Analysis of Agile and Multi-Agent Based Process Scheduling Model

Y. M. Malgwi[1,*], N. B. Shelleng[2], K. J. Danjuma[3]

[1]*Computer Science Department, Modibbo Adama University of Technology, Yola, - Nigeria*
[2]*Mathematical Sciences Department, Adamawa State University, Mubi, -Nigeria.*
[3]*Computer Science Department, Modibbo Adama University of Technology, Yola, - Nigeria.*

Abstract:- As an answer of long growing frustration of waterfall Software development life cycle concepts, agile software development concept was evolved in 90's. The most popular agile methodologies is the Extreme Programming (XP). Most software companies nowadays aim to produce efficient, flexible and valuable Software in short time period with minimal costs, and within unstable, changing environments. This complex problem can be modeled as a multi-agent based system, where agents negotiate resources. Agents can be used to represent projects and resources. Crucial for the multi-agent based system in project scheduling model, is the availability of an effective algorithm for prioritizing and scheduling of task. To evaluate the models, simulations were carried out with real life and several generated data sets. 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 work in project scheduling.

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

I. 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*. 2000).

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 this research is to develop a multi-agent based agile process scheduling model that is more efficient, flexible and has limited resource management capabilities.

II.   METHODOLOGY

Two mathematical models (Agile and Multi-agent based process scheduling) will be design to mathematically represent the XP process. Also, two algorithms that will implement the mathematical model will also be developed and simulated with real life and generated data sets to actualize their effectiveness. Visual C++ will be used as a tool in order to ensure smooth implementation. The models created will follow 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 framework will be based on agile scheduling problem solving which is modeled by a 6-phase closed-loop XP process shown in Figure 3.1. The 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.
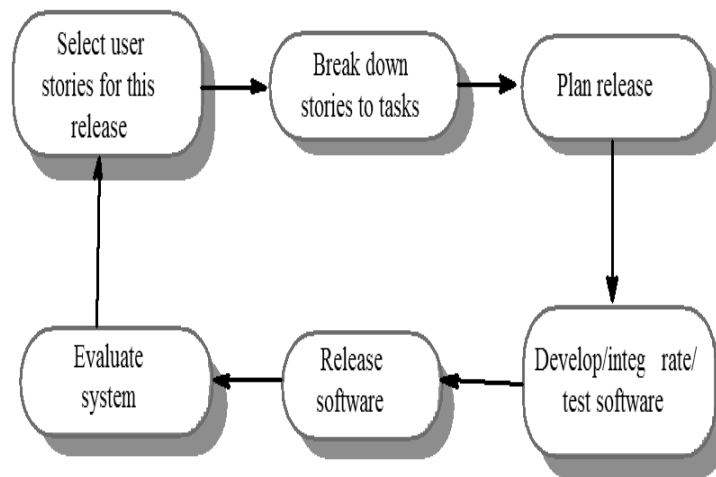

Figure 3.1: XP Process-cycle framework (Ian, 2009)

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.
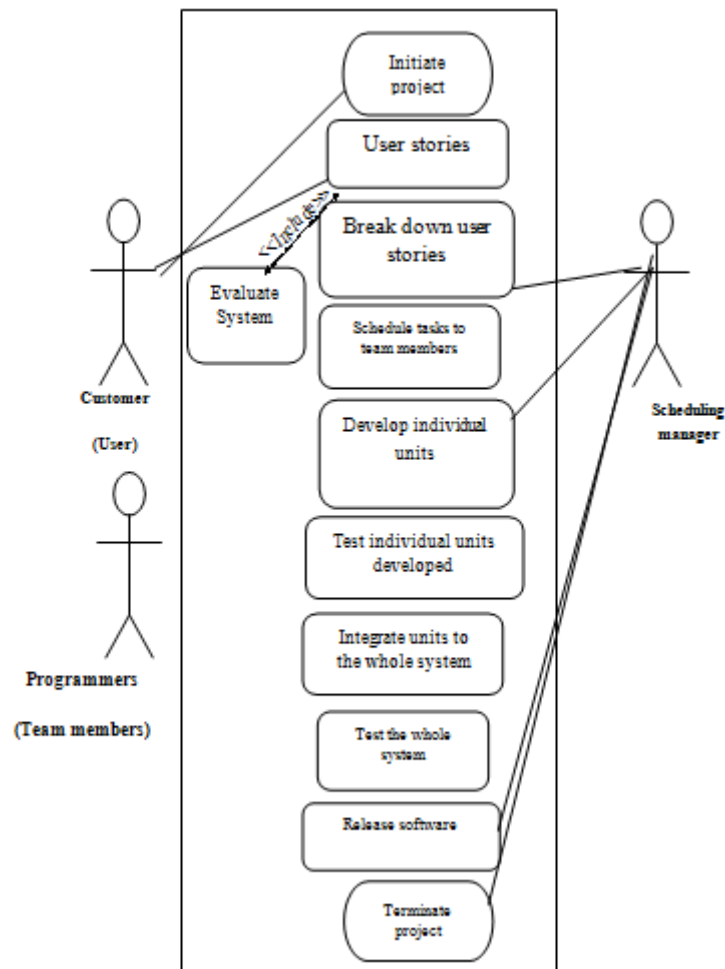
Figure 3.3: Use case diagram for XP processing (Fabio *et al*. 2004)

