Int. J. Pure Appl. Sci. Technol., 29(2) (2015), pp. 54-63

International Journal of Pure and Applied Sciences and Technology

ISSN 2229 - 6107

Available online atwww.ijopaasat.in

Research Paper

Multi-Agent Based Agile (XP) Software Development Process Scheduling Model

Y.M. Malgwi[1,*] and N.V. Blamah[2]

[1] Computer Science Department, Modibbo AdamaUniversity of Technology, Yola, Adamawa State, Nigeria

[2] Computer Science Department, University of Jos, Nigeria

* Corresponding author, e-mail: (yumalgwi@yahoo.com)

(Received: 3-7-13; Accepted: 17-7-15)

Abstract: *User requirements during software development keep changing due to evolving business needs. Most Users do not have clear vision about the specification of their requirements at the early stage. In such changing environment agile development methodology is suited. In this paper, a multi-agent based approach to process scheduling was adopted, where each activity is viewed as an autonomous and flexible agent process. Crucial for the multi-agent based system in project scheduling, is the availability of an effective model and algorithm for scheduling of task. The developed model (Multi-agent based System) provides an optimized and flexible agile process scheduling and reduces overheads in the software process as it responds quickly to changing requirements without excessive rework in project scheduling.*

Keywords: Agile Software development process methodologies, Extreme programming, Agents and Multi-agents.

Introduction

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams (Gonzalez & Pilar, 2009). It promotes adaptive planning, evolutionary development and delivery. (Hock, 2009).

The most popular definition of Agent was proposed by (Wooldridge & Jennings, 1995): An Agent is essentially a special software and hardware component that has the characteristics of autonomy, sociality, and reaction and pro-action. It is a computer system that is capable of independent (autonomous) action on behalf of its users or owner. It can provide different interaction interface for the outsiders, and even have the characteristics such as knowledge, belief, and intention and so on. (Pour *et al*. 2004).

Just as its name implies, Multi-agent System (MAS) is a system that consist of numbers of agents within an environment which interact with one another. It can be used to solve problems that are difficult or impossible for an individual Agent (Jennings, Sycara & Wooldridge, 1998).

In the most general case, agents can be acting on behalf of users with different goals and motivation. To successfully interact, they will require the ability to cooperate. Coordinate, and negotiate with each other, much as people do. Indeed, multi-agent systems enhance overall system performance, in particular along such dimensions as computational efficiency, reliability, extensibility, responsiveness, reuse, maintainability, and flexibility (Michael, 2002) & (Fabio *et al*. 200).

Lack of penetrations of the modern agile planning tools during software development process usually provides a ‘quick and dirty’ solution which is informally managed. Typically, informal managed planning factors are:

i) Scheduling of tasks and resources
ii) Relationship or Communication between entities.

Although the principles of agile development rely on communication instead of rigorous planning, this fact can be explained by the lack of easily applicable algorithm solutions. Informal approaches work well in smaller projects but not sufficient in larger projects. As the size and complexity increase, scheduling becomes a very complex process and advocate tool support.

As a consequence, optimized and flexible project plans are crucial issues from the economic considerations of both customer and developer’s side. These critics underline the importance of providing a more established model for agile process scheduling. In this article, the aim is to diminish these barriers and implement a model that schedules the agile process. The aim of the model is to provide an efficient and flexible system.

Materials and Methods

In this research, the materials used are gotten from journals and text books. A mathematical model for the Multi-agent based agile (XP) process scheduling will be design to mathematically represent the whole process. Also, algorithm to implement the mathematical model will also be developed and simulated with real life and generated data sets to actualize its effectiveness. Visual C++ is used as a tool in order to ensure smooth implementation. It is also important to point out that the model created follows the principles of branch and bound optimization technique in order to generate an optimized schedule for the XP processes stated earlier.

Problem-Solving Framework

The process of multi-agent based agile scheduling process problem solving can be modeled by a 6-phase closed-loop XP process shown in Figure 3.1. This process contains the following functions: Selection of user stories, breaking down of tasks, planning the release, Executing/developing the plan, Releasing the software and Evaluating the system by the user/ customer. The arrows represent the flow of data between each of these functions, which forms a continuous feedback mechanism.

