Towards effective AI-powered agile project management

Hoa Khanh Dam
University of Wollongong, Australia
hoa@uow.edu.au

Truyen Tran
Deakin University, Australia
truyen.tran@deakin.edu.au

John Grundy
Monash University, Australia
john.grundy@monash.edu

Aditya Ghose
University of Wollongong, Australia
aditya@uow.edu.au

Yasutaka Kamei
Kyushu University, Japan
kamei@ait.kyushu-u.ac.jp

***Abstract*—The rise of Artificial intelligence (AI) has the potential to significantly transform the practice of project management. Project management has a large socio-technical element with many uncertainties arising from variability in human aspects e.g., customers' needs, developers' performance and team dynamics. AI can assist project managers and team members by automating repetitive, high-volume tasks to enable project analytics for estimation and risk prediction, providing actionable recommendations, and even making decisions. AI is potentially a game changer for project management in helping to accelerate productivity and increase project success rates. In this paper, we propose a framework where AI technologies can be leveraged to offer support for managing agile projects, which have become increasingly popular in the industry.**

***Index Terms*—Software engineering, artificial intelligence, agile project management**

I. Introduction

Artificial Intelligence (AI) has started making a substantial impact to many parts of our society, and is predicted to disrupt how we produce, manufacture, and deliver. The rise of AI is empowered by the growth and availability of big data, breakthroughs in AI algorithms (e.g. deep learning), and significantly increased computational power. The pervasiveness of software products has resulted in a massive amount of data about software projects which AI techniques can leverage. We envision that AI will transform (software) project management practice in many aspects, from automating basic administration tasks to delivering analytics-driven risk predictions and estimation, facilitating project planning and making actionable recommendations. In this paper, we present a framework of how various AI technologies are adapted and integrated to support various areas of agile project management (*agile PM*).

Agile methods (e.g. Scrum) have been widely used in industry to manage software projects [1]. This relatively new approach to project management empowers software teams to focus on rapid delivery of business value to customers, thus significantly reducing the overall risk of project failures. Project management has thus witnessed a shift away from the traditional "waterfall" process and towards a more adaptive, agile model. The number of projects following agile has increased significantly in the recent years, not only in the software industry but also in other non-IT domains [2].

An agile project are centered around a *product backlog*, which is typically a collection of items to be completed in the project [3]. Items in a product backlog can be, for example, customer requirements for the product (user stories), requests for bug fixes, changes to existing features, and technical improvements. Product backlog is evolved through regular updates and refinement to ensure that it contains items that are relevant the project's scope and objectives, sufficiently detailed, and appropriately estimated. Important updates to the product backlog include adding or removing backlog items based on current needs, estimating the size of items, and refine large items into small fine-grained items.

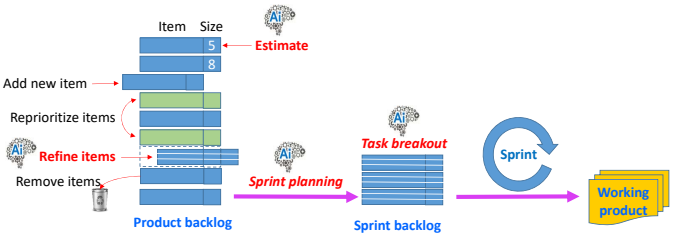

Fig. 1. A typical agile process

An agile project consists of multiple iterations (or alternatively referred as *sprints*). Each sprint is often a short period in which the team aims to complete a subset of items in the product backlog. Prior to a sprint, the team performs *sprint planning* to identify the goal of that upcoming sprint, and select items from the product backlog which they will complete to meet the sprint's goal. During sprint planning, many agile teams decompose each product backlog items into a set of *tasks*. These tasks and their corresponding product backlog items form the *sprint backlog*. The team then executes the sprint to complete items in the sprint backlog to deliver a potentially shippable product increment.

