Soc. Netw. Anal. Min. (2015) 5:1
DOI 10.1007/s13278-014-0242-0

ORIGINAL ARTICLE

Verification of lack of emergent behavior in extending a social network of agents

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

Received: 19 February 2014 / Revised: 10 October 2014 / Accepted: 27 November 2014 / Published online: 9 January 2015

Abstract The scalability of the system is of vital importance in the design of social networks. This research attempts to establish a comprehensive framework for analysis and validation of requirements and design documents for software systems. In previous work, we applied this framework to analyze the requirements of a social network of agents with respect to scalability of the system. In our approach, system requirements were expressed using scenario-based specifications. Scenarios are appealing because of their expressive power and simplicity. Moreover, due to the clear and concise notation of scenarios, they can be used to analyze the system requirements for general validity, lack of deadlock, and existence of emergent behavior. In this paper a methodology is presented to formally verify that certain scenarios do not emerge in the system's behavior. This methodology is devised to indicate whether or not the new requirements of the system are consistent with the current requirements in place. A larger prototype of a social network of MSA for semantic search is utilized to illustrate the developed methodology.

Keywords Software requirements engineering · Software verification tool · Ontology · Domain knowledge · Scalability of social networks · Formal verification · Scenario-based software engineering · Emergent behavior · Multi-agent systems

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

S. M. El-Sherif
e-mail: smmelshe@ucalgary.ca

B. H. Far
e-mail: far@ucalgary.ca

R. Alhajj
Department of Computer Science, University of Calgary, 2500 University Dr NW, Calgary, AB T2N 1N4, Canada
e-mail: alhajj@ucalgary.ca

R. Alhajj
Department of Computer Science, Global University, Beirut, Lebanon

1 Introduction

Modifying the scale of software applications is in general a non-trivial endeavor especially in social networks due to their rapid growth. As a social network increase in size, the need for additional functionality becomes evident. Therefore, these systems face a serious scalability challenge due to their lack of central control as well as their rapid growth which results in increased complexity of the system. To avoid introducing bugs into the system, it is highly beneficial to ensure the correctness and integrity of the system will be preserved after scaling. 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 (Goldenson and Gibson 2003).

Due to the integrated nature of social networks, it can be challenging to gather comprehensive and correct requirements for these software systems. Scenario-based specification is an effective and efficient way to describe the behavior of a variety of software systems such as multi-agent systems and distributed systems. Scenarios enable engineers and designers to describe system's functionality using the partial interactions of the system elements. There are several advantages of using scenarios such as

expressive power and simplicity. In Moshirpour et al. (2013) we demonstrated that scenario-based software engineering (SBSE) can be used to effectively represent the requirements of social networks. Furthermore, in Moshirpour et al. (2013) we used the methodology devised in this research to analyze the scalability of social networks.

There are two main ways of representing scenarios, namely, sequence diagrams (SD) developed by the object management group (OMG) (Unified Modeling Language Specification. version 2 2006) and message sequence charts (MSC) which were developed by the International Telecommunications Union (ITU) ("ITU: message sequence charts. Recommendation, International Telecommunication Union" 1992). In this research MSCs are used to represent scenarios.

In Moshirpour et al. (2013) a methodology to analyze the requirements of social networks, expressed using scenarios, to detect emergent behavior was introduced. Emergent behavior, also known as 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 (Casual Closure for MSC Languages 2005; Alur et al. 2003; Muccini 2003; Uchitel et al. 2002). 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, the methodology presented in (Moshirpour et al. 2013a, b) is extended to formally verify that certain scenarios do not emerge in the system's behavior after applying the changes in system requirements. To illustrate this methodology, it is applied to the extended case study of semantic search engine which is a social network of multi-agent systems. Furthermore, this paper introduces a software verification tool which was developed based on the devised methodology in this research.

The structure of this paper is as follows: in Sect. 2 some background on the scalability of social networks as well as the time management amongst the nodes of such systems is presented. Section 3 contains the case study of the social networks of multi-agent systems (MAS). The verification methodology is demonstrated in Sect. 4. In Sect. 5 the software verification tool is introduced and conclusions and future work are presented in Sect. 6.

2 Background

Some background knowledge with regard to scalability of social networks as well as the tie management amongst nodes in social networks is provided in this section.

2.1 Agent

The concept of agent has many different definitions. According to Tianfield, an agent can be defined as: "An autonomous entity package of a set of capable computational entities, three of which (for internal scheduling, problem solving and communication routing) are normative and others are optional" (Moshirpour et al. 2010).

2.2 Multi-agent system (MAS)

MAS can be defined as: "a loosely coupled network of problem solvers (agents) that interact to solve problems which are beyond the individual capabilities or knowledge of each problem solver" (Mousavi 2009). A MAS is therefore a collection of heterogeneous agents, each of which with its own problem solving strategy that is able to interact and coordinate with each other (Mousavi and Far 2008).

When an agent is implemented in a MAS, it interoperates with other agents; therefore the architecture of the agent must be able to distinguish between its own internal data *Datown* and the internal data of other agents *Datother* it communicates with (Daconta et al. 2003). At the same time, they must be able to understand each other even if they use different knowledge representations.

2.3 Ontology

Ontologies are used in artificial intelligence, knowledge engineering, education, e-commerce, and semantic web, etc. There are several definitions of ontology, but we will use the definition stated by *Daconta* in (Daconta et al. 2003) as it is more relevant to our work, "Ontology defines the common words and concepts (meanings) used to describe and represent an area of knowledge, and so standardizes the meanings. Ontologies are used by people, databases, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, counterterrorism, imagery, automobile repair, etc.) Ontologies include computer usable definitions of basic concepts in the domain and the relationships among them."

Daconta classifies ontologies according to their richness into four groups as presented in Fig. 1. The lower ontologies have weaker semantics and the upper ones have stronger semantics.

If two agents use the same ontology or are able to understand each other's ontology, communication between them is possible. However, this is rare. In this case they need a mechanism to understand each other. In this paper, we illustrate how a single learner agent can learn new concepts from different teacher agents.

