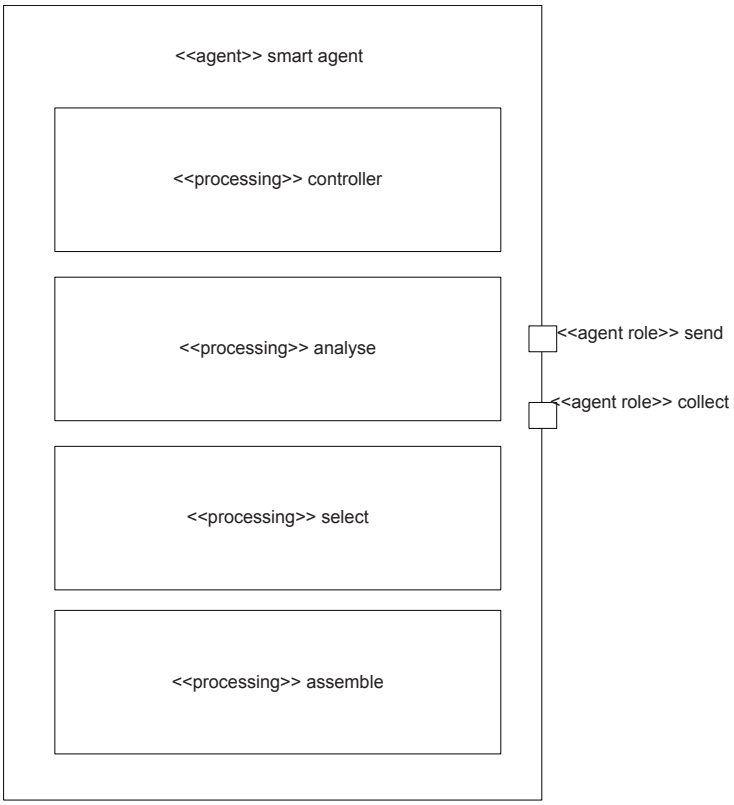

Fig. 1.6 Composite Structure Diagram to represent internal structure of Smart Agent.

user. These areas of processing are stereotyped using the <<processing>> stereotype. The two ports titled 'send' and 'collect' represent the roles played by the Smart Agent to send and receive data to other agents. They are stereotyped as <<agent role>>.

1.6.2 *Modeling Data Collection and Mining Experiments*

In Figure 7, a Use Case Diagram illustrates the different functions involved in conducting a medical experiment. The medical researcher usually selects a category of experiment to be carried out. Various categories exist and they range from clinical trials to public surveys. Researchers also conduct a literature review which helps in formulating a hypothesis, which will be refined and verified by the experiment. A system such as that described in

Figure 5 can assist the researcher in gathering relevant data. The researcher chooses a subject group and obtains their consent to participate in the experiment. There will be a functionality to assist the researcher to assign the subjects into different experimental groups according to whether or not the control variable is present. For example, if researching lung cancer, the control variable may be smoking. Subjects would be divided into two groups according to whether or not they smoke. Response variables would include symptoms leading to lung cancer. External variables could include things that may cause variation in the response variable, but that must be constant across the treatment and control group. For instance, it may be the climate where subjects live.

Based on the selected variables and existing knowledge in the field, questionnaires will be designed. The experimental subjects will complete the questionnaires and the data will be collected in this way. It is possible to design an online questionnaire system which will greatly facilitate systematic data collection and analysis. Once data has been collected, it would need to be cleaned and formatted to prepare it for analysis by data mining software. The purpose of the data mining software would be to establish patterns and relationships. In the case of lung cancer, the researcher may be looking to see if there is a causal relationship between smoking and lung cancer. This information would permit the researcher to evaluate the hypothesis formed at the beginning of the experiment. Finally, there would be functionality to record experimental conclusions. In this case, it would mean deciding that smoking causes lung cancer, or that it does not.