II. Challenges in Agile Project Management

Many tools have been developed to support agile project management such as Atlassian's JIRA Software[1], Axosoft[2], and Assembla[3]. Those tools allow agile teams to create and manage various agile artifacts such as user stories, product backlogs, sprints, and sprint backlogs. For example, they support teams in creating user stories and tasks, linking related tasks and user stories, assigning team members to tasks and issues, creating deadlines, setting priorities and estimates (e.g. story points). They also enable team members to see the amount of work required for individuals and teams during each sprint, track progress of the sprint and its associated user stories and tasks. They facilitate real-time information exchange and collaboration via centralised project information.

Although existing agile tools are useful, their support is limited to creating, managing, and tracking project artifacts, and visualising historical project data such as burndown charts and other agile reports. Current agile project management tools lack advanced analytical methods that are capable of harvesting valuable insights from project data for prediction, estimation, planning and action recommendation. Many decision-making tasks in agile projects are still performed by agile teams without machinery support. We identify a number of important areas in agile project management that remain challenging due to this lack of effective support.

*1) **Identifying backlog items**:* Items in the product backlog can be derived from different sources such as a requirement specification, new feature requests from customers, bugs reported by end users, previous bug fixes, discussions among agile teams (e.g. technical debts, design changes or action items from retrospective meetings), end users' reviews of the product, and even experiences from previous projects. It is difficult and time consuming for agile teams, especially product owners, to process this large amount of heterogenous data in order to identify and create new items for the product backlog. In addition, for each newly created backlog item, it is necessary to consider *inter-dependencies* between the new item and existing ones. This is challenging as a typical project has a large product backlog with more than 100 items.

*2) **Refining backlog items**:* Some items (e.g. user stories) in the product backlog are initially large, thus do not fit within a single sprint. Agile teams are often required to refine these large items into small ones such that they not only facilitate implementation but are sufficiently large, allowing stakeholders to understand business value [3]. There are typically three levels of refinement: (1) decomposing an epic into a number of user stories; (2) splitting user stories into small stories; and (3) breaking a user story into a number of specific project tasks. Different rules and guidelines have been proposed to help teams refine backlog items, but rules often overlap with or even conflict with one another. Teams struggle to refine backlog items and rely on their own intuition and experience.

[1]https://www.atlassian.com/software/jira

[2]https://www.axosoft.com

[3]https://www.assembla.com

*3) **Sprint planning**:* The key part of sprint planning is selecting a subset of items in the backlog which can realistically be accomplished by the team in the upcoming sprint to deliver a product increment. The customer expects the team to deliver what have been planned for a sprint, thus meeting this expectation is important in maintaining the customer's faith in the team's ability to deliver. Sprint planning is however highly challenging since many important factors must be considered, including items contributing toward the sprint goal, their priority and business value to customers, the dependencies among items, appropriate allocations to bug fixing and other technical work (e.g. resolving technical debts) and the availability of team members and the team's capacity. Risks impeding a sprint execution should also be forecasted and factored into a sprint plan. Sprint planning thus requires not only in-depth understanding of the current project and team but also experience learned from previous projects. Tool support is needed to manage complexities for large projects.

*4) **Pro-actively monitoring sprint progress and managing risks**:* As the sprint unfolds, the team needs to track sprint progress and manage risks. Current practices in risk management mostly rely on high-level guidance and subjective judgements. Predicting future risks is highly challenging due to the inherent uncertainty, temporal dependencies, and especially the dynamic nature of software. There is currently a gap in providing agile teams with insightful and actionable information about the current existence of risks in a sprint, and recommending concrete measures to deal with those risks.