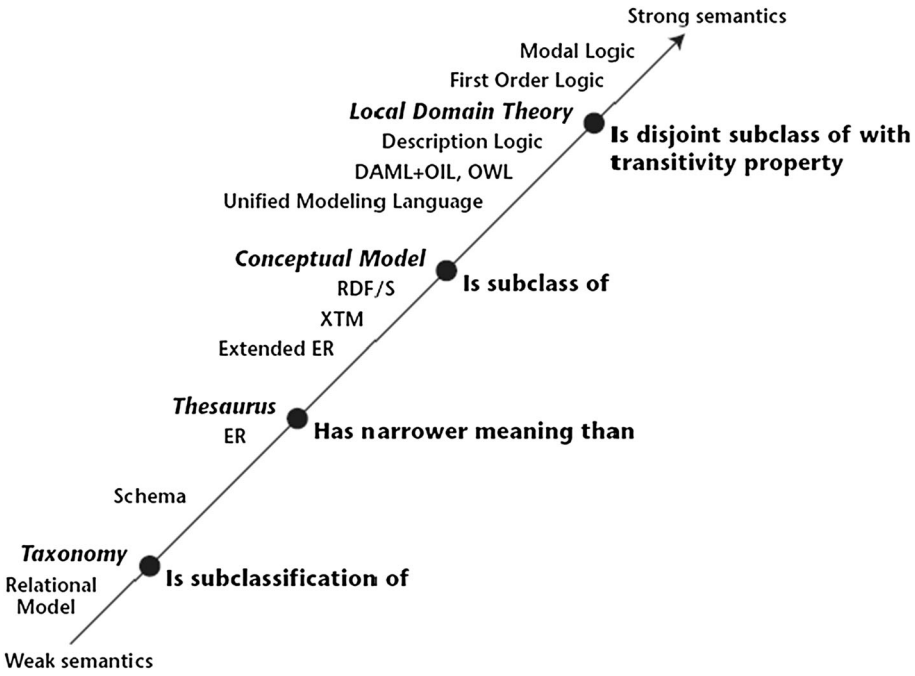

Fig. 1 The ontology spectrum (Daconta et al. 2003)

2.4 Agent communication

In order to let heterogeneous agents communicate with each other, a communication and a contents language are needed. The communication language, commonly known as Agent Communication Language (ACL), includes several constructs to help agents find each other and exchange messages. There are a number of ACLs specified in the IEEE FIPA (http://www.fiba.org) specifications. The contents language, on the other hand, handles the message contents and lets the sender and receiver break down the message to interpretable tokens. Some of the tokens are operands (i.e., concepts) and some are operators (i.e., how to manipulate concepts). Interpretation of the operands requires an ontology to assign the same meaning to the operands on the sender and receiver side.

2.5 Scalability of social networks

A simple paradigm to avoid the scalability challenge in social networks is with a fully distributed architecture of the network. It is not always possible to achieve this paradigm due to resource scarcity. There is always a tradeoff between functionality and future scalability.

There are several recent works in the literature which aim at solving the problem of social networks scalability. In Pujol et al. (2010), the authors propose a system to scale up a centralized social networks design without undergoing a costly transition to a fully distributed system. They take advantages of the structural properties of social networks to propose their paradigm. They call it one-hop replication (OHR). OHR utilizes some of structural characteristics of social networks. For instance most of information is one-hop away, and the topology of the network of connections among nodes displays a strong community structure. This system is composed of two components. The first component is the controller which is responsible for assigning users to servers. The second component is the middleware which ensures replication consistency.

Social Partition and Replication (SPAR) is another paradigm which is implemented in Pujol et al. (2010). SPAR is a middleware that leverages the social graph structure to achieve data locality and minimize replication at the same time. SPAR constricts all relevant data for a user on a server and guarantees that for all users' direct neighbors, data is co-located on the same server. In this paradigm, scalability is achieved by adding commodity servers with low memory and network I/O requirements.

On the other hand Loupasakis et al. (2011) introduces a decentralized scalable social network (eXO) to solve the problem of centralization in order to face the scalability challenge. eXO offers a fully decentralized social network with the ability to efficiently index and search globally for top-k users and content based on metadata information. The architecture of eXO provides also content replication as in P2P networks. eXO is based on distributed hashed table and it adds methods on top of it for efficient indexing and search and retrieval of users and content in the social networks scenarios.

2.6 Managing the ties between nodes

The strength of a tie is affected by several factors. Granovetter (1983) proposed four dimensions that may affect tie strength: the duration of the relationship; the intimacy between the two actors participating in the relationship; the intensity of their communication with each other; and the reciprocal services they provide to each other. In social networks of humans other factors such as socioeconomic status, educational level, political affiliation, race and gender are also considered to affect the strength of ties (Lin et al. 1981).

Structural factors, such as network topology and information about social circles, may affect the tie strength (Burt 1995). The work in (Gilbert and Karahalios 2009) suggests quantitative measures (variables) for tie strength including intensity variable, days passed since the last communication and duration. Another variable that may affect the strength of the tie is the neighborhood overlap variable (Onnela et al. 2007) which refers to the number of common friends the two actors have. The work in Petroczi et al. (2007) introduced mutual confidence between the actors of social networks. In El-Sherif et al. (2011) we propose a new methodology to calculate the strength of ties between agents in a social network using Hidden Markov Models (HMM) (Cappé et al. 2007).

We showed that tie strength depends on several factors: closeness factor, by measuring how close two agents are to each other (i.e., the degree of similarity between the two ontologies used by the two agents participating in the relationship); time-related factor, combines all time factors that affect the strength of the relationship (e.g., duration of the relationship, frequency of communication between the two agents, time passed since the last communication); mutual confidence factor, clarifying the nature of the relationship under measure, if it is a one-sided relationship or a mutual relationship. Then we built an HMM model to measure the strengths of ties between agents in a social network using those factors.

3 Case study: semantic search engine

A model for semantic search was presented in Moshirpour et al. (2013). This model utilizes a spiral workflow to incorporate both search and concept learning in the semantic search process (Far et al. 2009). The spiral workflow and its suggested scenario are shown in Fig. 1.