1.6.3 *General Practitioner (UML Examples)*

The system modeling for general practitioners is focused on the initial contact between doctors and patients and the establishment of preliminary diagnosis, prognosis and treatment of disease. It also includes following up to determine whether a recommended treatment has been effective and whether the patient has been cured of the disease. There is a health care system in most first world countries which includes a network of general practitioners and hospitals, both public and private. Usually, the initial point of contact for most patients who are ill is a general practitioner in a suburban clinic. After an initial consultation, the doctor usually makes some type of diagnosis and may ask the patient to give a sample of some type (for instance, blood or urine) which will be sent to a pathology lab for analysis. The results may refine or confirm the initial diagnosis. The

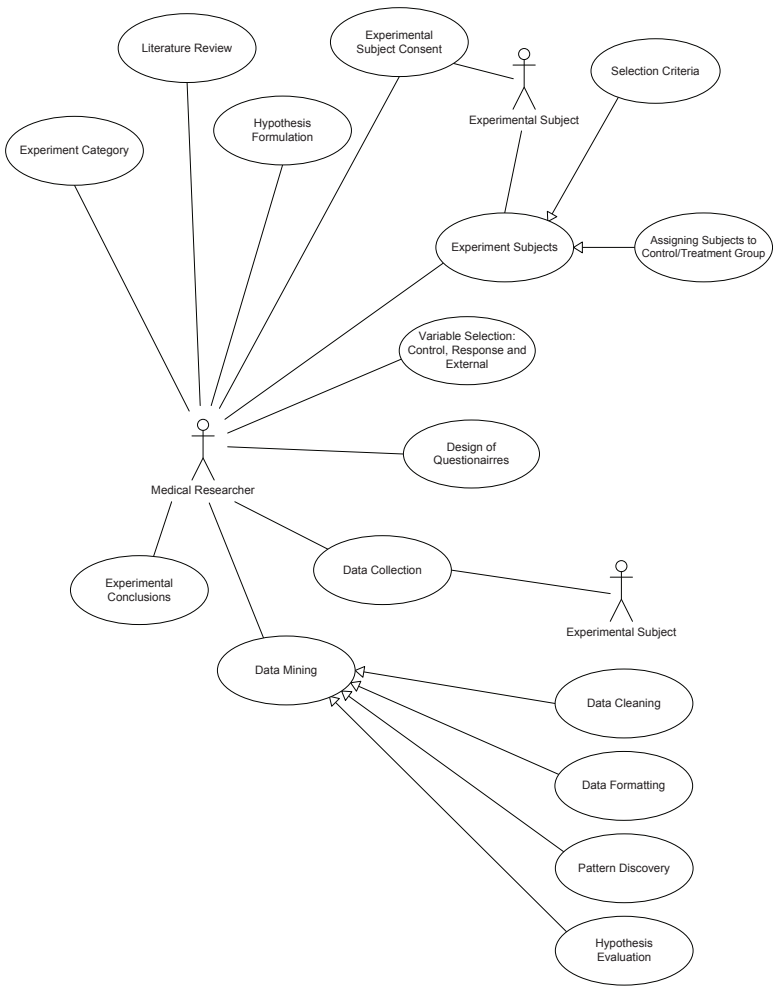

Fig. 1.7 Use Case Diagram representing the conducting of medical experiments including hypothesis formulation, data collection and data analysis.

doctor may also suggest some type of medical imaging technique for the same reason. One example of medical imaging would be a CT scan of the patient's brain to see if s/he has a brain tumor. If the patient has a certain type of disease, the general practitioner may recommend that s/he see a specialist. The specialist may recommend surgery, and that would require the patient to be admitted to hospital.

Finally, the doctor may recommend some type of therapy from, for

example, a clinical psychologist or physiotherapist. Whether the treatment modality is a drug, therapy or surgery, the goal of treatment is usually to cure the disease, or to manage the symptoms.

What follows in this section is a conceptual model for a standard consultation by a general practitioner.

There is a Use Case Diagram and also an Activity Diagram to model this scenario.

The Use Case Diagram for the general practitioner involves system functionality in a number of areas. Firstly, the doctor may use the system to perform a diagnosis. This functionality will be specialized by gathering both symptoms and signs. There will also be functionality related to what the doctor decides in relation to externals: medical imaging, sampling for analysis in a pathology lab, or referring the patient to a specialist, all of which will be all specialized forms of the ordering functionality. Once the diagnosis has been established, the system may assist the doctor with a prognosis and also prescribing treatment. Finally, the doctor may use the system to follow up on the effectiveness of the treatment and close the case. Both the patient and doctor are actively involved in the entire consultation process as reflected in the diagram.

