Task Coordination and Allocation Assistant

Project Proposal

David K.M. Mak

University of British Columbia

Vancouver, Canada

Table of Contents

1.0 Introduction

Over the past few years, agile development methods steadily gain popularity in the world of software development. Proponents of these lightweight methodologies contends that agile methods are better suited to the fast pace of information technology change than traditional plan-based development methods [1]. However, agile development processes may encounter difficulties in task coordination and allocation on large or distributed teams because of agile methods' inherent need for frequent, informal face-to-face communications. The purpose of the proposed project is to develop a task coordination and allocation method (TCAM) and a task coordination and allocation assistant (TCAA) tool for use in a distributed agile team environment.

The rest of this proposal is organized as follows: section 2 presents the current problem in detail; section 3 is a summary of work that is related to the proposed project; section 4 describes the TCAM, the TCAA tool, as well as the experimental methodology to test for the effectiveness of the tool; section 5 explains the benefits of the proposed approach; section 6 outlines the project plan; and section 7 consists of an outline of a master thesis which is based on the project.

2.0 Problem Statement

In a project, organizing tasks and activities is one of the main responsibilities of a project leader or project manager. The project manager needs to determine the tasks that are essential to the achievement of the next milestone (task coordination), and allocate those tasks to team members of different expertise and experience (task allocation). As software development projects are volatile in nature, the project manager needs to revisits those task coordination and allocation decisions frequently in order to account for exceptions, changing requirements and team member feedback. In the task coordination process, the project manager needs to consider milestones (immediate and long-term), state of the artifacts, risks associated with each activity, as well as feedback from team members. Workload, strengths, and area of work of each team members are criteria of consideration in the task allocation process.

2.1 Current Practices in Task Allocation and Resource Management

At present time, the processes of task coordination and task allocation rely on the project manager's intuition and experience. Some agile software development teams adopt the Scrum process [2] as standard practice. In the Scrum process, a team would keep a list of identified tasks, called a backlog. The backlog is constantly updated with new tasks as changes in the project occur. Tasks are prioritized and each team member would commit to a number of tasks. Short daily meetings, known as Scrum meetings, would be held throughout the course of the project. During a Scrum meeting, the project manager would collect feedback from team members. In addition, team members would get to know each other's progress and whether a member's progress is blocked because another task is not yet completed. After the Scrum meeting, the project manager would perform task coordination and allocation, as well as resolving issues that were raised during the Scrum meeting.

2.2 Deficiencies in Current Practices

There are a number of disadvantages in relying solely on the skill and experience of a project manager. The project manager may be inexperienced and require guidance for efficient task coordination and allocation. The project manager may have unintentionally overlooked an issue raised by a particular team member, and the issue is left unresolved. As a result, the team member may feel he or she has been mistreated and become discontented.

In addition, as with other agile approaches, Scrum processes often experience difficulties in distributed development environments or large teams [3]. Current technologies in video conferencing and instant messaging are not very effective in emulating face-to-face communications, which is one of the emphases of agile approaches. In addition, as the size of the project team increases, informal, face-to-face meetings such as Scrum meetings are not as effective. The author hopes that a tool which is based on the proposed method will be able to alleviate these difficulties by allowing a more effective way of providing feedback and reporting progress.

3.0 Summary of Related Work

The context of the proposed project relates to the areas of project management, workflow management, as well as multi-agent task delegation. This section summarizes the prior art in those areas.

3.1 Project Management Tools

In their 2001 paper [4], Lam and Maheshwari proposes a distributed software project management tool (DSPMtool). The work on DSPMtool draws from online services such as iTeamwork and onProject.com, which offers a web-based project management tool. However, DSPMtool is a closed system that can only be accessed within an organization, whereas the online services can be accessed from anywhere through the World Wide Web.

The DSPMtool is based on a task and team model. In the model, a task contains a name, the expected start and finish dates, priority, as well as textual information that is associated with the task. A team consists of a team name, a description, a team leader and a number of members. Tasks are assigned to teams or individual team members. Team members report the progress on their current tasks on the tool. The project manager would be notified if a team member has not update his or her progress, or if the progress of a task is behind schedule.

