Towards an Engineering of Multi-Agent Eco-Systems

Khan, A.Basit
Department of Computer and Information Science (IDI)
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
Email: basit@idi.ntnu.no

Mihhail Matskin
Department of Computer Science (IDI)
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway and The Royal Institute of Technology (KTH) Stockholm, Sweden
Email: misha@imit.kth.se

Abstract—Different approaches for engineering multi-agent systems have been proposed over time. However, there is a lack of attention towards the social aspects of multi-agent systems. In this paper we present an approach for engineering multi-agent systems, which considers multi-agent systems as digital ecologies. Different agent ecologies can work together and act as digital ecosystem. In such ecosystem the social aspects of the agents are well defined and all the subcomponents of the multi-agent system work together to achieve a fine integration of the whole ecosystem. Instead of focusing on internal model of software agents our methodology attempts to capture the dynamic and social behaviors of agents by focusing on the cooperation, coordination, negotiation and management attributes of the system. From implementation point of view, the proposed engineering methodology is grounded in our multi-agent framework and the concepts developed using this methodology can be mapped to implementation easily. The framework provides support for integration of web-services and uses peer-to-peer approach for resource discovering.

Index Terms—Multi-Agent; Software Engineering; AGORA; Ecology, Ecosystem; Services

I. INTRODUCTION

Agent-based computing has provided significant benefits both for academy and industry. The ongoing trend of applying software agents in many different domains is likely to continue. This is due to the fact that agents are used as a complementary approach to object-oriented style of software development. Triggered by wide scale adoption of software agents in many research domains (such as peer-to-peer system, grid systems [8], [10], [15], artificial intelligence [12], e-learning [2], [4] [etc...]) academy and industry have came up with many different frameworks which either use or propose different forms of multi-agent systems. Each domain has treated software agents differently and the use of software agents in many cases is dependent on the domain. In spite of advances in developing agent-based systems the question which got less attention till now is “how to properly engineer a multi-agent system?”. We believe that growing adoption of agents will require more general approach to engineering agent-based software systems. This is because of domains/technologies need to converge in order for them to be useful to each other and to the users. Furthermore, in order for an engineering approach to be adoptable, it should be supported by tools and grounded in a framework. These tools would allow to map concepts developed during design phase into corresponding elements of the framework. In this paper we propose a higher level multi-agent system engineering approach which considers multi-agent systems as a digital ecosystem consisting of one or more digital ecologies; such an approach allows the future changes/evolution and adapts readily to open environment such as Internet. While our approach pays due attention to the internal model of entities (i.e. Internal Model of Software Agents); this however, is not the main theme of our work. In this paper we are focusing more on the social aspects of the system as a whole. The work presented here is grounded in our multi-agent framework called AGent Oriented Resource mAnagement (AGORA) [3]. This means that we not only propose an agent engineering methodology, but we also provide tools supporting it.

The rest of this paper is organized as follows. In the next section we present our understanding of relationship between a digital ecosystem and ecologies. Next section provides introduction to AGORA multi-agent framework. Next we discuss in details our engineering methodology. The methodology is then elaborated by demonstrating how it is applied in a mobile learning system. Then we provide a brief overview of well established multi-agent based software engineering methodologies and relate them to our work. Lastly we conclude by briefly discussing the usefulness of our work in the research and development of digital ecosystems for small and medium size enterprises (SMEs).

II. ECOLOGICAL PERSPECTIVE OF DIGITAL ECOSYSTEM

Our methodology is based on the fact that digital ecosystem(s) can be viewed as a collection of one or more digital ecologies. This conceptualization is depicted in figure 1. The concept of digital ecologies is adopted from Nardi’s book “Information Ecologies: Using Technology With Heart” [9], where a digital ecosystem is defined as “...self-organization and evolutionary models from Biology to be applied to software, based on the assumption that such biological behavior of the software, if attained, is likely to optimize the catalytic function of the ICTs in question for socio-economic growth.
An information ecology is defined as an environment where users along with their social practices have harmonious relationship with technology and among themselves and there is a considerable balance among the actions of all those who exist in the information space. An information ecology is defined as a system of different sub components (i.e. ecologies).