III. An AI–Powered Agile Project Assistant

The above challenges and the serious lack of effective tools presents an opportunity for AI to significantly improve the practice of agile project management. AI-based tools are able to process massive amounts of data generated from software projects, harvest useful insights, and train to perform complex tasks such as estimating effort, task refinement, resource management, and sprint planning. Figure 2 shows our proposal for the architecture of an AI-powered agile project management assistant. The core of this AI system are an ***analytics engine***, a ***planning engine*** and an ***optimization engine***. These machineries depends on the ***learning representation engine*** to learn and generate representations of project data that are mathematically and computationally convenient to process. The ***conversational dialog engine*** converses with users and brings the support provided by the other engines to the users.

A. Representation learning engine

Agile project artifacts contain both structured and unstructured data. For example, backlog items may have structured attributes such as type and priority (which are easily extracted to form a vector representation), whereas product visions, sprint goals, description of backlog items, and communication among team members (e.g. comments on backlog items) are written in natural text. Codebases contain documentations such as release notes and comments written in natural text, and source code written in programming languages. Hence, the

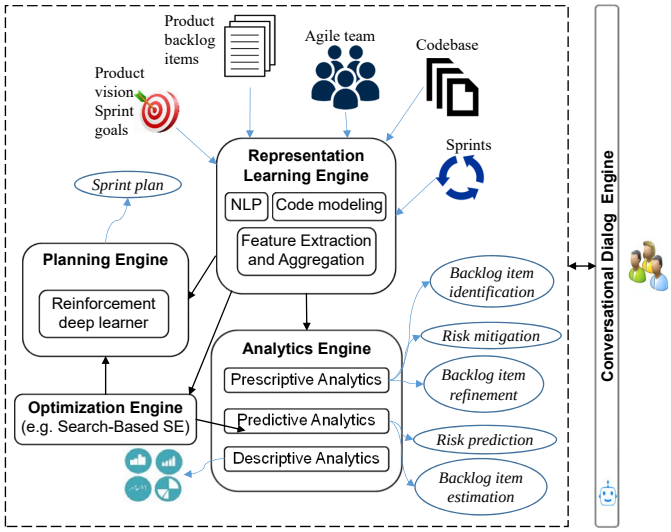

Fig. 2. The architecture of an AI-powered agile project management assistant

representation learning engine is an important component of this AI system, responsible for learning meaningful vector representations for each project artifact. These representations can automatically be learned from unlabelled data, and are then used by the other machineries in the AI system.

The representation learning engine has a `NLP` component which performs automatic analysis on project textual artifacts and then generates good representations of those artifacts. Traditional NLP techniques (e.g. Bag of Words) produce very high dimensional and sparse vector representations. By contrast, latest advances in deep learning-based NLP techniques [4] such as word2vec, paragraph2vec, Long Short-Term Memory (used in Google Translate), or Convolutional Neural Networks (used in Facebook's DeepText engine) are able to generate dense vector representations that produce superior results on various NLP tasks. Source code is another important source of project data. The `Code Modeling` component is responsible for learning meaningful vector representations which reflect the semantic and syntactic structure of source code. State-of-the-art statistical language modeling techniques, including deep learning models, have demonstrated their effectiveness for source code and thus can be leveraged here [5, 6].

We have leveraged the powerful deep learning architecture, Long Short-Term Memory (LSTM) to automatically learn vector representations for both backlog items and source code[4]. LSTM enables us to learn the semantics and syntactic structures, particularly the long-term dependencies, existing in both natural text and source code. We will extend these models to learn representations for other textual artifacts such as product visions, sprint goals, and developer communications.

Any useful AI machinery must take into account the capability and dynamics of agile teams. Obtaining a representation for a team requires modeling of its members (e.g. developers). A developer can be represented through the project artifacts they have involved with, such as the backlog items they have completed or the code they have written. The `Feature Extraction and Aggregation` extracts all the vector representations of the artifacts related to a developer, and learn to aggregate them to form a vector representation of the developer. A number of feature aggregation techniques proposed in recent work [7] which derive vector features of a sprints based on the features of the backlog items assigned to it. We will extend those aggregation methods to learn features for representing team members. This representation will be enriched with features representing work and social dependencies between team members, extracted from communication logs (e.g. comments or discussions on work items).