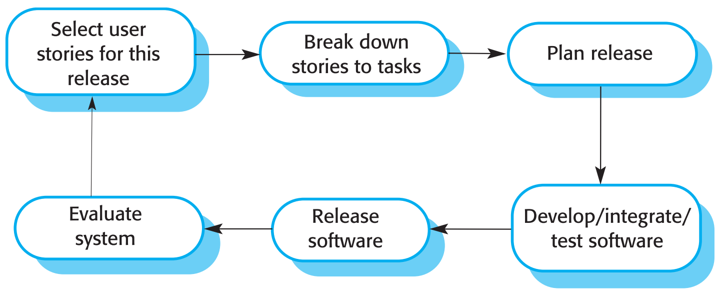
Select user stories for this release
Break down stories to tasks
Plan release
Develop/integrate/ test software
Release software
Evaluate system

Figure 3.1: XP Process-cycle framework (Ian, 2009)

Each of the functions in Figure 3.1 can be further decomposed to address the more specific problem of multi-agent based agile planning and scheduling.

The Analysis Phase

The analysis phase aimed to clarify the problem without any (or minimal) concerns about the solution. The analysis phase is carried out through a number of steps, described.

Use Case Diagram

Use cases been an effective way to capture the potential functional requirements of a new system. The use case was used in representing the XP process scenario that demonstrates how the system interacts with the external environment to achieve a specific goal. The *use case diagram* is produced as shown in Figure 3.2 below.

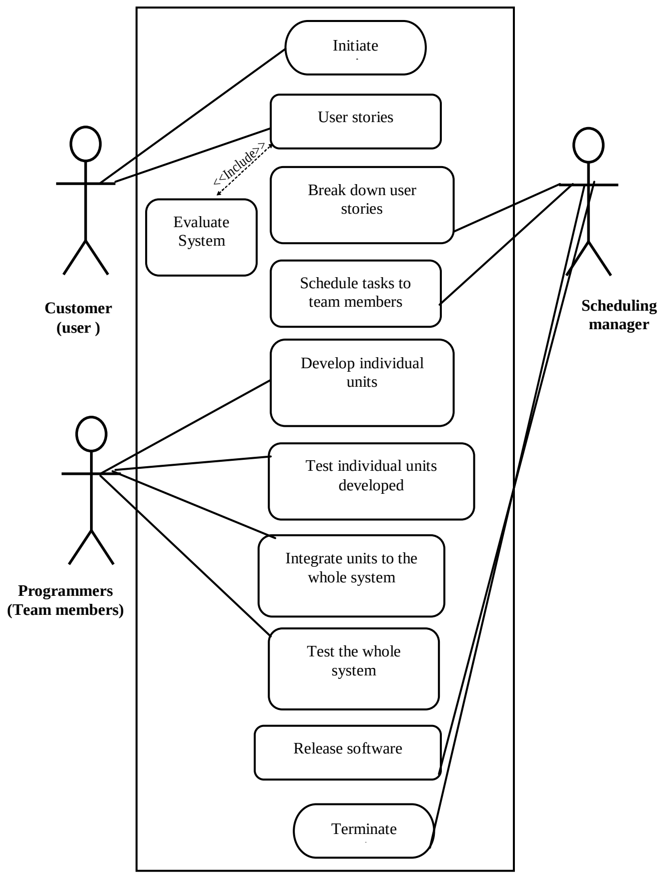

Figure 3.2: Use case diagram for XP processing

Initial Agent Types Identification

This identifies the main agent types and subsequent formation of a first draft of the *agent diagram*. The following rules are being applied;

i.) Adding one type of agent per actors.
ii.) Adding one type of agent per resource. By applying the above rules to the agile process case study, the initial diagram shown in Figure 3.3 is obtained.