Lam and Maheshwari contends that DSPMtool can help decreasing information seek time for project team members. Members of the project can retrieve information on a task at anytime because all tasks are viewable by every member of the project. The task description has a field that indicates the person who instantiated the task, hence new team members would know who to ask for information about that task. When tasks are assigned to a team, questions regarding that task are directed to the team. The team leader can either answer the question, or redirect the question to the team member who is directly responsible for the task.

DSPMtool attempts to address the issue of communication overhead in a distributed environment by providing as much information as possible in the task and team models. The task model used by the tool is similar to the Scrum process in the way that a task belongs to an individual or a group of individuals and the tasks are prioritized.

However, the DSPMtool has a number of shortcomings. Firstly, the tool requires explicit actions from the users to update the progress of their current tasks. The users may see the tool to be beneficial only to the project manager for task coordination and performance monitoring purposes; hence they fail to see the justification for the extra work. As a result, users may choose not to make use of the tool and render it obselete. This “opt out behaviour” problem is described in Jonathan Grudin’s 1994 article [5]. In addition, in the DSPMtool task model, tasks are modeled as discrete objects that do not have any explicit relationship to each other. The model does not have the notion of a process, in which tasks and their associated artifacts are interdependent while activities in software development are highly interdependent as changes made in the artifacts of one task would affect another task. The tool also did not address the issue of providing assistance for task coordination and allocation. With the DSPMtool, the project manager is required to aggregate and analyze information relating to tasks and team progress without any help from the tool.

3.2 Workflow Modeling

In a 2002 paper [6], Noll and Billinger propose a process modeling approach for coordinating work flows in a distributed, autonomous environment such as open source development. The paper describes an object oriented process modeling language (PML) and the architecture of a process enactment operating system (PEOS) to execute processes models that are written in PML. Under the model as described in the paper, the process would be broken down into fragments. Each actor in the process would run one fragment independently on his or her own PEOS. Coordination is done by defining triggering events for each fragment. When the trigger event (e.g. source code committed to the configuration management tool) occurred, the execution of the next "down flow" fragment would begin.

Noll and Billinger contend that the described approach has a number of advantages. Since the execution of fragments is automatically triggered by the completion of an "up flow" fragment (e.g. modification of a shared resource), explicit communication between PEOSes is not required for coordination. Hence, it is possible to coordinate actors who are not following an explicit process. However, this approach requires an adaptation of a new process model, instead of taking advantage of process models that may be already in use with the organization. In addition, since each fragment’s “trigger” is rigidly defined, the process cannot be changed easily.

3.3 Workflow Management Tools

Workflow management tools are developed out of the need for a more efficient way of collecting and passing on information as a process is executed [7]. Workflow management systems have been deployed in various fields, including business and administrative processes, as well as software development.

In their 1994 paper, Abbott and Sarin outlined a number of desirable design principles for workflow systems. The ones below are relevant to the context of the proposed project:

- A task would have a number of states, e.g. not started, in progress, finished, etc.
- Provide easy access to the contents of the tasks
- Tasks are allowed to manipulate multiple documents, or conversely, a single document can be shared among multiple tasks
- Support for status monitoring and reporting
- Support for process changes during the course of execution

3.3.1 InConcert

InConcert [8], [9] is an process-oriented workflow management tool which was developed by XSoft. It is developed based on aforementioned principles. Each task is modeled as a mutable object. Before InConcert can be used in a particular application, a number of objects, such as the process, users and artifacts, need to be defined. The InConcert suite provides a number of tools for managing users and various objects in the system. InConcert is extensible and can be customized in order to suit different applications.

Abbott and Sarin contend that InConcert was successfully applied in various fields, including software engineering. They reported that some users find InConcert easy to adapt because of its open and extensible architecture. However, Abbott and Sarin note that there are a number of improvements that can be made to InConcert, as well as other knowledge management systems. In order for the system to effectively handle exceptions, non-procedural interactions, such as work reassignment, request for rework or negotiate for a delay, must be integrated with regular procedural interactions in the system. In addition, a knowledge-based "planning" system should be in place to assist users in making revisions to the process after an exception has occurred. Abbott and Sarin also noted that it is essential for the system to support external activities (e.g. meetings) and synthesize the results of those activities with the data objects of the system.

