2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)

Towards Unified Software Project Monitoring for Organizations using Hybrid Processes and Tools

Eray Tüzün
Department of Computer Engineering
Bilkent University
Ankara, Turkey
eraytuzun@cs.bilkent.edu.tr

Çağdaş Üsfekes
ICT Department
HAVELSAN
Ankara, Turkey
cusfekes@havelsan.com.tr

Yagup Macit
ICT Department
HAVELSAN
Ankara, Turkey
ymacit@havelsan.com.tr

Görkem Giray
Independent Researcher
İzmir, Turkey
gorkemgiray@gmail.com

***Abstract*— Large-scale software development organizations generally carry out multiple software development projects simultaneously. Teams use various software development processes and tools to implement these projects. In this context, the main challenges of the practitioners are (1) keeping track of the status of a single project where hybrid set of tools exist for different software life cycle activities (2) effectively monitoring a consolidated status of multiple projects that use hybrid processes and tools. To address these challenges, it is vital to have a unified view of these projects independent from these hybrid processes and tools. To this end, we report on our preliminary experiences on the development of a unified project monitoring solution and a corresponding tool support based on the Essence framework's language and kernel. Our solution provides an up-to-date and unified view of projects by collecting data from various tools automatically as well as allowing manual data entry.**

Keywords—hybrid processes, hybrid tools, software process, project monitoring, project tracking, essence framework.

I. INTRODUCTION

For software development projects using a diverse set of application life cycle management (ALM) toolsets for different activities (such as source control, defect management, backlog management, build management etc.), it is crucial to collect data on the status of these various activities to effectively monitor a software development project. Generally, this data reside in different tools used for these various activities or manually provided by stakeholders. At an organization level, where multiple software development projects are considered, this picture becomes even more complex since more diverse methods, activities, tools, and teams are involved. In a typical large-scale company, the above scenario of having hybrid processes [1] and hybrid tools [2] is very common [3]. Therefore, a common ground is needed to consolidate and unify data coming from various tools and teams.

The Essence Framework [4] provides a process modeling language and an extensible framework that aims at giving a common framework for various processes, methods and practices of software development in general. This consolidation approach promises software development organizations a common ground for project/product development and monitoring. However, there is no direct operable automation infrastructure for this common ground. Our main goal in this study is to investigate a unified progress monitoring solution for software development organizations In this study, we report on our preliminary experiences on the development of a unified project monitoring solution based on the Essence framework's domain language.

In Section II, we provide a background related to project & portfolio management, software development methods, and Essence framework. In Section III, we provide three scenarios for problem motivation. In Section IV, we describe our approach to address the problems in the scenarios described in Section III. In Section V, we provide the initial results of our approach as a proof-of-concept tool. Finally, Section VI concludes the paper.

II. BACKGROUND

PMBOK Guide [5] defines "project management" as the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements. As the size of organizations increase, the number of simultaneous projects mostly increase. The collection of these projects makes up the project portfolio of that organization. Portfolio management refers to the centralized management of the projects of an organization to ensure the achievement of strategic objectives of that organization [5].

Medium and large-sized organizations employ many teams to carry out projects. These teams have a diverse set of set of skills and experiences. They use different software development methods and tools that fit to their culture as well as the requirements of their projects. One way of classifying methods is using a spectrum from plan-driven to agile. Agile manifesto [6] proposes some values and principles addressing both technical and organizational aspects of software development. The notion of "agile" constitutes an umbrella concept for some practices which propose activities, artifacts, roles, frameworks realizing these values and principles. This umbrella covers Scrum, eXtreme Programming (XP), Crystal, Lean Software, and Feature Driven Development (FDD) to name a few. The motivations behind agile methods are generally based on some shortcomings of older models of software development such as Waterfall, Spiral, and Unified Process. These older models are classified as plan-driven (or plan-and-document, plan based) methods. Although it is beneficial to classify the methods according to some criteria and try to understand these comparatively, we have to say that this classification is not precise and sharp. For instance, Unified Process emphasizes iterative and incremental development which is suggested by agile methods. We leave this discussion aside and focus on a common ground, on which we can unify these methods.