A digital ecology consists of the following main subcomponents as described in [9]:

1) **System**: it is to say that there is a relationship among the participants of the ecology and therefore whole system acts like a single entity. One participant getting affected or failing to function has an impact to the others and thus can result in disturbance of the system behavior.

2) **Diversity**: of skills and knowledge is another important aspect of an information ecology, a healthy information ecology is likely to have participants with diversity of skills and knowledge.

3) **Co-evolution**: means contiguous change in a balanced way. An information ecology is never static, there is always a change that allows the knowledge and tools inside ecology to evolve.

4) **Keystone Species**: Keystone species are those who are extremely important for the survival of the ecology. Unavailability of these species can be catastrophic for the ecology. In an information ecology these are the entities/services whose presence is necessary for the survival of the whole ecology.

5) **Locality**: whom the technology belongs to? what business, cultural region and users are using it? what is it used for? In an information ecology a particular technology is associated with a particular group. These entities define the meaning of technology, thereby putting technology directly under the control of it’s context of use.

We argue that an ecosystem shall not be considered as a monotheistic entity, instead one should take a perspective view to it as a system of different sub components (i.e. ecologies). This can be considered as an extension to the natural metaphor of an ecosystem. This view of information systems depicts the ground reality of the existing situation of the SMEs. A digital ecosystem may consist of many different businesses. The businesses in the digital ecosystem may be cooperative or competitive in their nature or they can be both depending on situation. However, when their social aspects (communication, coordination and negotiation) are well defined a balance can be achieved between cooperation and competition. Thus, allowing the ecosystem to function as a system of fine grained integration.

III. INTRODUCTION TO AGORA FRAMEWORK

The AGORA multi-agent framework [3] is designed using the principle of a marketplace. The core purpose of AGORA framework is to support collaborative activities in a distributed environment. An AGORA is an aggregation of four main subcomponents, figure 2 depicts the main components of an AGORA. These components are discussed below.

1) **AGORA Services**: are the main functional units provided by an AGORA. All the other subcomponents functionally are designed to ensure correct life cycle of the service provided by an AGORA.

2) **AGORA Managers**: are the manager agents of an AGORA. Each AGORA has three Managers. The first one is **Manager Agent** who controls access to all the services provided by AGORA. It also performs all management related activities of AGORA. The second manager agent is the **Coordinator Agent**. This agent is responsible for providing coordination support for the services and activity supported by AGORA. The main responsibility of this agent is to ensure that the rules of an activity are followed and if it detects any problem it is responsible for taking corrective actions. The last type of manager agent is **Negotiator Agent**. In many cases not all the activities (e.g. service invocation etc.) proceed as they are planed. If the corrective actions taken by **Coordinator Agent** are not able to achieve the normal
behaviors of involved parties, then it is the responsibility of Negotiator Agent to take actions to bring back the system to the normal state.

3) **AGORA node**: this node holds all the shared information about an AGORA. It mainly keeps the information about services provided by an AGORA and locators of manager agents. The Manager Agent provides access to the AGORA services. Other information kept in this node is the secret keys for private communications among the Manager Agents of an AGORA.

4) **AGORA registered agents**: are the external agents who may be interested to use the AGORA services and the activities provided by AGORA.

The main highlights of AGORA framework are (1) it is able to unify different kind of services such as W3C [14], FIPA [1] and RESTful services. (2) It allows peer-to-peer search for the services provided by different AGORAs. (3) It allows supporting all the mechanisms related to communication, coordination and negotiation for keeping control over the services provided by the AGORA.

There may be several AGORAs running concurrently in the system providing their own services. Each AGORA can be connected to other AGORAs through Parent-Child and/or registration mechanisms. This allows to search for services both in a central and P2P fashion. AGORA framework is deployed with base/default functionality of the main components and it allows the system developers to extend and override the functionality as they require.

IV. AGORA’S ENGINEERING METHODOLOGY

As discussed earlier we provide an ecological approach for engineering multi-agent systems. The concepts developed using this approach can be programmed using the AGORA framework discussed in the previous section. The proposed methodology employs two perspectives: **active and passive**.