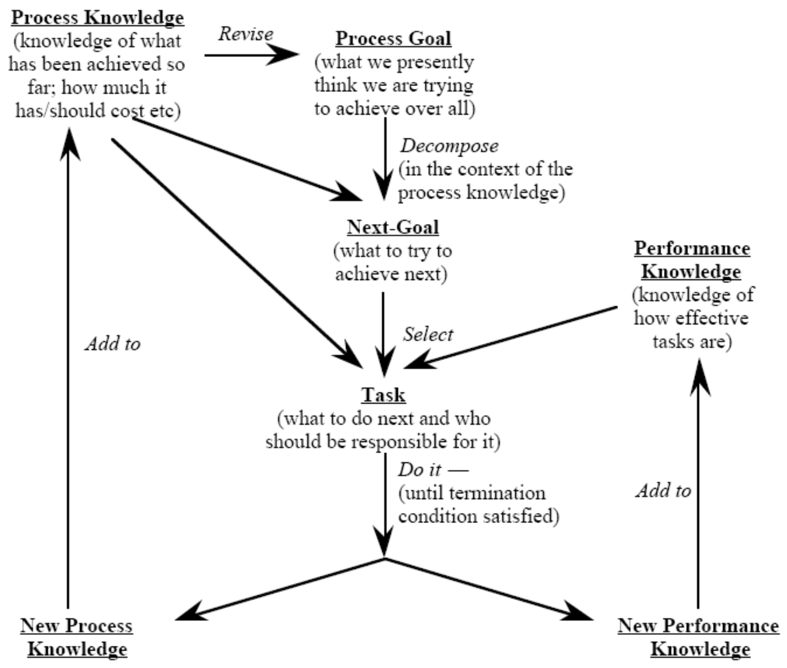
3.3.2 Agent-based workflow systems

A number of researchers in the field of automated agents and workflow systems have proposed the application of software agents in a workflow system. O'Brien and Wiegand [10] suggest that process management should be decentralized with the use of software agents. In their paper, they described a workflow system which an autonomous agent from one sub-team would negotiate with agents from other teams to acquire services. The agent would also manage resources of the team that it belongs to, as well as activating tasks and services to fulfill service requests. Dartflow [11] is a similar agent-based workflow system which features a web-based interface.

Proponents of agent-based workflow systems argue that reactive agents can coupe with changing circumstances (e.g. resource temporarily not available) by modifying resource and task plans. However, autonomous agents may not be able to deal with drastic changes (e.g. adding or removing an activity) in the process because they are programmed to execute a predetermined sequence of tasks. Hence, the autonomous agent paradigm may not fit well with software processes because software processes are often more volatile than administrative or business processes (e.g. applying for a bank loan). Furthermore, most tasks in a software process require actions by human actors, hence decreasing the usefulness of automated agents.

3.4 Knowledge-driven Task Delegation

In a 2002 paper [12], Debenham described the notion of a knowledge-driven process in the context of a multi-agent process. A knowledge-driven process is an iterative process where the process patron may make changes to the process goal according to the process knowledge generated from executed tasks (e.g. what has been done so far). Both process knowledge and performance knowledge are taken into account when task delegation is performed. In turn, the execution of the next round of tasks will generate more process knowledge, forming a feedback loop. The figure below illustrates a simplified view of a knowledge-driven process.


Process Knowledge (knowledge of what has been achieved so far; how much it has/should cost etc)
Revise
Process Goal (what we presently think we are trying to achieve over all)
Decompose (in the context of the process knowledge)
Next-Goal (what to try to achieve next)
Performance Knowledge (knowledge of how effective tasks are)
Add to
Select
Task (what to do next and who should be responsible for it)
Do it — (until termination condition satisfied)
Add to
New Process Knowledge
New Performance Knowledge

Debenham also proposed an automated, negotiation-based task delegation strategy. There are three steps to the strategy [13]. First, when a new task is created, an agent X_0 would put forward a "proposal" to agents $\{X_1,\ldots,X_n\}$, which are deemed suitable to perform the task and those agents would bid to perform the task. A bid consists of the agent's cost estimate, probability of success and constraints. X_0 would first eliminate the bids that violate process constraints, and select an agent from the set $\{X_1,\ldots,X_n\}$. The actual delegation strategy may be affected by factors such as payoff and corporate culture. However, Debenham's model cannot be directly applied to human-intensive software development. Each human actor, unlike software agents, has different skill sets and experiences. These differences must be taken into account in addition to the constraints that are considered in Debenham's model. In addition, the notion of feedback from the task performers is absent from Debenham's model.

