Received February 28, 2020, accepted March 13, 2020, date of publication March 19, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2981789

A Crisis Situations Decision-Making Systems Software Development Process With Rescue Experiences

ALIREZA NOWROOZI[1], PEYMAN TEYMOORI[2], TOKTAM RAMEZANIFARKHANI[2,3], MOHAMMAD REZA BESHARATI[4], AND MOHAMMAD IZADI[4]

[1]Department of Media Engineering, IRIB University, Tehran 1995614317, Iran
[2]Department of Informatics, University of Oslo, 0373 Oslo, Norway
[3]Department of Technology Systems, University of Oslo, 0373 Oslo, Norway
[4]Department of Computer Engineering, Sharif University of Technology, Tehran 1458889694, Iran

Corresponding author: Peyman Teymoori (peymant@ifi.uio.no)

The work leading to these results, by Toktam Ramezanifarkhani, was supported in part by the SCOTT - Secure Connected Trustable Things. SCOTT (www.scottproject.eu) was supported in part by the Electronic Component Systems for European Leadership Joint Undertaking under Grant 737422. This Joint Undertaking receives support from the European Unions Horizon 2020 Research and Innovation Programme and Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, The Netherlands, Belgium, and Norway.

ABSTRACT Previously, we have proposed a computational model for decision-making in crisis situations called C-RPD (Computational Recognition Primed Decision). In this paper, a software development process customized for Crisis Situations Decision-Making Systems (CSDMSs) is proposed. Agile processes can skillfully manage uncertainty in software requirements and some of their features like incremental development can solve some problems in developing CSDMSs. However, these processes do not provide comprehensive solutions for issues like the lack of enough knowledge about CSDMSs, very rapid changes, urgent need to overcome security challenges, high development unpredictability, and the performance test. Extreme Programming (XP) is one of the best and most widely-used agile processes. In this article, a customized version of XP called Crisis Situations Decision-Making Systems Software Development Process (CSDP) is proposed. Standing first and second in five national and international RoboCup rescue agent simulation tournaments from 2006 to 2010 bear witness to the efficiency of the developed software using CSDP. Relying on its characteristics, CSDP has been able to practically tackle the challenges of developing CSDMSs such as the lack of crisis-related knowledge and cumulative nature of crisis-related knowledge, difficulty of extracting knowledge, long development cycle, and sudden and frequent changes in system requirements.

INDEX TERMS Agile software development process, crisis management, crisis situations decision-making system, naturalistic decision making, recognition primed decision model, RoboCup rescue simulation agent benchmark.

I. INTRODUCTION

Although the term ‘‘crisis management’’ has been widely used in the literature, the terminology is ambiguous. Crisis management can be taken to refer either to managing a crisis after it has arisen–that is, intervening in a crisis situation–or managing in such a way that a crisis does not arise in the first place [1]. Allinson believes it is best to avoid using such a label since the word ‘‘management’’ implies that the process so labeled is envisioned as a solution to the problem of crises in general [1]. The term ‘‘crisis management’’ should be used only for the crises that have already arisen. Having studied different definitions of crisis in [1], we define the crisis as a critical situation which may dramatically turn into a more severe situation with lots of casualties and damages if it is not handled in an appropriate manner in operational settings (time and resource limitations) by emergency services (police, fire service and ambulance service). Since crisis management is a new and non-mature science [1], research studies have only focused on plans, rules, guidelines, procedures and sociology of crisis [2]. Any crisis always involves complex, uncertain, and unstable events and it is unpredictable

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

and sudden. Moreover, crisis in complicated communities such as smart infrastructures becomes an over-increasing challenge [3]; one of the main challenges is an urgent need to tackle critical situations such as security attacks and their effects on the whole system. Existing products are generally not homogeneous, and they are hardly able to make right decisions in such situations [4].

Timely and proper decision making in dealing with complex operational situations is a complicated task and dependent upon different factors of which time pressure, changing conditions, vague and shifting goals, vague and incomplete information and uncertainty are of great importance [5]. These factors alone can render analytical decision making approaches ineffectual [6]. Decision making in emergency situations is totally different from decision making in normal situations. In normal situations, a decision maker has got enough time at hand to weigh the advantages and disadvantages of different options which are fixed. Because of time pressure and changing conditions, we need a proper approach which can be effective in emergency situations. Unbounded rationality-based approaches analyze various options in order to come up with the most effective and appropriate ones [5]. Herbert Simon tackled this problem by introducing ''satisficing'' to the Bounded Rationality literature through blending ''satisfy'' and ''suffice'' [7], [8]. Generally, bounded rationality considers human decision making as an ability to find a satisficing solution rather than an optimum one.

There are two main problems in developing efficient systems (agent-based or decision-support) for decision making in crisis situations: how to decide in crisis situations and how to develop a software which is able to make right decisions in a crisis situation. To deal with the first problem which is related to the artificial intelligence field, we proposed a model called Computational Recognition-Primed Decision (C-RPD) model in [9].

We address the second question by the software engineering field, part of processes and process models for software development. To the best of our knowledge, there is not such a development process tailored for Crisis Situations Decision-Making Systems (CSDMSs). Effective software for these systems, however, cannot be developed using traditional and classic methods due to the lack of business-related knowledge and its cumulative nature, difficulty of extracting knowledge, dynamicity and extensibility of system, and highly frequent changes in system requirements.

Agile processes, on the other hand, can be customized to satisfy the development needs of CSDMSs. Extreme programming (XP) which is a popular agile software development process can respond more effectively to customer requirements and tries to develop a high quality software at the same time [10]. To use a process in a certain application, it needs to be customized for that application. Since XP has been the most widely used and the most successful process in similar systems [11], it has been used as the basis for customization in our project and particular needs in developing CSDMSs have been added to it. The results of implementing the customized process in rescue simulation environment are presented here.

Our extensive experiences of developing such systems reveals that developing complex operational software using current approaches faces serious challenges and it is vital to resort to newer and different approaches. In addition, using our proposed computational model, C-RPD, in software development and our successful results in practice (see our final results in RoboCop Rescue competitions as an example of CSDMSs [12]–[16]), confirms that C-RPD can be effectively utilized as a decision making engine in a software development process to overcome challenges of developing CSDMSs. *It should be noted that in this article we are not addressing the issue of how CSDMS should decide in crisis situations; we are presenting a customized software process to develop CSDMSs.* To avoid misunderstandings, we use the terms ''development system'' and ''resulting system''. By development system we mean a system which is under realization in a process and is not completed yet. Resulting system is the final release of the system and can be run in operational environments. About the operation of CSDMSs is touched upon but our main concern is their software process aspects. Because the proposed process depends on the concepts of C-RPD, just a brief introduction of C-RPD and its basis will be presented in the following Section. More explanations are available at [9].

The reasons behind customizing agile processes to develop CSDMSs are: lack of domain knowledge, cumulative nature of domain knowledge, difficulty of extracting knowledge, long development cycle, the vagueness of requirements, sudden and highly frequent changes in requirements. The vagueness of requirements and frequent changes are by-products of the first two items. Owing to their incremental structure, agile processes have great potential for developing adaptive operational systems. The current methodologies and even agile processes for developing CSDMSs suffer some drawbacks which will be discussed in the following. In contrast to traditional systems, in such systems customers are not aware of *Problem Scenarios* (User Stories) as aware as they are expected in current software processes, and they cannot help determine CSDMS problems, challenges and requirements. Instead of expecting customer to write about requirements, user stories are an agile approach to requirements focus on talking about them (extracting requirements in conversations with customer). Therefore, extracting user/problem stories and determining requirements is a complex process which has not been delineated in agile systems. These drawbacks are clearly felt when we find out that unbounded solutions are not effective and we need to look for bounded rationality-oriented satisficing approaches. Due to lack of real testing situations, determining the truth of user stories is difficult and without simulation, gaming meetings or mental simulation is practically impossible. In such situations, it is necessary to monitor events in order to find an appropriate simulator or pattern catalog for any new problem and similar CSDMSs in order to strengthen the evaluation of our CSDMS.

System stability, which is defined as a system's ability to continuously function well in or above an acceptable period is one of the key features of CSDMS and must be achieved gradually during the development process. Using performance test activity which effectively tests each system release through simulating different *problem scenarios* we ensure the convergence of the process for developing a stable system and protects us against being caught up in an unexpected real *problem situation*. Performance activity test allows us to correct these bugs before the real *problem situations*. The efficiency of such a process needs to be assessed properly as well. Our novelties in this paper are summarized as follow:

- We present a novel software development process to develop Crisis Situations Decision-Making Systems (CSDMSs); this overcomes the challenges of developing these systems including the high degree of unpredictability and unknown/vague requirements.
- We customize the XP process because it can offer some useful practices in such situations, and we believe CSDMSs can be best developed using an agile process.
- We adopt best practices of useful psychological and computational decision-making methods in crisis situations such as C-RPD [9]. This indeed enhances our proposed process beyond the state of the art.
- We present the lessons we learned of applying our process. Our experiences confirm some of XP practices; however, we discuss further the practices that should be adopted carefully or need more discussion.
- We evaluate our proposed process under very competitive software development situations: we apply the process to develop software for several national and international RoboCup rescue competitions, i.e. one of the best examples of CSDMSs, and show how our process succeeds in developing outstanding applications with top rankings. We see this as a very useful method of quantifying the quality of our developed software and its development process.

The paper is organized as follow: Section II reviews the related work and discusses C-RPD. In Section III, we present our proposed process, CSDP, and in Section IV the benchmark of CSDP is presented. Section V summarizes the lessons we learned. Finally, Section VI concludes the paper.

II. LITERATURE REVIEW

Agile software development is a member of current and future trends of software business, platforms and ecosystems (see [17] and [18]). Nowadays, there are a number of agile software development methods such as Scrum, eXtreme Programming (XP), Feature-Driven Development (FDD), Adaptive Software Development (ASD), Dynamic Systems Development Method (DSDM) [19]. In a study conducted in 2008 by Dyba, it was found out that about 76 percent of research studies were on studying XP [20]. In 2017, 30 percent of reviews found in the agile software development literature were about XP and Scrum [21]. XP is a popular process which is employed in many projects. However, it is utilized mostly in small or medium-sized projects [22]. Investigating the efficiency of XP due to its ever-increasing popularity is one of the great challenges in software engineering research studies [23].

A. REVIEW OF XP

XP is a lightweight software process, which deals with vague or changing software requirements. In this process, the team size is fewer than 10 people who are expected to be in the same physical location [11]. It is organized as four framework activities: planning, design, coding, and testing. XP allows an agile team to create subsequent software releases delivering features and functionality that have been described and then prioritized by the customer [24]. The XP development paradigm is the object-oriented approach. Four important characteristics of XP are as follows:

- Creating a spike solution: to find responses to the technical and hard design.
- Release planning: in such sessions, a plan that shows the general framework of the project is presented for release. The detailed plan of each iteration is developed just before that iteration.
- Iterative development: this can add agility to the development process. In this phase, the development is divided into 1 to 3 week phases.
- Acceptance test: these tests are constructed based on User Stories. During each iteration, first, selected user stories for that iteration are turned into the acceptance test. Then, the customer specifies the scenarios which can be used to determine the success of each User Story.