A. **Active perspective**

In this perspective the multi-agent system is considered as a set of active system entities interacting among themselves and with external entities. The focus here is on considering dynamic/social aspects of the system at the runtime. The behavior of the system is considered from the point of view of the user/agent who will be using the system (i.e. the occasion and reasons when external entities would want to interact with the system). Generally speaking during active system perspective participants and cooperative points are defined. In this perspective we consider the following two steps:

1) **Understanding the SYSTEM**: During the first step the cooperation point or the cooperative activities the system will engage in are identified. In principle, we analyze overall social aspects of the system to be designed. In this step the interested partners (i.e. Agents) are considered as the central point of the activity. This allows previewing the system’s functionality from the users point of view. The major focus here is to identify the possible activity partners and the points of cooperations. Such cooperative points can be identified by understanding the domain/business for which the system is being designed. For example, a possible cooperation point between a financial bank and a retailer would be when the retailer want to verify a credit card, or to make a transaction from the credit card’s holder account to the retailer’s account. Thus the cooperative activities are identified based on possible scenarios. The result of this step is a list of all perceived/possible points of cooperation.

2) **LOCALIZING system’s dynamics**: During the second step coordination and negotiation support required for each cooperative activity is identified. We consider negotiation and coordination as the vital aspects of any activity in multi-agent systems. This is depicted in figure 3. The coordination and negotiation aspects surrounds the cooperative activity and they are required in order to execute an activity in a perceived way. These activities are considered as cooperative nodes which are mapped to AGORAs and they may require coordination and negotiation support from other sub-components of their respective AGORAs.

![Fig. 3. Conceptualization of management, coordination and negotiation support for a cooperative activity](image)

In our framework, while the Manager agent is responsible for providing access to the AGORA services, the Coordinator and Negotiator agents are responsible for providing coordination and negotiation support. In terms of implementation the required coordination and negotiation support is presented as services supported by Coordinator and Negotiator agents. In this way when a cooperative activity is executed it is possible for an agent to know who takes cares about coordination and negotiation.

The AGORAs identified in this step can be created for different applications as needed at the runtime and many instances of such AGORAs can be started in the system. These AGORAs are called **application level AGORAs**. This step also sets the requirements for the plication services (i.e. **Required Application Services**) which are needed to perform the cooperative activity. This means that we make a differentiation between the services [see section IV-B1] required to perform a cooperative activity and the services required to coordinate and negotiate about a cooperative activity. The AGORAs identified during this step are further elaborated from the passive perspective (see below). By virtue of this
step the system’s dynamic/social behavior is localized to the type activities it needs to perform.

More cooperative points can also be identified and added later during the system’s life time in order to adopt to changes. This is necessary because of it may be difficult to envision all the possible applications supported by the system at the design time. Therefore, such step can be performed many times during the system’s life time.

B. Passive Perspective

In the passive perspective the main goal is to define services which the system should have in order to execute cooperative activities identified in the pervious phase. Another goal of this perspective is to identify the system’s components who would be the providers or hosts of the required services. At this stage the system is treated as a static collection of functionalities. Whole process is divided into the following main steps as it is depicted in figure 4.

1) **Defining the system’s DIVERSITY:** During the first step the main purpose is to derive the list of **Required Application Services** which are needed in order to execute the cooperative activities. The requirements for the types of operations for these services are set in the previous phase (see section IV-A2). These services are called application services as they allow the system to perform its function in a cooperative activity. These services are mapped into the Manager Agent of respective AGORAs identified in the previous step. The list of these services is further refined by iterating against the required functionality. The output of this phase is the minimum number of services which are necessary for the system to function.

2) **Identifying system level services for CO-EVOLUTION:** During this step more services are identified to extend the system’s functionality for future adaptability and extensibility. While the main purpose of the AGORAs and services identified in the previous two steps were to perform the application level functions, the purpose of these services is to perform system level operations. Therefore these are generic services which can be used without consideration of their application/invocation context. Our understanding is that, the application level services are specific to their respective applications and it is wise not to mix the system level functionality with application service. Consequently more services and applications can be supported by the system when the core services are available. This makes the system more capable of supporting new applications and activities. The services of this phase are called **system level services**. They provide basic/common system level functionality and allow more complex applications to be built on the top of the basic functionality allowing the system to evolve for future changes. An example of such service can be a service which checks the credentials of a user.