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.
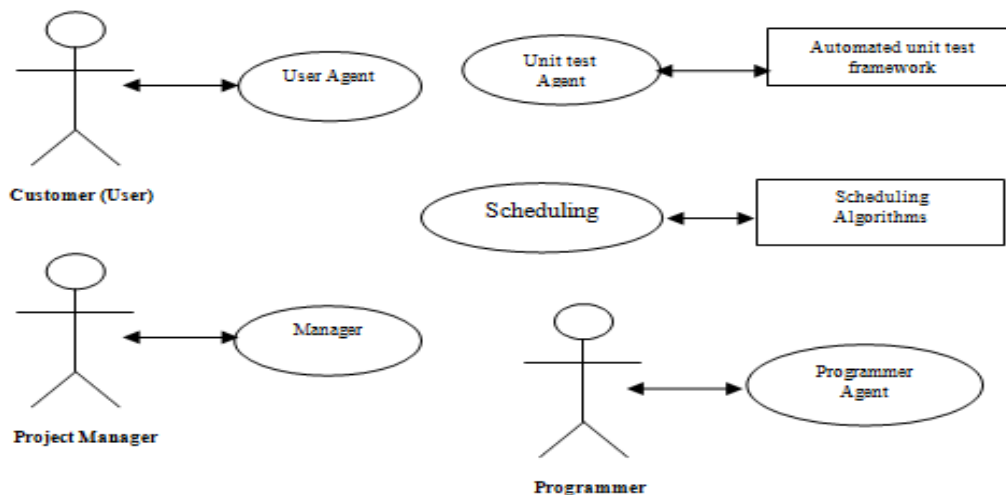

Figure 3.4: Agent Diagram for XP (Fabio *et al*. 2004)

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

Responsibility table for Agents in XP Process

Agent Type	Responsibilities
User agent	1. **Initiates the project** 2. **Provides requirements inform of user stories** 3. **Evaluates the system at each iteration**
Manager Agent	1. **Gets requirements from user agent inform of stories** 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 agent.** 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**

Table 1: Responsibility identification table.

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.
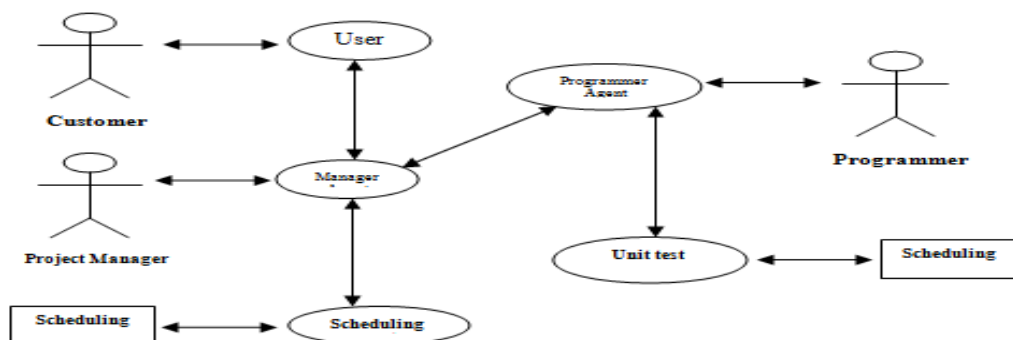

Figure 3.5: Agent diagram for XP processes depicting (Fabio *et al*. 2004) 2004) Acquaintance.

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.