Fig. 1 shows how team members work with each other in XP. To be brief, we do not explain XP further. For more information see [20], [24], [25].

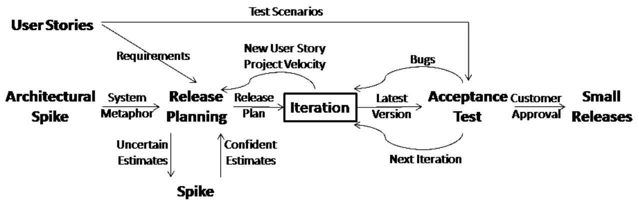


FIGURE 1. The XP Project (adopted from [25]).

B. NATURALISTIC DECISION MAKING (NDM)

Classical decision making strategies generally define a set of principles and criteria to choose the best (i.e., optimal) from a number of choices [5]. In other words, classical methods usually base these criteria on Expected Utility Maximization approach where options are chosen or removed based on priority and appropriateness. Despite the fact that these methods have yielded good results [9], research has shown that in real-world scenarios, they have not been very useful in providing assistance for decision makers [9].

The subject of Naturalistic Decision Making (NDM), with two decades of application, shows how people use their expertise to make choices in operating conditions and it has led to the emphasis on specialization [9]. NDM paradigm believes that instead of coming up with different options, rating them, and then choosing the most suitable one, the experts should place emphasis on situation recognition and awareness. There is no doubt that Herbert Simon's works can be considered as the beginning of these research studies by introducing Bounded Rationality solutions. Some of the well-developed theories of NDM paradigm are: Explanation Based Reasoning Theory [26], Image Theory [26], and RPD model [27]. See [9] for more information on NDM.

C. BRIEF DESCRIPTION OF RPD

The RPD model which was introduced by Klein and his colleagues models the behavior of experts in operational settings [27]. This cognitive psychological theory is based on field studies of fire commanders while extinguishing fire. RPD believes that professional decision makers using their past experiences rely on situation assessment, and while observing a situation recall the solution with no comparison between different solutions [28]. In tough and complex situations, an experienced decision maker may evaluate the course of action (CoA) by a mental simulation to ensure success. For more information see [9], [28], [29]. RPD, beyond proposing a model of expert decision making, introduces a novel psychological approach which has been used to train US Naval officers especially when facing stressful situations. The best RPD models in computer science are: Fuzzy RPD [30] and its applications [31], R-CAST [32] and its applications [33], and C-RPD [9].

In [9], we have developed a computational form of RPD on which our agents' decision making method is based (Fig. 2). Explaining C-RPD model and how agents make decisions is beyond the scope of this article. For more information refer to [9] which explains the implementation and assessment of rescue simulation agents based on C-RPD. RPD and C-RPD have been mentioned in this article only due to the great impact of the NDM paradigm on our proposed model CSDP, which will be introduced in the next section.

III. THE PROPOSED PROCESS: CSDP

In this section, a customized process called CSDP is presented for developing Crisis Situations Decision-Making Systems (CSDMSs). Fig. 3 shows the highest level of the process. Fig. 4 and Fig. 5 expand detailed actions of the CSDP activities 'User Story Extraction' and 'Iteration' presented in Fig. 3. Fig. 6, 'Development', is a sub-activity of CSDP Iteration procedure and Fig. 7, 'Collective Code Ownership' is a sub-activity of CSDP Development procedure. Actions and activities with thick borders in the figures show our new proposed items.

CSDP is a customized version of XP. The most important change in CSDP compared to XP is the addition of a new activity for extracting User Stories from the available sources on decision-making in crisis situations. User stories are crisis situations/scenarios in CSDMSs. Further in the following

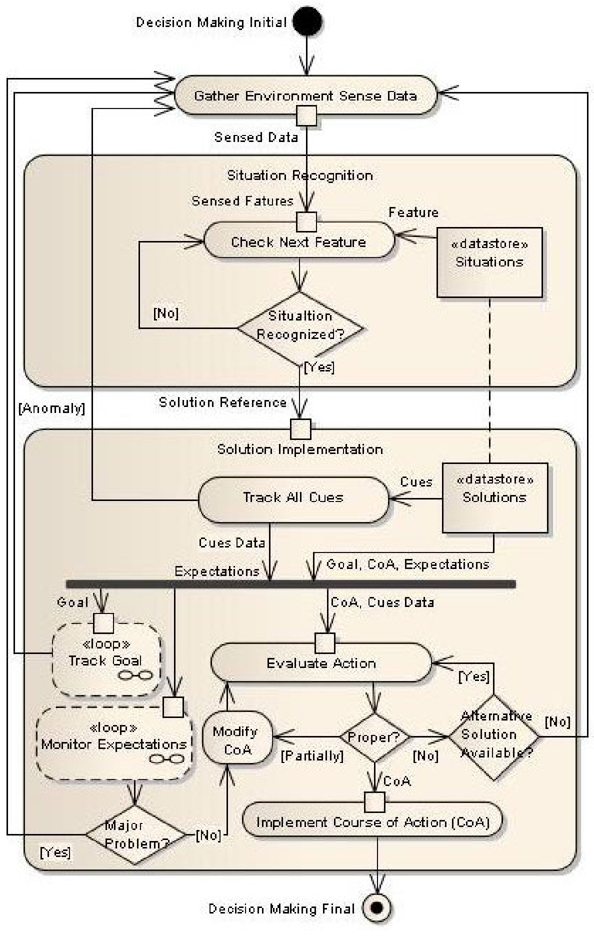


FIGURE 2. The C-RPD model.

TABLE 1. CSDP Terms.

Term	Description
Crisis Situation	The specification of a crisis situation
Crisis Solution	The solution given for a crisis situation
Crisis Scenario	Crisis situation plus solution
Crisis Simulator	A system which simulates crisis scenarios
Simulated Crisis Scenario	The simulated crisis scenario plus the details of simulation results
Similar CSDMSs	Systems into which crisis situation is fed and then they provide solutions step by step or as a whole

sections, first, a number of definitions are given, then CSDP is presented through explaining its five components, namely 'roles', 'artifacts', 'activities', 'phases', and 'iteration'. Each respective section focuses only on new ideas or changes made to XP. For more on other aspects of XP, visit [34]. Feedback loops and simulation activities are two special components of this process. These new proposed items help the process to overcome difficulties and managing complexities which are natural for Crisis Situations Decision-Making Systems.

A. THE DEFINITION OF NEW TERMS

Table 1 presents the definition of crucial crisis-related terms in CSDP. Since CSDMSs cannot be tested in the real world

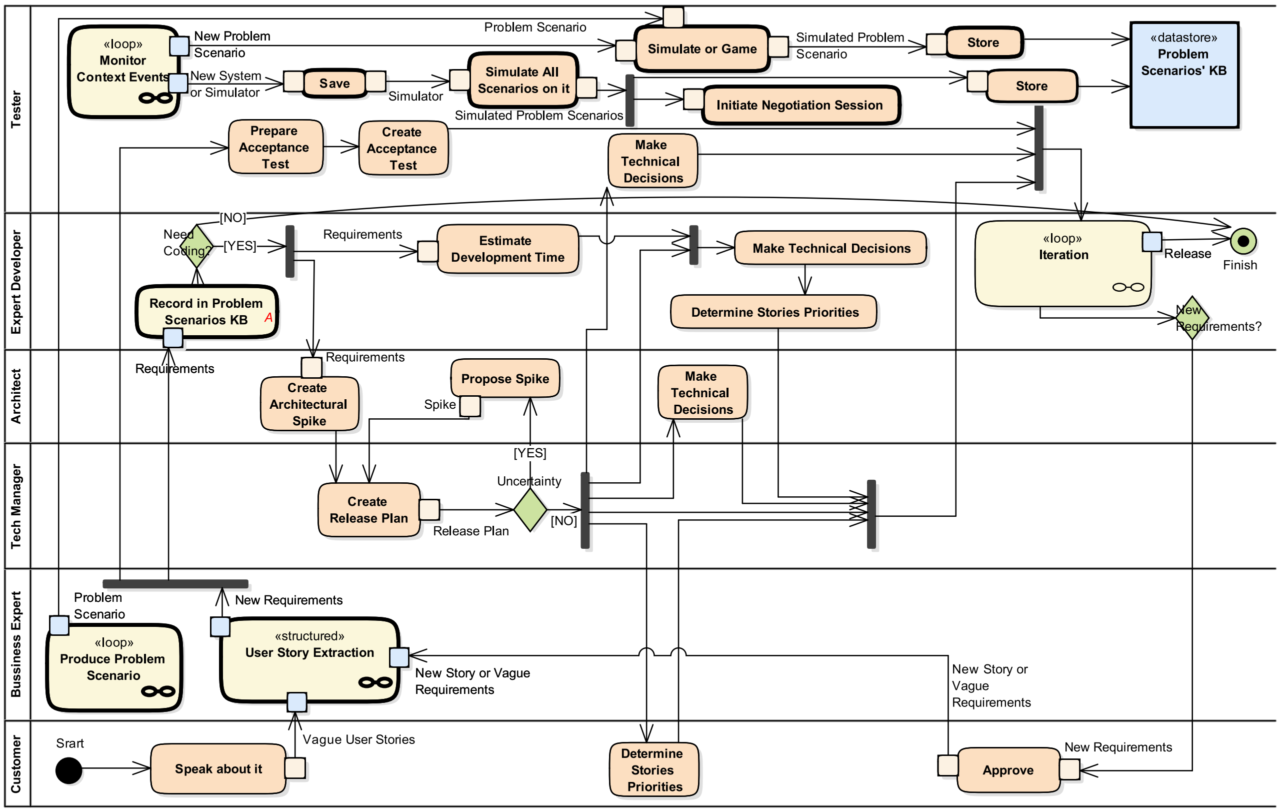


FIGURE 3. **Crisis situations decision-making systems software development process (CSDP).**

situations, simulators play a key role in developing such systems in two ways: first, they help to turn imagined crisis solutions into satisficing solutions. Second, they assist continuously in assessing the efficiency and effectiveness of the system in the course of its development by simulating it in different crisis situations. Similar CSDMSs can be used in two ways:

- The similar CSDMS merges itself with simulator and at any moment receives the new crisis situation from the simulator (gets feedback from simulator) and gives the next step of crisis solution accordingly to be executed and simulated by the simulator. This cycle continues up to the end of the crisis. The initial and the middle status of the crisis and their respective solutions make the crisis scenario.
- The similar CSDMS receives initial crisis situation as an input without resorting to the simulator and produces its own solution as an output. Since CSDMS does not get feedback from the environment in this method, CSDMS is not aware of the middle status of the crisis and the quality of solution is lowered as a result. After CSDMS produces the crisis solution, a simulator needs to be used in order to assess the solutions attained through this method. It goes without saying that the first method is much more efficient and effective.

B. ROLES

In this section, Business Expert, Technical Manager and Expert Developer roles which play a vital role in CSDP are explained. Other roles (Architect, Developer, and Tester) are either exactly similar to the roles defined in XP or their activity is a little different which will be explained in their respective section.

