A Hybrid System for Detection of Implied Scenarios in Distributed Software Systems

Anja Slama
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
anja.slama@ucalgary.ca

Fatemeh Hendijani Fard
Department of Computer Science
University of Calgary
Calgary, Canada
fhendija@ucalgary.ca

Behrouz Far
Department of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
far@ucalgary.ca

***Abstract*—Distributed software systems (DSS) are usually open-ended systems used in different domains such as robotics, energy, health, etc. Multi-agent system (MAS) are a sub-class of DSS. In DSS, maintaining consistency between the system iterations is a complex and expensive task that requires coping with requirements changes and systems upgrading. The interactions, complexity and decentralized communication between components of the DSS may emerge an unwanted behavior. An unwanted behavior, known as Emergent Behavior (EB) or Implied Scenario (IS), could lead to irreversible damages. Thus, detecting IS at an early stage of the system development is needed to decrease the cost of maintaining the system. This work focuses on verification of DSS that its requirements modeled using Message Sequence Chart (MSC). The system verification focuses on the detection of IS using two already proposed different approaches. This article presents the combination of the two approaches by improving the usability of the tool presented in the first approach and the catalogue presented in the second approach. This combination allows the detection of new implied scenarios not detected using the cited approaches separately.**

***Index Terms*—Distributed Systems, Multi-agent systems, Implied Scenarios, Message Sequence Chart**

I. Introduction

Distributed Software Systems (DSS) and Multi-Agent System (MAS) are used in a wide spectrum of domains. MAS are a sub-class of DSS. The size of DSS as well as the lack of central control together contribute to complexity of DSS' behavior.

Scenario-based specification is used to model the interaction between the DSS components that provide the overall system functionality. As each scenario presents a partial specification of the system, the identification of unexpected interaction patterns may not be obvious. These unexpected patterns exhibit system behaviors that do not conform to the system requirement and design. These patterns are known as Implied Scenarios (IS) or Emergent Behavior (EB) [1].

The detection of presence of implied scenarios will usually lead to detect potential system failures at run-time that may prevent irreversible damages to the system and reduce the overall system development and maintenance cost. Thus, there is a need for an analysis tool to inspect the interactions of the components and their impact on the overall behavior of the system.

DOI reference number: 10.18293/SEKE2018-195.

In this paper, we present a technique that combines two already existing methodologies, which are based on the analysis of Message Sequence Charts (MSC). This work aims to detect potential ISs and alert the developer to address the issue that does not conform to the concurrent nature of distributed systems. The advantage of the combination of the two methodologies is to get benefit from the automatic analysis of the MSCs using the LTSA tool [2], in order to detect the classical types of IS and also those defined in the catalogue of implied scenarios developed in our research group [3]. Unfortunately, the two techniques in their current form are incompatible. In this work, we compare both methodologies and we propose a technique to exploit the results of the extension of the LTSA tool considering the catalogue of implied scenarios.

II. Background and Related Work

A. Preliminaries

In this section, we explain the different types of implied scenarios and we review the concepts used in this work.

a) Implied Scenarios (IS): An implied scenario is a behavior of the distributed system that does not conform to the system requirements. We can investigate the system behavior in two levels: component level, and system level [3]. For the component level, the implied scenarios can be categorized into four main classes:

- CLEB-I (Shared states)
  The existence of a shared state of one component in different scenarios. The states occurring after these shared states make the behavior of the component non-deterministic.
- CLEB-II (Respond to different components)
  When a component receives the same message from two other different components, bringing it to the same state, an implied scenario could occur if the component receiving the messages loses track of the senders or receivers' information. In this case, the component could be confused in which MSC to proceed.
- CLEB-III (Local branching)
  This could occur when the component is active, i.e. can

initiate sending a message, in one of the MSCs included in the branch of its high level MSc (hMSC) and where no condition is specified to trigger its functionality in the next MSC.

