Empirically Modelling Approaches to Project Team Management

0534192

Abstract

This paper examines the potential use of Empirical Modelling (EM) to model the work of project teams, in order to support the study and analysis of team management techniques. An approach to represent a project, a team and associated management aspects is introduced. To illustrate the use of EM tools in this context, the Website Development Model is presented and discussed. Within the model, examples of different management approaches are presented and their influences on the simulated team's performance are compared. The model provides a basis for feedback on EM tools and scope for future extensions. Potential applications, such as in education and research are discussed, as well as limitations that would need to be addressed in future work. In conclusion, EM is found to provide a solid basis for building models which support analysis and understanding of approaches to manage project teams.

1 Introduction

Nowadays, most real-life projects are run by teams of people. 'Teamwork' is an often used keyword in any business activity. A lot of research has been done on team management techniques and numerous books have been published (Lewis, 2004). A deeper understanding of how teams should best be managed seems to be of interest to anyone who wishes to improve their team's performance and, eventually, their project's success.

However, managing a project team is a rather dynamic process and involves complex properties and interactions. To gain a deeper understanding of these techniques, a purely theoretical study may not be sufficient. Ideally, we could complement the learning process with a practical, 'hands-on', experience. However, this may not always be possible in a study environment. It would be beneficial for the learner to be able to test different management techniques on various types of teams and projects in a simulated learning environment and to be able to interact with the scenario.

This paper explores the potential of Empirical Modelling (EM) to simulate a project team and its use to analyse and study management techniques. It presents ways in which the project team, its members and management aspects could be represented. A model of a website development project is presented as an example to illustrate the use of EM. Potential applications of the EM approach, as well as potential obstacles are then discussed. Finally, we conclude that models built using EM have the potential to provide a useful enhancement to learning about project team management.

2 Modelling Project Teams

2.1 Application of EM Principles

In EM, the main princliples of observables, agents and dependencies enable us to create computer-based models to aid our own understanding and *sense-making* of a given system (Beynon, 2006). An important notion in EM is its focus on the experience of the models as we create and interact with them, as opposed to a set of prescribed *apriori* specifications and expected behaviours which are to be implemented. In EM, a model is a dynamic and evolving process, whose features may be defined and refined in real time as the user progresses in their own experience and understanding. This interaction in turn motivates the user to further explore and enhance the model (Beynon, 2004).

A project team can be seen as a multi-agent system with interactions, decision rules and actions. From an EM perspective, we may exploit the notion of agents, observables and dependencies to represent features of the project, team and rules governing operation of the team. The constructed model could then simulate different team management approaches by specifying a set of rules. The user of the model could make adjustments to these rules, simulate

different scenarious and observe the team's performance.

2.2 Project Team Representation

This section presents a high-level description of the features of a project team that may be relevant in a modelling study. This exercise will support the building of a specific model. We can distinguish three main components to form the basis of the model: the project, the team and a set of rules managing the operation of the team with regards to the project. The level of detail contained within each component may vary with the complexity of the model.

2.2.1 The Project

A project can be described as having:

Properties: list of tasks, completion time, deadline, etc.

Type: by application area (e.g. software development, research, construction), by environment (e.g. corporate, small business, university)

Project properties set out the scope and content of the project, as well as measurable success criteria (completion deadline). The types of the project may determine the rules associated with the model, relating to the organisation of the team, task dependencies, decision rules, etc.

Within the task list, a *Task* may consist of: task name, assignee, status (outstanding, in progress, done), duration, due date, priority, preceding task, etc.

A task can be seen as the basic unit of interaction between the project and the team.

2.2.2 The Team and Members

A team consists of a number of team members, each of whom can be described with:

Properties: name, skills, time capacity, busy/idle flag, individual task list

Type: Regular member, team leader

Possible activities: working on a task, idle, management activities, others.

A team member is a unit of resource, which can be assigned to a task and their properties determine how this will be done. They can also act as an active agents with associated decision powers. Team members may have special priviliges, such as creating tasks or management the team.

2.2.3 Management Rules

Rules associated with the project team govern the overall team operation. They specify how project tasks are to be completed, how they are assigned, how team members interact, etc. They make use of one or more observables in the system (e.g., properties of the tasks, team mebers, etc.) to trigger an action.