B. Analytics engine

The analytics engine aims to provide decision support in the following aspects:

1) ***Descriptive analytics****:* Most existing agile project management tools support this basic level of analytics: data visualization via reports, dashboards, and scorecards. Common agile reports such as burndown charts, velocity charts, and sprint reports are created by summarizing what happened using historical project data and presented to the users in an intuitive and easily interpretable manner. Knowing what happened (e.g. team velocity from sprint to sprint) is useful, but *diagnosing* why something happened (e.g. why the team's velocity dropped significantly in some sprints) is even more useful. AI equipped with machine learning can augment descriptive analytics by discovering patterns, identifying anomalies and detecting "unusual" events.

2) ***Predictive analytics****:* Most existing agile tools are not yet capable of providing this advanced level of analytics. Two challenging areas are effort estimation and risk prediction that are specifically for agile contexts. Machine learning techniques are suited to build prediction models. For example, recent work [8] used deep learning to estimate the size of user stories through learning a team's previous estimates. Estimation tools could be used as a decision support system and takes part in the existing estimation process (e.g. planning poker) or in a completely automated manner. Forecasting future risks requires the capability of processing large amounts of historical project data, memorizing a long history of past experience, and inferring the current "health" state of the project. Recent work has moved forwards in this direction to predict delay risks [9] or sprint delivery risks [7].

3) ***Prescriptive analytics****:* This is the most advanced level in the project analytics stack. Using the results from descriptive analytics and predictive analytics, prescriptive analytics recommends the best course of actions for agile teams in a specific situation. We identify here three important areas in agile PM that prescriptive analytics would be useful:

- *Backlog item identification:* Using the NLP component in the representation learning engine, prescriptive analytics will automatically process and extract new backlog items from different data sources such as a requirement specification, new feature requests from customers, bugs

[4]References omitted due to double-blind review requirement.

reported by end users, previous bug fixes, discussions among agile teams (e.g. technical debts, design changes or action items from retrospective meetings), end users' reviews of the product, and even experiences from previous projects. It will also able to recommend inter-dependencies between new item and existing ones using machine learning and representation learning.

- *Backlog item refinement:* prescriptive analytics will suggest how a user story is split into a smaller user stories or how a user story is decomposed into tasks. Learning decompositions is highly challenging since it requires a background knowledge. It is still a new topic in AI and machine learning.
- *Risk mitigation:* Using results from predictive analytics, prescriptive analytics recommends a course of actions to take advantage of a future opportunity or mitigate a future risk and shows the implication of each decision option.

C. Reasoning capability

Reasoning is the capacity to infer new knowledge by algebraically manipulating existing knowledge base to respond to a query [10]. Traditionally, it works on symbolic knowledge representation through the means of induction or deduction. This permits domain knowledge provided by agile teams (e.g. project rules). Recently, deep neural reasoning offers an alternative for producing answers from sub-symbolic (vector) representation [11], which are output of the representation engine. Inferred knowledge can be put back to enrich the knowledge base. The reasoning capability of our AI system is provided by two engines: planning and optimization.

1) Planning engine: Planning for a sprint can be formulated as an AI planning problem in which the initial state is the state of the project and the product prior a sprint, a goal state is specified in the sprint's goal, and selecting items from the product backlog can be viewed as the act of choosing plan operators to be executed from the initial state to a goal state. The planning engine needs to consider a range of input such as the existing product backlog items, the sprint's goal, the existing codebase, the team's capacity and previous performance in previous sprints, and the duration of the sprint. These data are often not formally expressed. The representation learning engine has convert them into vector representations but further formal encoding would be needed.