Semantic search depends on understanding the meaning of the concepts used in the context of other words. Thus, it then tries to retrieve the related documents to these concepts. The foundation of semantic search is the semantic interoperability which is the main ingredient for notation extraction from the search phrase. Utilizing social networks in this system provides great flexibility; in particular, when 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 proposed in Moshirpour et al. (2013), we assume that in a society of n multi-agent systems; MAS_1, MAS_2, …, MAS_n, each multi-agent system, MAS_i, controls a repository R_i. Each repository uses an ontology that consists of a set of concepts and some documents to represent examples of these concepts. The architecture of our system is illustrated in Fig. 2. Each concept C in our

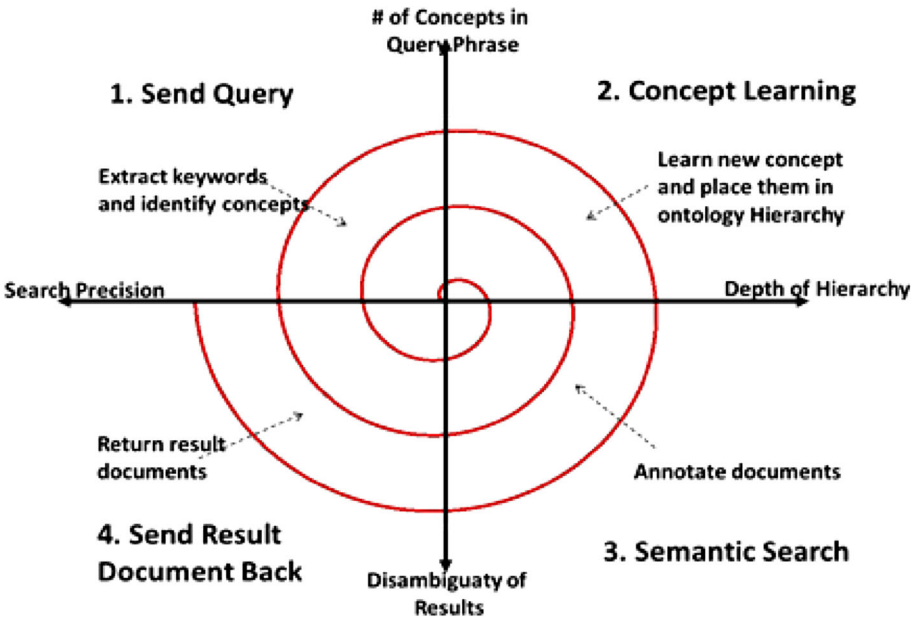

Fig. 2 Spiral workflow between semantic search and concept learning

system possesses supporting examples. In addition, each agent has its own ontology (O_i) to represent the concept C.

The key requirements for our proposed semantic search system are given below:

1. Software agents from different MASs must be able to communicate with each other and be able to exchange information.
2. MASs must be responsible for organizing data in their own repository by annotating documents in their local repositories.
3. MASs must be able to reorganize their local repositories based on updates of concepts.
4. Agents from different MASs must be able to cooperate with each other to learn and teach new concepts.
5. MASs should be able to hide the complexity of learning process and semantic search from the user.

Figure 3 shows different agent roles in each MAS in our prototype system (Moshirpour et al. 2013). In this system, agents of different MASs are the nodes in a social network. They connect with each other by a tie. These roles can be defined as follows (Fig. 4):

- *Query Handler* This role involves accepting a 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 broadcasting it to all neighbor agents.
- *Concept Learner* This role involves maintaining and confirming newly learnt concepts, including creation of taxonomies of domain of interest. It is also responsible for broadcasting these concepts to all group members to teach the concepts to the local agent. Moreover, the concept learner rearranges local repositories with the newly learned concepts.
- *Document Annotator* This role involves annotating the documents in the 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) that communicate with the current agent.
- *Tie Manager* This role involves keeping track of common concepts between peers and the interactions that occur between those peers in the learning process. This allows the change of the strength of a tie between peers dynamically. It is also responsible for setting the initial strength of the relationship between agents.

Figure 5 is a flow diagram of the concept learning process. The steps of this process are as follows:

- The concept learning process is initialized by an agent in one MAS (learner in this case). The learner sends a request to all MASs it is related to in a social network. They become teachers in this case. The learning request contains information available about the concept to be

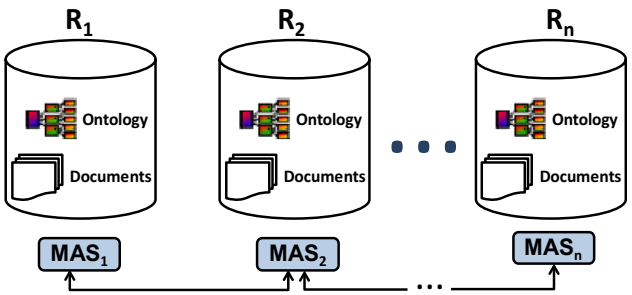

Fig. 3 Semantic search system architecture

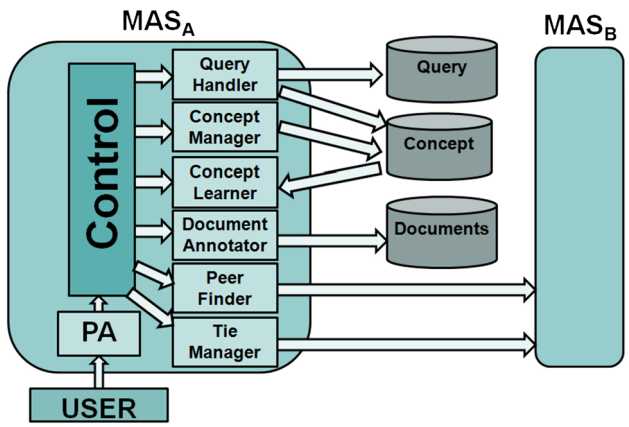

Fig. 4 Agent roles within the semantic search engine

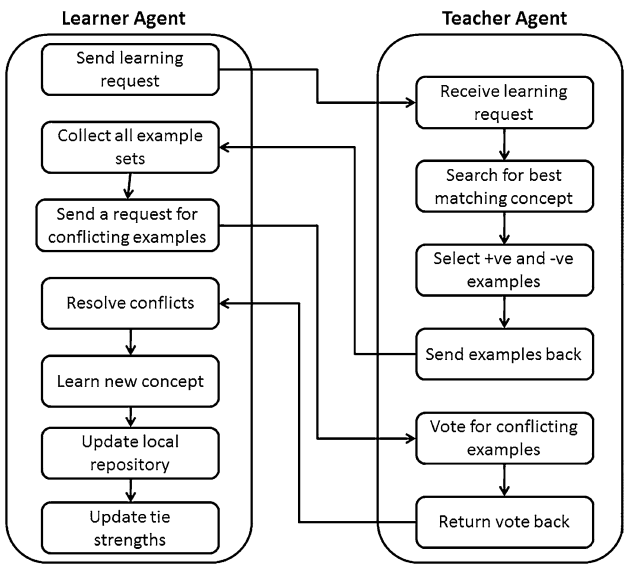

Fig. 5 Flow diagram of tasks in the concept learning module

learnt (e.g., its name, identifying keywords, annotation information, etc.).