The Essence framework [4][7] provides a common language and a domain model of software development, which form a basis for modeling development methods. With the use of the Essence framework, it is possible to model a wide set of software development methods [8], including plan-driven methods such as Unified Process [4], and agile methods such as Scrum [9]. The Essence framework also proposes a way to monitor the overall state of a project and select activities for progress ending up with a concrete guidance for software development teams.

978-1-7281-3393-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSSP.2019.00023

III. PROBLEM MOTIVATION

A. Case Description

We first describe our motivation in two different yet related scenarios. Scenario 3 is the combination of the first two scenarios. Based on these scenarios, we formulate two research questions.

1) One Project – Hybrid Tools:

In a typical software development project, many different tools are potentially used in different life cycle activities. Typical life cycle activities in a project can be requirements management, source code control, design, defect management, test management, project management, build management etc. We witness a "Cambrian explosion" of DevOps and ALM tools in the recent years as described by Kirsten [2]. This heterogenous set of tools brings the problem of integrating and coordinating these toolsets to keep track of project status. On top of the commercial tools per each life cycle activity, there might be some custom tools that are developed in-house as well. Especially for development teams that choose their tools per life cycle activity based on best-of-breed approach, the development environment will become a very heterogeneous environment including a variety of different toolsets. These life cycle activities and their corresponding tools should be integrated and synchronized to work properly. In the presence of hybrid tools for different life cycle activities, it might become very difficult to track the progress of a single project.

2) One Company – Hybrid Processes:

In a typical software development company, many projects coexist. These projects may vary in size, their adopted tools per each life cycle activity, and adopted development methods and practices (plan-driven or agile). Even a single project might use a hybrid method [1] such as water-scrum-fall. This further complicates monitoring at project level. From a higher management point of view, there is a need for keeping track of the progress of all projects in a unified manner.

3) One Company – Hybrid Processes and Hybrid Tools:

The third scenario is the combination of Scenario 1 and Scenario 2. In this scenario, a company has multiple number of projects with each project applying different type of software development methods. Moreover, each one of these projects might potentially use a different toolset per each life cycle activity.

B. Research Questions

Aligning with the three scenarios described above, our research goal is to investigate a unified progress monitoring approach for software development organizations. In line with our research goal, we have defined the following research questions for our study:

RQ1: How can we achieve progress monitoring for a project that uses variety of different tools?

RQ2: How can we achieve unified portfolio monitoring in a company, where projects are applying different software development methods?

IV. OUR APPROACH

In the following subsections, we first give a brief overview of how Essence framework is potentially suitable for addressing the hybrid process and tools scenario that we described in the previous section. Although the Essence Framework could potentially address majority of these problems in theory, the solution would be totally manual and will not be practical to use in an industry setting. In the next subsection, we have described how we can utilize Essence framework to come up with a conceptual solution. In the final subsection, we explained the details related to the practical implementation of the conceptual solution.

A. Using Essence Framework to address Hybrid Processes & Tools

The use of different process models and practices to develop software results in extra efforts due to the different steps, outputs and control points in the development and management activities. The Essence framework provides a common ground modeling software development methods. The Essence Kernel, as seen in Fig. 1, models seven vital dimensions of every software development project. These dimensions are named as "Alpha" in Essence language and are organized under three areas of concern, namely, customer, solution, and endeavor.