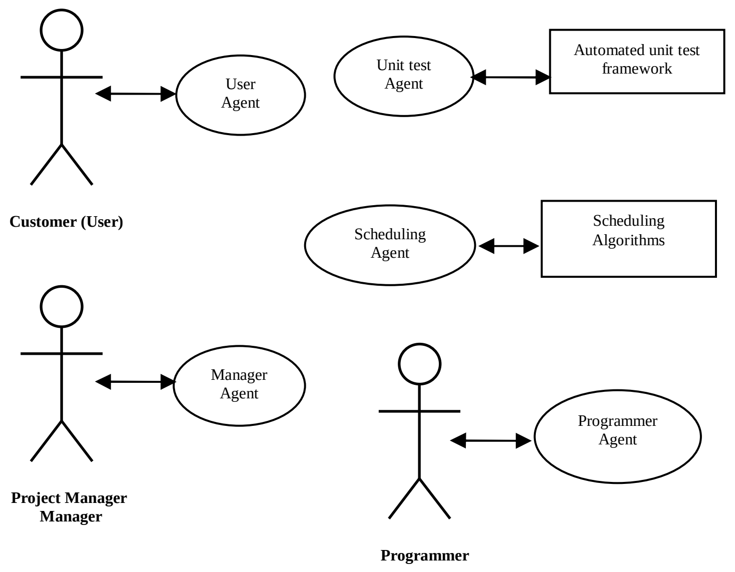

Figure 3.3: Agent Diagram for XP Processes

With reference to Figure 3.3 above, the agent diagram includes four types of elements:

1. Agent Types: The actual agent types, represented by circles.

2. Humans: People that must interact with the system under development, Represented by the UML actor symbol.

3. Resources: External systems that must interact with the system under development, represented by rectangles.

4. Acquaintances: Represented by an arrow linking instances of the above elements, specifying that the linked elements will have to interact in some way while the system is in operation.

Responsibilities Identification

In this step, for each identified agent type, an initial list is made of its main responsibilities in an informal and intuitive way. The artifact resulting from this process is the *responsibility table*.
The following rules are applied in this step:

i) The initial set of responsibilities was derived from the use cases identified in figure 3.3 above.
ii) The agents' responsibilities were considered.

By applying the above rules to the agile process scheduling case study, the consideration of the XP Process agent is initiated and Table 1 is produced.

Table 1: Responsibility table for Agents in XP Process

Agent Type	**Responsibilities**
User agent	1. Initiates the proje 2. Provides requirements inform of user stories 3. Evaluates the system at each iteration
Manager Agent	1. Gets requirements from user agent inform of stori 2. Breaks down user stories 3. Retrieves the relevant Scheduling agent 4. Tracks the activities of the Team Members 5. Releases software 6. Terminates the project
Scheduling Agent	1. Retrieves tasks and resources from manager age 2. Schedules tasks to resources.
Programmer Agent	1. Gets tasks from Scheduling agent. 2. Develops individual units 3. Retrieves the relevant Unit Test agent 4. Provides status to manager agent 5. Integrates units to the whole system 6. Tests the whole system
Unit tests Agents	1. Receives completed units from programmer agent for testing. 2. Carries out unit testing

Acquaintances Identification

In this step, the focus was on who needs to interact with whom and the agent diagram (Figure 3.2) is updated by adding proper acquaintance relations connecting agents that need to have one or more interactions. An obvious acquaintance relation in the XP process case study is required between different XP process agents.

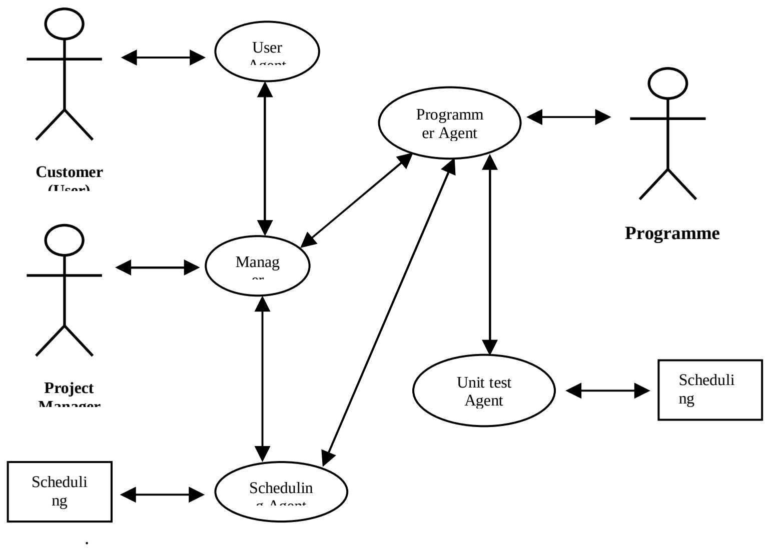

Figure 3.4: Agent diagram for XP processes depicting Acquaintance

Multi-Agent Planning and Scheduling Process

As shown in Figure 3.5, these steps consist of:

i.) Formulating or receiving from another source (users/customers).
ii.) Structuring this objective in a form where it can be easily decomposed into a partially-ordered set of sub-problems or jobs.
iii.) Surveying the environment for available agents and services that may be used to complete these jobs.
iv.) Mapping jobs to available services or sets of services that is capable of completing them.
v.) Determining the allocation of jobs to agents, such that the resulting schedule is optimized according to user-defined parameters.
vi.) Forwarding this solution to the appropriate agents for Evaluation.

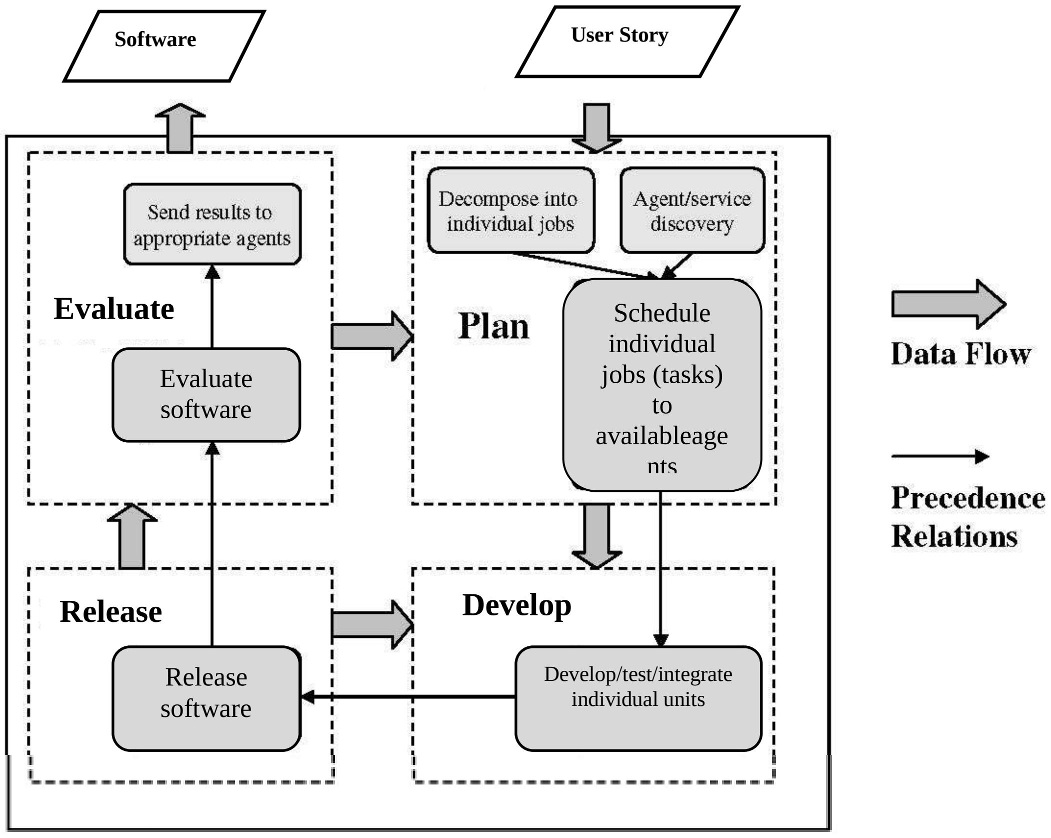
Software
User Story
Send results to appropriate agents
Evaluate
Evaluate software
Decompose into individual jobs
Agent/service discovery
Plan
Schedule individual jobs (tasks) to availableage nts
Data Flow
Precedence Relations
Release
Release software
Develop
Develop/test/integrate individual units

Figure 3.5: Multi-agent planning and scheduling process in the context of the closed-loop architecture (Mark.2003)

It is important to note that the small arrows between the specific tasks in Figure 3.5 represent *precedence relations*, while the large arrows between the high-level functions represent continuous *data flow* and feedback between these phases. It should be noted, however, that the feedback provided by the data flow could represent precedence relations at a higher level itself in the context of this architecture, as one phase may not be allowed to begin its next cycle until it has received data from another phase.