1) BUSINESS EXPERT

In developing CSDMSs, the customer does not either know what he actually needs or if he does so he does not know whether what he needs is logical, useful and practical in computing machines because crisis management knowledge is non-mature and complex. The execution team does not know what to do either. In such situations, the business expert or the business expert team using their knowledge and experience on crisis performs the crucial task of guiding the Customer and the execution team appropriately. Business expert (crisis expert) is expected to give an expert response to any crisis-related questions or problems of the development-team.

Although a business expert is usually at the customer-side, in CSDMSs because of lack of expertise knowledge, he can also be a development team member who has gained ample experience and expertise through working on CSDMSs. A business expert can also be a third party who is involved in the project.

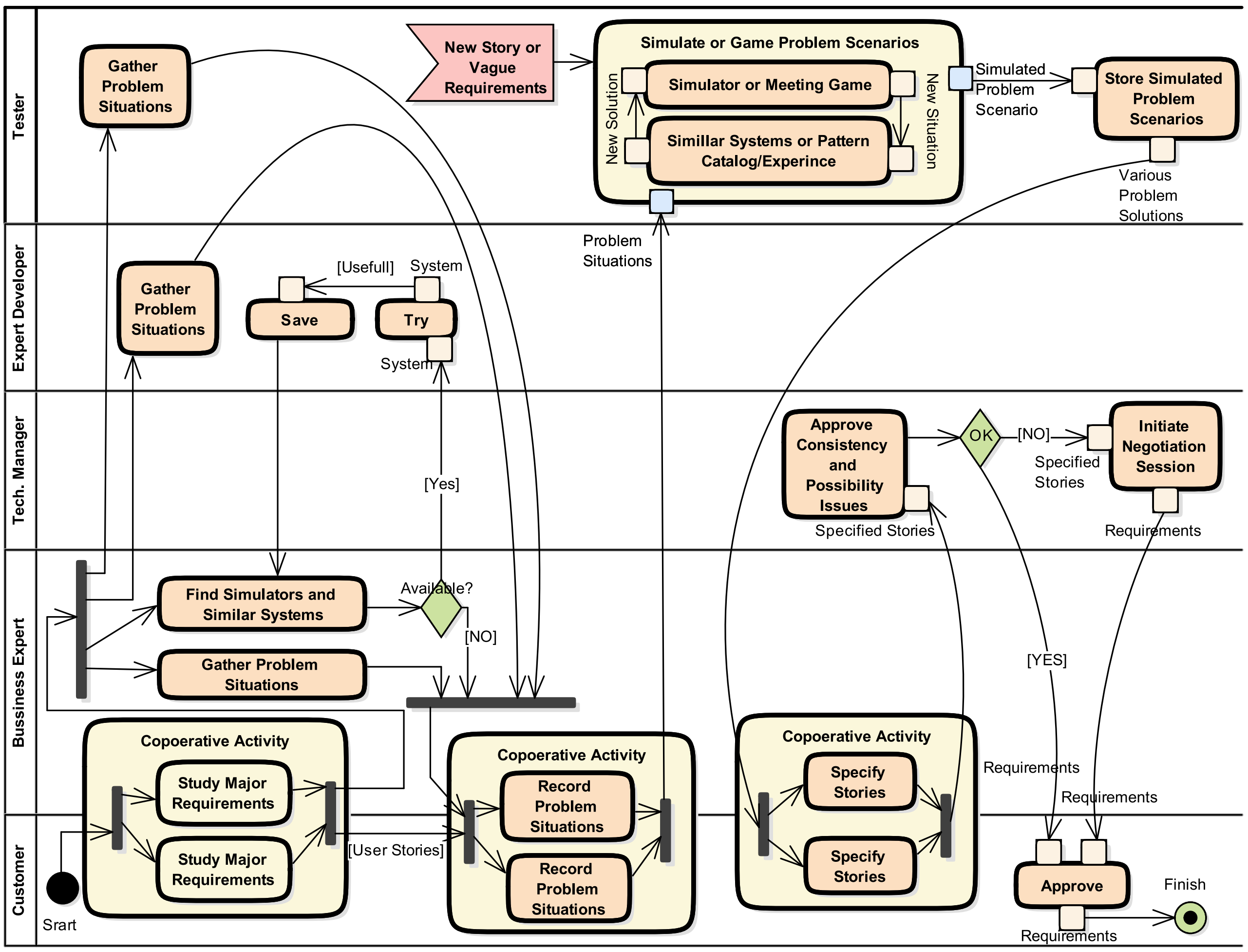

Tester
Expert Developer
Tech. Manager
Bussiness Expert
Customer
Gather Problem Situations
New Story or Vague Requirements
Simulate or Game Problem Scenarios
Simulator or Meeting Game
Simillar Systems or Pattern Catalog/Experince
New Solution
New Situation
Simulated Problem Scenario
Store Simulated Problem Scenarios
Various Problem Solutions
Problem Situations
Gather Problem Situations
[Usefull]
System
Save
Try
System
[Yes]
Approve Consistency and Possibility Issues
Specified Stories
OK
[NO]
Specified Stories
Initiate Negotiation Session
Requirements
Find Simulators and Similar Systems
Available?
[NO]
Gather Problem Situations
[YES]
Copoerative Activity
Study Major Requirements
Study Major Requirements
[User Stories]
Copoerative Activity
Record Problem Situations
Record Problem Situations
Copoerative Activity
Specify Stories
Specify Stories
Requirements
Requirements
Approve
Requirements
Finish
Srart

FIGURE 4. CSDP user story extraction procedure.

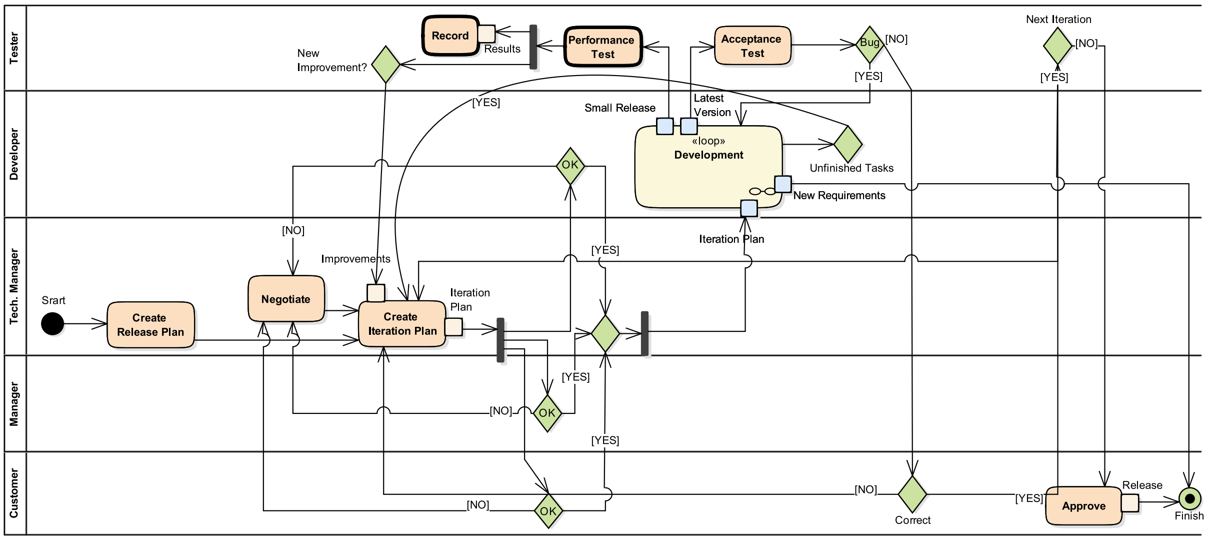


FIGURE 5. CSDP iteration procedure.

2) TECHNICAL MANAGER

The technical manager role is in fact a more complete form of the planner role discussed in XP. In addition to planner activities, a technical manager plays another activity role as well in order to be fully aware of the project's technical issues and team's moment-by-moment status and their problems.

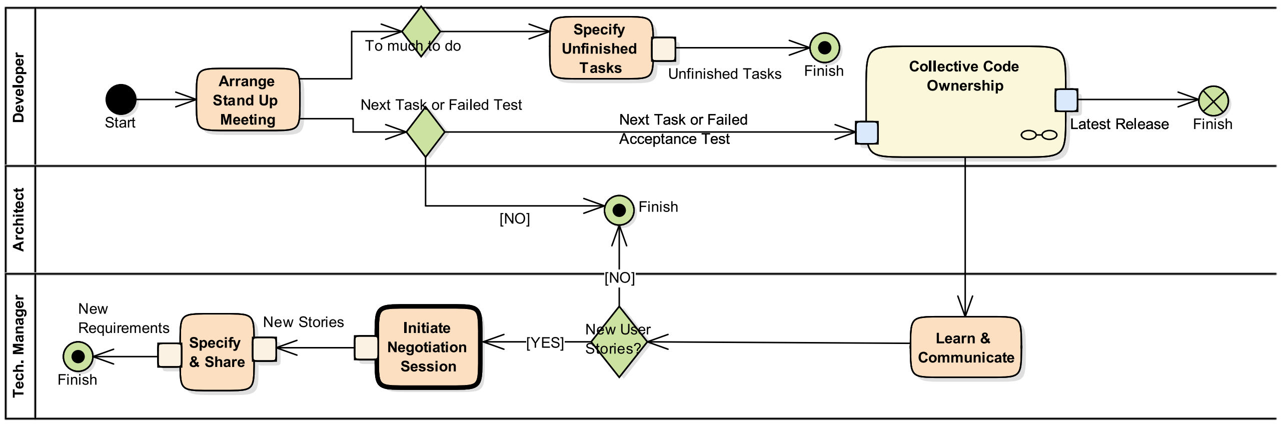


FIGURE 6. CSDP development procedure.

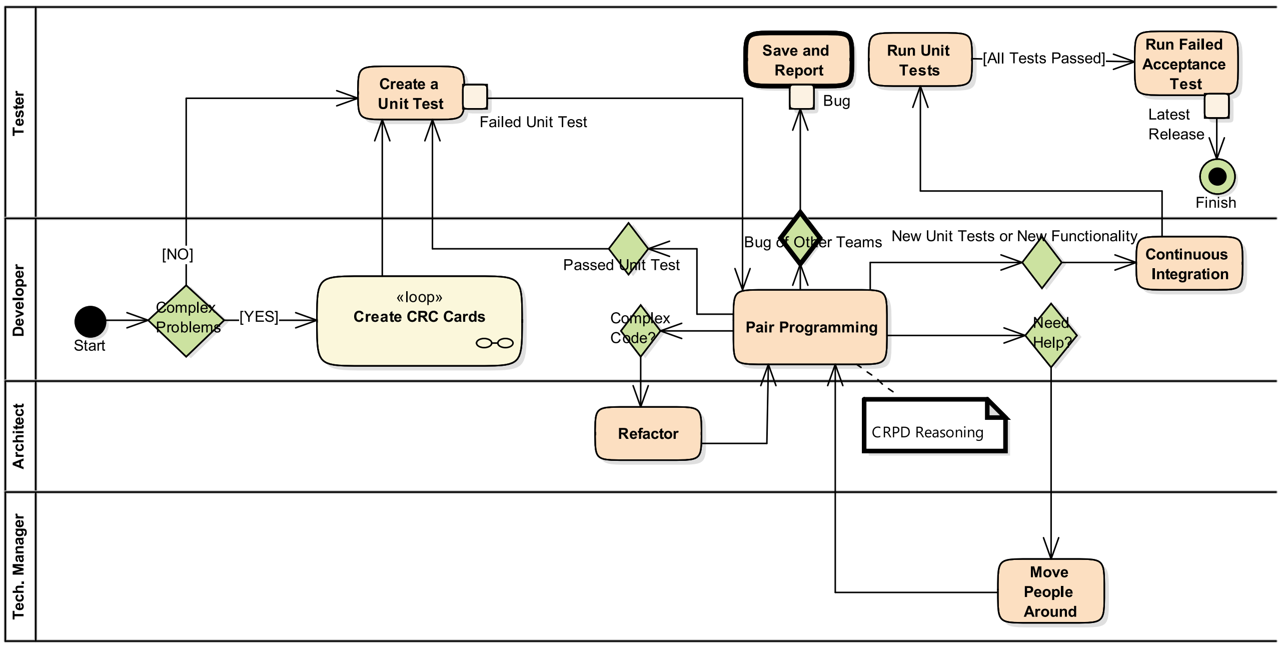


FIGURE 7. CSDP collective code ownership procedure.