Since the AGORA framework allows adding new AGORAs and services in the system dynamically the system can evolve as changes and new requirements occur in the future.

3) **Identifying KEYSTONE AGORAs:** In the last step the **System Level AGORAs** are identified. These are AGORAs which would be the providers of the system level services identified in the previous step. The number of identified AGORAs may vary depending on the nature of the system and the number of identified services. At least one instance of each such AGORA can exist in the system. Collectively all the AGORAs identified in this step define the core architecture of the agent system. Some examples of such AGORAs could be ontology manager AGORA, repository manager AGORA etc. These AGORAs are the providers of the core functionality of the system.

V. ELABORATION USING A USE CASE

This section elaborates the methodology discussed in the pervious section, by using it for engineering a multi-agent system for mobile learning called FABULA [5], [13]. The architecture of the system is discussed in details in [4]. Our intention here is only to highlight the engineering methodology and it’s application. Figure 5 provides an overview of FABULA multi-agent system.

Core description of FABULA can be summarized as follows:

1. **FABULA system is a service based learning support system for informal mobile learning with city as the learning context.** The system should be able to support learning which happens in LEARNING GROUPS, it should be possible for the members of learning groups to COMMUNICATE WITH EACH OTHER. The system shall provide services in order for learners to COLLABORATE WITH EACH OTHER. The system shall be intelligent to TAKE INTO ACCOUNT THE CONTEXT of the learner in order to adopt the system behavior according to the places/spaces which surround the learner. Such intelligence

---

1The phrases in caps are intended to highlight the main system’s functionality
shall be built over ONTOLOGICAL REASONING AND SEMANTIC INTEROPERATION. The system shall MANAGE AND RECOMMEND LEARNING CONTENT to the learner based on the nature of activity the learner is engaged in. System shall be able to TAILOR DIFFERENT LEARNING ACTIVITIES based on the interests of learners, these interests are maintained by the system through user profiles.

A. Applying the engineering approach

Based on the description presented in the previous section, we now apply our approach to construct the multi-agent system.

- **Understanding the SYSTEM (Active Perspective):**
  From the problem description we identify cooperative activities where participants may require coordination or negotiation support. We have identified five such activities and four kinds of AGORAs. These activities mainly include 1) **Authoring Activity**: for allowing the learners to create learning artifacts together. 2) **Collaboration activities**: to allow users to concurrently work over and discuss about learning artifacts. 3) **Communication activities**: To allow the learners to communicate with each other (text and voice). 4) **The group management activities**: to manage different learning groups.

The fifth type of activities consists of application specific activities which are intended for unforeseen applications which may be supported by FABULA later.

- **LOCALIZING system’s dynamics (Active Perspective):**
  For each cooperative activity we identify the coordination and negotiation support. Because of space limitation we will not discuss all possible cooperative/negotiative points, but just briefly mention some of the most important ones. For example, in case of Collaborative learning activity, it is required that the state of concurrently edited learning artifact stays consistent and shall not be corrupted, therefore ensures coordination of concurrent changes. Furthermore, if two learners belong to same learning group, but provide different answers to a question, then it shall be possible for these learners to negotiate about their understanding.

This cooperative/negotiative support is mapped to AGORAs and to coordination and negotiation services for these AGORAs. The following AGORAs are identified to be the hosts of services required to support the categories of activities discussed above: 1) **Group AGORA**: One such AGORA is created in the system every time a new learning group is created. 2) **Space AGORA**: To represent and manage the social aspects of vicinity where learning takes place. 3) **Application Specific AGORA**: To host application specific services. 4) **Application Dependent AGORA**: they are the AGORAs which are not defined prior to the system, but may be added in order to extend the system functionality.