- CLEB-IV (Race conditions)
  The race condition occurs when the behavior of the component depends on the order of receiving messages from other components.

b) Message Sequence Chart (MSC): Message Sequence Charts (MSCs) are usually used to formalize and model the requirements of distributed software systems. MSCs are standardized by telecommunication standardization of the International Telecommunication Union (ITU). It models the interaction between the system components [4].

c) Labelled Transition System Analyser (LTSA): LTSA is a tool used to automatically check and analyze models in order to confirm the expected behavior from the designer and implementer's viewpoint [5].
LTSA can translate the MSC chart to Finite State Processes (FSP) definition, a form of state machine description, used to draw the correspondent states machine. This tool offers different features to support verification of properties including safety and progress check [6].

B. Related Work

Among the model-based analysis approach, many researches have been performed so far. Process divergence and non-local branching choice were identified as under specification in MSC that could cause IS [11]. This research has been extended by generating state machines from the scenario specification in order to detect implied scenarios [12]. The formal methods tool, Spin, has been introduced to verify models for distributed software design [13]. Another approach aimed to detect implied scenarios considering the order between the events specified in the scenario specification [14]. In another work causality among events and using ontology to detect IS has been studied [15], [16].
Fard proposed a catalogue to categorize IS according to their reason of occurrence and devise the solution repositories accordingly. This methodology was based on state machines and social network analysis [3].

III. Methodology

In this work, we use the LTSA [2] and the catalog of implied scenarios [3] together. In the following, we compare the effectiveness of each as well as our approach that combines both, to find out implied scenarios.

A. Modeling System behavior using MSCs

The first step is modeling system interaction in the form of MSCs. The MSCs are translated to Finite State Processes (FSP) in order to model the required behavior and then compiled in the form of Labelled Transition System (LTS). For each component, the state machine of the parallel execution of the component FSPs is generated. The approach behind this phase is fully described in [2].

B. Detection of Implied Scenarios

Based on the generated FSMs, a verification of the existence of implied scenario is realized based on the LTSA tool. Moreover, to detect further type of implied scenarios, we analyze the traces generated by the LTSA taking into consideration the catalogue of IS. The method used to detect IS in this latter is based on extracting identical states for each agent from the send matrices obtained from the MSCs [10].
To detect the different type of ISs, explained in section II.A, we start by analyzing the set of shared states of the process detected by the LTSA tool. Performing a progress and a safety check on a selected component of the system, the tool generates the related trace based on the MSC modeling messages. The analysis consists of fetching the pattern defined by the catalogue to detect the IS.

IV. Experiment

A. Objectives and research questions

In this work, we seek to answer the following questions:

- RQ: How effective are these approaches to detect Implied Scenarios? We apply the approach on three case studies and we compare the number and the type of ISs detected based on the same case studies. The type of ISs corresponds to the classes of IS presented previously in the section II.A.

B. Case studies specification

To answer the research question, we use and evaluate the three case studies presented below;

a) Fleet Management System: The requirements of the Fleet Management System are locating drivers, considering different itineraries and approximately calculating the departure or arrival time for a minimum commute time [8]. The requirements are demonstrated in Fig. 1 to 4

b) Greenhouse System: This case represents the interaction between the agents of the Greenhouse Multi Agent System to control the greenhouse's environment by managing the available resources. The modeling of this system is composed by two MCSs as presented in [8].

c) Online Auction System: This case study represents interaction between the six agents (Controller, Auctioneer, Registrar, Seller, Buyer and Credit Associate) in order to assure the sell and the buy of books online according the rules defined by the auction hosting authority [9].
Table 1 is a recapitulative table of the case studies considered in this work.

TABLE I
SUMMARY TABLE OF THE CONSIDERED CASE STUDIES

Case Study	# MSCs	# Transitions	# Processes
Feet Management	4	37	8
Greenhouse	2	20	8
Online Auction	6	100	6

C. Evaluation measurement

We use two evaluation measurements which are the type of implied scenarios according to the catalogue and the number of implied scenarios detected.