We can broadly divide the rules in two categories: task-oriented and member-oriented, although it has to be noted that these may overlap sometimes. Task-oriented rules may govern e.g. the task flow and prioritisation. A task-flow rule might be: "If a task has a predecessor task, it cannot be started until the predecessor task has been completed."

Team-member-oriented rules may specify task assignment rules and decision powers of members. In order to assign decision making powers to a team leader, we could create the following rule: "If a member is working on a low-priority task and other members are waiting for a high-priority task to be completed by him, the team leader may re-assign the task to another member." The complexity of such rules can be as high as we allow our model to be and some rules may require additional features of the project or of the team members to be introduced.

By setting up an appropriate set of rules, the model could simulate a variety of team management approaches, ranging from an authoritarian leader-based system, a democratic consesus-based system to an individualist leader free system.

3 The Website Development Model

As an example of the use of EM to simulate a team project, a model for a website development project has been created.

3.1 Model Scenario

The main aims of the modelled project are to develop a website, an underlying database and a reporting tool which will be integrated into the website. The development is broken down into 17 separate tasks. An excerpt from the project task list is shown in Figure 1. There is a team of four developers, each having a specific skill-set (web development, database, server).

#	*Task name*	*Duration*	*Predecessor*	*Skills*
…				
4	Learn about data structure	3 h	-	-
5	Layout design	4 h	-	Web
6	Database design	4 h	4	Database
7	Reporting tool design	7 h	4	Database
…				

Figure 1: Tasks in the Website development project

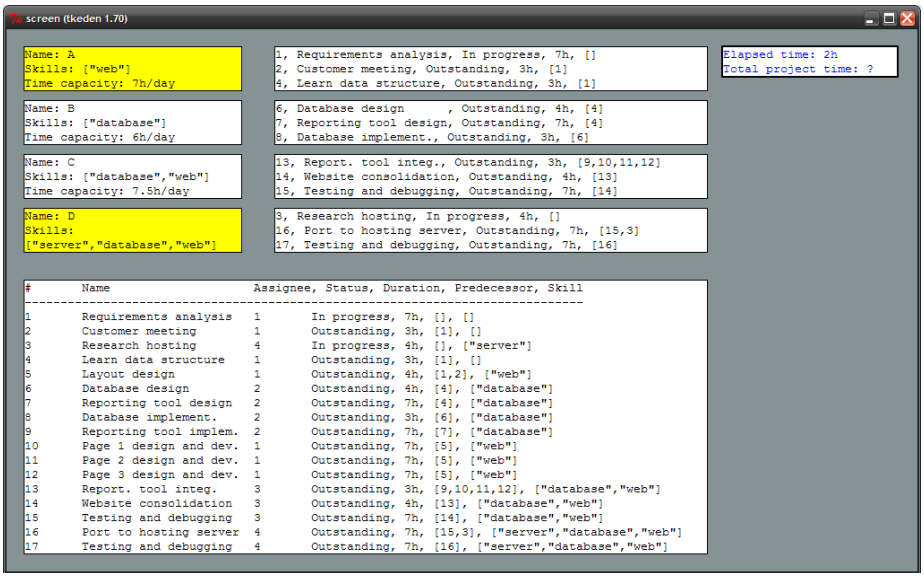

Figure 2: Main screen of the Website Development Model

For the purpose of building a model of this scenario, the following assumptions have been made:

- Each team member in the model may only work on one task at a time. Also, one task may only be assigned to one member.
- Each team member has an associated set of skills, which enable him to work on tasks which require these skills. When this criterion is met, a task can be assigned to the member and will appear in his individual task list.
- A member can be in two possible states: idle or busy. When he is idle, he will start working on the next task in his task list (in a FIFO-order).
- Each task may have an associated predecessor task(s). If so, then the predecessor task(s) need to be completed before work on the dependent task can be started.
- Each task has an associated duration. A member is assumed to be working on a task from the moment he takes up the task for the specified duration, after which the task will be completed.
- The project is considered to be completed when all tasks from the task list are completed.
- The overall completion time is the main metric for the team's performance.

3.2 Modelling in `tkeden`

The model has been created in the EDEN software tool, specifically in the `tkeden` edition. Figure 2 shows the main window of the model. The majority of the model is created using the EDEN definitive notation. The visual part of the model is built using the SCOUT definitive notation. Since the visual part of the model focuses mainly on observing textual information, SCOUT windows have been fully sufficient for the model. More advanced graphics might be introduced in future extensions of the model; however, they are not crucial to communicate the model behaviour to the observer.