- **Defining the system’s DIVERSITY (Passive Perspective):** For each kind of cooperative activity the services are identified. The services identified for collaboration support are as follows: (1) Social network representation to the user. (2) Place representation and management (e.g. available users in the learning environment). (3) Management and maintenance of social relationships among different learning networks. (4) Management of places where learning takes place.

- **Identifying system level service for CO-EVOLUTION (Passive Perspective):**
  By analyzing the results of the active system perspective and main functional requirements eight main categories of system level services arranged in two different layers were identified for the FABULA system. Each category consists of different services, important categories and their respective services.

  1) **Community/Group Management Services**: This category of services manages all the groups which are created in the system. These services also ensure that the information about different groups is persisted in the system for later use.

  2) **Information/Content Retrieval and Management Services**: These services are responsible for managing the learning content in the system. (1) Service for storing, retrieval, organization and searching the content in the content repository. (2) Cataloguing Service, (3) Content Archival Service, (4) Digital rights management Service, (5) Services for content reputation and recommendation (social content filtering and collaborative filtering).

  3) **Meta-Data and Semantic Services**: (1) Metadata creator service, (2) Terminology service, (3) Ontology query service, (4) Semantic rea-
soning service.

4) **Application/Learning Activity Composition Services**: These services allow the system to dynamically create different learning applications from the services available in the system

5) **User Management Services**: These services allow the system to manage profiles for the users.

Other three categories of system level services are event management services, security services and discovery services.

- **Identifying KEYSTONE AGORAs (Passive Perspective):**
  
  After identifying the services, the next step is to identify the system level AGORAs which would provide the services. Without going into the details of each AGORA identified for FABULA system – we only provide the names of these AGORAs\(^2\). (1) Context Manager AGORA, (2) Group Manager AGORA, (3) Ontology Manager AGORA, (4) Learning Applications AGORA, (5) Repository Manager AGORA, (6) User Manager AGORA.

**VI. RELATED WORK**

Different multi-agent system engineering approaches have been proposed over time. We briefly discuss only two representative methodologies. Our aim is not to provide a detailed analysis/overview of all agent methodologies but instead we discuss how our approach is positioned among the selected representative methodologies. The following two methodologies are discussed: GAIA [7], AAII [6].

GAIA methodology adopts a stepped process, while it focuses on the individual aspects of an agent. The first step is to find what different roles an agent may have. Each role has four different attributes responsibility, permissions, activity and protocols. These four attributes have further attributes. Once the roles are defined, they are then mapped to agents and agents are instantiated.

AAII also focuses on the individual model of the agent, and divides the agent into two different models: internal and external models. The internal model is concerned how agent performs its function internally. This mainly concerns the reasoning aspects. The External model focuses on the relationships between agent.

In short, existing approaches to engineering software agents mainly focuses on the internal design of individual agents, defining the services of the agents and their communication protocols. These methodologies target the internal architecture of agent such as reasoning agent, learning agent, Belief Desire Intentions (BDI) based agent, formal methods for verifying the correctness and liveness properties.

The ecological view allows to look at the agent system from the external point of view and focuses more on the social aspects. The resultant multi-agent architecture is designed form the scratch by keeping in mind the services, negotiation, coordination and management aspects of the software agent.

All the identified agents are necessary for the system to operate and no extra functionality is added to the system unless its needed. The whole system acts as a coherent unit of fine grained integration.

**VII. CONCLUSION**

The multi-agent engineering methodology presented here is general enough to be applicable to different domains. Our approach does not look at individual agents as the main system entities to be designed, instead it inspects the cooperative activities which the agents should perform and then defines the agents which are needed to perform the cooperation. Instead of adopting a bottom up approach to solve a problem it takes a top down approach, with the negotiation and coordination as the main points of attention. If adopted this methodology would allow a business to define its own multi-agent system (i.e. ecology) according to the nature and type of the business and let it be part of a larger digital ecosystem which could be managed by the AGORA framework.

**VIII. ACKNOWLEDGMENT**

This research is funded by Norwegian research council NFR-project (176841/S10).

**REFERENCES**


\(^2\)The self descriptive names should provide some background information and nature of these AGORAs and the services they host