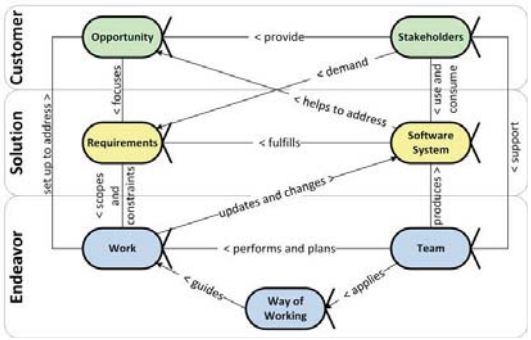

Fig. 1. Essence Kernel

According to Essence framework, the progress of a software development project can be tracked by following the state changes in these Alphas. A team completes some activities to change the state of an Alpha. Each state has a checklist item list and the team works to complete these items. This common base is suitable for monitoring projects in which various methods are used. Moreover, it is possible to collect data from a diverse set of tools to check the status of a checklist item. Essence framework uses cards to show the states of Alphas and the checklist items of a state.

B. Conceptual Solution

In real life projects, the use of Alpha State Cards, which is the recommendation of Essence framework for monitoring a project, has operational challenges. The manual use and maintaining the state values of these alpha state cards brings an unnecessary overhead for practitioners. In order to overcome this challenge, automatically collecting the data for project monitoring from the tools will help the team in focusing their core activities and enable the formation of a project monitoring dashboard that reflects an up-to-date status of a project or a project portfolio.

The first column of Fig. 2 shows that Essence language defines a list of connected states for an Alpha and a list of checkpoints for each state. The second column illustrates that Essence framework visualizes the Alpha states and checklist items with physical cards. The last columns shows how a checklist item can be associated with a tool. Such an integration enables automatic status check for a checklist item. For some of the items, there might not be a corresponding tool, hence a manual data entrance on the status of that item might be required.

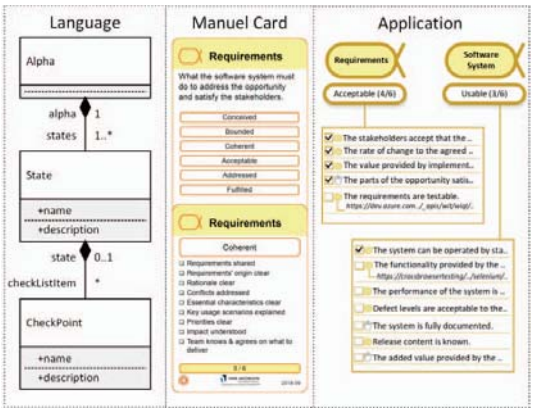

Fig. 2. Alpha state transitions in Essence Language, physical cards and our application

In a software project, checkpoints can be tracked through activities based on DevOps tools. For example, "The requirements are testable" checkpoint that can be retrieved as an information from a DevOps tool. In this example, it can be checked with a REST query on a DevOps tool to determine whether all the requirements' state are ready for test as seen in Fig. 2. Even if a hybrid DevOps toolkit is used for a project, the value of each checkpoint can be retrieved from the corresponding DevOps tool and the Alpha status information can be managed.

C. *Tool Development*

Based on the conceptual solution provided in the previous subsection, we developed a project monitoring tool based on Essence framework.

Fig. 3 displays the architecture of this tool. The architecture consists of four tiers: web client, web server, essence block and database. In web client tier, we used JavaScript, CSS and Bootstrap (a third party UI framework) and AJAX for the communication between client and server tiers. In web server tier, we used ASP.Net MVC with C# language. In data tier, we used Microsoft SQL Server and T-SQL.

In the first two layers of the essence tier, the language and kernel provided by the Essence Framework are located. On the third layer, ontologies of software development practices are transformed into essence language elements with a systematic approach [8]. In the fourth layer, the Essence elements obtained from the practices are combined and the kernel is expanded or modified. In the last layer, the runtime elements of the method used for a product execution are modeled.

At the time of writing this paper, we have integration plugins for the DevOps tools such as Team Foundation Server (Azure DevOps), Atlassian, Selenium, ASANA, and GitHub. We developed an integration layer to access DevOps tools. Each DevOps tool works as a plugin under Essence platform. Only a Rest API service URL is needed for the communication between a DevOps tool and our tool. One Rest API service can be defined for a single checklist item. This service can be triggered automatically or manually.

