Chapter 17
Detecting Emergent Behavior in a Social Network of Agents

Mohammad Moshirpour, Shimaa M. El-Sherif, Behrouz H. Far, and Reda Alhajj

Abstract An effective and efficient approach in designing software systems to describe system requirements is using scenarios. A scenario, commonly shown as a message sequence chart or a sequence diagram, is a temporal sequence of messages sent between system components. Scenarios are appealing because of their expressive power and simplicity. Moreover due to the clear and concise syntactic of scenarios, they can be used to analyze the system requirements for general validity, lack of deadlock, and existence of emergent behavior. Emergent behavior or implied scenarios are specifications of behavior that are derived from compiling of all requirements together but are not explicitly specified in the set of scenarios. Although emergent behavior is not necessarily unwanted, nevertheless it is useful for system designers and engineers to be aware of its existence. Defining requirements using scenarios and conducting consequent analysis has been done for distributed systems as well as multi-agent system. In this research the requirements of a social network are described using scenarios. The scenarios are then used to detect emergent behavior using a systematic methodology. This is illustrated using a prototype of a social network of MAS for semantic search that blends the search and ontological concept learning.

M. Moshirpour (✉) · S.M. El-Sherif · B.H. Far
Department of Electrical and Computer Engineering, University of Calgary, 2500 University Drive NW, Calgary, T2N 1N4, AB, Canada
e-mail: mmoshirp@ucalgary.ca; smmelshe@ucalgary.ca; far@ucalgary.ca

R. Alhajj
Department of Computer Science, University of Calgary, Calgary, AB, Canada

Department of Information Technology, Hellenic American University, Manchester, NH, USA

Department of Computer Science, Global University, Beirut, Lebanon
e-mail: alhajj@ucalgary.ca

T. Özyer et al. (eds.), *The Influence of Technology on Social Network Analysis and Mining*, Lecture Notes in Social Networks 6, DOI 10.1007/978-3-7091-1346-2_17,

17.1 Introduction

Social networks have received enormous international interest in recent years, especially after the ubiquitous use of the internet as a communication medium [1]. Social network is a set of individuals and relationships between them. It can be represented as a set of actors (nodes or agents) that have one or more kind of relationships (ties) among them. The importance of the social network is due to its effect in diffusion of information among actors included within the network [2]. In this paper, when we use the term social network, we are referring to the abstract meaning of this term and not the social networks in popular culture such as Facebook or twitter. In other words, by social networks we mean a set of nodes that can communicate with each other via a network where they may have different types of relationships (ties) and different levels of strength of these relationships depending on several factors. The strength of ties represents how close the nodes are with respect to one another. We propose a semantic search framework that is based on a multi-agent system (MAS). Each MAS controls a document repository and each repository has a different ontology. The agents cooperate with each other to select the best subset of documents suitable for a semantic search query sent by a user. A query is in the form of searching for keywords in the context of one or more concepts. We use social networks in our system to communicate between agents from different MAS, which allows for proper handling of the concepts in the query. This allows agents to find peers that may have close but slightly different concepts in their respective ontologies.

Due to the integrated nature of social networks, gathering comprehensive and correct requirements for such software systems can be challenging. Scenario-based specification is an effective and efficient way to describe the behavior of a variety of software systems such as multi-agent systems (MAS) and distributed systems. Scenarios enable engineers and designers to describe system's functionality using the partial interactions of the system elements. In this research, the approach of scenario-based software engineering (SBSE) has been followed to represent the requirements of social networks.

There are two main ways of representing scenarios, namely, Sequence Diagrams (SD) developed by the object management group (OMG) [3] and Message Sequence Charts (MSC) which were developed by the International Telecommunications Union (ITU) [4]. In this paper MSCs are used to represent scenarios.

There are several advantages of using scenarios such as expressive power and simplicity. However, there are also several challenges such as weak partial ordering semantics that may result in missing some behavioral requirements particularly for software systems with distribution of control such as distributed systems, MAS and social networks. For instance, because each scenario gives a local and partial story of interaction between two or system components, the challenge is how the behavior of the whole system can be constructed from those scenarios and more importantly whether the derived behavior is acceptable or not. In the analysis of social networks, the interacting system components are individual nodes.

The model which describes the behavior of each system component (i.e. node for social networks) is usually called *behavioral model,* and the procedure of building the behavioral model from a scenario-based specification, is called *synthesis of behavioral models*, or simply, *synthesis process*. A commonly used model for behavioral modeling of individual components is the state machine. There are several reports on the procedure of converting a set of scenarios to a behavioral model expressed by state machines [5–10]. In the synthesis process, one state machine will be built for each node. The state machine includes all the interactions of a particular node based on the messages that it receives or sends. Theoretically, the behavior of the network can be described by the union (parallel execution) of all the state machines of the individual nodes.

One of the challenges during the synthesis process, is *implied scenarios* [11–14], also known as *emergent behavior*. An implied scenario is a specification of behavior that is in the synthesized model of the system but is not explicitly specified in the set of scenarios. This usually happens when several autonomous components need to handle a joint task as a group in a shared environment where control is also distributed. Although emergent behavior is not always unwanted, it is extremely useful for system designers and engineers to be aware of its existence. In this paper we use an example of a social network of MAS for semantic search to demonstrate the ideas.