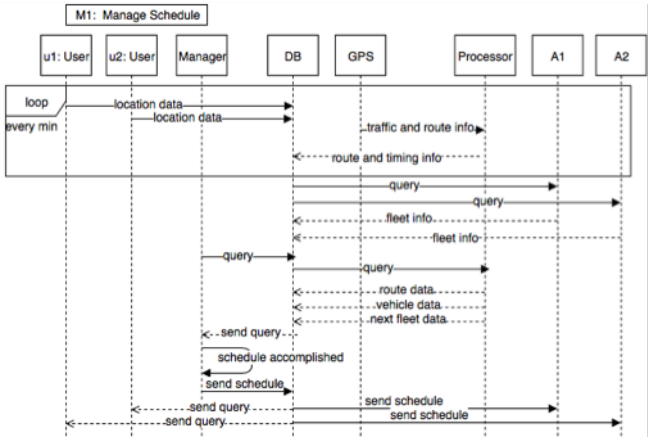

Fig. 1. MSC M1-Schedule management

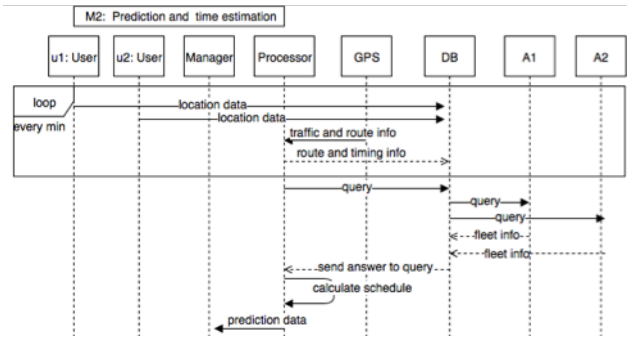

Fig. 2. MSC M2-Prediction and time estimation

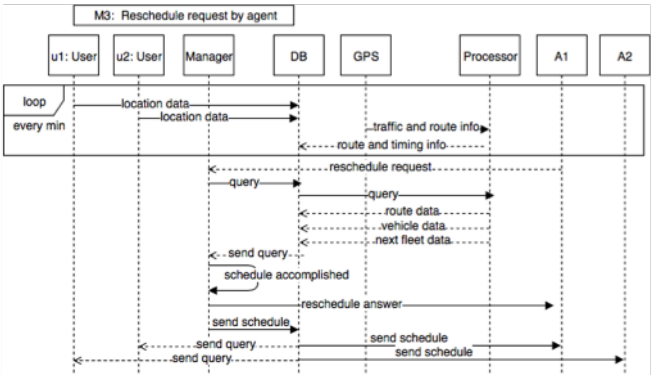

Fig. 3. MSC M3-Reschedule request by Agent

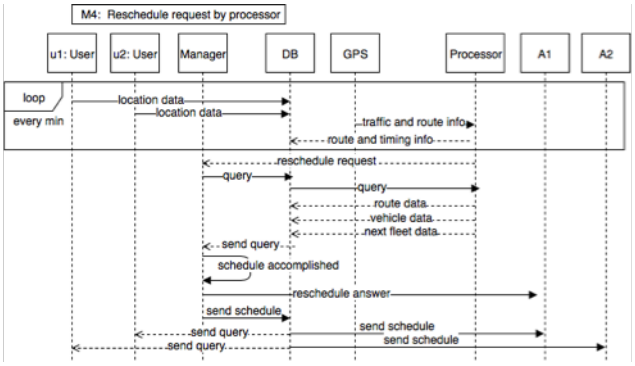

Fig. 4. MSC M4-Reschedule request by Processor

D. Procedure

The experiments conducted in this study consist of the detection of implied scenarios from the system modeling. Fig. 5 shows the different components of the system used during this process.

The verification process of the system consists of modeling graphically the system in the form of MSC using LTSA-MSC [17]. MSCs are converted to an FSP specification. LTSA tool checks the system for implied scenarios and safety violation based on this latter. The next step consists in analyzing the traces produced by the LTSA tool to detect further implied scenarios.