- Each teacher receives the learning request.
- Each teacher searches its local repository for the best matching concept. In order to select the best matching concept, each teacher calculates for each concept in its local repository the value of $\mathrm{sim}(q_{\mathrm{spec}}, C_{\mathrm{best}})$ which is the ratio between number of examples that satisfy the search keywords and total number of examples that represent each concept. The best matching concept is the one with the highest value for $\mathrm{sim}(q_{\mathrm{spec}}, C_{\mathrm{best}})$.

$$\mathrm{sim}(q_{\mathrm{spec}}, C_{\mathrm{best}}) = \mathrm{MAX}_{\forall i} \times \frac{\text{number of examples satisfy search keywords in concept } C_i}{\text{total number of examples for concept } C_i}$$

- After finding the best matching concept, each teacher agent selects some positive (+ve) and negative (−ve) examples to represent this concept.
- The teachers send their example sets to the learner agent.
- The learner collects all example sets from all teachers.
- The learner tries to detect if there are any conflicts between the examples. By conflicts we mean that some examples are considered as positive examples by some teachers while at the same time being considered as negative examples by other teachers. In order to resolve this conflict, the learner sends another request to all teachers to vote against examples of conflict (i.e., decide if those examples are positive or negative examples according to their ontologies).
- All teachers vote against the examples of conflict.
- All teachers send the votes back to the learner.
- The learner collects all votes and resolves the detected conflicts.
- After resolving all conflicts, the learner uses the positive and negative example sets to learn the new concept.
- The learner updates its local repository by adding the newly learnt concept to its proper location in the ontology.
- The learner updates the strengths of all ties between it and all teachers based on the interactions occurred between them during the learning and the closeness between their ontologies.

Here we need to explore the concept learning process from the perspective of a learner. We explore the initialization of the learning process, the main steps in learning new concept from other agents and updating strength of ties between the learner and teachers.

We need also to apply the factors that affect tie strength in a normal social network as describe earlier in Sect. 2.6 of this paper. For the closeness factor, we can measure the closeness between the multi-agent systems by measuring similarity between their ontologies. The intensity variable can be represented in our system by the number of messages traded between two agents and how many concepts are learnt from each other which brings their ontologies closer to each other.

The dependency factor can indicate how much agents still depend on each other in learning new concepts or searching for keywords. The neighborhood overlap can be reflected as the overlap of neighborhood circles (i.e., number of common neighbors) of two agents. In most social networks, the relationship between members is asymmetric. That means the strength of ties is not necessarily equal in both directions. The same consideration is valid in our system.

The system is initialized by a user sending a query statement to one MAS in our system. The query handler tokenizes this statement to detect concepts and it cooperates with other agents to figure out if there are any new concepts to be learnt. The concept learning process is initialized when new concepts are identified. It is achieved by concept learner agent as indicated in Fig. 6.

The concept learning process itself is described in Fig. 7. This figure shows interaction done by the leaner and one teacher (B) only. This process is repeated for all detected peers.

As the system is developed further, the strengths of ties between the learner and all peers need to be updated. It is assumed that the strength of ties between each two nodes in the network (i.e., each MAS) depends on the following criteria (see Fig. 8):

(a) Number of times they have been able to successfully cooperate (frequency of interactions).
(b) Number of peers they have in common.
(c) Closeness between ontologies.

Figure 9 shows undesired behavior, where the strength of tie between the learner and a teacher (peer D) is too low that the learner cannot depend on that teacher to learn new concepts. Although, the learner did not validate the strength of the tie and send a learning request to that teacher.

4 Verification methodology

By modifying the functionality of the social network of MAS, it is highly desirable to verify the system's behavior. The methodology to detect emergent behavior in the requirements of this system was presented in Moshirpour et al. (2013a, b). A solution to verify the system against particular unwanted behavior, such as the one depicted in Fig. 9 is presented in this section.

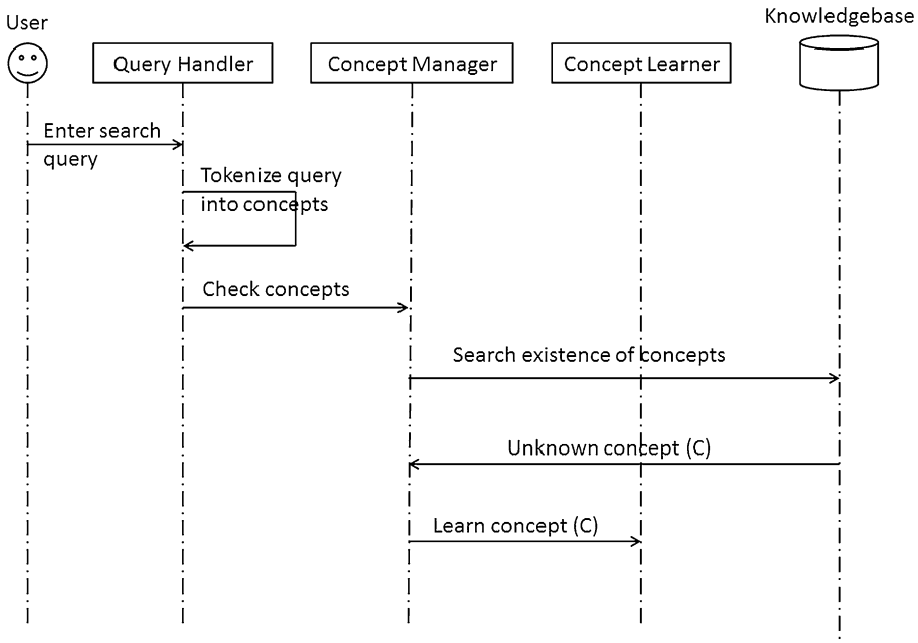

Fig. 6 The initialization of concept learning process based on search query entered by a user