In addition, the plan needs to be executed in a manner that is *robust* and *resilient* to changes. The challenge is not only to be flexible enough to deal with immediate impediments to the sprint execution, but to also anticipate future states of affairs that might impede sprint execution or the achievement of the sprint goal. Impediments to the successful sprint execution can appear in many forms. For instance, a task might not be completed by the due date, preventing other dependent tasks from being started. Hence, the relationship between a sprint plan and its operating environment can been seen as adversarial. Recent successful work in deep reinforcement learning (e.g. [12]) can be thus leveraged to build this part of the planning engine.

2) Optimization engine: The optimization engine helps the planning engine to compute the optimal set of actions given a certain situation. For example, it can be used to compute the optimal selection of backlog items for the upcoming sprint given multiple constraints and objectives. It can also be used for hyper-parameters tuning of machine learning models used in the analytics engine. Search-based software engineering techniques can be leveraged here to build the optimization engine.

D. Conversational dialog engine

The conversation dialog engine is envisioned to converse meaningfully with agile teams. It is a form of a software chatbot [13], acting as an interface between the users and the remaining part of the AI system. The chatbot can be asked different types of questions, such as "Show me your estimate of this user story" or "Can you help split this user story?". Through conversations with the users, it receives input and requests, and passes them to relevant engines in the system. Future chatbots can be trained end-to-end [14] and person-specific instead of task-specific [15].

IV. Next Steps

We are developing prototype tools to realize each component of the proposed AI-powered agile project management assistant. We plan to first evaluate it using our existing dataset of 150 open source projects. We will also collaborate with our existing industry partners to perform an evaluation on commercial software agile projects. We however believe that AI will assist, not substitute, human teams. Individuals, interactions, and collaboration are still the key elements of project success. AI can serve as a distinctive accelerator for agile teams and thus help increase project success rates.

References

[1] R. Hoda, N. Salleh, and J. Grundy, "The rise and evolution of agile software development," *IEEE Software*, 2018.

[2] CollabNet and VersionOne, "The 12^{th} annual state of agile report," Tech. Rep., 2018, https://explore.versionone.com/state-of-agile.

[3] M. Cohn, *Agile estimating and planning*. Pearson Education, 2005.

[4] C. D. Manning, "Computational linguistics and deep learning," *Computational Linguistics*, 2016.

[5] M. Allamanis, E. T. Barr, P. Devanbu, and C. Sutton, "A survey of machine learning for big code and naturalness," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, p. 81, 2018.

[6] M. Tufano, C. Watson, G. Bavota, M. Di Penta, M. White, and D. Poshyvanyk, "Deep learning similarities from different representations of source code," in *Proceedings of the 15th International Conference on Mining Software Repositories*, ser. MSR '18. New York, NY, USA: ACM, 2018, pp. 542–553.

[7] M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose, and J. Grundy, "Predicting delivery capability in iterative software development," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 551–573, June 2018.

[8] M. Choetkiertikul, H. K. Dam, T. Tran, T. T. M. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[9] M. Choetkiertikul, H. K. Dam, T. Tran, and A. Ghose, "Predicting the delay of issues with due dates in software projects," *Empirical Softw. Engg.*, vol. 22, no. 3, pp. 1223–1263, Jun. 2017.

[10] L. Bottou, "From machine learning to machine reasoning," *Machine Learning*, vol. 94, no. 2, pp. 133–149, 2014.

[11] H. Jaeger, "Artificial intelligence: Deep neural reasoning," *Nature*, vol. 538, no. 7626, p. 467, 2016.
[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[13] C. Lebeuf, M. Storey, and A. Zagalsky, "Software bots," *IEEE Software*, vol. 35, no. 1, pp. 18–23, January/February 2018.
[14] I. V. Serban, A. Sordoni, Y. Bengio, A. C. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models." in *AAAI*, vol. 16, 2016, pp. 3776–3784.
[15] S. Kottur, X. Wang, and V. Carvalho, "Exploring personalized neural conversational models." in *IJCAI*, 2017, pp. 3728–3734.