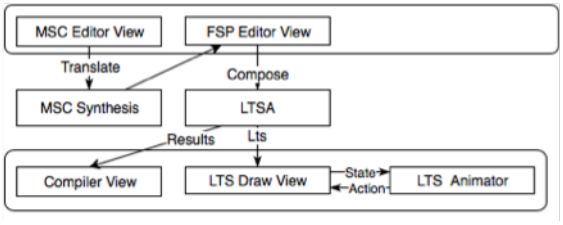

Fig. 5. Different components of the LTSA tool used during the process of detection of IS

E. Results

To answer the research question, we compare the number and the type of IS detected. Table II shows the results.

TABLE II
THE DETECTED ISS IN FLEET MANAGEMENT SYSTEM

Type/Approach	Uchitel	Fard	Combined approach
CLEBI	No IS found	No	Yes
CLEBII	No IS found	Yes	Yes
CLEBIII	No IS found	No	No
CLEBIV	No IS found	No	No
Deadlock	No deadlocks/errors	N/A	Yes

We apply the same procedures on the other case studies presented in Sect.IV.B to get the number of IS detected in each use case as shows the Table III.

TABLE III
THE NUMBER OF IMPLIED SCENARIOS DETECTED IN EACH CASE STUDY

Type/Approach	Uchitel	Fard	Combined approach
Feet Management System	1	1	3
Greenhouse System	1	1	3
Online Auction System	1	N/A	4

Fig. 6 shows the relation between the number of IS detected and the number of communication messages between the different processes. According to [18], there are a strong correlation between the number of detected IS and the number of messages. Whereas, there are a negative correlation between the number of implied scenarios detected and the number of processes. The probability of emergence of ISs is related to the grow of systems in complexity.

F. Discussion

Table III summarizes the number of detected ISs in the considered case studies. The efficiency of our work to detect

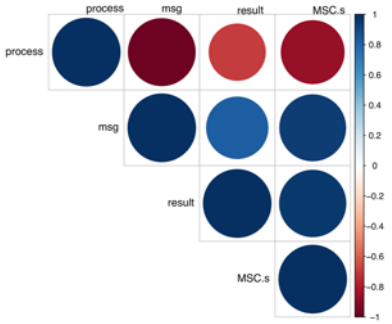

Fig. 6. Correlation between the results and the input data

ISs is based on a hybrid ensemble of the two previous approaches. We show that this method has the ability to detect more IS than Uchitel's and Fard's approaches in the three case studies. The reasons are that we take advantage of the automatic detection of shared states generated by the LTSA tool in order to facilitate the detection of EBs defined by Fard's work and that the approach of this latter do not detect deadlocks. Additional case studies may be needed to build a stronger database of the anti-patterns. By combining the two methodologies through the case studies, we have shown that we can achieve more efficiency in term of the number and type of Implied Scenarios detected.

G. Threats to the validity of the study

The approach proposed in this work is based on the LTSA tool. The traces generated by the LTSA tool directly impact the number and the type of IS detected. Testing the LTSA tool, we found out that the number of states generated is reduced and limited to a maximum number set by the tool itself to avoid the explosion of states. In future work, we have to further investigate the effect of reducing the states on possibly missing some of the implied scenarios. Moreover, the differentiation between asynchronous and synchronous communications is not considered.

V. CONCLUSION AND FUTURE WORK

The detection of implied scenarios in requirement and design phase is important for several reasons such as reduction of the project cost, time as well as the prevention of the occurring of unwanted behavior that could have irreversible damages to the systems environment and/or users. Assuring conformity of the system behavior to its requirements is not a trivial task in open ended systems. Therefore, a tool to maximize the detection of IS before the implementation is an immediate need. In this work, we targeted the detection of a greater number of ISs by combining two existing methodologies. The first challenge, was the fact that the explosion of states generated by the LTSA while defining different relation between the MSCs composing the system. The second challenge was to prove that the detected ISs form the system modeling really exist when monitoring the software if the system is developed and we are keen to work further on this challenge in future. Our overall goal is to automatically detect implied scenario at the requirement level using various forms of communication diagrams. Our technique was a combination of the two existing approaches. We are currently working on a platform for verification, testing and monitoring multi-agent systems using the JADE platform. We plan to extend this work by integrating the generation of the MSCs of the system from the execution trace or from the code within this tool. Moreover, we target the implementation of a plugin for Eclipse IDE that graphically shows the implied scenarios on top of the MSC diagrams.