The structure of this paper is as follows: in Sect. 17.2 some background on social networks, particularly about the relationships between nodes is presented. The case study for the social networks of MAS for semantic search is given in Sect. 17.3. In Sect. 17.4 system behavioural modeling is explained and the detection of indeterminism and emergent behaviour is discussed in Sect. 17.5. Finally conclusions and future work are presented in Sect. 17.6.

17.2 Social Networks

As mentioned previously, a social network is a set of individuals and relationships between them. Social networks are often shown using graphs which consist of several nodes representing actors of the network (such as agents) and arrows or lines representing the relationships between the actors [15] as shown in Fig. 17.1.

This section contains a brief overview on the structure of social networks and the measurement of tie strengths which of particular interest to this research.

17.2.1 Connection between Nodes

One of the most important features of a social networks is the connections between its nodes. It is necessary to identify the way the actors are embedded within a relation. There are many different types of relations. The most important types

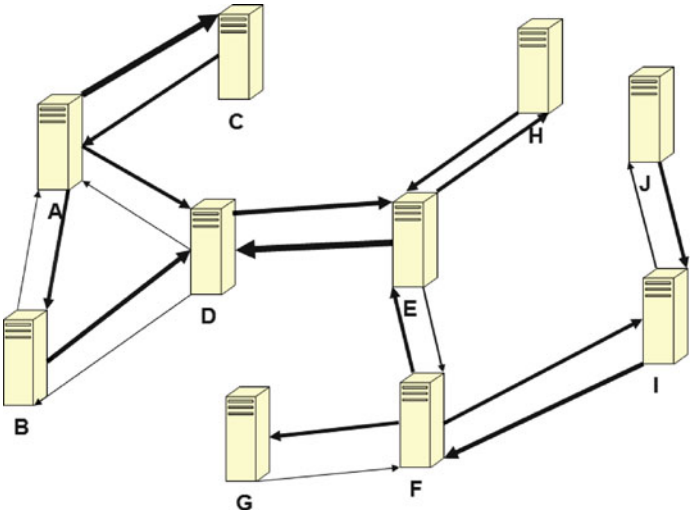

Fig. 17.1 Example of agents connected in a social network. The *arrow head* represents the direction of the relationship and the *thickness of the line* represents the strength of the relationship

of relation are the *dyad* relation (two actors involved in the relation) or the *triad* relation (three actors involved in the relation). In the dyad relation between A and B there are four possibilities; A has a relation with B, B has a relation with A, neither A nor B has relation with each other or both A and B has a mutual relation.

The *size* of the network is the number of nodes within the network. The *density* of the network is the ratio between the existing connections of the network to all possible connections. The number of possible connections within a network is $k(k-1)$, where k is the number of the nodes in the network in the case of the directed graph or the symmetric network. In the case of an asymmetric network, the number of possible connections equals $(k(k-1))/2$ because in this case the connection between A and B is the same as the connection between B and A. Note that when the number of the nodes increases linearly, the number of possible relations within the network increases exponentially.

The *degree* of a node is the number of connections in which this node is involved. In the case of a directed network, there are two types of degrees, *in-degree* and *out-degree* representing the number of relations towards and outwards from the node respectively. The degree of a node is a very important property of a node. The in-degree represents how powerful the node is. It corresponds to the amount of knowledge it has. The larger the in-degree the more knowledge it has. The out-degree represents how influential this node is. The larger the out-degree of a node the more it affects the surrounding nodes.

The *reachability* of an actor to another is the ability of one actor to reach another one by any sequence of steps. In directed graphs there may be a case that A can reach B but B cannot reach A. In symmetric data if two actors cannot reach each other so there will be a kind of division within the network.

The *distance* between two nodes is the number of steps required to exchange information between them. This is an important property of nodes in a social network because it specifies how effective a node is and how far knowledge at a node can propagate.

17.2.2 Tie Strength in Social Networks

Most social networks only consider if there is a relationship between two members or not. They do not consider the strength of the ties between members. Recently more attention has been paid towards this property (i.e. the tie strength). The strength of the tie is affected by several factors. Grannovetler [16–18] proposed four dimensions that may affect the tie strength:

- The duration of the relationship
- The intimacy between the two individuals participating in the relationship
- The intensity of their communication with each other
- The reciprocal services they provide to each other

In other literature, such as in the works of Wellman and Wortley [19], it is argued that emotional support strongly affect the strength of the tie between any two members in a social network. Other factors, such as socioeconomic status, educational level, political affiliation, race and gender are also considered to affect the strength of ties [20]. Moreover the structural factors, such as network topology and information about social circles may affect the tie strength [21].

17.3 Case Study: Social Networks of MAS for Semantic Search

Traditional search engines depend on the number of occurrence of given words in documents. Semantic search depends on understanding the meaning of the concepts used in the context of other words. It then tries to retrieve the related documents to these concepts. The backbone of semantic search is the semantic interoperability which is the main ingredient for notation extraction from the search phrase. Using social networks in this system provides great flexibility; especially in dealing with concepts in ontologies. It allows MAS to understand the meaning of the same concept even though its definition might be slightly different in each agent's ontology.