4.0 Approach

The proposed project looks to will build on the concepts introduced by the literature in section 3 and devise a method for task coordination and delegation, as well as building a tool to demonstrate the method's effectiveness. This section describes the goals and details of the approach, as well as the experimental method in the project.

4.1 Goals of the Approach

The aim of the proposed study is to solve the problems mentioned in section 2. More specifically, the proposed study has the following goals:

1. The proposed method would assist the project team in determining the most pertinent tasks and the team members who are most suited to perform those tasks.
2. Allows project managers to concentrate on handling exceptions and assisting team members.
3. Improve communications between team members on a distributed team
4. Provide an efficient way for team members to give feedback on their progress

and any obstacles that they encounter.

5. Create awareness among team members by allowing them to see each team member’s current tasks.

Some additional goals for the proposed tool:

1. The tool shall require as little extra work (other than initial data input) on the user’s part as possible.
2. The tool shall be able to be integrated into other project management tools
3. The tool shall have means of preventing abuse by any stakeholder

Conformance to requirements 1 and 3 will be demonstrated with a qualitative experiment to be described below.

4.2 Process Model-based Task Coordination and Allocation Method

There are two sources of input variables in the proposed task coordination and allocation method (TCAM): information within the software process model, as well as attributes and feedbacks of stakeholders in the project. The TCAM would aggregate those information and conflicting goals would be weighted against each other to determine the tasks that are pertinent to the success of the project. The TCAM would then determine the best candidates for performing the tasks based on each team member’s skills, experience, workload and preferences.

The following is a high level dataflow diagram of the proposed method:

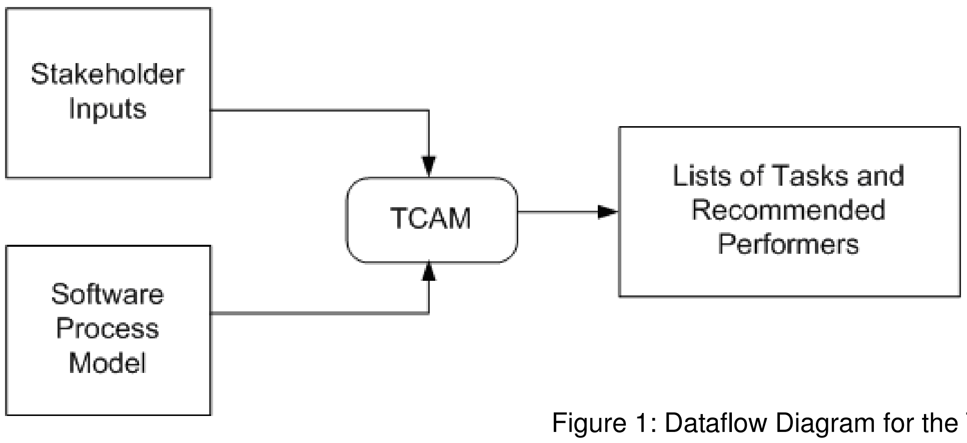


Figure 1: Dataflow Diagram for the TCAM

The following sub sections would first introduce the Software Process Engineering Meta-model (SPEM), describe different input variables to the TCAM, and in turn describe the proposed methodology to process the input information.

4.2.1 Introduction to SPEM

A software process model consists of a set of activities (and artifacts that are associated with each activities), relationships between activities, a set of roles, as well as a number of goals at each milestone. The proposed method would use the process model to extract information on the activities, the artifacts, activity relationships and goals and sub-goals of a process instance.

The TCAM would make use of process models that are based on the Software Process Engineering Meta-model (SPEM) [14] with view of accommodating other families of process models in the future. SPEM is a high-level specification of software models which is developed by members of the industry and adopted by the Object Management Group. Unified Modeling Language (UML) is used as the notations of the meta-model. In simple terms, a SPEM-compliant process model would contain (but not limited to) the following elements:

- ProcessPerformer: defines the performer of a *WorkDefinition*
- *WorkDefinition*: defines a task or a set of activities
- *WorkProduct*: an artifact which is produced, consumed, or modified by a process
- *Precondition*: a condition that must be satisfied before the associated *WorkDefintion* can be executed, expressed as a state of a *WorkProduct*
- *Goal*: the desired state of a *WorkProduct*

Each of these elements has a number of attributes, which contain information about associations between elements (e.g. the WorkDefinition coding is associated with the WorkProduct source code) and other characteristics of an element, such as whether the WorkProduct is a formal deliverable.