This Activity Diagram models the process of the doctor's consultation. It begins with the doctor recording the patient's signs and symptoms. The doctor formulates a hypothesis and then has a decision to make. S/he will usually either order additional tests or analyze the test results to confirm his/her hypothesis, or the doctor will refer the patient to a specialist, who then takes over from the general practitioner. If the doctor is to follow through with the patient to closure, s/he will confirm a diagnosis, indicate the prognosis to the patient and prescribe treatment. Finally, s/he will close the case, thereby terminating the consultation process.

1.7 Possible Applications

In terms of medical research, multi-agent systems have the potential to provide evidence that will help the researchers obtain relevant information and progress with knowledge more rapidly. The information obtained through the intelligent GHDO-based multi-agent system can also assist in the formulation of hypothesis and provide foundational knowledge that will help obtain new knowledge in the field through the conduction of various medical experiments. An experimental process has been shown in which

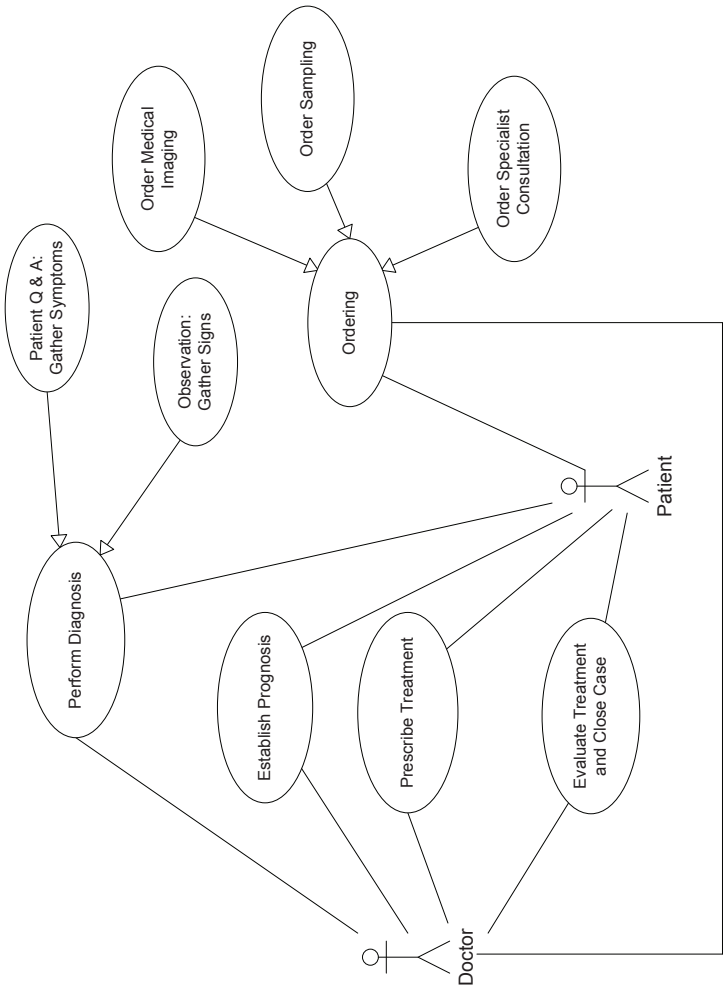

Fig. 1.8 Use Case Diagram for general practitioner consultation.