In our framework, we assume that in a society of n MAS ($\text{Mas}_1, \text{Mas}_2, \ldots \text{Mas}_n$), each MAS ($\text{Mas}_i$) manages a repository R_i and consists of different agents, each with its own responsibilities. These repositories use different ontologies to represent their knowledge. The ontologies used within the repositories are not necessarily the same. Also, agents do not need to understand concepts in the same way. The most important concept here is the ability of agents to understand each other.

In our framework, we try to make agents from different MAS able to communicate with each other and to understand each other by interacting through a social network. In this social network, the strengths of relationships (ties) between agents are not the same.

The tie strength between agents is a good indicator of how close these agents are to each other. It helps in both semantic search and concept learning modules.

In the learning module, when the user sends a search query to a local agent, the local agent checks for new concepts that are not defined in the repository. If the local agent found new concepts, it asks its peers/neighbors for this new concept. These selected agents are now teacher agents. The choice of teacher agents depends on the tie strengths between the local agent and all its peers. The teacher agents teach the local agent the new concept by sending some illustrative examples representing this concept. If conflicts occur during the learning process, the local agent depends more on the examples sent by teachers with which it has stronger relationships (i.e. higher tie strengths).

The semantic search module starts when the local agent is sure that all new concepts in the search query are learnt correctly. The local agent annotates its own repository using keywords in the search statement. The annotation procedure re-categorizes the repository by conforming to concept hierarchy. Then the local agent searches the annotated repository for the search query and returns the results (called local results). At the same time the local agent sends the search query to all its neighbors (in this case they are called remote agents). Each of these remote agents searches its own repository by annotating it with the search keywords. Then each remote agent returns more results (called remote results). After gathering all the results of the search from all remote agents, the local agent tries to rank the results. The purpose of this ranking is to decide the order in which the local agent delivers the results to the user. The raking process depends also on the tie strength between the local agent and each of remote agents. The stronger the tie between them, the higher a rank is given to results returned from this particular agent.

17.3.1 Semantic Search Process

We have devised a spiral workflow to incorporate both search and concept learning in the semantic search process [22]. The spiral workflow and its suggested scenario are shown in Fig. 17.2. On one hand, search engines should be capable of responding to the requests according to agreements with the concept learning module. On the other hand, annotation procedures of search engines can be done on the fly based on the obtained concepts instead of depending on fixed predefined ontological concepts. This view exposes the intrinsic relationship between concept learning and semantic search in a heterogeneous environment. In such an environment, concept learning and semantic search are treated equally as basic roles, involved in the process which support each other to achieve their own goals by enriching the set of ontological concepts and reducing ambiguity of the search, respectively.

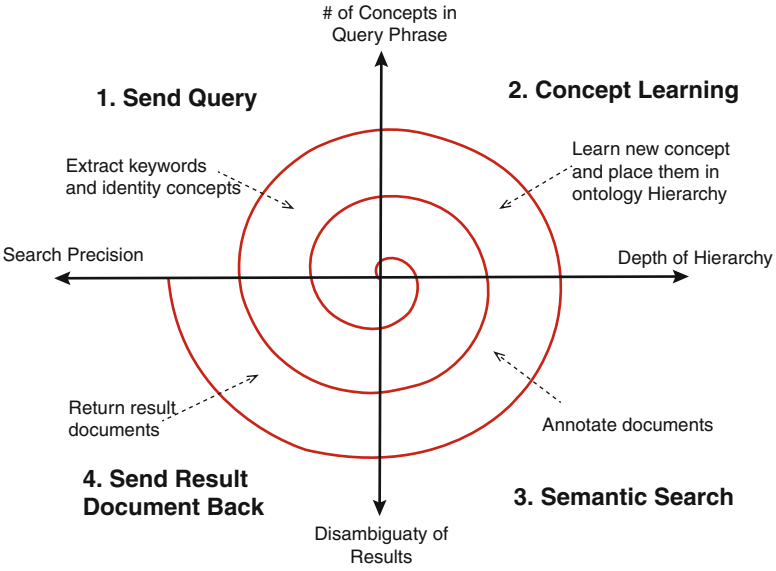

Fig. 17.2 Spiral workflow between semantic search and concept learning

Following the spiral process, concept learning module and semantic search take actions alternately.

The scenario of this process is as follow:

- The system is initiated by the user when he/she sends the search query to one agent (we call it local agent).
- Concepts are extracted from the search query.
- The concepts in the search query are compared by those defined with the ontology in the local repository. This comparison is essential to enable the local agent to find out the new concepts in the search query that are not defined in its local repository.
- If new concepts are found in the search query, the local agent requests the other agents (remote agents) to teach it these new concepts. (This step represents the initialization of the concept learning process.)
- The concept learning mechanism is continued until the local agent learns the new concepts adequately.
- After learning all new concepts, the concept hierarchy in local repository is reorganized to add the new concepts in their proper position.
- The annotation procedure is then performed on the fly on all the concepts in the local repository; old and newly learned concepts. UIMA [23] is used to enable search and classification within each document repository.
- In the same time, the local agent broadcasts the search query to all remote agents to search their local repository and send back the result documents.

The local agent collects the returned documents from remote agents and ranks them using social networks before retrieving them to the user.