As described in Section 2, the model has three main components: the project, the team and set of rules. In the main window, the team perspective consists of windows showing individual members' properties and task lists. Project information consists of a project task list, elapsed time and overall completion time. We will now discuss how functionality of the model was implemented in `tkeden`.

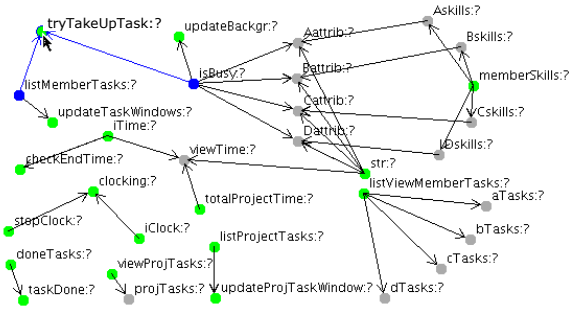

Figure 3: Dependency Graph

3.2.1 Use of Observables and Dependencies

The project task list is stored in the *list* data type, which allows for dynamic access and operations with individual tasks. Similarly, each task is represented as

a *list* of task attributes (e.g., task name, assignee, duration, predecessor, skills required). Some of these attributes, such as 'required skills' may have multiple values, therefore they are also stored as a *list*. This forms a hierarchical *list* structure of the task list that makes operations on any level very convenient.

Each team member has an associated individual task list, which is represented in the same way as the overall project task list. In this way, tasks can be freely passed between the tasks lists without any conversion.

It has been considered to implement the task list using the `eddi` notation for databases within EDEN. However, this would only allow for 2-dimensional data storage, as opposed to 3 dimensions, which are used in the project task list. The members' task lists are themselves stored in a *list*, allowing for a dynamic number of members to be maintained and hence making it a 4 dimensional *list*. Ideally, an object-oriented data representation could be used to improve this issue.

There is a number of dependencies below the surface of the model (see Figure 3 for a dependency graph built using the DMT tool). Dependencies include building string representations of the task lists, updating the SCOUT windows. Simulated time is used to track the duration of tasks in progress and to measure the overall project time. It is implemented by dependencies on the real time and applying a conversion. The simulated time currently starts at '0 hours' and increases the number of elapsed hours by 1 simulated hour per 5 real-world seconds.

3.2.2 Representing Management Rules

The team management rules have been implemented using procedures, which are usually triggered by the simulated time, change in the project task list or other flags. The procedures then make use of functions to handle specific actions.

For example, the procedure to 'try to take up a task' is triggered every time a member's task list or a member's 'idle/busy' flag changes. In simple terms, if a member becomes idle *or* is already idle and a new task appears in his task list, he will take up the first task in his task list and start working on it. That is, another function will be called which, among other things, starts up the task duration counter.

On a higher abstraction level, two project management styles have been implemented, which govern the way tasks are assigned to team members: a leader-centred style and a individualistic (self-governing) style. These styles require one or more functions and procedures to achieve the desired behaviour. Their operation is detailed in the following section.

3.3 Management Approaches and Results

In order to evaluate the model as a tool to simulate project team management techniques, 3 management scenarios have been designed. Two of these scenarios are run automatically by the model, one involved user interaction.

Authoritarian (Leader-centred): This scenario involves the notion of a team leader, who solely decides which tasks each of the team members will be assigned to. In this way, each task in the project can be assigned at the beginning of the project, creating a specific plan that will be followed by all members.

In the current implementation, the leader uses a 'greedy' approach to assign a task to the first team member, who has the required set of skills for that task. However, in a possible extension, the team leader could take other factors into account, such as load balancing.

Individualistic (Self-governing): In this scenario, team members themselves choose tasks they will work on. They do so as they go along in the project, one task at a time. Whenever a team member finishes their task, they will search for the first available task that matches their skill-set.

This scenario would correspond to a small-scale project perhaps in a high school or university area.

Intuitive Assignment by the User: In this scenario, project tasks have been assigned to individual team members manually by intuitive consideration. The criteria for the assignment of tasks were decided by the author of the model according to his personal experience in a website development project.