4.2.2 Stakeholder Attributes and Feedback

In this proposal, the project manager, project team members, users of the system, as well as members of the general public are broadly referred to as stakeholders. Stakeholder attributes and feedback are essential in making task coordination and allocation decisions because a software project is executed mostly by human actors and the resulting product will likely be used by human users.

4.2.2.1 Team Member Attributes

Members of a software development team may have varied skill sets and experience. Given two team members' workloads are identical, it is preferable to assign the task to the team member whose skills and experience is more closely matched to the requirements of the task.

However, allocating tasks by individual skill set is not sufficient. A team may have one or two members who are much more skilled and experienced in most areas than the rest of the team. Skills-based assignment of tasks would result in most (if not all) tasks being assigned to the minority on the team. Therefore, the workloads of the team members have to be taken into account when tasks are allocated. The importance or risks associated with the task would also affect the task allocation decision. An important and/or a particularly risky task should be assigned to a team member who is more experienced in that type of task. In addition, the task allocation method should also consider the issue of different areas of expertise within the system in development. For instance, a team member who has been working solely on sub-system A should not be assigned to work on sub-system B (while leaving work on sub-system A undone) unless it is necessary because the team member would incur overhead in learning about B before he or she can start working. Therefore, the TCAM should consider what tasks each team member performed in the past in order to determine which team member is an expert in a particular sub-system.

4.2.2.2 Team Member Feedback

One of the goals of the proposed method is to allow effective feedback from team members. At times, team members may find their progress obstructed by another not yet completed task (assigned to another member or otherwise). For example, a change in the requirements specification is needed before the test cases may be updated in order to prevent requirements drift. Although artifact dependencies are reflected in the process model, the fact that a team member is waiting for the task to be completed should add more weight to the priority of the task.

The TCAM should also take a team member's preference into account. For example, some team members may prefer working with source code over working with documentation. When possible, the TCAM should recommend tasks that are of interest to the team member. Factoring team members' preferences would help increase job satisfaction and morale.

4.2.2.3 Users and Other Stakeholder Input

After the first iteration of the project, some executable code may be delivered to the users and clients. Therefore, users and clients may request for new features or bug fixes. Users and clients may prioritize those requests according to their business needs. As users and clients gain information through prototypes and early releases, they may find that their list of priorities may change over time. Although inputs from users and clients may not affect the day-to-day task allocation decisions, they have important effects on the general direction of the project; therefore they should be taken into account.

Other stakeholders refer to people who are not directly involved in the project, such as government regulatory agencies, independent standards body, the media, as well as members of the general public. Even though they may not be users of the system, they take interest in the project. For instance, when a security flaw of an operating system is reported by the media, the project team must attempt to eradicate the problem immediately.

4.2.3 Evaluation of Inputs

Each of the aforementioned input variables represents the goals of each stakeholder of the project. At times, these goals may conflict with each other. Hence, before a task allocation decision can be made, those conflicting goals must be evaluated against each other. The evaluation process can be done using one of the methods described below.

4.2.3.1 Analytic Hierarchy Process

Analytic hierarchy process (AHP) [15] is a widely used methodology in the field of business decision support. It is effective in multi-criteria evaluation where the decision maker need to examine both quantitative (such as time and monetary resources) and qualitative (such as skills and experience of a team member) factors.

The AHP decompose the decision problem into pair-wise comparisons between two courses of action. The corresponding attributes of each course of action can be compared against each other on a relative scale. The results of the pair-wise comparisons are normalized and averaged to produce an overall rating among a number of alternatives. In the context of the proposed task coordination and allocation method, the AHP can be used to evaluate the importance of a task, as well as finding a the most suitable team member to work on a task based on his or her skills, experience and workload.

4.2.3.2 Task Delegation Using Fuzzy Numbers

Shen, Tzeng, and Liu proposed a multi-criteria task assignment model [16] in the area of workflow management systems. The model is based on fuzzy numbers and linguistic variables. Qualitative measures are described on a linguistic scale, and the scale is mathematically represented as fuzzy numbers. Shen, Tzeng and Liu's model consists of five criterions: capability, work load, task similarities, and relationship between team members. The attributes of the team members are compared against each other to produce a relative rating for each criterion. In turn, the ratings are added up to produce an overall ranking.