Problem Formulation and Decomposition

In the multi-agent framework that we are considering, there are sets of agent in XP process, each with a different set of available abilities and services. A single agent is given an objective to complete, possibly from another agent, and it wishes to take advantage of the resources provided by these other agents in the XP process to complete the objective more efficiently. The planning agent's first step is to decompose its objective into a number of tasks that can be allocated to other agents in the XP process and completed in parallel.

However, there are often many possible problem formulations for a given objective and choosing the best way to decompose the objective which may depend on the structure of the agent organization and the number of different service types provided by these agents. In the scenario that this research addresses, the agent's main goal is to choose the job precedence that produces the schedule with the most possible user stories completion within a fixed – time.

Mathematical Model for the Multi-Agent Based System

Problem Variables

For easily formulating a mathematical model for the multi-agent based system representing our agile software development process (more precisely extreme programming). It is inimical we identify our constraints decision variables and then formulate our objective function.

Input Parameters

j is the index of all set of available agents A. Each agent is indexed numerically denoted by *i*.

Where: P- Immediate job precedence matrix

$$P_{j'j} = \begin{cases} 1 & \text{if user story j directly precedes j} \\ 0, & \text{otherwise} \end{cases}$$

Q- Full user story priority matrix

$$Qj'j = \begin{cases} 1, & \text{if user story } j' \text{ comes anytime before j} \\ 0, & \text{otherwise} \end{cases}$$

B- Agent ability matrix

$$B_jA = \begin{cases} 1, & \text{if agent A provides the services needed to complete story j} \\ 0, & \text{otherwise} \end{cases}$$

A possible mathematical model for the multi-agent based agile software development process scheduling can be in the form formulated below.

Maximize $z = \sum_{j=1}^{m} \sum_{a=1}^{n} B_{j'a} .\ X_j\ 'a$

Subject to:

$B_j'a = 1 \qquad \forall\ j \in W, a \in A$

$X_j{}' a = 1 \qquad \forall\, j \in W$

$X_j{}' a + X_j{}' a \leq 1 \qquad \forall\, (j'j) \in W, a \in A$

Where:

$X_j{}' a = 0$ or 1, $\quad X_j{}' a \in \{0, 1\}$

$B_j{}'a = 0$ or 1, $\quad B_j{}' a \in \{0, 1\}$

Algorithm

A suitable algorithm for implementing the mathematical model formulated above is provided below:

Require: $(I^R \in N)$

$a \in A, B_j{}' a \in \{0, 1\}$

$j \in W, Q_{j'j} \in \{0, 1\}, P_{j'j} \in \{0, 1\}$

Ensure: $\forall j \,\exists!\, W$ and $\forall a \,\exists!\, A$ $(X_j{}' a \in \{0, 1\})$

```
1:   repeat
2:   $a \in A \subseteq A^*$
3:   $V^R \leftarrow \sum_j B_j{}'a: a \in A$
4:   $K, l'_k$
5:   $C_k \leftarrow V^R * l'_k$
6:   $X \leftarrow$ Schedule (Q, A, B)
7:   until X is satisfying
8:   return X.
```

Results and Discussions

Results

The model was tested using different data sets to determine its efficiency and flexibility. The data set used consist of seven actual deals that were collected, these includes Collateral evaluation (RA), Risk assumption (R_B), Ukrainian deal flow I (R_C) & II (R_D), Romanian deal flow I (R_E), II(R_F), and III (R_G).

These are indexed R_A to R_G respectively (gotten from the back log of IRIS application developed by multi logic Ltd) (Akos, 2011). All releases had same project members (16 programmers) iteration length (2 weeks), iteration velocity (30 story point), domain, customer and development methodology (XP) but were differentiated by user stories, iteration counts, length of each iteration and release duration.
In order to create more flexibility, the user of system input the different services provided by the available agents (programmers, managers, users, test agent and so forth). The model creates an optimal combination of all the different services. The propose multi-agent based agile process scheduling creates a more optimal schedule thereby having a higher Max Value as dully shown in table 4.1 below.