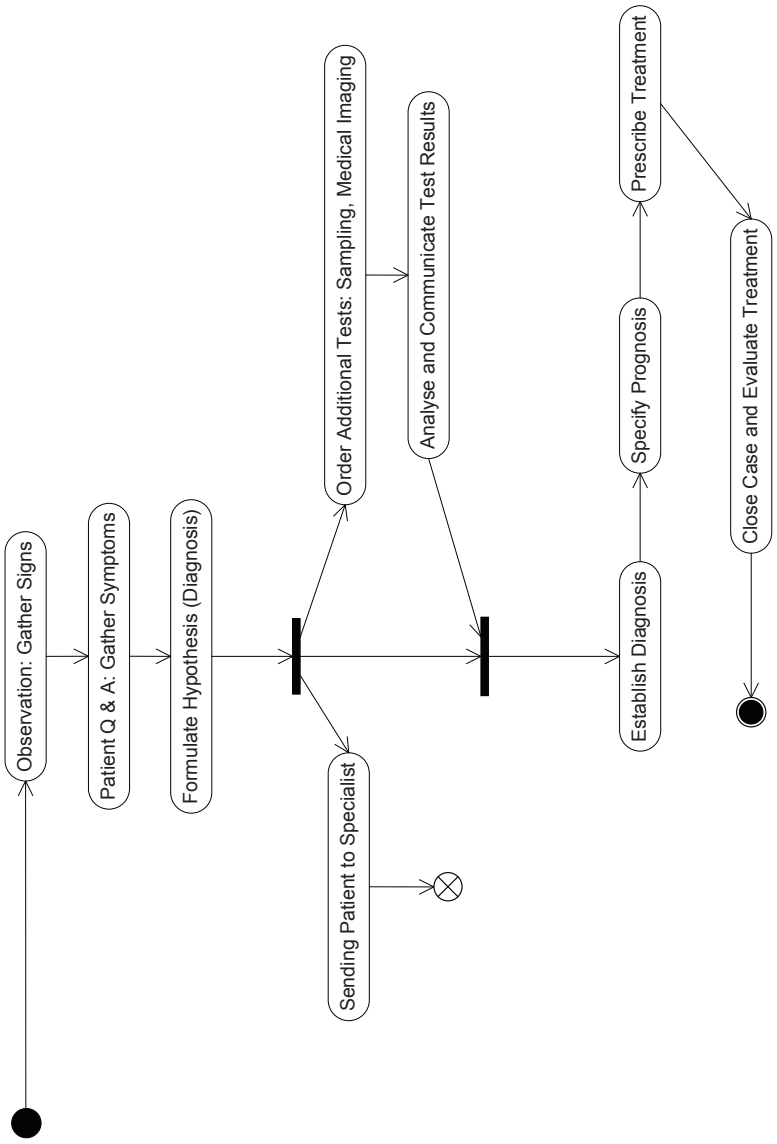

Fig. 1.9 Activity Diagram for general practitioner consultation.

data is obtained through questionnaires and analyzed with the help of data mining algorithms. Data mining algorithms can analyze experimental data and find patterns that a human researcher may easily miss. Additionally, the implementation of a multi-agent system also helps increase the control and simplifies the standard experimental process.

In terms of a standard consultation, the implementation of multi-agent systems has numerous advantages such as controlling the standard process of consultation, assisting the doctor to diagnose the patient illness, supporting decision-making processes in regard to requirements for pathology tests, medical imaging or specialist referrals and enabling data collection that will help doctors to improve the standard consultative process.

It is hardly conceivable that a software system can be designed to replace a human agent. Nevertheless, some interplay between the researcher/doctor and the system may refine the process being used and provide support for decision-making at various stages.

1.8 Conclusion

This chapter has established a methodology called DYNASTAT which provides a conceptual overview of multi-agent systems, a selection of agent characteristics modeled by the methodology and also a discussion of what needs to be modeled in particular for each agent characteristic.

After the presentation of this methodology, the authors chose UML 2.2 as a modeling language to realize DYNASTAT for a particular system in the multi-agent domain. We explained four UML 2.2 diagram types including: the use case, composite structure, and activity and sequence diagrams. A particular multi-agent system was then chosen from the medical domain in order to model these UML 2.2 diagrams.

Many of the areas of weakness in the current research have been addressed with a coherent example to permit the reader to follow a particular line of thought. It is to be noted that this chapter does not venture into the more complex paradigm of assigning roles to agent groups. The complexities of basic agent modeling have been explored before going further in order to build a solid foundation. Nevertheless, more complex paradigms in multi-agent systems such as modeling the environment, effectiveness of agents, change and adaptation of agents and implementation of agent systems may be considered in future work published by the authors.

Bibliography

[1] (2009). Omg unified modeling language (omg uml), superstructure, version 2.2, URL `http://www.omg.org/spec/UML/2.2/Superstructure`.