4.2.3.3 Other Evaluation Methods

Other potentially useful methods for evaluating and processing inputs include using genetic algorithms and machine learning techniques to use past histories as a basis for future allocation decisions, as well as using recommender systems to locate experts in certain areas of the project. These methods will be further investigated during the course of the project in order to evaluate their usefulness in the context of the proposed project.

4.3 Task Coordination and Allocation Assistant

To demonstrate the practicality of the aforementioned method, a task coordination and allocation tool (TCAA tool) will be developed in Java. The TCAA tool would consist of two major components: task recommender and task information viewer.

4.3.1 Task Recommender

The task recommender would be responsible for extracting information from the process model (e.g. parsing an XML representation of the model), processing information, and produce a list of recommended tasks for each team member. The information processing sub-component of the task recommender would be an implementation of the aforementioned task allocation method. On the user interface level, a team member can select any number of recommended tasks and place them into his or her personal work list or backlog. For tasks that required more than one performer, the tool should allow multiple team members to add that task into their work list. The task recommender would collect task preferences information by keeping track of the selections made by each team member. Team members can choose to view each recommended task's rating on various categories. The ratings are displayed as a way to provide reasoning for why the task is recommended to the team member. In addition, a team member can update the progress on his or her current tasks, as well as reporting any “blocks” that he or she encountered.

4.3.2 Task Information Viewer

The purpose of the task information viewer is to facilitate information sharing and generating a sense of awareness among distributed team members. In the task information viewer, all members on the project team would be able to view other members’ current work list, available tasks, as well as dependencies among tasks. By viewing the relationships between tasks, a team member can easily identify any potential blocking situations as well as locating experts in a certain sub-system. The following figure illustrates a mock-up of the task information view screen.