Table 4.1: Optimized Multi-agent based agile release plan values

	SC	IC	IL	RD	NA	PA	Max=DV
R_A	33	4	2	8	10	3	50
R_B	25	3	2	6	10	4	38
R_C	27	5	2	10	10	4	48
R_D	27	4	2	8	10	5	48
R_E	53	4	2	8	10	3	57
R_F	26	4	2	8	10	4	47
R_G	53	5	2	10	10	3	56

Keys

Max = DV - Maximum Deliverable Values

SC -	User story Count	NA -	Number of Agents
IC –	Iteration Count	SA –	Type of Agents Services
IL –	Iteration Length	PS -	Priorities of service provided by agents
RD –	Release Duration		

Discussion of Results

The main concept of agile process scheduling is based on multiple knapsack optimization technique. The proposed multi-agent based system covers a wide-ranging release scheduling (fixed time).

The multi-agent based system made it possible to adapt an efficient global optimization algorithm for more flexibility and smooth iterations. The algorithms strives to prevent resources overload-which often yields increasing delivery risks, and prevent resource underload-which captivates economically, and badly utililized iterations.

The results reveal that the outcome of the research is an extension of readily available scheduling tools which helps collected the process scheduling data (user stories, required effort, team velocity, etc) Therefore, with this extension, it is believed that one can produce a flexible and efficient process scheduling system easily based on the collected data.

The method also indicates that it requires no much time because of the good communications or relationship between entities. It expresses dependencies between deliverables features, as it produces optimal schedule within seconds. However, the actual major difference between the two systems (agile process scheduling and Multi-agent based process scheduling) is the higher quality schedule realized (avoiding underload and overload), as the multi-agent based system produces a better resources utilization and make it possible to re-schedule the process anytime within seconds in order to support the what-if-analysis.

Conclusion

The proposed model gives the main parameters of the typical agile process scheduling space (such as objectives and constraints) and presents an optimization model that can be realized by optimization tools or by implementing the suggested custom-made algorithm.

The goal of this thesis has been to implement a multi-agent based agile process scheduling model for solving the most difficult classes of these problems. Additionally, this approach provides more informed and established decisions with application of what-if analysis (rescheduling the release by altering its parameters).

The findings of this result reveal that:

i.) An agent provides an interoperable interface to an arbitrary system and/or behaves like a human agent working for some clients in pursuit of its own agenda.

ii.) The multi-agent systems can model complex systems and introduce the possibility of agents having common or conflicting goals. These agents may interacts with each other both directly (clicking on the environment) or directly (via communication).

iii.) The system is optimal and flexible with limited resources management capabilities.

Based on the findings of this research, the Multi-agent based systems may be recommended for instance in the field of telecommunication systems where large distributed networks of inter connected components which need to be monitored and managed in real time.

References

[1] B. Fabio, C. Giovanni and G. Dominic, Developing Multi-Agent System with JADE, John Wiley & Sons Ltd, 2004.

[2] G. Pour, F. Yao and C. Yu, A mobile agent-based architecture for mobile systems supporting distributed software project management, *IEEE Soft COM*, (2004).

[3] M. Hock, Review of agile methodologies in software development, *International Journal of Research and Reviews in Applied Sciences*, 1(1) (2009), 6-8.

[4] M. Wooldridge, An Introduction to Multi-Agent System, John Willey & Sons Ltd, 2004.

[5] N. Jennings, K. Sycara and M. Wooldridge, A roadmap agent research and development, *Int.Journal of Autonomous Agents and Multi-Agent Systems*, 16(1998), 62-87.

[6] R. Gonzalez and P. Pilar, Some findings concerning requirements in agile methodologies, *Product-Focused Software Process Improvement*, 32(4) (2009), 171-184.

[7] S. Akos, Conceptual scheduling model and optimized release schedule for agile environment, *Information and Software Technology*, 53(2011), 574-591.

[8] S. Ian, Software Engineering (Ninth Edition), Pearson Education, Inc., Publishing, Addison – Wesley, New York, 2009.

[9] M. Wooldridge and N. Jennings, Intelligent agents: Theory and practice, *Knowledge Engineering Review*, 10(1995), 115-152.