V. PROOF-OF-CONCEPT STUDY

A. *Example Scenario*

We provided a proof-of-concept scenario to illustrate the usage of our solution. In this scenario, the organization has four projects/products as shown in Fig. 10. Due to space constraints, we discuss the tool and process configurations of only Project B (Table 1) and Product C (Table 2). Project B is a more traditional project that uses a waterfall methodology. This project uses MS Project for project management, IBM Doors for Requirements Management, Atlassian Jira for defect management, SVN for source control, and Selenium for Test management.

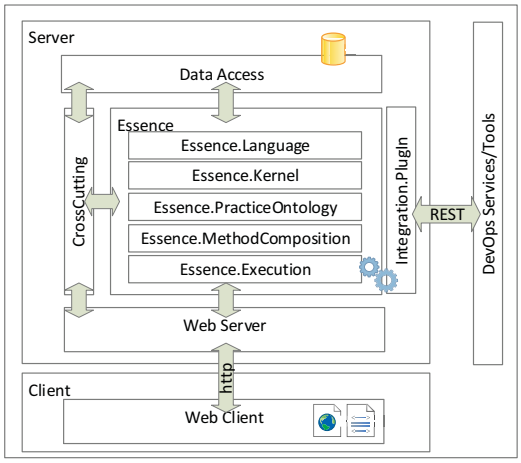

Fig. 3. Architecture for the project monitoring tool

TABLE 1. PROCESS AND TOOL INFORMATION RELATED TO PROJECT B.

Process	Waterfall
Project Management	MS Project
Requirements Management	IBM Doors
Source Control	SVN
Defect Management	Atlassian Jira
Test Management	Selenium

Whereas Product C is using Scrum as a development methodology. The project uses ASANA for project management, MS TFS for requirements management, Git for source control. The defect management and test management solutions are same with Project B.

TABLE 2. PROCESS AND TOOL INFORMATION RELATED TO PRODUCT C

Process	Scrum
Project Management	ASANA
Requirements Management	Microsoft TFS
Source Control	Git
Defect Management	Atlassian Jira
Test management	Selenium

The data required for a tool integration are divided into two categories as shown in Fig. 4. In "Data Source" category, some service information like address, authentication type, username, password, schedule date and feeding type are defined. In "Decision Model" category, service output details like output type (JSON, XML), token name, value member, data type, expected value and comparison type are defined.

In Fig. 4, we present the checklist items of the Requirements Alpha for its "Coherent" state. One of the checklist item is "The priority of the requirements are clear". To check this automatically, we query MS Team Foundation Server (Requirements Management solution for Product C) using the query displayed in Fig. 5. MS TFS uses WIQL (Work Item Query Language) to query work items.

The output of this query is a JSON string as shown in Fig. 6. The output has two work item objects that holds 'id' and 'url' attribute. 'asOf' attribute shows the execution date of the query that corresponds to 'Schedule' field in Figure 4. 'queryType' and 'queryResultType' attributes show that the

executed query result is a flat requirement work item list in MS TFS.

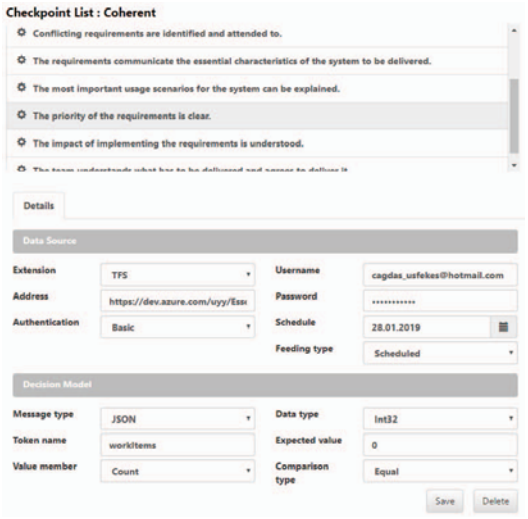

Fig. 4. DevOps tool service integration

```
SELECT [System.Id], [System.Url] FROM WorkItems
WHERE [System.TeamProject] = @project
AND [System.WorkItemType] = 'Requirement'
AND [Microsoft.VSTS.Common.Priority] = ''
```