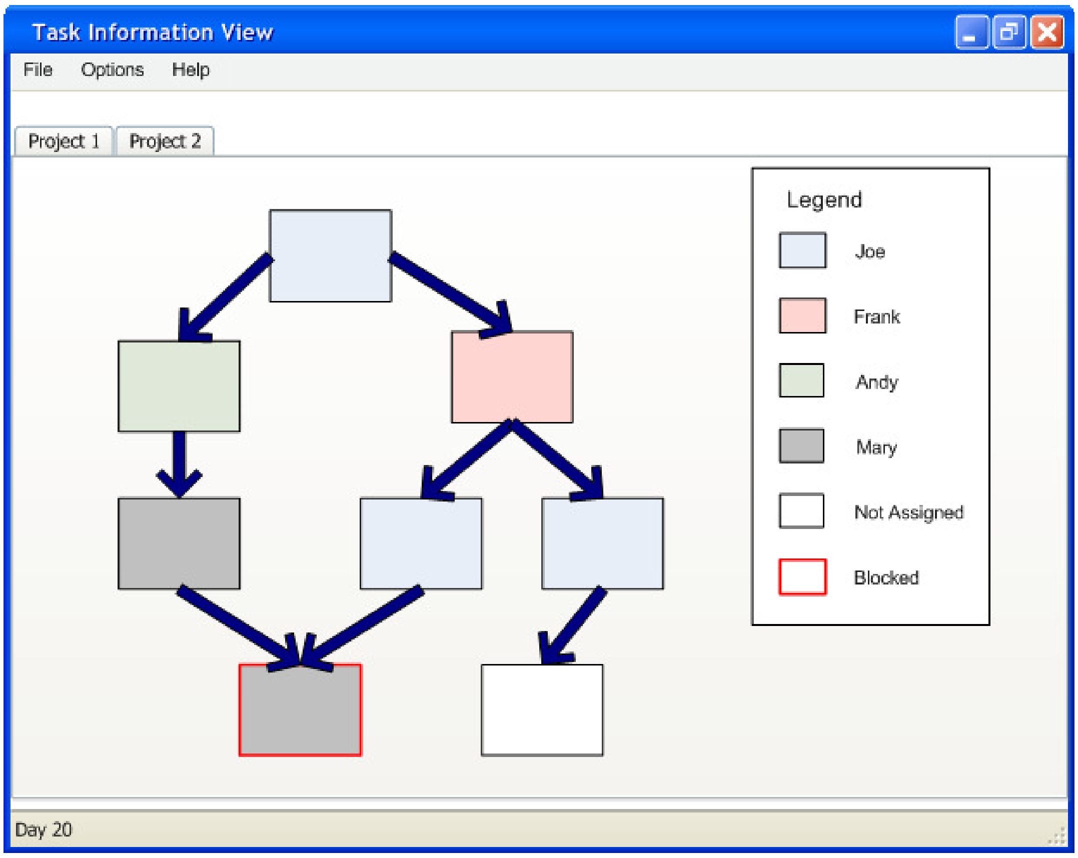


Figure 2: A screen mock-up of the task information view

4.3.3 Integration with Other Software Development Tools

In the initial prototype of the TCAA tool, users may need to input most of the information manually (e.g. having to load an external application and click on certain buttons in a GUI in order to signal the completion of a task). This is not desirable because it would create inconvenience for team members and the aforementioned opt out behaviour would likely to occur. Therefore, the long term goal for the TCAA tool is to be integrated with existing software development tools, such as integrated development environment (IDE), project management tools, project logging tools, or defect tracking tools. The TCAA tool would capture information as users perform actions on those software development tools (e.g. committing changes to code in order to fix a bug), hence reducing the need for users to explicitly enter information for the purpose of using the TCAA tool.

4.4 Experimental Methodology

A two part experiment will be conducted in order to test for the proposed method and tool's effectiveness.

4.4.1 Simulation of the Task Coordination and Allocation Method

An object-oriented simulator will be constructed to test the task coordination and allocation method in a controlled environment. The simulator would consist of an information processing component, as well as a number of objects that emulate the behaviour of various users. The simulator would be able to parse textual scripts as input in order to decrease the number of recompilation required in between simulations. The simulator will be used to calibrate the weighting of each input. In addition, extreme situations, such as all team members always choose the lowest ranked tasks, will be simulated using the simulator in order to check the method's robustness.

4.4.2 Qualitative Testing

Computer-based simulation is not enough to demonstrate the practicality of the TCAM in a real-world environment. Therefore, a qualitative study will be conducted using a prototype of the TCAA tool. A team of student testers will be asked to use the tool in a software development project over a period of three months. Throughout the course of the study, the testers will be asked to give feedback on the following areas:

- Convenience: whether the tool created a lot of extra work
- Effectiveness: whether the tool is effective in reflecting the feedback of the team members
- Improvement: how can the tool can be further improved

The testers' feedback will be analyzed and suggestions for changes will be incorporated into later releases of the TCAA tool.

5.0 Benefits of Approach

5.1 Issues Addressed

The proposed method and tool will help agile software development team in a number of ways. Project leaders can use the tool as an aid to make task day-to-day coordination and allocation decisions and spend more time in other important tasks such as handling exceptions and providing assistance to team members. Project team members would be able to select tasks that they would like to work on, which would improve morale. In addition, through the task information view, team members become aware of what others are doing and organize their work to achieve better coordination among the team.

5.2 Effects on Distributed Teams

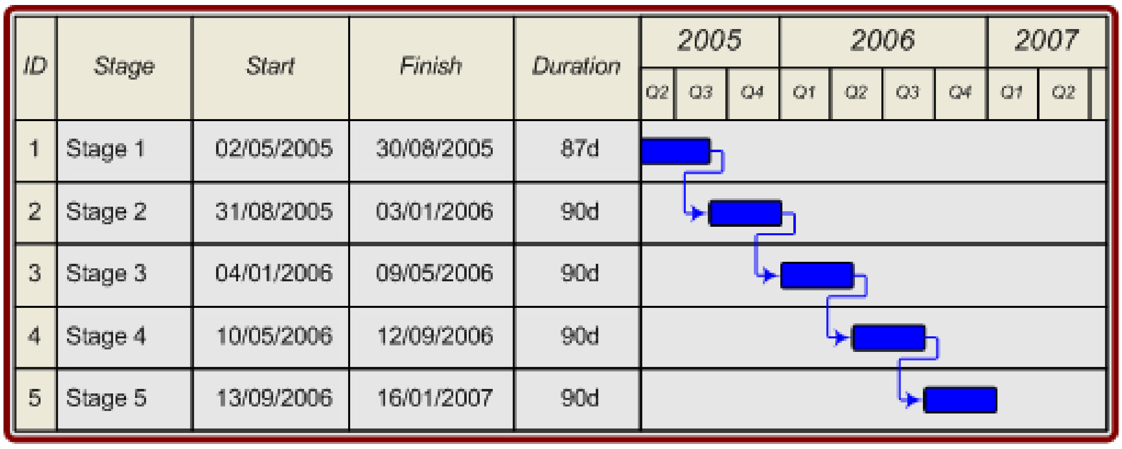
The proposed tool would also be useful for distributed teams, where informal face-to-face communication is not always possible. With the task information view, distributed team members would be able to get updates on other team members' progresses easily. In addition, the TCAA tool allows for effective feedback on their progress and reports any blockages that they encounter. Hence, the tool can be used as an alternative to Scrum meetings, as such meetings are hard to organize when the team is not collocated.