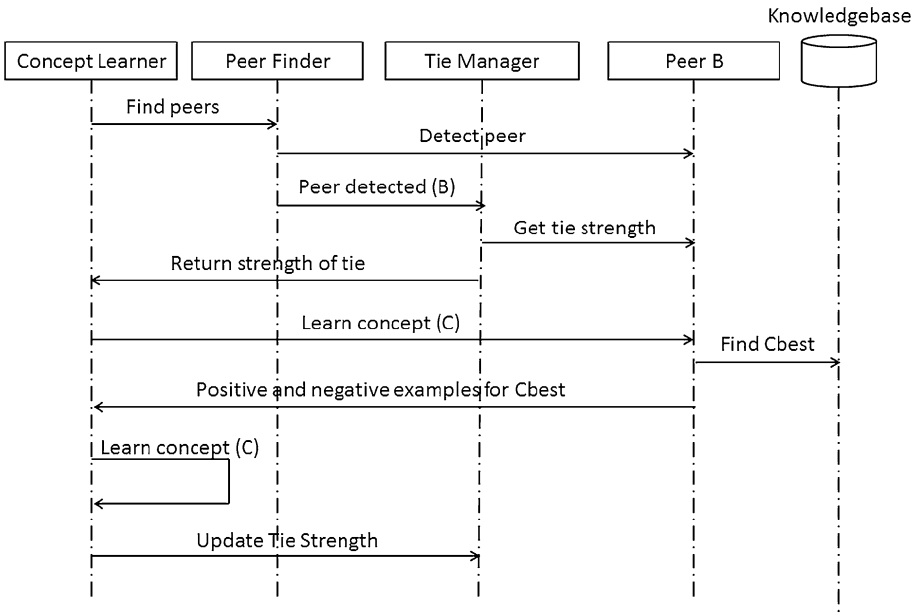

Fig. 7 Steps of concept learner process

This solution takes two different sets of scenarios which are expressed using MSCs as follows:

A. A set of MSCs containing scenarios describing the system's behavior (Figs. 6, 7, 8)
B. A set of illegal scenarios which are undesirable to occur (Fig. 9)

By having these two sets of scenarios this methodology verifies whether or not scenarios in set B can be derived from scenarios of set A. In other words this methodology ensures that system's behavior does not contain scenarios from set B. This methodology is divided into two parts of constructing the behavioral model (Moshirpour 2011; Moshirpour et al. 2010) and ensuring the lack of invalid scenarios in the built models, as presented in the following subsections.

4.1 Behavior modeling

The model which describes the behavior of each system component (i.e., node for social networks) is usually called a *behavioral model*, and the procedure of building the behavioral model from a scenario-based specification, is called *synthesis of behavioral models*, or simply, *behavior modeling* (Moshirpour 2011; Moshirpour et al. 2010). State machines are commonly used to represent behavioral

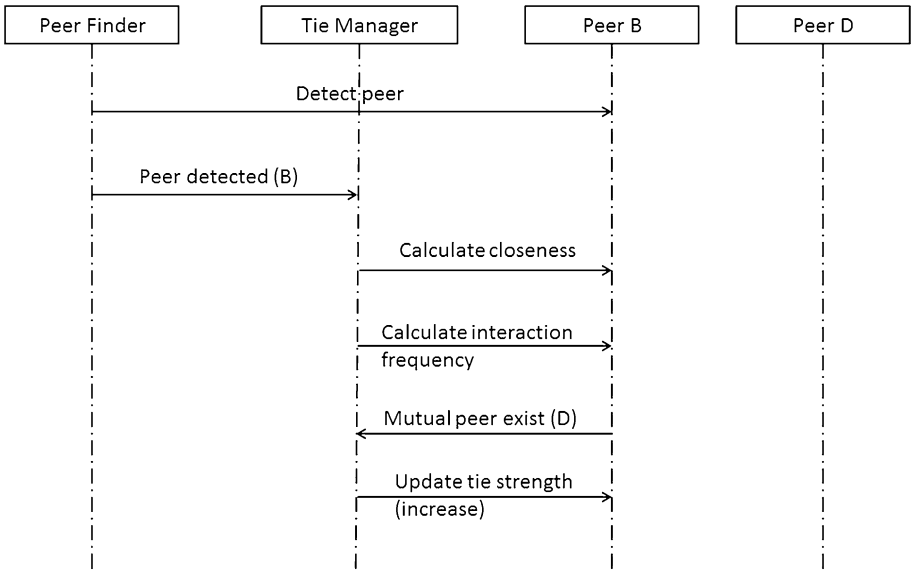

Fig. 8 Updating strengths of ties between a learner and one of the teachers (Peer B)

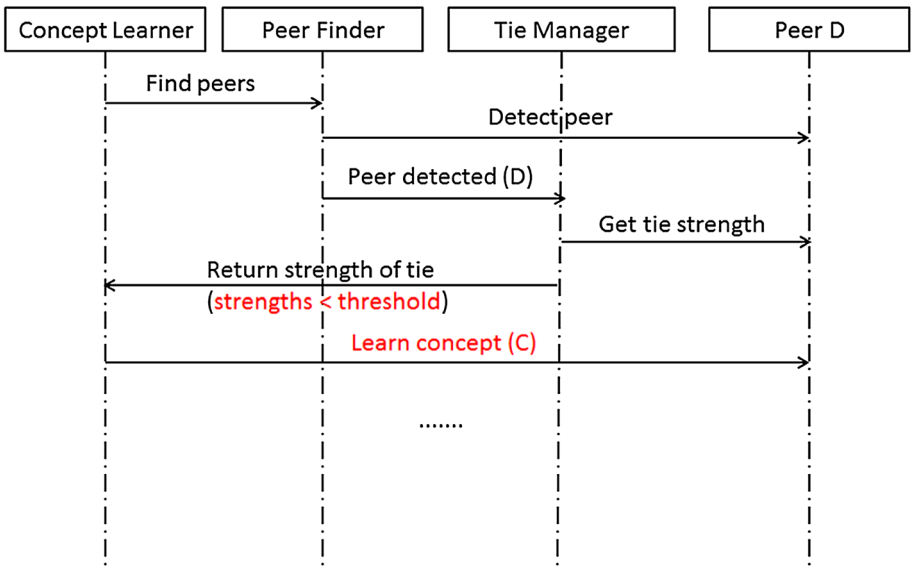

Fig. 9 Undesired behavior of concept learning system

models (Harel and Kugler 2002; Kruger et al. 1999; Makinen and Systa 2001; Uchitel et al. 2003; Whittle and Schumann 2000, 2006). 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. The behavior model for the initial tie-management mechanism, shown in MSCs 1 and 2 is depicted in Fig. 10 (Moshirpour et al. 2013).

To verify that the system's integrity is preserved after adding the new functionalities, the new behavior model for the system is constructed as shown in Fig. 11.