He is expected to be the first person who becomes informed of the problems facing the team and rapidly comes up with a solution to prevent the project from being halted or slowed down. To perform this cooperative activity properly, the technical manager not only needs to have a constant contact with team leaders and members but he must hold regular sessions with all development teams as well. This is similar to the Scrum meetings. He must hold on-demand sessions in order to find solutions when they arise. Moreover, a technical manager is required to keep the set of all User Stories compatible in order to prevent system divergence. In a nutshell, a technical manager ensures that the project moves toward project goals.

3) EXPERT DEVELOPER

An expert developer is a professional developer who has both enough experience in programming and software development and is an expert in dealing with crises as well. He understands what the business expert states. An expert developer who is recommended to be the leader of his own team adopts two different roles, namely crisis expert and developer.

C. ARTIFACTS

XP artifacts which have been either changed or presented in a new framework in CSDP are explained in the following. These artifacts are: user stories and documents.

1) USER STORIES

Due to the importance of crisis situation seriousness, severity has been added to user stories. When the system is under stress and the sources are limited as a result, severity enables developers to prioritize their tasks appropriately. Severity is a great help in prioritizing user stories for development, too. The more sever the user story (crisis scenario) is, the better it is implemented from the beginning.

The representation used in the RPD model for stories has been proposed and used as a framework for stories. This representation has been designed in a way that is compatible with human abilities and recognition. Reference [27] supports this claim. Following this representation, we define user stories as consisting of severity, situation and solution (Table 2). Severity and situation refer to its seriousness and state, respectively. Solution which consists of goals, cues, expectation and Course of Action (CoA) refers to the solution of this state. CoA is defined as a story consisting of 5 plus/minus 2 scenes along with the reason for transition from the previous scene to the next one (Table 3). The reasons for presenting this representation for modeling experience have been given by Klein and his colleagues in [27].

TABLE 2. User story structure.

Severity	Situation	Solution			
		Goal	Cue	Expectation	CoA

2) DOCUMENTS

The trace of changes and decisions made in the course of conducting the project might be lost in the absence of proper documentation and the project can be slowed down. Inability to test CSDMSs in real-world situations necessitates reliance on simulation results. In addition, since crisis management and modeling knowledge is a new evolutionary and cumulative science with lots of improvements and new findings, the completion of current simulators and the emergence of new ones in future can improve or even contradict our knowledge on which we based our development. In this condition, feeding new improved knowledge to the system can be achieved more effectively by having thorough knowledge about the traces of previous increments. Although abundant documentation is not possible due to the agility of XP, the brevity of documentation itself is a merit if it can be used to meet the system needs. The proposed system frees the developers from the need for traditional-style documentation which has always annoyed them. The proposed documentation can be done by inserting comments into the code and filling specific tables. Tables used in CSDP are of three types: versions table, performance test table and charts for the efficiency of versions.

- Versions table: upon adding a user story which results in a new version, the developer is expected to fill one of the rows of the table with the Table 4 header. Having this table, the development team with little documentation exactly knows what it has done in each version and what the results have been. It should be noted that pseudo-code column is filled only with the link for pseudo-code or the link for implemented code file related to this particular story so that the developer can easily recall the implementation of this story in future references. In future references, the developer can manipulate previously-developed codes on condition that he remembers the condition for which the code was scripted. Since the recall process is a time-consuming one, anything like the mentioned column from the below table which can expedite this recall process can be of great help. It is highly recommended that a version be developed for each story. However, if the development team decides to include a few stories in a version, it can change the table to suit its needs.
- Performance test table: results of the performance test activity must be recorded for each release. Although performance test results are an estimated assessment of the development job, keeping them provides a great help with analyzing the process and revealing the mistakes made in user stories. To achieve this, the development team must prepare a system-assessment-parameters table and for each release fill one of its rows.
- Versions efficiency chart: a tester is required to draw a chart using performance test results and update it for each release. Since performance test results table is a table full of numbers, it is difficult to understand it at a glance. Therefore, this chart is drawn to make it easy to observe the system efficiency. This chart facilitates understanding the impact of user stories and the amount of development process progress. In addition, it helps monitor the efficiency improvement rates and see whether it has decreased or increased. Moreover, this chart immediately reveals wrong strategies or contradictions in user stories and corrects them.

D. ACTIVITIES

The new activities which have been proposed in CSDP and the altered XP activities are explained in this section. First, new tasks are briefly explained in Table 5. Next, more complex tasks which need more explanation are presented. Major and complex activities include user story extraction activity, context events monitoring activity, performance test activity and observing bugs of other development teams' activity.

1) USER STORIES EXTRACTION ACTIVITY

In traditional systems, the customer can easily express his stories and requirements, or even document them. But this is not the case in CSDMSs. In agile development, it is usually assumed by most processes that the customer can express his stories to some extent, which can be completed through talking with the developer and holding some sessions. In developing systems that are supposed to operate in complex operational situations, it is not possible to extract requirements easily. Familiarity with domain, its characteristics, and factors involved with it is an expertise and needs to be learned academically. Therefore, the extraction of user stories requires a procedure which is dependent on business expert role. In other words, user stories for CSDMSs are not just customer-dependent but they depend on other roles such as business expert and the tester. These issues will be discussed

TABLE 3. Course of action structure.

Scene1	Transition1	Scene2	Transition2	Scene3	Transition3	Scene4	Transition4	Scene5

TABLE 4. Framework of versions Table.

	Human Description of		Crisis Scenario						Score	
Crisis Class	Problem	Solution	Crisis Severity	Crisis Situation	Crisis Solution	Considerations	Pseudo Code	Version No	Before	After

TABLE 5. CSDP new tasks.

Role	Task	Description
Business Expert, Expert Developer and Tester	Gather Crisis Situations	Crisis scenarios are collected (Fig. 3).
Customer	Speak about it	In crisis management, the customer is not fully aware of stories to tell them. The customer talks about the system so that stories can be extracted in three phases (Fig. 3).
Business Expert	Produce Crisis Scenario	Business expert produces crisis scenario using his expertise (Fig. 3).
	Find Simulators and Similar CSDMSs	The tester finds the simulators and similar CSDMSs in order to complete its simulations tools.
Customer in cooperation with Business Expert (this is a cooperative activity)	Study Major Requirements	The customer and the business expert in a joint activity and cooperatively determine basic system requirements.
	Record Crisis Situations	Customer and business expert in a joint activity and cooperatively record crisis situation.
	Specify Stories	Customer and business expert in a joint activity and cooperatively specify stories.
Technical Manager	Approve Consistency and Possibility Issues	The technical manager studies the consistency of news stories with previous ones and checks their possibility. Approved user stories in this activity are called requirements in the rest of CSDP.
	Initiate Negotiation Session	If the technical manager does not approve the consistency and possibility of new stories, he holds a negotiation session with his manager, expert developer, business experts and the customer to study the issue. This activity is also done when new stories are being learned.
Expert Developer	Try	The technical manager checks the operability and the validity of the program and simulators found by the business expert.
	Save	Saves the program and the simulator.
	Record in Crisis Scenarios KB	User stories and the crisis scenario are inserted into the Knowledge Base (Fig. 3).
	Need Coding?	Expert developer checks to see whether the crisis scenario needs new basic functionalities which have not been coded yet or all requirements have been predicted and added to the system (Fig. 3).
Tester	Monitor/Watch Events of the context	The tester monitors the events of the context in order to find new similar system, new simulator, and new crisis scenario (Fig. 3).
	Simulate Crisis Scenario	The tester simulates the crisis scenario with a similar system in order to learn more about its different aspects (Fig. 3).
	Initiate Negotiation Session	The tester requests a negotiation session with the technical manager, expert developer and business expert whenever he encounters a new system or a new simulator (Fig. 3).
	Performance Test	The tester administers the performance test to the new release.
	Save	Saves the new program and the simulator (Fig. 3).
	Store	Records new crisis scenarios in the database (Fig. 3).
	Record	The numerical and textual test results are recorded in the database or table.
	Save and Report	The bugs detected by developers are saved and reported to the development team.

in the following. Another problem is the fact that the expert knowledge extraction is a difficult task.

The procedure for extracting user stories has been presented in Fig. 4. Before explaining the actions defined in this activity, it is necessary to define a new concept: cooperative activity. In performing three actions, namely, ‘‘studying major requirements’’, ‘‘writing system situations’’, and ‘‘specifying stories’’ the user stories extraction activity needs the cooperation of customer and business expert. To perform these activities, business expert, and customer temporarily join forces in order to complete the cooperative activity. Unfortunately, UML currently does not have a symbol to represent this type of activities. In order to demonstrate these activities, we produced activities which cover two lanes. Input is guided to both roles. Then, the action is performed simultaneously in both roles, and finally the outputs are joined. This does not mean that performing this activity is parallel in both roles.

The following explains the procedure for extracting user stories shown in Fig. 4. The procedure starts with the cooperation of customer and business expert to study major requirements in order for the project manager to identify the customer’s basic requirements. After this action, business expert or a team of business experts collects *problem situations* and *problem scenarios* and finds similar CSDMSs and useful domain problem simulators simultaneously. *Problem situations* indicate the situations that the resulting system will face. The *problem situations* help us to specify user stories (problem stories) as follows. Similar CSDMSs introduce us different solutions for each *problem situation*.

Problem simulators allow us to simulate and test various *problem solutions* for each *problem situation* in order to find a satisficing *problem scenario*. Parallel with this activity, tester independently performs *problem scenarios* collection action. Similar CSDMSs and problem simulators which have been found by business expert are delivered to expert developer to test and save them for future uses. It should be noted that first, business expert and customer extract user stories as *problem scenarios*, and then all the functionalities which must be provided by the system are extracted from *problem scenarios* as *problem scenarios* requirements. In fact, working on *problem scenarios* results in the extraction and the identification of system requirements.