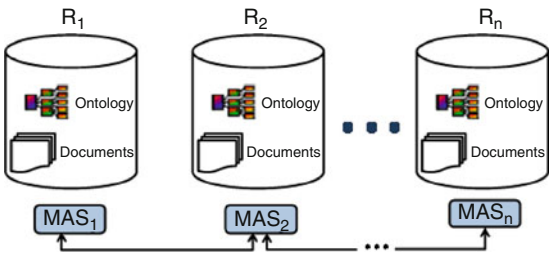

Fig. 17.3 The system architecture

17.3.2 Semantic Search Infrastructure

In the previous section, we describe the importance of the concept learning module in the semantic search system. The main obstacle in the concept learning system is the complexity of ontological heterogeneity solution. In order to solve this problem we propose to leverage of the power of a multi-agent system to use it in the infrastructure of our semantic search system. A great social network of MAS is used to set up the backbone elements that communicate with each other. As shown in Fig. 17.3, each MAS controls a repository that uses an ontology to cover a specific area of knowledge. Each repository contains some documents that are related to concepts defined within the ontology. Using MAS allows the system to hide the search complexity from the user.

17.3.3 Prototype System Architecture

Figure 17.4 shows different agent roles in each MAS in our prototype system. These roles are defined below.

Query Handler: This role involves accepting search query and processing it by extracting concepts from it. Also it is responsible for broadcasting the query statement to all the neighbor repositories.

Concept Manager: This role involves finding the new concepts in the search query and broadcast it to all neighbour agents in order to be learnt.

Concept Learner: This role involves maintaining and confirming newly learnt concepts; including creation of taxonomies of interested domain. It is also responsible of broadcasting these concepts to all group members to share searching for it. Moreover it rearranges local repositories with the newly learned concepts.

Document Annotator: This role involves annotating the documents in local repository and filtering them according to the search keywords, then returning back the filtered documents.

Peer Finder: This role involves detecting cooperative peers (agents) in the social network that communicate with the current agent with a relationship.

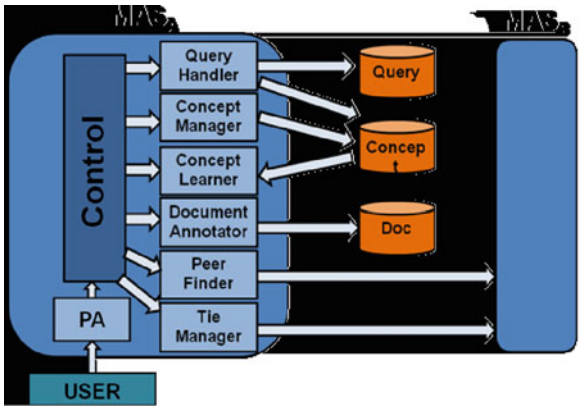

Fig. 17.4 Roles of different agents

Tie Manager: This role involves keeping track of common concepts between peers in the social network and the interactions which occur between those peers in the learning process. Tie manager is able to change the strength of tie between peers dynamically. It is also responsible for setting the initial strength of the relationship between agents.

17.4 System Behavioral Modeling

The requirements for the social network of MAS defined in the previous section are described using partial message sequence charts (pMSCs); defined formally in Definition 1.

This system consists of a large social network of multi-agent systems. Each MAS contains a repository and connects to other MAS on the network for concept learning purposes. The structure of the social network is ever-changing as MAS continue to strengthen or weaken the ties among them based on their commonalities. The managing of the ties is done by two agents of "peer finder" and "tie manager" which are part of each MAS in the network as shown in Fig. 17.4. For the purpose of this case study, it is assumed that the strength of ties between each two nodes in the network (i.e. each MAS) depends on the following criteria:

1. Number of times they have been able to successfully cooperate
2. Number of peers they have in common

Criteria (a) indicates that MAS which have the most concepts in common and continue to have a working relationship will have stronger ties. On the other hand (b) is derived from the social network concept that for node A to be balanced in its relationship (or friendship) with B and C, then A must believe that B and C are peers.

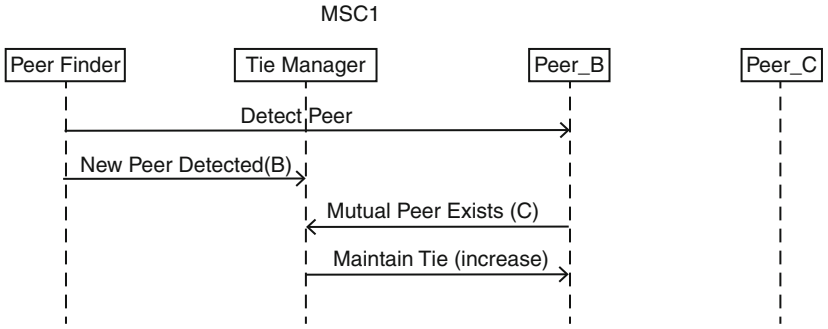

Fig. 17.5 The tie between MAS A and B are established based on their mutual peers