[2] da Silva, V., Noya, R. and de Lucena, C. (2005). Proceedings of the 4th international conference on autonomous agents and multiagent systems, aamas, in *Using the UML 2.0 Activity Diagram to Model Agent Plans and Actions*, pp. 594–600.

[3] da Silva, V., Noya, R. and de Lucena, J. (2004). *Software engineering For multi-agent systems II : research issues and practical applications*, chap. Using the MAS-ML to model a multi-agent system (Springer), pp. 129–148.

[4] de Maria, B., da Silva, V., Noya, R. and de Lucena, C. (2005). Visualagent: A software development environment for multi-agent systems, in *Proceedings of the 20th Brazilian Symposium on Databases and 19th Brazilian Symposium on Software Engineering, SBBD*.

[5] Fonseca, J., Mora, A. and Marques, A. (2005). Mamis - a multi-agent medical information system, in *Proceedings of 3rd IASTED International Conference on Biomedical Engineering, BioMED*.

[6] Hadzic, M. and Chang, E. (2005a). Medical ontologies to support human disease research and control, *International Journal of Web and Grid Service* **1**, pp. 139–150.

[7] Hadzic, M. and Chang, E. (2005b). *Self-organization and Autonomic Informatics*, chap. Ontology-based multi-agent Systems Support Human Disease Study and Control (IOS Press), pp. 129–141.

[8] Huang, J., Jennings, N. and Fox, J. (1995). An agent-based approach to health care management, *International Journal of Applied Artificial Intelligence* **9**, pp. 401–420.

[9] Kavi, K., Kung, D., Bhambhani, H., Pancholi, G. and Kanikarla, M. (2003). Extending uml for modeling and design of multiagent systems, in *Proceedings of the 2nd International Workshop on Software Engineering for Large-Scale multi-agent Systems, SELMAS*.

[10] Malyankar, R. and Findler, N. (1998). A methodology for modeling coordination in intelligent agent societies, *Computational & Mathematical Organization Theory* **4**, pp. 317–345.

[11] Merelli, E., Culmone, R. and Mariani, L. (2002). Bioagent - a mobile agent system for bioscientists, in *Proceedings of the Network Tools and Applications in Biology Workshop Agents in Bioinformatics, NETTAB*.

[12] Moreno, A. and Isern, D. (2002). A first step towards providing health-care agent-based services to mobile users, in *Proceedings of the 4th International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pp. 589–590.

[13] Odell, J., Van Dyke, H. and Bauer, B. (2001). *Agent-Oriented Software Engineering* (Springer-Verlag).

[14] Parka, S. and Sugumaran (2005). Designing multi-agent systems: a framework and application, *Expert Systems with Applications* **28**, pp. 259–271.

[15] Rao, A., Georgeff, M. and Kinny, D. (1996). *Agents Breaking Away*, chap. The methodology and modeling technique for systems of BDI agents (Springer Berlin / Heidelberg).

[16] Ren, Z. and Anumba, C. (2004). Multi-agent systems in construction state of the art and prospects, *Automation in Construction* **13**, pp. 421–434.

[17] Rumbaugh, J., Jacobson, I. and Booch, G. (2004). *The Unified Modeling Language Reference Manual (2nd Edition)* (Addison Wesley).

[18] Srinivasan, P., Mitchell, J., Bodenreider, O., Pant, G. and Menczer, F. (2002). Web crawling agents for retrieving biomedical information, in *Proceedings of International Workshop on Agents in Bioinformatics, NETTAB-02*.

[19] Tweedalea, J., Ichalkaranjeb, N., Sioutisb, C., Jarvisb, B., Consolib, A. and Phillips-Wren, G. (2007). Innovations in multi-agent systems, *Journal of Network and Computer Applications* **30**, pp. 1089–1115.

[20] Ulieru, M. (2003). *New Directions in Enhancing the Power of the Internet*, chap. Internet-enabled soft computing holarchies for e-Health Applications (Springer), pp. 131–166.

[21] Yan, Q., Shan, L., Xin-Jun, M. and Zhi-Chang, Q. (2003). Romas: A role-based modeling method for multi agent systems, in *International Conference on Active Media Technology*.