After these three parallel actions, customer and business expert in a cooperative activity using the collected *problem situations* determine the situations that the system might encounter at run-time. Later, tester finds various solutions to each *problem situation* through operational decision-makers. To determine the efficiency of the problem scenarios, they are simulated on various simulators which were found and tested in the previous phase. In this phase, different solutions to each *problem situation* are extracted and the simulation results are collected and recorded in the respective database for future uses. It should be noted that finding solutions to a problem situation in such systems is not as easy as finding solutions to a classic problem. Moreover, since executing an ineffective solution in operational settings involves a high risk, it is vital to study the solutions to each situation to come up with the satisficing solution. Due to the fact that solutions are studied by simulators, it is obvious that possessing more and better simulators can provide us with more precise solutions. Therefore, it is clear to see the importance of finding different simulators.

After simulations, different solutions for each situation are judged by customer and business experts in a cooperative activity in order to come up with a satisficing solution and its alternatives. It must be pointed out that we do not look for an optimal solution in a *problem situation*. In other words, an optimal solution refers to a solution which can solve the problem rapidly and appropriately in operational settings considering the time pressure, changing conditions, incomplete information, resource limitations, and work pressure. Next, technical manager studies the consistency and practicality of different solutions gained from the previous activity and confirms the user stories which are now called requirements. If there are no inconsistencies and impracticality, this process finishes by producing requirements. In case there are such problems, technical manager holds a negotiation session with business expert and expert developer of the development team and the current ill-prepared solutions are turned into a collection of consistent and practical solutions. Due to the operational settings limitations and different constraints that influence the functionality of the system, such conditions are common in developing CSDMSs. In other words, there are a lot of trade-offs which prevent the implementation of solutions and user stories in their original form. In such cases, we need to find solutions which are compatible. In the end, customer confirms the determined requirements. The framework for knowledge representation is the same as the one presented in “artifacts section”.

2) MONITORING CONTEXT EVENTS ACTIVITY

Each business domain knowledge is progressing competitively and from time to time a new and influential open-source or commercial advancement in simulation and business logic is released and these advancements can reveal new aspects of the domain and can be a great help in developing such systems. For this reason, it can be seen in Fig. 3 that a new process for tester has been defined through which tester constantly monitors context events in order to find new systems and simulators and new scenarios as well. Upon facing a new *problem scenario*, tester records it and stimulates it on existing simulators to check its functioning and then saves the solutions to this situation. In addition, whenever a new simulator is found, it is necessary to simulate all the existing *problem scenarios* on that simulator, to record the simulation results and on tester’s request to hold a negotiation session headed by technical manager in presence of expert developers and tester. Due to the complexity of the complete simulation, no simulator is capable of fully modeling domain and each simulator can reveal new aspects of the problem which are expected to be carefully studied in the negotiation session when a new simulator is found. When simulator or similar system is not available, other light form of them can be used: doing simulation by gaming or mental simulation, and initiate meetings and using problem catalogs instead of using similar systems. Moreover, owing to the fact that the system functionality is dependent upon the completeness of possible scenarios which are added to it during system development, a new action called “producing *problem scenario*” has been defined for the business expert so that he can produce new *problem scenarios* in addition to the current scenarios through utilizing its expertise and delivering them to the development team to be implemented. The advantage of the system you develop over other existing systems lies in the very *problem scenarios* produced by business experts.

3) PERFORMANCE TEST ACTIVITY

The ability of the system to continue functioning in operational conditions is the greatest challenge. Due to the unpredictable nature of the real-world operations, it is impossible to fully ensure the flawless functioning of the system. However, it is essential to constantly do our best in simulating *problem situations* in order to design a robust system. Robustness is insured by adding an activity called ‘performance test’ to the iteration procedure. The performance test activity is a semi-automatic activity which for each small release fully tests efficiency of released version through both machine and expert human assessment. The results of this assessment are saved in order to have a system performance record during its development, and to be able to refer to it when it is needed. After each performance test activity, if the need for an improvement is felt in the system, it is delivered to technical manager to produce release plan for that improvement.

It must be noted that each performance test activity and acceptance test activity is conducted for every single release and version, respectively. Performance test activity automatically exposes the system to a comprehensive collection of *problem scenarios* which have been collected previously and are updated continuously by tester. As a result, system behavior is simulated and assessed automatically. Besides the automatic assessments, the domain expert also assesses the system behavior by observing and analyzing the simulations in order to reveal what has been overlooked by the automatic assessment so that a precise system evaluation is ensured. Human assessment of the system, which is unavoidable, boosts the project costs and slows down the development process. Therefore, any attempt which facilitates this assessment like providing assessment charts or pinpointing the sensitive simulation times to guide the observer directly to those points and to eliminate the need for a complete search will be of great help.

4) OBSERVING BUGS OF THE OTHER DEVELOPMENT TEAMS

In the Collective Code Ownership process presented in Fig. 7, programming teams are required to report the bugs belonging to other system development teams to the tester so that he is able to record the exact information about the bugs and report them to the respective development team. This helps to achieve stability rapidly in the developed version by preventing the teams from wasting their time due to an unknown bug. CSDMSs, especially agent-based ones, have a complicated structure which are generally tackled with parallel designs. Therefore, in such systems various modules are executed in parallel. Programmers are well aware of this programming nightmare that how a bug in a module can have an adverse effect on the functioning of other modules in a parallel system and even can cause whole system failure. However, finding that bug is very difficult due to the cascading effect and finding it needs an exhaustive trace. In such cases, the activities of all teams can come to a halt due to such bugs. The teams cannot resume work until the bugs are found and corrected. As a result, a lot of time is wasted from all development teams.

E. PHASES

The only change made to XP phases was dividing ''iteration to release'' phase into two parts. The first phase is called framework development and the second phase is called software increments [9]. The first phase release (developed framework) is called ''core product'' in Software Engineering and it is known as ''system shell'' in knowledge-based systems in AI with a little different view. The two proposed phases follow the same cycle with the only difference that in the framework development phase, it is expected that a version be rapidly developed, which contains the framework and the template of the whole project. The output of this phase is a functioning code but it might not have any intelligent mechanism to respond to any of the *problem situations*. It might not be able to comprehend *problem situations* either. This phase which like the next phase needs the user stories and other activities is just designed to inform and guide the thinking and the implementation of the code by developers to prevent them from becoming confused and slowing down the system development process as a result. To develop such a framework, expert developers or domain expert(s) need to possess experience in developing such systems or to have worked with available similar CSDMSs, if any. If the project is developers or domain expert's first experience, the development is performed without the framework development phase, but in the following projects this phase is placed at the start of the project. This phase has been proposed to overcome XP's problem with big projects and to expedite performance test. This phase should not be longer than two weeks. Software increments phase is the same as XP ''iteration to release phase''.

F. ITERATION

Iteration procedure of this process is the same as iteration procedure in XP. However, it is necessary to explain concepts like 'basis code' and its impact on merging the codes. XP wishes to be able to add each code to project immediately after development so that other development teams can work on this new version. In other words, development teams work on a version which is updated at any moment. This is not recommended for CSDMSs.

In CSDMSs, there might be many conflicts between system components. As a result, adding any code and components can influence the system functionality and efficiency, make it more difficult to find bugs, and slow down the development process. Therefore, we have introduced performing performance test activity (see Fig. 5) on the releases (a release is issued for a few iterations) to make sure the release functions well in the first place and then use it as a basis code in the ensuing iterations of the development and to the following developments on this platform. This helps programmers who always attribute the bugs to other teams to be certain that their platform has been tested well and look for the problem in their own codes and correct it. The authors of this article experienced many of these sorts of bugs which belonged to other teams' codes and faced slowing down or even the halt of the development.

IV. RESCUE AGENT SIMULATION BENCHMARK

This section is allotted to studying the effect of using CSDP in developing fire brigade agents which will decide in rescue agent simulation environment. Rescue is a league of RoboCup competitions explained in Section IV-A. The equivalents of CSDP terms (introduced in Table 1) in rescue simulation are presented in Table 6. In this section, first, rescue agent simulation environment is introduced and then in four sections which are titled strategic principles, user stories, documentation, and results of the competitions, our activities in this environment are explained within CSDP framework.

TABLE 6. Rescue terms definition.

CSDP Term	Rescue Term
Crisis Situation	City Map
Crisis Solutions	The course of actions done by each team in each crisis situation (map)
Crisis Scenario	The map and the course of actions done
Simulated Crisis Scenario	The details of the crisis scenario simulation in rescue environment
Crisis simulators	Rescue agent simulation environment
Similar CSDMSs	Other rescue teams' codes

Statistics related to development process, if mentioned without the competition name, are related to our last experience (SOS team in world championship 2009 in Graz, Austria).

A. BRIEF DESCRIPTION OF RESCUE SIMULATION ENVIRONMENT

The stimulus for the RoboCup-Rescue project was the great Hanshin-Awaji earthquake which struck Kobe on January 17th, 1995 causing more than 6500 casualties, destroying more than 80,000 wooden houses and directly or indirectly affecting more than 1 million people [9], [12]. Rescue agent simulation environment which is the platform of the rescue agent simulation league, one of RoboCup tournaments, is a popular and suitable evaluation benchmark for rescue strategies in crisis situations. It is a large scale multi-agent test-bed which consists of five components: Kernel, Agents, Simulators, GIS and Viewers. The environment simulates an urban earthquake on a designed city map, and then three types of agents, firefighters, search and rescue units and the police try to handle the crisis. Each city map has the following entities: roads, buildings, civilians, cars, refuge, fire brigades, fire stations, ambulance teams, ambulance centers, the police forces and the police stations.

In this environment, agents can make decisions on their own. By sending appropriate command to the kernel, agents can perform a variety of actions like sending messages to each other, shouting, moving, searching, rescuing people from rubble, treating the injured, extinguishing fire or performing any other task which is within their ability and responsibility. Competitors should program agents properly so that they can send suitable commands in operational settings (time limitations and resources failures) to the kernel in each simulation cycle. The team with the highest score is announced as a winner. For additional information see [12].

B. THE STRATEGIC PRINCIPLES OF THE DESIGN AND ANALYSIS OF CSDMS

There are two strategic principles in CSDP: 'loose coupling' and 'satisficing'. The whole development team (including customer) should understand these fundamental principles perfectly, practice them frequently and obey them automatically. These are as follows.

The ability of the entities to avoid halting due to lack of information to make decisions in operational settings is called Loose coupling. When there is lack of new information, they should be able to use the available information and response quickly instead of wasting time by waiting for new updated information. The entities like firefighters, rescue units, and special teams which are involved in crisis management are well aware of this fact and do not wait for orders or confirmations as far as possible and try to solve the problem by using the available information and making decisions. It should be noted that such teams have been prepared through giving necessary permissions and practicing. That is why their work style is different from other services. The most noticeable difference between crisis management and traditional management is the instability of the system resources and the necessity for instant decision-making. Not receiving information is very common in crisis situations. Traditional systems are generally designed by assuming that there is no failure or at best the systems will report the failures. However, in any of the cases, when there is a failure the system does not work and the support team is needed to solve the problem. We definitely cannot rely on the support team in a crisis situation, and in most cases, lack of instant decision-making will cause irreversible damages and make the crisis unmanageable. A loosely coupled system does not postpone decision-making until the arrival of the new information and makes decisions based on the available information. In a crises situation usually it is not possible to gather all the data so we should do our best with whatever we have right now. It is true that the decision might not be the best one but as pointed out in [27] the crisis control through instant decision-making is satisfactory. The optimization-oriented thinking leads to tightly coupled system. As an experience of this principle we should say that most rescue teams develop tightly coupled systems. These teams usually experience agent inaction in competitions. Agent inaction brings the team a zero score. This can have a very detrimental effect on the team's morale. A team facing agent inaction can overcome the problem to some extent through managing agent decision-making time in already met situations in rather a long time. However, a determined map designer can cause agent inaction for even experienced traditional teams through complicating the maps like Kobe 4 or Random Large by adding more fires with greater extent, more blocks, and more citizens.