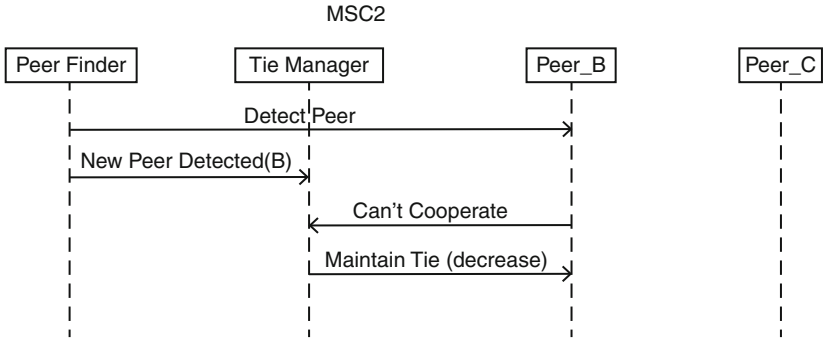

Fig. 17.6 The relation between MAS A and B is decided based on their ability to cooperate on a given query

In the following case study, three arbitrary MAS of A, B and C are selected from the social network. It is assumed that ties already exist between A and C as well as between B and C; however there are no relations between A and B. The following scenarios expressed using MSCs, define the behavior of the network in identifying peers. It is important to note that these scenarios have been devised from the perspective of MAS_A.

MSC1 shown in Fig. 17.5 illustrates a scenario where the ties between MAS A and B is established based on their mutual peer; Peer_C. Alternatively, MSC2 shown in Fig. 17.6, displays a scenario where the relations between MAS A and B are decided based on their ability to cooperate on a given query. The scenario shown in MSC1 (Fig. 17.5) typically occurs when each MAS attempts to update its ties with other nodes on periodic bases, whereas the scenario in MSC2 (Fig. 17.6) would occur when the ties with other MAS are updated as the result of a sent query.

However there could emerge a scenario where the periodic update of the ties of a MAS would coincide with the update of the ties triggered by sending a query. Such a scenario can be derived from the scenarios shown in MSCs 1 and 2 (Figs. 17.5 and 17.7 respectively) and is illustrated in MSC 3 (Fig. 17.7).

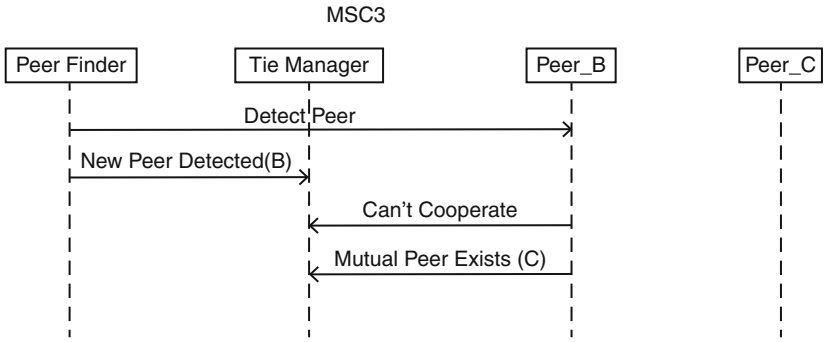

Fig. 17.7 Possible emergent behavior

17.4.1 Definitions

In this section, we give some definitions related to MSC notation based on a subset of ITU definitions for MSC [1,4,7].

Let P be a finite set of agents in a distributed system (with the total number of agents $p \geq 2$) and C be a finite set of message contents (or message labels) that are passed among the agents. Let $\sum_i = \{i!j(c),\ i?j(c) |\ j \in P \backslash \{i\},\ c \in C\}$ be the set of alphabet (i.e. events) for the agent $i \in P$, where $i!j(c)$ denotes an event that sends a message from agent i with content c to agent j, whereas $i?j(c)$ denotes an event that is received by agent i a message with content c from agent j. The set of alphabet will be $\Sigma = \bigcup_{i \in P} \Sigma_i$ and each member of Σ is called a message.

In the following, we try to capture a causal relationship between a message and its predecessors by defining partial Message Sequence Chart (pMSC).

Definition 1 (Partial Message Sequence Chart). A partial Message Sequence Chart (pMSC) over P and C is defined to be a tuple $m = (E, \alpha, \beta, \prec)$ where:

E is a finite set of events.

$\alpha : E\ \Sigma$ maps each event with its label. The set of events located on agent i is $E_i = \alpha^{-1}(\Sigma_i)$. The set of all send events in the event set E is denoted by $E! = \{e \in E \mid \exists i, j \in P,\ c \in C\ :\ \alpha(e) = i!j(c)\}$ and the set of receive events as $E? = E \backslash E!$.

$\beta :\ F!E?$ is a bijection mapping between send and receive events such that whenever $\beta(e_1) = e_2$ and $\alpha(e_1) = i!j(c)$, then $\alpha(e_2) = j?i(c)$.

$\prec$ is a partial order on E such that for every agent $i \in P$, the result of $\succ$ on E_i is a total order of its members and the transitive closure of $\{(e_1, e_2) | e_1 \prec e_2, \exists i \in P : e_1, e_2 \in E_i\} \cup \{(e, \beta(e)) | e \in E\}$ is a partial order of the members of E.

The partial order $\succ$ captures casual relationship between the events of a pMSC. This causality basically represents two things. First, a receive event cannot happen without having its corresponding send event happened before. Second, a receive

(or send) event, cannot happen until all the previous events, which are causal predecessors of it have already been accomplished. Obviously, if all the send events have their corresponding receive events (i.e. as defined by the function β), the structure is called a Message Sequence Chart or simply an MSC. In other words, an MSC has the same structural components as a pMSC, except that β is defined for $F! = E!$.