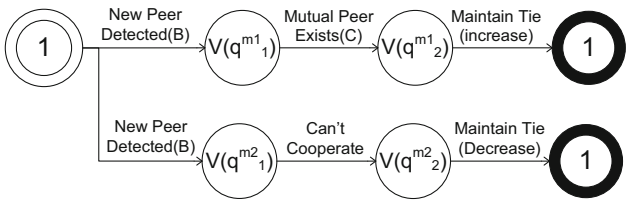

Fig. 10 The union of state machines built from MSCs 1 and 2

4.2 Domain knowledge

After synthesis of behavior models for each system component, the state values for the models are to be calculated. To do this, an invariant property of the system called semantic causality is used:

Definition 1 (*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 component i has to keep the result of the operation of $m|_i[j]$ in order to perform $m|_i[k]$.

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 domain experts. In other words, we let the current state of the component to be defined by the messages that the component needs in order to perform the messages that come after its current states.

Following the definition of semantic causality, the domain theory of the system is constructed as follows:

Definition 2 (*domain theory*) The domain theory D_i for a set of MSCs M and component $i \in P$ is defined such that for all $m \in M$, if $m|_i[j] \xrightarrow{se} m|_i[k]$ then $(m|_i[j], m|_i[k]) \in D_i$.

This solution provides an automated approach to build the domain knowledge for the system using an ontology-based approach as presented in detail in Khurshid et al. (2013) and Moshirpour et al. (2013). The ontology consists of the static and dynamic views (Khurshid et al. 2013; Moshirpour et al. 2013). The static view of ontology is much like a tree structure, where the elements are components of the system and are related to each other in this ontology based on their hierarchy within the system. The dynamic view of ontology represents the interactions between system components. This view describes the aspects of the system which can change with time. Figure 12 depicts the dynamic view of the ontology for the social network of MAS, which has been constructed using a deterministic finite state automaton.

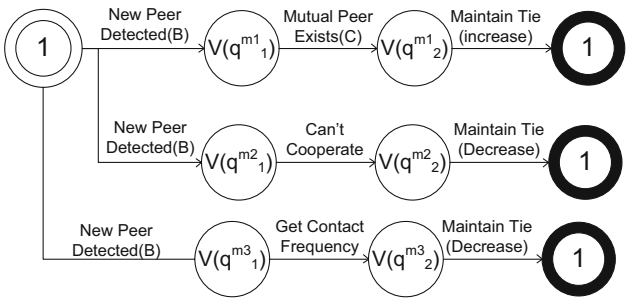

Fig. 11 Behavior model of the system after scaling

The states of this automaton are the categories which were established in the static view of the ontology. Both the static and dynamic ontology of the system are constructed automatically using the system scenarios and are used in the automatic extraction of the domain knowledge of the system (Khurshid et al. 2013; Moshirpour et al. 2013).

All required system knowledge should be present in the full set of MSCs. Thus, for each MSC i in MSCs, we find a non-mutually exclusive partial set of knowledge. For each component c in i, we consider adding c under the "System" node in the static ontology. If a similar node d exists under "System", we refrain from adding c to the ontology. Otherwise, we add c.

This algorithm assumes two things:

1. All knowledge in the system is present as components in the full set of MSCs; and
2. Any knowledge hierarchy within the system is irrelevant.

To gather knowledge about the behavior of the system, we use the knowledge present in the static ontology, as well as the MSCs. There are three steps to this technique:

1. Add states,
2. Determine next-state relationships, and
3. Determine finality.

Step 1—add states For each node n in the static ontology under the "System" node, add a state s to the dynamic ontology.

Step 2—determine next state For a message m in MSC i, we note f as the component from which m originates from, and t as the component to which m is sent. Find the state s' that relates to f, and the state s'' that relates to t. We will now consider choosing s'' as a next state to s'. If s'' is already a next state to s', do not add s'' to s' as a next state. Otherwise, add s'' as a next state to s'. Repeat this process for all messages m in all MSC i in the full set of MSCs.

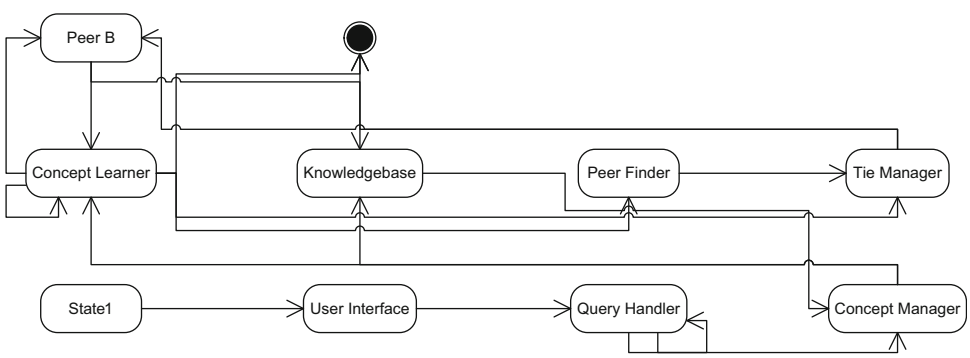

Fig. 12 Dynamic ontology of the system

Step 3—finality For every state s in the dynamic ontology, if the state has no next-states, then mark it as final state. If the state is not a next state to any state, then mark it as initial state. The final dynamic ontology generated by Emergent Behavior Detection Tool (EBD) is shown in Fig. 12.

4.3 Detection of emergent behavior

Implied scenarios or emergent behavior appear when there is 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 FSMs obtained through behavioral modeling. A definition for identical states is needed for detection of emergent behavior. 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 Whittle and Schumann (2000) proposes the assignment of global variables to the states of eFSMs by the system engineers. 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 Moshirpour et al. (2010) and Mousavi (2009) which make use of an invariant property of the system called semantic causality is followed.

By making use of semantic causality (Definition 1), the state values can be calculated as follows:

Definition 2 (*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 $\Sigma_i \cup \{1\}$ such that $v_i|\left(q_f^m\right) = m|_i[f-1]$, and for $0 < k < f$ is defined as follows:

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 $k \leq l < f$
3. $v_i|(q_k^m) = 1$, if none of the above cases hold.

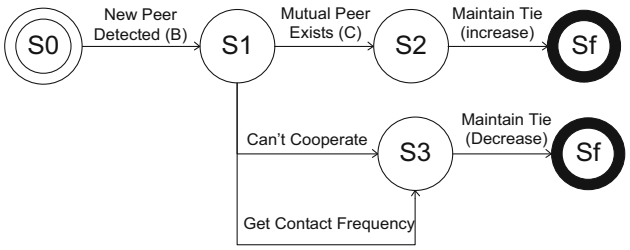

Fig. 13 Resulted FSM after merging identical states

By calculating all state values, equivalent states are merged as shown in Fig. 13 (Moshirpour et al. 2013).