All development team members from technical and project managers to designers and developers must strive to find and implement satisficing solutions in a loosely coupled manner. Our experiences show that teams make attempts to stand first in competitions through optimizing small components while they must look for satisficing solutions and make sure to achieve the highest efficiency level through satisficing thinking. The short-sightedness of trying to optimize all decisions and elements will annoy the team and prevent it from achieving its main goal. The most important problem which results from such optimum-oriented thinking is agent inaction in a rescue competition. Act local, think global thinking aims at correcting such viewpoints. All such problems are rooted in developers' optimization mentality. Since predictions do

TABLE 7. Rescue framework of versions Table.

Map	Problem		Situation	Solution				Considerations	Pseudo code	Version	Score	
	Description	Solution		Goal	Cue	Expectation	CoA				Before	After

TABLE 8. Rescue framework of performance test results.

BAP	FMAT	FDT	AAMT	TC	HR	PAMT	FS

not come true in operating conditions and calculations take more than expected time, no order is sent from agent to kernel in 500 milliseconds time he had.

As it was explained in 'artifacts section', what we call documentation in the project is recording the trace of the actions done and the decisions made in the course of developing the project so that it becomes possible to track changes. Documentations are generally done in the form of table notes and code comments. Three templates used in rescue documentation are as follows:

- Versions table: for each version, the developer is expected to add a row to the table similar to the Table 7 which is a customized version of Table 4 for rescue simulation environment.
- Performance test results table: upon each performance test which is conducted for each release a row is added to the Table 8 which is a performance test table for rescue simulation environment. The full forms of abbreviations in the Table 8 are as follows:
  - BAP: Burnt Area Proportion
  - FMAT: Fire agents Mission Accomplishment (Quenching all fires without the possibility of catching fire again) Time
  - FDT: Fire Death Toll
  - AAMT: Ambulance Agents Mission accomplishment Time
  - TC: Total Casualties
  - HR: Civilians' Average Health Ratio
  - PAMT: Police Agents Mission accomplishment Time
  - FS: Final Score
- Versions efficiency chart: to facilitate understanding the data presented in the Table 8, it is necessary to draw a chart. The chart helps us to understand efficiency changes easily. Average efficiency of versions produced in this project has been presented in Fig. 8 in chronological order. The horizontal axis in each point shows a selected version of a group of increments and the vertical axis indicates the efficiency of that version which has been normalized in the range of [0, 1]. Efficiency is a measure which is calculated using (1). To normalize the efficiency, (2) was used. As it can be seen, the efficiency is on the rise, especially in the beginning and in the end. In the middle of the chart where the increase in efficiency is slow, the user stories of the crises for the specified cities were being developed.

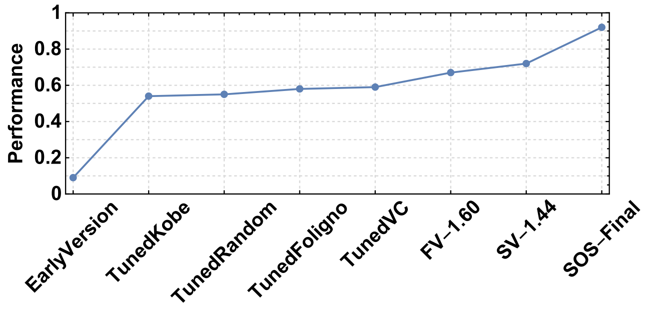


FIGURE 8. Average performance of 8 sets of increments during the development process.

$$\text{Score} = \frac{1}{37}\Big(10(1-\text{BAP})+5(1-\text{FMAT})+6(1-\text{FDT}) + 8(1-\text{AAMT}) + 3(1-\text{PAMT}) + 2(1-\text{TC}) + 2\text{FS} + \text{HR}\Big) \quad (1)$$

$$x_{\text{normalized}} = \frac{1}{\max(x) - \min(x)}\Big(x - \min(x)\Big) \quad (2)$$

A brief point about system training and system bias must be pointed out here. Test teams usually classify the crises and then use them for training the system. For instance, in rescue competitions, first we developed the system using the crisis scenarios to an unclassified collection of cities (versions 1.xx collection has been released), then the system was developed using the maps of certain cities (versions 2.xx to 5.xx were developed using the maps of Kobe, Random Foligno, and VC, respectively). Finally, versions 6.xx, 7.xx and 8.xx were developed using an unclassified collection of maps. Except for the first special collection, other developments using the special collections do not show a considerable increase in efficiency. Some developments even show a little decrease in efficiency. Two points must be mentioned regarding efficiency.

First, assessment criteria are not very exact and flawless and finding a suitable assessment criterion for rescue agent simulation environment is an open problem. Assessment criteria are expected to indicate the rises or declines and not just provide exact assessment. Due to this fact, finalists in competitions compete against each other using different maps. The most exact assessment can be done using an expert who can analyze competitor's code behavior. However, because of the personal nature of such assessments it is not viable in formal competitions.

Second, training, the system using certain maps biases the system toward those crises. Although new crisis scenarios are added to the system and it is expected that the system efficiency increase, system bias toward these scenarios causes the general system settings to become unadjusted and the efficiency of the system will decrease when it faces the map of other cities. It should be noted that the comprehensive set of test maps is fixed for all of the tests. In these conditions, previous experiences must be put together appropriately to achieve an acceptable efficiency. This has been done in the last phase (version 8.xx collection) which contains new and unmet cities. For version 8.xx collections, in most cases, new experiences were not added to the system. Instead, previous experiences were well adjusted with each other, which boosted system efficiency rapidly. In other words, system has gained enough knowledge from the previous versions.

C. ACHIEVEMENTS

CSDP has been developed and customized using our experience in rescue agent simulation competitions in the five years from 2006 to 2010. Our major achievements are as follows: Standing second with Sharif University's Impossibles in 2007 US competitions in Atlanta [13]. The time between the formation of the team and gaining the successful result was just nine months. Standing first with Amirkabir University's SOS team in the Graz world championship in 2009 [14]. In addition, we stood first in the national Khwarizmi RoboCup tournament 2009 and 2010 and international Iran Open 2010 with SOS team. For Khwarizmi tournament see [15]. Iran Open 2010 tournament results are available at [16].

V. LESSONS LEARNED

In this section, the lessons learned from our experiences of applying CSDP will be presented. First, we will look at these lessons from an XP perspective and later our other experiences will be presented. It should be noted that we use RoboCup rescue as an example of CSDMSs, and our results and achievements apply to the wide range of CSDMSs.

A. LESSONS IN APPLYING XP PRACTICES

In the following, we discuss our success or failure experiences in the RoboCup rescue simulation for those XP practices that were challenging in the new context.

1) THE PLANNING GAME (SUCCESSFUL AND UNSUCCESSFUL EXPERIENCES)

The main player here is the technical manager who is present in all key points in system development and studies all user stories to check their compatibility and priority. The technical manager is responsible for reaching tradeoffs when dealing teams' needs. He reaches the best solution through negotiation sessions. He also makes risky decisions in competitions using his experience and expertise. These decisions range from working on code stability in round intervals to implementing a novel user story. The success or failure of such vital decisions completely depends on technical manager's expertise and experiences.

2) METAPHOR (SUCCESSFUL DIFFICULT EXPERIENCE)

Explaining how the system works through using just one story and sharing it in order to create an appropriate understanding of the project among team members is impossible at least in complicated systems like CSDMSs. What we did to achieve this goal was adding the framework development phase (version 0.xx) to the system. This release is developed using a few (not just one) expert stories in a short time. By providing workflow of the whole system, classes' template, properties and functions, the available code is the same as metaphor. To keep its efficiency, this metaphor needs to be updated along with the system development. The metaphor can be kept updated during system development through using meaningful names and Scrum meetings. We gained useful experiences regarding metaphor in the new definition (framework development).

3) SIMPLE DESIGN (SUCCESSFUL EXPERIENCE)

The four finalist teams' codes are released in rescue agent simulation competitions. However, rarely can a team's code be studied and developed. This is because their codes are complicated, disorganized, and monolithic. The RPD strategy helped us develop a simple, organized and modular code despite being sizable. Due to this feature, code's functions can be easily found and altered.

4) TESTING (SUCCESSFUL EXPERIENCES)

Without designing a method for testing the code, developers and the technical manager cannot become aware of the status of development and bugs cannot be detected. Furthermore, the divergence of CSDMSs is quite possible due to the complexity of the knowledge involved in such systems. Therefore, development teams must be fully aware of this fact. This awareness cannot be gained without having a procedure for testing the code. All RoboCup teams are aware of this fact and have plans to test the code. However, developers do not usually welcome such plans. We ensured this convergence through using automatic and the human testing together.

5) PAIR PROGRAMMING (SUCCESSFUL AND UNSUCCESSFUL EXPERIENCES)

Our five-year experiences taught us that pair programming is dependent on programmers, their characters, and the degree of their matching with each other and cannot be forced on them. Both programmers must be interested in pair programming and suggest it themselves. Adding a new member to the team is very time-consuming and rescue teams usually avoid this despite the fact that they greatly need it. New member's pair programming with an expert programmer can expedite the new member's mastery of the code so that he can independently play a vital role in developing the code. We had both successful and unsuccessful experiences in this regard. In our successful experience, the programmers themselves were eager to do pair programming as a result we were able to add a new programmer to our team even a short time before the competition.

6) COLLECTIVE OWNERSHIP (SUCCESSFUL EXPERIENCES)

We were able to achieve this ambitious goal through utilizing a concept called basis code. Each developer could make any changes to the code any time he wished. However, these changes were available to other developers after they were approved for each release through conducting the performance test. Before each release, all developers worked with the older version of basis code whose stability had been approved completely.

7) ON-SITE CUSTOMERS (SUCCESSFUL AND UNSUCCESSFUL EXPERIENCE)

We believe that on-site customer is a bit exaggerated. Although there must be a close relationship between the customer and the developer and the customer must be available to respond to all questions and vague issues, the relationship between the customer and the development team is a trade-off, which can boost the quality and agility on the one hand, but can have a negative effect on the development team's peace of mind on the other hand. Therefore, based on our experiences there is no doubt that there must be a relationship, but the way this relationship is formed must be dealt with carefully and appropriately.

8) SIT-TOGETHER (VERY SUCCESSFUL EXPERIENCES)

Sitting together is very important and effective. However, more than enough presence of manager and the technical manager among programming team members like on-site customer can disturb the team and decrease efficiency. Therefore, based on our experiences avoiding the over-use and under-use of sit-together can make it effective.