Definition 2 (Projection). The projection $m|_i$ for agent i in MSC m, is the ordered sequence of messages which correspond to the events for the agent i in the pMSC m. For $m|_i$, $\|m|_i\|$ indicates its length, which is equal to the total number of events of m for the agent i, and $m|_i[j]$ refers to jth element of $m|_i$, so that if e_j is the jth interaction event for agent i according to the total order of the events of i in m, then $\alpha_m(e_j) = m|_i[j-1]$, $0 \prec j \prec \|m|_i\|$. In $m|_i$, we call every element $i!j(c)$, $i, j \in P$, $c \in C$, a send message and every element $i?j(c)$, a receive message.

For example, the projection for the agent QH in MSC1 in Fig. 17.2 will be "QH!CL(send concept) ".

Definition 3 (Equivalent Finite State Machine for a projection). For the projection $m|_i$, we define the corresponding deterministic finite state machine $A_i^m = \left(S^m, \Sigma^m, \delta^m, q_0^m, q_f^m\right)$ such that:

S^m is a finite set of states labelled by q_0^m to $q_{\|m|_i\|}^m$.
Σ^m is the set of alphabet
q_0^m is the initial state
$q_f^m = q_{\|m|_i\|}^m$ is the final state (accepting state)
δ^m is the transition function for A_i^m such that $\delta\left(q_j^m, m|_i[j]\right) = q_{j+1}^m$, $0 \leq j \leq \|m|_i\| - 1$. Thus the only word accepted by A_i^m is $m|_i$.

Note that scenarios can be treated as *words* in a formal language, which is defined over send and receive events in MSCs. Then, a *well-formed word* for an agent is one that for every receive event there exists a send event in that word, which in fact captures the essence of definition given for a pMSC (Definition 1). On the other hand, a *complete word* for a agent is the one that for every send event in it, its corresponding receive event also exists in it. In practice, a system designer must look for complete and well-formed words for each agent, which is not necessarily an easy task. For any MSC m in the set of MSCs M, any sequence ω of m, obtained from a sequence of events in m that respects the partial order of the events defined for m, is called a linearization of m, and is a word in the language *L(M)* of M.

17.4.2 Constructing Behavioral Models

As mentioned in the previous section, scenario based specification is an efficient and effective way to represent system requirements for software systems. However as each scenario only partially describes system's behaviour, scenario based

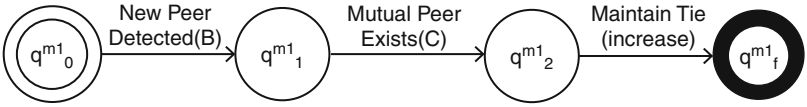

Fig. 17.8 eFSM for the Tie Manager agent of MAS_A in MSC1

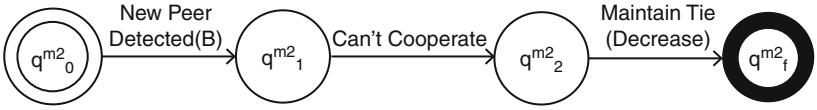

Fig. 17.9 eFSM for the Tie Manager of MAS_A in MSC2

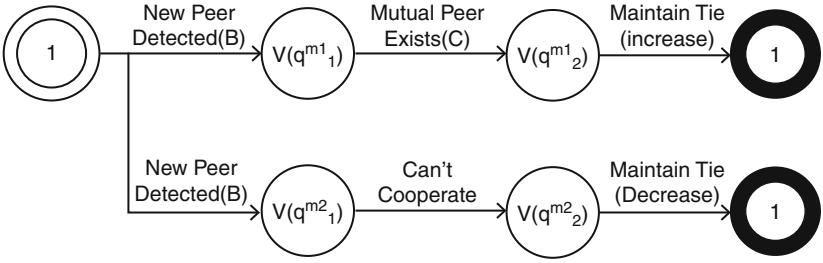

Fig. 17.10 The union of eFSMs built from MSCs 1 and 2

specifications are subject to deficiencies such as incompleteness and contradictions. Therefore having a methodology which can systematically discover system design errors before implementation will result in enormous savings in time and cost. The first part of this approach, which is the synthesis of state machines from message sequence charts is described in this section and is demonstrated using the example of the semantic search system.

The first step is to construct behaviour models for individual system component using finite state machines (FSMs). As explained earlier, the procedure of constructing FSMs from message sequence charts (MSCs) is referred to as behaviour modeling. For any system component *i* of a partial message sequence chart (pMSC) defined in Definition 1, an equivalent finite state machine (Definition 3) can be constructed. For the example of semantic search system, behaviour model of the tie manager (TM) agent of MAS_A is demonstrated. Figures 17.8 and 17.9 show the eFSMs that are constructed for the TM agent in MSC1 (Fig. 17.5) and MSC2 (Fig. 17.6) respectively.

To complete behavior model for the TM agent another FSM, which is the union of its corresponding eFSMs from different scenarios that contain TM is built and shown in Fig. 17.10.