3.3.1 Team Performance Results

Team performance has been measured as the total time taken to complete the project. The three scenarios have been tested in the Website Development Model and the results[1] are as follows:

Intuitive Assignment	57 hours
Authoritarian (Leader-centred)	66 hours
Individualistic (Self-governing)	54 hours

4 Discussion

From the model point of view, there is a large scope for possible extensions and additions. It implements only a portion of the features described in Section 2.2. In terms of modelling the team members, more

[1] Time in hours corresponds to the simulated time within the model

complex descriptions could be introduced, as well as member interactions, which would more closely model real-life scenarios. There is indeed a tremendous complexity to the human character, behaviour and interaction. The extensions could include the notion of learning (either through experience or training), suggesting new ideas and new tasks and others. From the project point of view, numerous extensions seem obvios: enabling multiple members to work together on a task, introducing more metrics to measure the team performance, such as member utilisation. Tasks could also be extended by adding priorities and enabling team members to re-prioritise when necessary. On the management level, more complex and diverse management approaches could be modelled.

Regarding the use of EM tools, the overall experience has been positive. There are some areas for improvement, though. On the whole, the EM tools proved sufficient and useful in this task, mainly because of the ease to add and change definitions, in real time. In some cases, however, it would be beneficial to be able to use an object-oriented approach when storing data having a hierarchical structure and requiring a level of abstraction. The addition of DOSTE to EM might help to solve this problem (Beynon, 2009). Another peculiarity of EDEN is the occasional lack of a debugging facility, which makes the testing/debugging process challenging at times.

4.1 Potential Applications and Limitations

Applications of EM in education have already been proposed in the past (Beynon, 1997; Boyatt, 2006). Interactive models would be beneficial for studying and analysing how project teams work, how they can be managed and what influences their performance. These models could be used in education, and perhaps in training courses in a business environment. They could also be employed in management and organisational research.

There are, however, some limitations to a potential wider application, which may need to be tackled. Firstly, it may prove difficult for any non-programmer to create their own models or to extensively change the existing ones. There may need to be an additional layer of abstraction to allow features or rules to be added or removed with simple commands. Alternatively, ready-made models may have to be distributed, allowing only pre-set simulations to be run. Secondly, a limitation of these models would be their inherent simplified view of the real-life scenarios. The models would need to show a sufficient degree of sofistication to be accepted by the user.

5 Conclusion

This paper has presented a potential use of EM to model project teams and simulate the management aspects of project team work. A high-level approach to represent the features of a project, a team and management rules for the modelling purposes has been introduced. The Website Development Model created for the purpose of this project using EM tools has been presented and discussed.

Overall, EM seems to provide a solid basis for building complex models, which could reinforce the study, analysis and learning about project team management. However, in order to achieve this goal and to enable a wider adoption of these models, more work would need to be done to create sofisticated and user-friendly models that more closely resemble real-life situations. It has to be noted that this task in itself, in accordance with the philosophy of EM, might be a way of gaining deeper understanding of the peculiarities of team management.

6 References

Beynon, W.M., Russ, S. and McCarty, W. (2006). *Human Computing: Modelling with Meaning.* Literary and Linguistic Computing 21(2), 2006, 141-157.

Beynon, W.M., Roe, C. *Computer support for constructionism in context.* In Proc. of ICALT'04, Joensuu, Finland, August 2004, 216-220.

Beynon, W.M. *Empirical Modelling for Educational Technology*. Proc. Cognitive Technology '97, University of Aizu, Japan, IEEE, 54-68, 1997.

Beynon, W.M. (2009, Oct 19). *DOSTE and EDEN in EM* [Online]. (URL http://www2.warwick.ac.uk/fac/sci/dcs/research/em/teaching/cs405/dosteandem.pdf). (Accessed 24 Jan 2010).

Boyatt, R., Harfield, A., Beynon, W.M. *Learning about and through Empirical Modelling.* In Proc 6th IEEE International Conference on Advanced Learning Technologies (ICALT 2006) Kerkrade, The Netherlands, July 2006, 662-666.

Lewis, J. P. (2004). *Team-Based Project Management.* Beard Books, 2004.

Roe, C., Beynon, W.M. *Empirical Modelling principles to support learning in a cultural context.* In Proceedings of 1st International Conference on Educational Technology in Cultural Context, University of Joensuu, Finland, 2002, pp 151-172.