9) ENERGIZED WORK (SUCCESSFUL EXPERIENCE)

Since money plays a key role in energizing people in profit-centered projects, project managers might wrongly think that this is the case in such projects. What drives the team ahead is the satisfaction the team members derive from feeling effective and useful. Since we worked on a nonprofit-centered project, we understand the importance of such feeling. Therefore, a successful manager is the one who always satisfies the team mentally and makes them feel useful. This factor does not have a financial cost, but it is very influential.

10) STORIES (VERY SUCCESSFUL EXPERIENCES)

We are well aware of this concept and have been able to form a good relationship with it. This concept is one of the key concepts in RPD. Agent requirements were identified in rescue simulation environment through these user stories (problem scenarios).

11) WEEKLY CYCLE (SUCCESSFUL AND UNSUCCESSFUL EXPERIENCES)

Our team members were still university students whose academic studies took precedence over the RoboCup activities. Therefore, their working hours could not be regular and continuous, so we could not issue weekly releases. However, intervals between our releases were considerably shorter than other teams' intervals. In other words, in case we could have worked on the project full-time, we could have had releases every weeks. In our recent experience which was a four-month full time activity for Graz world championship we issued two releases each week.

12) TEST-FIRST PROGRAMMING (SUCCESSFUL EXPERIENCES)

In CSDP, user stories can be viewed as system tests. For this reason, tests are determined while extracting system requirements so that the system can be tested automatically and then the results can be delivered to the human observer to be analyzed. Complete automation of these tests is not possible due to the fact that rescue simulation is a complex system. We definitely need an expert human observer to analyze the results and system behavior. What can be done for the human observer to reduce his workload and facilitate the analysis is processing the results. The design and implementation of a fully automatic tester subsystem especially because of great changes in kernel and simulators in rescue environment is a giant and costly task which was not within our cost-benefit criterion. We have achieved good results through combining automatic testing and human observer analysis, which had a lower cost.

13) INCREMENTAL DESIGN (VERY SUCCESSFUL EXPERIENCES)

As mentioned before, our process model is completely based on incremental development. In developing rescue simulation environment agents, we have designed and developed 'new functionalities' and 'performance improvements' increment by increment. The statistics regarding these increments have been presented in Section IV-B. This kind of development has been introduced in [9] as an approach for developing expert systems.

14) MOVE PEOPLE AROUND (SUCCESSFUL AND UNSUCCESSFUL EXPERIENCES)

Moving people can increase their concentration because they will face new environments. However, this can reduce efficiency due to their lack of expertise in the new environment. What is annoying in moving people is the fact that a team of experts is by no means equal to an expert team. A team consisting of semi-expert members can be more expert than a team consisting of fully expert members. Forming an expert team rests on the manager's and technical manager's shoulders. Qualities like friendship among members and being a perfect match play a vital role in this regard. Considering the above issues and our failures, it is highly recommended that movements be done, if necessary, with the members' consent and with care. It is also suggested that the new member start as a pair programmer in order to adapt more rapidly and efficiently with the new environment.

B. OTHER LEARNED LESSONS

The other learned lessons are: a functioning version, member self-reliance, new member entry, swift changes, key roles, unpredictability, keeping the atmosphere friendly, keeping the team members motivated. These are explained in the following.

1) A FUNCTIONING VERSION

In multi-agent systems, due to the effect of different components on each other, it is not of much use to assess them individually and the whole system needs to be assessed in order to find out the functionality of each component. To achieve this, the development team as soon as possible must come up with a basis version of the software which works with minimum requirements. Through doing this, the team will be able to assess well each of the following increments and ensure the convergence and efficiency of the system. For this reason, we divided the development process into two phases: 'basis version development phase' and 'training and increasing experiences phase'. While other teams remain unaware of the effectiveness of their activities for a considerable period of time and have to make major changes in their codes, our development time will be expedited through this method.

2) MEMBER SELF-RELIANCE

The authors believe that an agile method cannot achieve its goals unless expert members are included in the team and are given a free hand in forming their tasks. Although we believe in a hierarchical structure of the system, we also have a firm belief in giving freedom to members to solve problems on their own so that their needs for referring to higher levels are kept to a minimum. This is also supported by [35]. However, this method can cause system divergence which can be guided toward an acceptable trade-off by the technical manager's expertise.

3) NEW MEMBER ENTRY

RoboCup teams usually need to add new members due to various reasons, especially the instability of the team structure and the highs and lows which lie in this kind of activity. However, teams prefer to continue work with the existing team because adding new members is time-consuming and difficult. In CSDP, it is highly recommended that the new member start as a pair programmer and cooperate with one of the expert members of the team. Our experience with pair programming was a successful one so that a new member could have an effective role in code development through receiving just a short training.

4) SWIFT CHANGES

Sudden major changes in RoboCup competitions are common. The following two examples bear witness to this fact. First, less than a month before 2007 Atlanta competition the organizing technical committee decided that a third of crisis situations be designed and run in a communication-less manner. Even experienced teams could not handle this sudden change. However, we turned this threat into opportunity using our flexible development methods and stood second. Second, in Iran Open 2009, it was revealed that crisis situations included citizens surrounded by fire. Our team, which was capable of handling such situations, could have gained more-than-expected results if the technical committee had not given in to teams' pressure to change the presented scenarios. Therefore, the capability to respond quickly and appropriately to swift changes gives a team the ability to overtake the experienced teams and achieve more-than-expected results. The capacity of CSDP for providing such capability has been proven in practice.

5) KEY ROLES

In our experiences, we have witnessed the vital role of business expert and technical manager in the failure and the success of the teams. Owing to the fact that domain knowledge science is new and immature, the customer usually does not know what he needs. In such cases, it is the responsibility of the business expert to guide and to inform the customer of these needs using his expertise and experience. In other words, the business expert and the customer jointly determine customer needs. One of the most import reasons that the customers are not satisfied at the end of the project is ignoring the role of business expert in this regard. We have experienced such problems in projects other than RoboCup rescue simulation.

The following reveals the importance of technical manager's role. Although the difficulty of forming a rescue team from scratch is by no means comparable to receiving a strong and already-formed team, the team which we had formed in just a year and had achieved success in world championships could not gain any noticeable success even after two years from the time we left despite all the efforts put into the team by the new technical manager. The technical manager is the one who is fully aware of the team's technical issues and moment-by-moment status of the teams. He is the first person who becomes informed of the obstacles and immediately comes up with solutions to prevent slowing down or the halt of the whole project. These decisions are made in on-demand negotiation sessions held by the technical manager. The technical manager is expected to hold regular weekly Scrum meetings with all development team members. There is no doubt that the technical manager creates and ensures project's convergence. Therefore, a project must be started and its feasibility be checked with the close cooperation of business expert, technical manager, and customer and it must be guided to an end by the full supervision of technical manager.

6) UNPREDICTABILITY

Developing CSDMSs, like problematic situations, is unpredictable and does not proceed as planned due to lack of domain knowledge, its vagueness and cumulative nature. 'Approve consistency and possibility issues activity' has been added to CSDP due to the importance of paying attention to this issue. This issue makes a big difference between CSDMSs and all other systems.

7) KEEPING THE ATMOSPHERE FRIENDLY

As it has been mentioned in agile process-related papers, we must make sure that the atmosphere is not unfriendly. In our experiences, all team members were friends and classmates who were familiar with each other's temperament. However, offences and improper behavior and their adverse impact on the team should not be overlooked especially when a new member enters the team. The new member might be unaware of atmosphere among team members and their jargons. Keeping the atmosphere friendly is the responsibility of the business and technical manager who must identify factors which have a bad effect on the friendly atmosphere while avoiding the interference with the relationship among team members.

8) KEEPING THE TEAM MEMBERS MOTIVATED

Although motivation is crucial in the project success and new business/non-commercial projects usually start with high individual and team motivation, little by little the motivation diminishes. Manager and technical manager are responsible for preserving the individual and team motivation which will boost efficiency and expedite reaching the goals. We have experienced the job satisfaction as the best motivator in our non-commercial project. Moreover, a systematic literature review on motivations in software engineering at [36] says that the most commonly cited motivator is the job itself. We have found that the following factors play a key role in keeping the team motivated: clear goals and tasks, personal interest, good management, technically challenging which cannot be done by team members' classmates, autonomy, trust, respect, experiencing and achieving professional capabilities, and manager attempts to provide appropriate working conditions. For a list of what motivates software engineers and a model of motivation take a look at [37].

VI. CONCLUSION

An agile process customized for developing crisis situations decision-making systems was introduced in this article. We chose Extreme Programming (XP) as the basis process because it is a well-developed agile process and it was compatible with our needs as well. The proposed process called CSDP is the customized version of XP. Performance test activity, cooperative activity, and user story extraction activity have been added to this process for the first time in this article.

The customer alone cannot extract user stories, which are called crisis scenarios in this article, as he does in common systems. That is why a new activity called cooperative activity and different roles for extracting user stories have been proposed. To perform the cooperative activity, two roles must become a team and do the tasks cooperatively. We suggest that a new symbol or stereotype be added to UML for this role. The existing systems either can do their tasks or cannot whereas intelligent systems provide a range of responses. Since CSDMSs deal with human life, we have introduced a new task called performance test to make sure the system works properly in operational settings. This test monitors system development and helps the development team to detect the errors and problems and correct them.

CSDP is the result of the authors' experiences from taking part in rescue agent simulation division of RoboCup competitions from 2006 to 2010. CSDP is an agile and swift process which makes the development team capable of responding to sudden changes in the shortest possible time. In addition, CSDP could dominate the vagueness of requirements which is the result of crisis-related knowledge vagueness and its cumulative nature. Form our CSDP code-development case study, the most important lessons we learned are as follows: small releases, stories, incremental design, sit-together, understanding the importance of two roles: business expert and technical manager, gaining the ability to give immediate responses to sudden changes, learning how to add and adapt new members, developing simple designs, and test first programming.

As future directions, CSDP could be improved by preparing a scenario-simulation facility toolbox. Such a toolbox must contain a sort of service-components which support systematic, formal and rigorous scenario-simulations. For example, it is a choice to use Reo Coordination Language [38] and its facilities [39] for specification, modeling, verification [40] and simulation of agent coordination protocols. Additionally, to overcome security-critical situations that are one of the most challenging areas today, we aim to customize and extend CSDP to mitigate the attacks impact on the system, for example, by considering access control violation in the underlying modeling language [41]. Moreover, to prevent security vulnerabilities in software systems, we intend to investigate the CSDP effectiveness in secure software development to provide security by design to practically enforce GDPR [42].

ACKNOWLEDGMENT

The authors would like to give special thanks to late Professor Caro Lucas for his useful suggestions in writing this article, and acknowledge our appreciation to Aramik Markari, Babak Hashemi, Faraz Falsafi and Ehsan Soleimani of faculty of Mathematics and Computer Science at Amirkabir University of Technology as the development team of the PolyteCS team (2008-2009) and S.O.S. team (2009-2010). They are deeply grateful to Ahmad Bourghani Farahany, Mohammad Salehe, Hassan Nikain, Seyyed Hossein Mortazavi, Mostafa Vafadoost and Moslem Habibi of faculty of Computer Engineering at Sharif University of Technology as the development team of the Impossibles 2007 team. They also wish to thank Pouya Esfandiar of faculty of Computer Engineering at Sharif University of Technology as a developer of the Impossibles 2006 team.