17.5 Detection of Emergent Behavior

Emergent behaviour occurs when there exists a state, in which the component becomes confused as to what course of action to take. This happens when identical states exist in the union of eFSMs obtained through behavioural modelling. A definition for identical states is needed for detection of emergent behaviour. To achieve this we must first have a clear procedure to assign values to the states of the eFSMs. This is a very important step and is performed differently in various works. For instance, [9] proposes the assignment of global variables to the states of eFSMs by the system engineer. However the outcome of this approach is not always consistent as the global variables chosen by different system engineers may vary. Therefore to achieve consistency in assigning state values, the approaches of [24,25] which make use of an invariant property of the system called semantic causality is followed.

Definition 4 (Semantic causality). A message $m|_i[j]$ is a semantical cause for message $m|_i[k]$ and is denoted by $m|_i[j] \xrightarrow{se} m|_i[k]$, if agent i has to keep the result of the operation of $m|_i[j]$ in order to perform $m|_i[k]$.

For example, in MSC1 in Fig. 17.5, message "New Peer Detected (B)" is a semantic cause for message "Mutual Peer Exists (C)". As semantic causality is an invariant property of the system and is part of the system's architecture and the domain knowledge, it is independent of the choices made by the system engineers. In other words, we let the current state of the agent to be defined by the messages that the agent needs in order to perform the messages that come after its current states. Thus in order to evaluate state values of the resulting FSM, a domain theory which consists of the domain knowledge of the system must be constructed as defined formally in Definition 5.

Definition 5 (Domain theory). The domain theory D_i for a set of MSCs M and agent $i \in P$ is defined such that for all $m \in M$, if $m|_i[j] \xrightarrow{se} m|_i[k]$ then $(m|_i[j],\ m|_i[k]) \in D_i$.

Continuing with the above example, since the message "New Peer Detected (B)" is a semantic cause for message "Mutual Peer Exists (C)", both messages are part of the domain theory. However building the domain theory can be very time consuming. Therefore as a part of this systematic approach, building a light domain theory is introduced. The concept of light domain theory is closely tied to the calculated state values as defined in Definition 6. Using this definition, it becomes evident only states with the same incoming transitions have the potential to exhibit indeterministic behaviour which have the same incoming transitions. Assigning state values to states of eFSMs is done by making use of semantic causality as defined in Definition 6.

Definition 6 (State value). The state value $v_i|(q_k^m)$ for the state q_k^m in eFSM $A_i^m = \left(S^m,\ \Sigma^m,\ \delta^m, q_0^m, q_f^m\right)$ is a word over the alphabet Σ_i U $\{1\}$ such that $v_i|\left(q_f^m\right) = m|_i\ [f\ -\ 1]$, and for $0 < \mathrm{k} < \mathrm{f}$ is defined as follows:

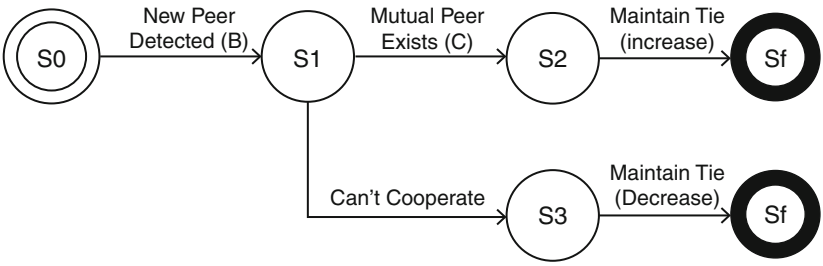

Fig. 17.11 Resulted FSM after merging identical states

1. $v_i|(q_k^m) = m\,|_i\,[k\,-\,1]v_i|(q_j^m)$, if there exist some j and l such that j is the maximum index that $m|_i\,[j\,-\,1] \xrightarrow{se} m|_i\,[l]\,,\ 0 < j\ < k,\ \ k \leq l\ < f$
2. $v_i|(q_k^m) = m\,|_i\,[k\,-\,1]$ if case (i) does not hold but $m|_i[k\,-\,1] \xrightarrow{se} m|_i[l]$, for some k ≤ l < f
3. $v_i|(q_k^m) = 1$, if none of the above cases hold

Getting back to our systematic approach to find emergent behaviour, by considering the resulting FSM in Fig. 17.10, we select pairs of states with the same incoming transitions and evaluate their state to look for identical states. Figure 17.11illustrates the constructed FSM as the result of the merging of identical states.

As it is shown in Fig. 17.11, state S1 is where the tie manager of MAS_A falls into confusion. That is, as it is illustrated in MSC3 (Fig. 17.7), TM will not be able to distinguish whether or not it should increase the tie with MAS_B or decrease it. Therefore as a result of the systematic approach in detecting emergent behavior in MAS, the system engineers is notified of such possible scenarios and is able to make modifications where necessary.

17.6 Conclusions and Future Work

Scenario based specification is an efficient and effective approach to illustrate the requirements of software system. Aside from their simplicity and expressive powers, scenarios can be used to analyze the requirements and design of software systems. Some of the failures in software systems can be directly attributed to their design. Research suggests that detection of failures and removal of faults during field use of a system is about 20 times more expensive than detection and removal in the requirement and design phase [26]. Unfortunately, manual review of the design documents may not efficiently detect all the design flaws due to the scale and complexity of the system. Therefore devising an automated and systematic methodology to analyze system requirements is greatly beneficial.