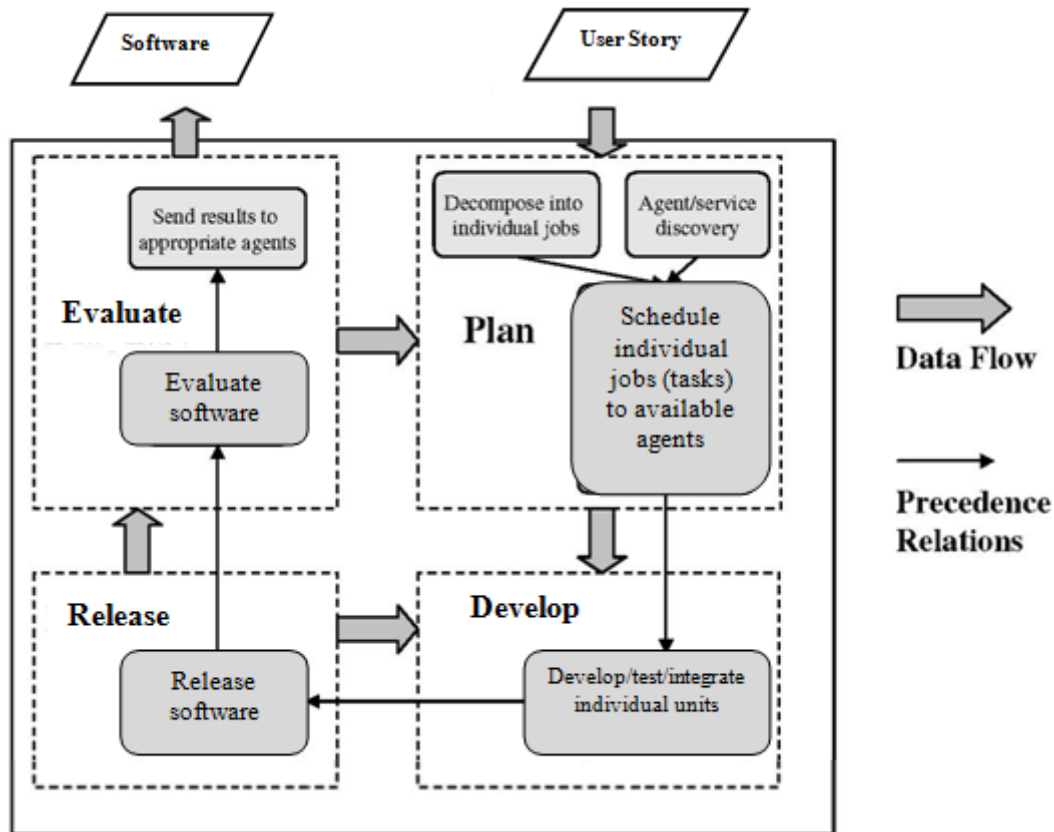vi.) forwarding this solution to the appropriate agents for Evaluation

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

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

III. MATHEMATICAL MODEL FOR AGILE PROCESS SCHEDULING

Problem Variables

Given a number of user stories $j (j \in W : |W| = n)$, with resource demand W_j and iterations k ($k \in j : |j| = o$) with different iteration velocities C_k within a release. Because we use fixed –time scheduling $o \triangleq [FT | \sum l_k^l] \le n$ Where FT is the fixed-time duration of the release and l_k^l is the length of iteration k.

Lets assign each user story into one iteration so that the total required effort in iteration k does not exceed C_k and the number of iterations used as a minimum-while both precedence (matrix) $P_{jj'} \in \{0,1\}$ (where $P_{jj'} = 1$ if j precedes j′ otherwise $P_{jj'} = 0$) and coupling relations (matrix) $C_{jj'} \in \{0,1\}$ (Where $C_{jj'} = 1$ if j is coupled with j', otherwise Cjj′ = 0) hold.

A possible mathematical model can be in the form formulated below.

Maximize $\quad z = \sum_{k=1}^{m} \sum_{j=1}^{n} P_j X_{kj}$ …………………………….. (1)

Subject to:

$\sum_{j=1}^{n} W_j X_{kj} \le C_k Y_k$ ……………………………………….. (2)

$K' - K \ge P_{J'J} : \quad X_{k'j'} = X_{kj} = 1$ …………………......... (3)

$[1/(|K' - K| + 1)] = C_{J'J} : \quad X_{k'j'} = X_{kj} = 1$ ……………… (4)

$\sum_{k=1}^{m} X_{kj} \le \quad \forall j \exists! k$ ………………………………….. (5)

Where: $Y_k = 0$ or 1, and $X_{kj} = 0$ or 1.