REFERENCES

[1] B. W. Blanchard, "Guide to emergency management and related terms, definitions, concepts, acronyms, organizations, programs, guidance, executive orders & legislation: A tutorial on emergency management, broadly defined, past and present," U.S. Federal Emergency Manage. Agency, United States. Federal Emergency Manage. Agency, Washington, DC, USA, 2008.

[2] E. L. Quarantelli, *What is a Disaster? A Dozen Perspectives on the Question*. Abingdon, U.K.: Routledge, 2005.

[3] T. Ramezanifarkhani and P. Teymoori, "Securing the Internet of Things with recursive InterNetwork architecture (RINA)," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Mar. 2018, pp. 188–194.

[4] T. Ramezanifarkhani and J. Noll. Assessment of Technology. (2019). *SCOTT:D22. v2.0, 2018-07-11, Secure COnnected Trustable Things (SCOTT) Public Deliverable*. [Online]. Available: https//:www.scottproject.eu

[5] C. E. Zsambok, L. R. Beach, and G. Klein, "A literature review of analytical and naturalistic decision making," Contract N66001-90-C-6023 for the Naval Command, Control Ocean Surveillance Center, San Diego, CA, USA, 1992.

[6] G. Klein, *Decision Making in Complex Military Environments*. Fairborn, OH, USA: Klein Associates Inc, 1992.

[7] H. A. Simon, "A behavioral model of rational choice," *Quart. J. Econ.*, vol. 69, no. 1, p. 99–118, Feb. 1955.

[8] G. Barros, "Herbert A. Simon and the concept of rationality: Boundaries and procedures," *Brazilian J. Political Economy*, vol. 30, no. 3, pp. 455–472, 2010.

[9] A. Nowroozi, M. E. Shiri, A. Aslanian, and C. Lucas, "A general computational recognition primed decision model with multi-agent rescue simulation benchmark," *Inf. Sci.*, vol. 187, pp. 52–71, Mar. 2012.

[10] M. Fritzsche and P. Keil, "Agile methods and CMMI: Compatibility or conflict?" *e-Inform. Softw. Eng. J.*, vol. 1, no. 1, pp. 1–8, 2007.

[11] S. W. Lee, H. K. Kim, and R. Y. Lee, "Enterprise process model for extreme programming with CMMI framework," in *Computer and Information Science*. Berlin, Germany: Springer, 2008, pp. 169–180.

[12] RoboCup Official Website. (2011). *RoboCup Rescue Agent Simulation League*. [Online]. Available: http://www.robocuprescue.org/documentation.html.

[13] RoboCup Rescue Results. (2007). *Atlanta World Championship Competitions*. Atlanta, United States. [Online]. Available: http://wiki.robocup.org/wiki/Rescue_Simulation_League and http://www.robocuprescue.org/wiki/index.php?title=Agent2007results

[14] RoboCup Rescue Results. (2009). *World Championship Competitions*. Graz, Austria. [Online]. Available: http://wiki.robocup.org/wiki/Rescue_Simulation_League and http://www.robocup2009.org/165-0-results

[15] RoboCup Rescue Results. (2010). *Khwarizmi RoboCup Competitions*. Tehran, Iran, [Online]. Available: http://robotic.irost.org

[16] RoboCup Rescue Results. (2010). *Iran Open RoboCup Competitions*. Tehran, Iran, [Online]. Available: http://www.iranopen2010.ir/Default.aspx?tuabid=141&language=en-GB

[17] P. Abrahamsson, J. Bosch, S. Brinkkemper, and A. Mädche, "Software business, platforms, and ecosystems: Fundamentals of software production research (Dagstuhl seminar 18182)," Schloss Dagstuhl-Leibniz-Zentrum fuer Inform., Wadern, Germany, Dagstuhl Rep. 18182, 2018, vol. 8, no. 4, pp. 164–198.

[18] M. Schaffnit, "Digital ecosystems," in *Digital Business Development*. Berlin, Germany: Springer, 2020, pp. 53–71.

[19] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, "Agile software development methods: Review and analysis," 2017, *arXiv:1709.08439*. [Online]. Available: http://arxiv.org/abs/1709.08439

[20] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.*, vol. 50, nos. 9–10, pp. 833–859, Aug. 2008.

[21] R. Hoda, N. Salleh, J. Grundy, and H. M. Tee, "Systematic literature reviews in agile software development: A tertiary study," *Inf. Softw. Technol.*, vol. 85, pp. 60–70, May 2017.

[22] H. Altarawneh and A. E. Shiekh, "A theoretical agile process framework for Web applications development in small software firms," in *Proc. 6th Int. Conf. Softw. Eng. Res., Manage. Appl.*, Aug. 2008, pp. 125–132.

[23] R. Mordinyi, E. Kühn, and A. Schatten, "Towards an architectural framework for agile software development," in *Proc. 17th IEEE Int. Conf. Workshops Eng. Comput. Based Syst.*, Mar. 2010, pp. 276–280.

[24] R. S. Pressman, *Software Engineering: A Practitioner's Approach*. London, U.K.: Palgrave Macmillan, 2005.

[25] (2019). *Extreme Programming, A Gentle Introduction*. [Online]. Available: http://www.extremeprogramming.org.

[26] R. Hastie and N. Pennington, "Explanation-based decision making," in *Judgment and Decision Making: An Interdisciplinary Reader*, T. Connolly, H. R. Arkes, and K. R. Hammond, Eds. New York, NY, USA: Cambridge Univ. Press, 2000, pp. 212–228.

[27] G. Klein, *Sources of Power Cambridge: How People Make Decisions*. Boston, MA, USA: MIT Press, 1998.

[28] G. A. Klein and R. Calderwood, "Decision models: Some lessons from the field," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 5, pp. 1018–1026, 1991.

[29] G. Klein, R. Calderwood, and A. Clinton-Cirocco, "Rapid decision making on the fire ground: The original study plus a postscript," *J. Cognit. Eng. Decis. Making*, vol. 4, no. 3, pp. 186–209, Sep. 2010.

[30] Y. Ji, R. M. Massanari, J. Ager, J. Yen, R. E. Miller, and H. Ying, "A fuzzy logic-based computational recognition-primed decision model," *Inf. Sci.*, vol. 177, no. 20, pp. 4338–4353, Oct. 2007.

[31] Y. Ji, H. Ying, P. Dews, A. Mansour, J. Tran, R. E. Miller, and R. M. Massanari, "A potential causal association mining algorithm for screening adverse drug reactions in postmarketing surveillance," *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 3, pp. 428–437, May 2011.

[32] X. Fan, S. Sun, M. McNeese, and J. Yen, "Extending the recognition-primed decision model to support human-agent collaboration," in *Proc. 4th Int. joint Conf. Auto. Agents multiagent Syst. (AAMAS)*, 2005, pp. 945–952.

[33] X. Fan, M. McNeese, B. Sun, T. Hanratty, L. Allender, and J. Yen, "Human–agent collaboration for time-stressed multicontext decision making," *IEEE Trans. Syst., Man, Cybern.-A, Syst. Hum.*, vol. 40, no. 2, pp. 306–320, Mar. 2010.

[34] K. Beck, *Programming Explained: Embrace Change*. Reading, MA, USA: Addison-Wesley, 2000.

[35] T. Dybå, "Special section on best papers from XP2010," *Inf. Softw. Technol.*, vol. 53, no. 5, pp. 507–508, May 2011.

[36] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in software engineering: A systematic literature review," *Inf. Softw. Technol.*, vol. 50, nos. 9–10, pp. 860–878, Aug. 2008.

[37] H. Sharp, N. Baddoo, S. Beecham, T. Hall, and H. Robinson, "Models of motivation in software engineering," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 219–233, Jan. 2009.

[38] F. Arbab, "Reo: A channel-based coordination model for component composition," *Math. Struct. Comput. Sci.*, vol. 14, no. 3, pp. 329–366, Jun. 2004.

[39] S.-S.-T. Q. Jongmans and F. Arbab, "Centralized coordination vs. partially-distributed coordination with reo and constraint automata," *Sci. Comput. Program.*, vol. 160, pp. 48–77, Aug. 2018.

[40] M. Izadi, A. Movaghar, and F. Arbab, "Model checking of component connectors," in *Proc. 31st Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 1, Jul. 2007, pp. 673–675.

[41] T. R. Farkhani and M. R. Razzazi, "UML-based representation of provision-based access control," in *Proc. 2nd Int. Conf. Inf. Commun. Technol.*, Apr. 2006, pp. 3605–3610.

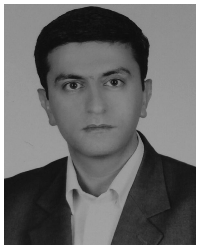
[42] S. Tokas, O. Owe, and T. Ramezanifarkhani, "Language-based mechanisms for privacy-by-design," in *Privacy and Identity Management. Data for Better Living: AI and Privacy*, F. Friedewald, M. Önen, E. Lievens, S. Krenn, and S. Fricker, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-42504-3_10.

ALIREZA NOWROOZI held a postdoctoral position at the Sharif University of Technology, and is a co-founder of four IT startups. He is currently an Assistant Professor of computer engineering with the Media Engineering Department, IRIB University. He is also a Consultant, advising government and private sector-related industries on innovative information technologies. He is a specialist in artificial intelligence, cognitive science, software engineering, IT security, and blockchain. Also, he is a recipient of some national and international rewards.

PEYMAN TEYMOORI received the B.S. and M.S. degrees in computer engineering from the Ferdowsi University of Mashhad and Amirkabir University of Technology, Iran, in 2001 and 2004, respectively, and the Ph.D. degree in computer engineering from the University of Tehran, in 2013. He was a Visiting Researcher at the Gwangju Institute of Science and Technology, South Korea. He is currently a Research Fellow at the Network and Distributed Systems Group, Department of Informatics, University of Oslo, Norway. His research interests include computer networks and software development processes.

TOKTAM RAMEZANIFARKHANI received the M.Sc. and Ph.D. degrees in information security from Polytechnic Tehran (Amirkabir University of Technology). She is currently a Postdoctoral Research Fellow at the Department of Informatics, University of Oslo. Her research interests include the wide area of information security, including but not limited to software and language-based security, network security, security mechanisms and protocols, formal methods, security measurement and assurance. She is involved in several projects working with industrial and academic partners.

MOHAMMAD REZA BESHARATI received the B.Sc. degree in software engineering from the Sharif University of Technology, Tehran, Iran. He is currently pursuing the Ph.D. degree in software engineering with the Sharif University of Technology, where he is also a member of the Distributed and Multiagent Systems Lab, Department of Computer Engineering. His main research areas of interest are logic in computer science, semantics, software engineering, software development and evolution, distributed systems, and theory of computation.



MOHAMMAD IZADI received the B.Sc., M.Sc., and Ph.D. degrees in computer engineering and another M.Sc. degree in philosophy of science, all from the Sharif University of Technology, Tehran, Iran, and the Ph.D. degree in computer science from Leiden University, The Netherlands. He is currently an Assistant Professor and the Head of the Distributed and Multiagent Systems Lab, Department of Computer Engineering, Sharif University of Technology. His main research areas of interest are logic in computer science, semantics, game theory, distributed algorithms, and theory of computation.