Furthermore, in this paper a method to identify the exact cause of implied scenarios is provided, so that by capturing it, implied scenarios can be detected and removed. This method is novel in the sense of formalization of the cause of implied scenarios. We believe that this is the main reason for some shortcomings and conflicts in the current works, as they have been revealed in [25, 27].

In this research we devised and demonstrated a method to detect and remove design flaws that may lead to emergent behaviors in social networks. These techniques were illustrated using a prototype of a social network of multi-agent systems for semantic search. Due to the lack of central control in social networks, the requirement gathering and design of such systems can be difficult. Thus the presented methodologies can be used to systematically validate the requirements of social networks.

In this research the requirements of social networks were analyzed with a component level perspective. For future work, the requirements of these systems can be analyzed with a system-level outlook. Furthermore since emergent behavior is not necessarily a negative quality of the system, the presented methodologies can be utilized to discover implied scenarios which do not cause problems for the system.

References

1. Scott, J.: Social Network Analysis: A Handbook, 2nd ed. Sage, London/Thousands Oaks (2000)
2. Hanneman, R.A., Riddle, M.: Introduction to Social Networks Methods. Sage, London/Thousand Oaks (2005)
3. Unified Modeling Language Specification. Version 2. Available from Rational Software Corporation, Cupertino (2006)
4. ITU: Message Sequence Charts. Recommendation, International Telecommunication Union (1992)
5. Harel, D., Kugler, H.: Synthesizing state-based object systems from lsc specifications. Int. J. Found. Comput. Sci. **13**(1), 5–51 (2002)
6. Kruger, I., Grosu, R., Scholz, P., Broy, M.: From mscs to statecharts. In: Rammig, F.J. (ed.) Distributed and Parallel Embedded Systems. Kluwer, Boston (1999)
7. Makinen, E., Systa, T.: MAS – an interactive synthesizer to support behavioral modeling in UML. In: ICSE 2001, Toronto (2001)
8. Uchitel, S., Kramer, J., Magee, J.: Synthesis of behavioral models from scenarios. IEEE Transaction on Software Engineering, Feb 2003, pp. 99–115
9. Whittle, J., Schumann, J.: Generating statecharts designs from scenarios. In: ICSE, Limerick (2000)
10. Whittle, J., Schumann, J.: Scenario-based engineering of multi-agent systems. In: Agent Technology from a Formal Perspective, 3d ed. Springer, London (2006)
11. Adsul, B., Mukund, M., Kumar, K.N., Narayanan, V.: Casual closure for MSC languages. In: FSTTCS, pp. 335–347. Hyderabad, India (2005)
12. Alur, R., Etessami, K., Yannakakis, M.: Inference of message sequence charts. IEEE Transaction on Software Engineering, July 2003, pp. 623–633
13. Muccini, H.: Detecting implied scenarios analyzing nonlocal branching choices. In: FASE 2003, Warsaw (2003)

14. Uchitel, S., Kramer, J., Magee, J.: Negative scenarios for implied scenario elicitation. In: 10th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2002), Charleston (2002)
15. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press, Cambridge/New York (1994)
16. Granovetter, M.S.: The strength of weak ties. Am. J. Sociol. **78**, 1360–1380 (1973)
17. Granovetter, M.S.: Getting A Job: A Study of Contacts and Careers. University Of Chicago Press, Cambridge (1974)
18. Granovetter, M.S.: The strength of weak ties: a network theory revisited. Sociol. Theory **1**, 201–233 (1983)
19. Wellman, B., Wortley, S.: Different strokes from different folks: community ties and social support. Am. J. Sociol. **96**, 558–588 (1990)
20. Lin, N., Ensel, W.M., Vaughn, J.C.: Social resources and strength of ties: structural factors in occupational status attainment. Am. Sociol. Rev. **46**, 393–405 (1981)
21. Burt, R.: Structural Holes: The Social Structure of Competition. Harvard University Press, Cambridge (1995)
22. Far, B.H., Zhong, C., Yang, Z., Afsharchi, M.: Realization of semantic search using concept learning and document annotation agents. In: Proceeding of Twenty-First International Conference on Software Engineering and Knowledge Engineering (SEKE), pp. 164–169. Boston, USA (2009)
23. Lally, A., Verspoor, K., Nyberg, E.: Unstructured Information Management Architecture (UIMA) Version 1.0 (OASIS, 2008) (2008)
24. Moshirpour, M., Mousavi, A., Far, B.: Detecting emergent behavior in distributed systems using scenario-based specifications. In: International Conference on Software Engineering and Knowledge Engineering, San Francisco (2010)
25. Mousavi, A.: Inference of emergent behaviours of scenario-based specifications. In: Department of Electrial and Computer Engineering, vol. PhD, University of Calgary, Calgary (2009)
26. Goldenson, D.R., Gibson, D.L.: Demonstrating the impact and benefits of CMMI: an update and preliminary results. CMU/SEI-2003-SR-009, Pittsburgh, Oct 2003
27. Mousavi, A., Far, B.: Eliciting scenarios from scenarios. In: Proceedings of 20th International Conference on Software Engineering and Knowledge Engineering (SEKE 2008), San Francisco, 1–3 July 2008