$X_{kj} = \{ \quad 0, \text{ if otherwise assigned to iteration k}$

$Y_k = \{ \quad 0, \text{ otherwise iteration k is used}$

C_k is a positive integer

$W_j \le C_k \qquad \text{for } \forall k, j$

IV. 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*.

P- Immediate job precedence matrix

Where:

$P_{j'j} = \{ \quad 1 \quad \text{if user story j directly precedes j}$
$\qquad\qquad 0, \quad \text{otherwise}$

Q- Full user story priority matrix

$Qj'j = \{ \quad 1, \quad \text{if user story j′ comes anytime before j}$
$\qquad\qquad 0, \quad \text{otherwise}$

B- Agent ability matrix

$B_j A = \{ \quad 1, \quad \text{if agent A provides the services needed to complete story j}$
$\qquad\qquad 0, \quad \text{otherwise}$

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

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

Subject to:

$B_j{}'a = 1$	$\forall j \in W, a \in A$
$X_j{}' a = 1$	$\forall j \in W$
$X_j{}' a + X_j{}' a \le 1$	$\forall (j'j) \in W, a \in A$

Where:

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

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

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.

V.   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 agile process schedule as shown in table 4.1 provides an optimal iteration and release schedules for the combination of an already specified constraint. The user does not have the privileged to modify the number or type of participants that can take part in the project. This however does not mean that the schedule provided is not optimal, but provides a maximized priority schedule as shown in table 4.1 below.

	IC	NP	IL	RD	NT	NUS	Max=DV (Z)	Release Values
R_A	20	10	10	7	10	33	400	12,800
R_B	24	10	11	9	12	25	480	11,520
R_C	22	10	12	10	11	27	440	11,440
R_D	23	10	10	8	10	27	460	11,960
R_E	21	10	9	8	12	43	420	17,640
R_F	20	10	8	9	10	26	400	10,000
R_G	24	10	10	11	11	43	480	20,160

Table 4.1 Optimized agile Iteration and Release Schedule values

Keys   Max = DV -   Maximum Deliverable Values
IC – Iteration Count   RD– Release Duration
NP – Number of Programmers   NT – Number of Task
IL – Iteration Length   NUS – Number of User story

For the multi-agent based agile process scheduling, we take the same data sets as that of the agile process scheduling. In order to create more flexibility, we allow the user of system to 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. Here, we propose that the multi-agent based agile process scheduling will create a more optimal schedule thereby having a higher Max Value as dully shown in table 4.2 below.

	IC	NA	IL	RD	NT	SPA	NUS	Max=DV (Z)	Release Values
R_A	20	5	10	7	10	12	33	3,000	2,880.000
R_B	24	7	11	9	12	15	25	3,300	2,376.000
R_C	22	8	12	10	11	13	27	3,600	2,808.000
R_D	23	7	10	8	10	14	27	3,000	2,340.000
R_E	21	8	9	8	12	12	43	2,700	3,402.000
R_F	20	8	8	9	10	13	26	2,400	1,800.000
R_G	24	6	10	11	11	10	43	3,000	3,780.000

Table 4.2 Optimized Multi-agent based agile Iteration and Release Schedule values

Keys

Max = DV - Maximum Deliverable Values
IC – Iteration Count RD – Release Duration
NA - Number of Agents PA - Priorities of service provided by agents
IL – Iteration Length NUS - Number of User story

VI. 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 utillized 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.

VII. 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 research has been to implement and analyze the agile and the multi-agent based process scheduling models for efficiency and flexibility. Additionally, the approach provides a 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 interconnected components which need to be monitored and managed in real time.

REFERENCES

[1]. B. Fabio, C. Giovanni & G. Dominic, Developing Multi-agent system with JADE, John Wiley & sons Ltd, 2004.

[2]. G. Pour, F. Yao & C. Yu, A Mobile Agent-Based Architecture for Mobile Systems Supporting Distributed Software Project Management. IEEE SoftCOM, 2004

[3]. M. Hock, Review of agile methodologies in software development. International journal of research and reviews in applied sciences, volume 1, issue 1, 2009

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

[5]. F. T. Mark, (2003). Optimization techniques for task allocation and scheduling distributed multi-agent operations. A Master thesis submitted to department of electrical and computer science Massachusetts institute.

[6]. N. Jennings, K. Sycara & M. Wooldridge, a Roadmap Agent Research and Development. Int. Journal of Autonomous Agents and Multi-Agent Systems. 16 (1998) 62–87.

[7]. R.Gonzalez & P. Pilar Some findings concerning requirements in agile methodologies. Product-focused software process improvement, 32 (4), (2009) 171-184pp.

[8]. S. Akos, Conceptual scheduling model and optimized release schedule for agile environment, information and software technology.53 (2011)574-591.

[9]. S. Ian, Software Engineering, ninth edition, Pearson Education, Inc., publishing, Addison –Wesley: New York.

[10]. W. Michael & N.Jennings, (1995). Intelligent Agents: Theory and Practice, Knowledge Engineering Review, 10(1995) 115-152.