VI. ACKNOWLEDGMENT

This work is partially supported by a grant from NSERC, Canada.

REFERENCES

[1] J. Chakraborty, D. D'Souza, and K. N. Kumar, "Analysing message sequence graph specifications," in Lecture Notes in Computer Science, 2010, vol. 6415 LNCS, no. PART 1, pp. 549-563.
[2] S. Uchitel, "Incremental Elaboration of Scenario-Based Specifications and Behavior Models Using Implied Scenarios," 2003.
[3] F. H. Fard, "Detecting and Fixing Emergent Behaviors in Distributed Software Systems Using a Message Content Independent Method," UNIVERSITY OF CALGARY, 2016.
[4] S. A. Chernenok and V. A. Nepomniaschy, "Analysis and verification of message sequence charts of distributed systems with the help of coloured Petri nets," Autom. Control Comput. Sci., vol. 49, no. 7, pp. 484-492, 2015.
[5] G. N. Rodrigues, D. Rosenblum, and J. Wolf, "Reliability Analysis of Concurrent Systems Using LTSA," in ACM/IEEE International Conference on Software Engineering (ICSE) Companion, 2007, pp. 63-64.
[6] H. Foster, S. Uchitel, J. Magee, and J. Kramer, "LTSA-WS: A Tool for Model-Based Verification of Web Service Compositions and Choreography," 28th Int. Conf. Softw. Eng. (ICSE 06), pp. 771-774, 2006.
[7] F. H. Fard and B. H. Far, "Detection of implied scenarios in multiagent systems with clustering agents' communications," Proc. 2014 IEEE 15th Int. Conf. Inf. Reuse Integr. IEEE IRI 2014, pp. 237-244, 2014.
[8] F. H. Fard and B. H. Far, "Detection and verification of a new type of emergent behavior in multiagent systems," in INES 2013 - IEEE 17th International Conference on Intelligent Engineering Systems, Proceedings, 2013.
[9] F. H. Fard and B. H. Far, "A method for detecting agents that will not cause emergent behavior in agent based systems - A case study in agent based auction systems," in Proceedings of the 2012 IEEE 13th International Conference on Information Reuse and Integration, IRI 2012, 2012.
[10] F. H. Fard and B. H. Far, "Detecting a certain kind of emergent behavior in multi agent systems applied on MaSE methodology," Can. Conf. Electr. Comput. Eng., pp. 03, 2013.
[11] H. Ben-Abdallah and S. Leue, "Syntactic detection of process divergence and non-local choice in message sequence charts," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 1217, pp. 259-274, 1997.
[12] H. Muccini, "Detecting Implied Scenarios Analyzing Non-local Branching Choices," in International Conference on Fundamental Approaches to Software Engineering, 2003, pp. 372-386.
[13] G. J. Holzmann, "The Model Checker SPIN," Ieee Trans. Softw. Eng., vol. 23, no. 5, pp. 279-295, 1997.
[14] I. G. Song, S. U. Jeon, A. R. Han, and D. H. Bae, "An approach to identifying causes of implied scenarios using unenforceable orders," Inf. Softw. Technol., vol. 53, no. 6, pp. 666-681, 2011.
[15] Mousavi Abdelmajid, "Inference of emergent behaviours of scenario-based specifications," University of Calgary, 2009.
[16] M. Moshirpour, "Model-based Analysis of Software Requirements for Distributed Software Systems," 2016.
[17] S. Uchitel, R. Chatley, J. Kramer, and J. Magee, LTSA-MSC: Tool support for behaviour model elaboration using implied scenarios, vol. 2619. 2003.
[18] R. Richard Taylor, EDD, "Interpretation of the Correlation Coefficient: A Basic Review." JDMS, 1990.