As it is shown in Fig. 9, state S1 is where the tie manager of MAS_A falls into confusion. That is, as it is illustrated in MSC3 (Fig. 6), 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.

Upon the identification of cases of emergent behavior, a new set C is constructed to contain their related behavioral models. Therefore, if a behavior model built based on scenarios in set B does not match a behavioral model in set C, it is verified that the system will not contain that particular illegal scenario. Conversely, if a behavior model constructed based on the scenarios of set B is equal to the behavioral model in set C, the verification has failed. By comparing the FSM in Fig. 13 with the behavior model constructed from set B it becomes evident that the illegal scenario of set B can potentially emerge from the scenarios of set A.

5 Software analysis tool

In order to enable the efficient and effective use of this methodology in real world projects, it needs to be automated in a software package. In this section, the analysis of system requirements using such a tool is presented.

The user can start the analysis by making a project and importing their MSCs. Currently, the Visio version of

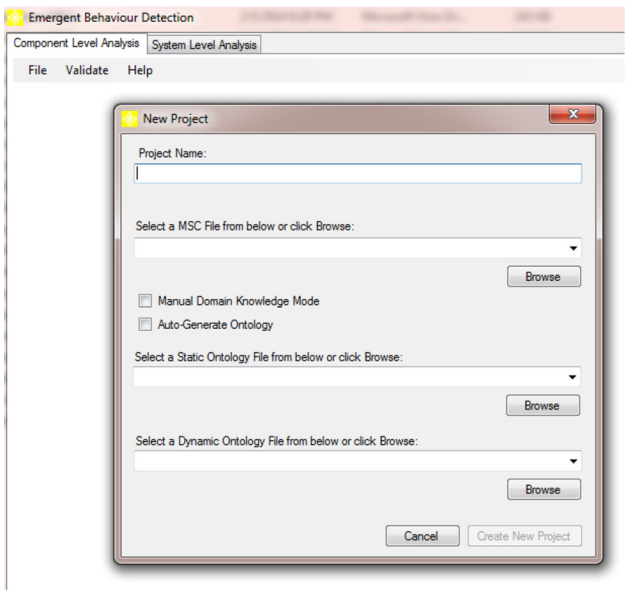

Fig. 14 Create Project Window for EBD

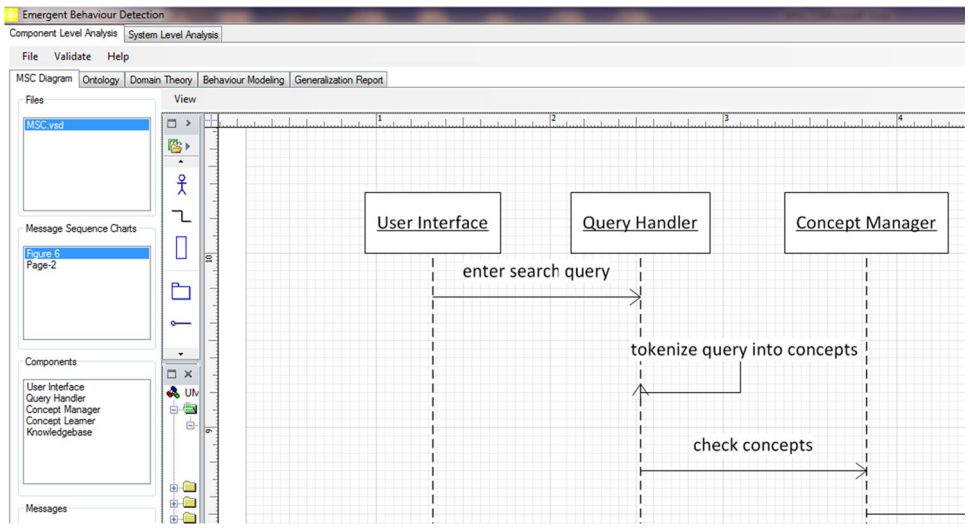

Fig. 15 Screenshot of the tool's user interface; displaying an imported MSC

EBD, supports all Visio programs up to and including the Office 365 version. The user can choose to import system ontologies or to simply have EBD auto-generate them as shown in Fig. 14. Upon importing the MSCs, the user can browse through them or change them in them as desired in the EBD window Fig. 15.

As can be seen from the snapshot of the tool's graphical user interface, upon importing a design project from Microsoft Visio, the data boxes of the GUI are populated automatically with data. By clicking any of the imported MSCs, the components of that MSC will be shown in the *Component* subsection of the GUI and the actual MSC will be shown in the *Selected Diagram* area.

Consequently, by selecting a component, the messages associated with that component will be shown in the *Message* subsection of the GUI. The synthesis of the behavior model, explained in Sect. 4 of this paper is conducted immediately upon importing related MSCs. The result of the built FSMs is shown in the *Constructed FSMs* subsection of the GUI. By clicking on the title of any of the FSMs the constructed figure will be shown in the selected diagram area as shown in Fig. 15.

EBD generates two types of system ontologies namely static and dynamic ontology. The static view of ontology is much like a tree structure, where the elements are components of the system and are related to each other in this ontology based on their hierarchy within the system. The static ontology here is used to generate knowledge about the system (Fig. 16). At this point, by clicking the validate design button, the methodology commences. Upon completion of the analysis, the user will be presented with a report outlining the areas in which unwanted behavior could occur.

The results of this analysis are then presented to the user in a detailed report outlining all possible areas of emergent behavior.

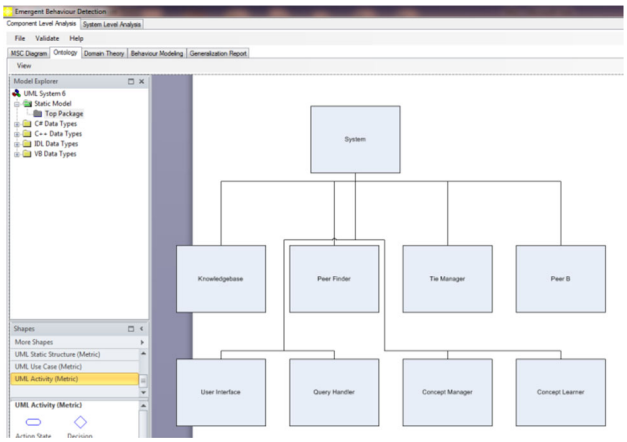

Fig. 16 Screenshot of the tool's user interface; displaying the static structure of the system

6 Conclusions and future work

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 (Goldenson and Gibson 2003). 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.