6.0 Project Plan

The proposed method and tool will be developed as an integral part of a Master of Applied Science thesis. The project will be divided into five stages:

- Stage 1: Research in task allocation and related methodology, identifying heuristics as input variables to the method
- Stage 2: Implementing the task coordination and allocation method and simulator; simulation and calibration
- Stage 3: Design and implementation of the task coordination and allocation assistant tool
- Stage 4: Qualitative study on the tool; modification of the tool based on feedback
- Stage 5: Writing and final preparation of the thesis

Stage 3 of the project will require the help of student helpers in implementing the tool's GUI. Stage 4 of the project will require the help of students to serve as testers for the tool. The Gantt chart below illustrates the planned schedule of the project.



ID	Stage	Start	Finish	Duration
1	Stage 1	02/05/2005	30/08/2005	87d
2	Stage 2	31/08/2005	03/01/2006	90d
3	Stage 3	04/01/2006	09/05/2006	90d
4	Stage 4	10/05/2006	12/09/2006	90d
5	Stage 5	13/09/2006	16/01/2007	90d

Figure 3: Gantt chart of the project schedule

7.0 Thesis Structure

The tentative outline for the thesis is as follows:

1. Introduction
2. Context of the study
3. Current State of Practices / Review of Literature
4. Problem Description
5. Hypothesis
6. Validation Method
7. Method and Algorithm Description
8. Benefits of Approach
9. Conclusion

10. References

11. Appendices

References

[1] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, pp. 64, 2002.

[2] L. Rising and N. S. Janoff, "The Scrum software development process for small teams," *Software, IEEE*, vol. 17, pp. 26, 2000.

[3] D. Turk, R. France, and B. Rumpe, "Limitations of Agile Software Processes," presented at Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, 2002.

[4] L. Hai Eric and P. Maheshwari, "Task and team management in the Distributed Software Project Management Tool," 2001.

[5] J. Grudin, "Groupware and social dynamics: eight challenges for developers," *Commun. ACM*, vol. 37, pp. 92-105, 1994.

[6] J. Noll and B. Billinger, "Modeling Coordination as Resource Flow: An Object-Based Approach," in *Proceedings SEA '02*. Cambridge, MA, 2002, 2002.

[7] K. R. Abbott and S. K. Sarin, "Experiences with workflow management: issues for the next generation," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. Chapel Hill, North Carolina, United States: ACM Press, 1994, pp. 113-120.

[8] S. K. Sarin, "Workflow and data management in InConcert," 1996.

[9] S. K. Sarin, "Object-oriented workflow technology in InConcert," 1996.

[10] P. D. O'Brien and M. E. Wiegand, "Agent based process management: applying intelligent agents to workflow," *The Knowledge Engineering Review*, vol. 13, pp. 161-174, 1998.

[11] T. Cai, P. A. Gloor, and S. Nog, "DartFlow: A Workflow Management System on the Web using Transportable Agents," May 1996.

[12] J. Debenham, "Who does what in a multi-agent system for emergent process management," 2002.

[13] J. Debenham, "Delegating responsibility in a multiagent process management system," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. Melbourne, Australia: ACM Press, 2003.

[14] "Software Process Engineering Metamodel Specification," Object Management Group, 2005.

[15] E. H. Forman and M. A. Selly, *Decision by Objectives*: World Scientific, 2001.

[16] M. Shen, G.-H. Tzeng, and D.-R. Liu, "Multi-criteria task assignment in workflow management systems," 2003.