In the work presented in Moshirpour et al. (2013), 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. This research indicates that this is the main reason for some shortcomings and conflicts in the current works, as they have been revealed in Mousavi (2009)

and Mousavi and Far (2008). In Moshirpour et al. (2013) we demonstrated our devised method to detect and remove design flaws that may lead to emergent behaviors in social networks.

In this paper, the methodology presented in Moshirpour et al. (2013a, b) is extended to verify the lack of existence of particular illegal scenarios in the system. This goes a long way towards ensuring the correctness and integrity of the design of social networks after applying the changes in system requirements. 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 terms of scalability. Furthermore, this paper introduces a software tool which was developed based on the devised methodology.

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.

Acknowledgments This research was partially supported by a grant from Natural Sciences and Engineering Research Council of Canada (NSERC). M. Moshirpour would like to thank NSERC, Alberta Innovates-Technology Futures (AITF), and the Electrical and Computer Engineering department, University of Calgary, Alberta, Canada, for their logistics and financial support.

References

Alur R, Etessami K, Yannakakis M (2003) Inference of message sequence charts. IEEE Trans Softw Eng:623–633

Burt R (1995) Structural holes: the social structure of competition. Harvard University Press, Cambridge

Cappé O, Moulines E, Rydén T (2007) Inference in hidden Markov Models. Springer

Casual closure for MSC languages (2005)

Daconta MC, Obrst LJ, Smith KT (2003) The semantic Web. A guide to the future of XML, web services and knowledge management. Indianapolis

El-Sherif SM, Far B, Eberlein A (2011) Calculating the strength of ties of a social network in a semantic search system using Hidden Markov Models. In: International Conference on Systems, Man and Cybernetics (SMC). Anchorage, Alaska, pp 2755–2760

Far BH, Zhong C, Yang Z, Afsharchi M (2009) 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

Gilbert E, Karahalios K (2009) Predicting tie strength with social network. In: International conference on Human factors in computing systems. Boston, pp 211–220

Goldenson DR, Gibson DL (2003) Demonstrating the impact and benefits of CMMI: an update and preliminary results. CMU/SEI-2003-SR-009

Granovetter M (ed) (1983) The strength of weak ties: a network theory revisited. In: Sociological theory

Harel D, Kugler H (2002) Synthesizing state-based object systems from lsc specifications. Int J Found Comput Sci

ITU: message sequence charts. Recommendation, International Telecommunication Union (1992)

Khurshid B, Moshirpour M, Eberlein A, Far BH (2013) An automated ontology generation technique for an emergent behavior detection system. In: 14th International Conference on Information Reuse & Integration (IRI 2013). San Francisco, pp 380–387

Kruger I, Grosu R, Scholz P, Broy M (1999) From mscs to statecharts. In: Rammig FJ (ed) Distributed and parallel embedded systems. Kluwer Academic Publisher

Lin N, Ensel WM, Vaughn JC (1981) Social resources and strength of ties: Structural factors in occupational status attainment. Am Sociol Rev 46:393–405

Loupasakis A, Ntarmos N, Triantafillou P (2011) eXO: decentralized autonomous scalable social networking. In: The 5th Biennial Conference on Innovative Data Systems Research (CIDR). Asilomar, California, pp 85–95

Makinen E, Systa T (2001) MAS—an interactive synthesizer to support behavioral modeling in UML" presented at the ICSE 2001. Toronto

Moshirpour M (2011) Model-based detection of emergent behavior in distributed and multi-agent systems from component level perspective. In: Master of Science Department of Electrical and Computer Engineering. University of Calgary, Calgary

Moshirpour M, Eberlein A, Far BH (2013) Automated construction of system domain knowledge using an ontology-based approach. In: The 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013). Boston

Moshirpour M, El-Sherif SM, Alhajj R, Far BH (2013) Detecting emergent behavior in a social network of agents. In: Özyer T, Rokne J, Wagner G, Reuser AHP (eds) The influence of technology on social network analysis and mining. Springer, pp 339–409

Moshirpour M, El-Sherif SM, Alhajj R, Far BH (2013) Analyzing the scalability of a social network of agents. In: The 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013). Niagara Falls, Canada

Moshirpour M, Mousavi A, Far BH (2010) Detecting emergent behavior in distributed systems using scenario-based specifications. In: Presented at the Proceedings of the International Conference on Software Engineering and Knowledge Engineering. San Francisco Bay, USA

Moshirpour M, Mousavi A, Far BH (2010) Detecting emergent behavior in distributed systems using scenario-based specifications. In: International Conference on Software Engineering and Knowledge Engineering. San Francisco Bay

Mousavi A (2009) Inference of emergent behaviours of scenario-based specifications, Ph.D. thesis Ph.D. thesis, Department of Electrial and Computer Engineering, University of Calgary

Mousavi A, Far B (2008) "Eliciting Scenarios from Scenarios," presented at the proceedings of 20th International Conference on Software Engineering and Knowledge Engineering (SEKE 2008). San Francisco Bay

Muccini H (2003) Detecting implied scenarios analyzing nonlocal branching choices. In: Presented at the FASE. Warsaw, Poland

Onnela JP, Saramaki J, Hyvonen J, Szabo G, Lazer D, Kaski K, et al. (2007) Structure and tie strengths in a mobile communication network. Natl Acad Sci United States Am 104

Petroczi A, Nepusz T, Bazso F (2007) Measuring tie-strength in virtual social networks. Off J Int Netw Soc Netw Anal 27

Pujol JM, Siganos G,Erramilli V (2010) The little engine(s) that could: scaling online social networks without Pains. In: ACM SIGCOMM. New Dehli, India, pp 375–386

Unified modeling language specification. version 2. (ed) (2006) Available from Rational Software Corporation. Cupertino, CA

Uchitel S, Kramer J, Magee J (2002) Negative scenarios for implied scenario elicitation. In: Presented at the 10th ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2002). Charleston

Uchitel S, Kramer J, Magee J (2003) Synthesis of behavioral models from scenarios. IEEE Trans Softw Eng:pp 99–115

Whittle J, Schumann J (2000) "Generating statecharts designs from scenarios," presented at the ICSE. Limerick, Ireland

Whittle J, Schumann J (2006) "Scenario-Based Engineering of Multi-Agent Systems," in agent technology from a formal perspective, 3rd edn. Springer, London