Fig. 5. WIQL query for MS TFS

As defined under "Decision Category", when the number of workitems (token name) equals (comparison type) to zero (expected value), the checklist item for requirements priorities ("The priority of the requirements are clear") will be completed. Since there are two workitems as shown in Fig. 6, the state of this checklist item is not completed (denoted with ⚙ in Fig. 4).


```
{
  "queryType": "flat",
  "queryResultType": "requirement",
  "asOf": "2019-01-28",
  "workItems": [
    {
      "id": 81,
      "url": "https://dev.azure.com/uyy/2362ac953da5/_apis/wit/workItems/81"
    },
    {
      "id": 82,
      "url": "https://dev.azure.com/uyy/2362ac953da5/_apis/wit/workItems/82"
    }
  ]
}
```


Fig. 6. Sample JSON data obtained with REST API

In addition to Requirements Alpha, we need to check some other dimensions to monitor a project. As an example we can check the status of the resulting software system. Fig. 7 shows three checklist items of "Usable" state of "Software System" Alpha. The first two items are related to the test management tool, whereas the third one is related to the defect management tool.

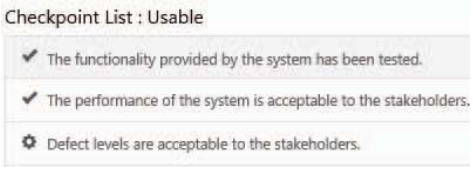

Fig. 7. Sample of checklist items for Usable state of Software System Alpha

We used a test tool (Selenium) available on the cloud (crossbrowsertesting.com) for the first two checklist items corresponding to functional and performance tests. The results of the tests can be automatically retrieved via a REST API.


```
{
  "selenium_test_id": 17439460,
  "selenium_session_id": "a2c4fab6-f02e-4a94-845c-e495a4165cf9",
  "start_date": "2019-02-04T11:27:46.000Z",
  "finish_date": "2019-02-04T11:28:00.000Z",
  "test_score": "pass",
  "selenium_version": "3.4.0",
  ...
  "show_result_web_url": "https://app.crossbrowsertesting.com/selenium/17439460"
  "show_result_public_url": "https://app.crossbrowsertesting.com/public/ibb45afb
  ...
}
```


Fig. 8. Sample test result JSON data obtained with REST API

Fig. 8 shows the result of a test as a JSON string. The value of the field "test_score" is "pass". When all the functional and performance tests are passed, both of the first two checklist items will be successfully completed.

Related to the defect management, the users executed all the test scenarios and reported their findings using the defect tracking tool (Atlassian Jira). One of the checklist items of this state is "Defect Levels are acceptable to the stakeholders" as shown in Fig. 7. Here we want to check, whether all the bugs that are marked as "critical" have been fixed. To achieve this, similar to the example for MS TFS, we used a JQL (Jira Query Language) query to check for this information.

```
project = "PRODUCT C"
AND issueType = Bug AND priority IN (Blocker, Critical)
AND (resolution != Fixed OR resolution IS EMPTY)
```

Fig. 9. JQL query for Jira

We query for Blocker and Critical issues in Product C that have not been resolved yet using the query shown in Figure 9. The result of the workitem count in JSON file is equal to 0, this means that since all the major issues (the issues that are blocker or critical) have been successfully resolved, thus this checklist item is successfully completed.

B. Answers to Research Questions

For RQ1, we investigated if we can effectively monitor the status of a project where hybrid set of tools exist for different software life cycle activities. In a proof-of-concept scenario, we provided examples where the status information come from various set of tools for different lifecycle activities.

For RQ2, we wanted to explore if we can effectively monitor consolidated status of multiple projects that use hybrid processes and tools. Originally Essence framework provide monitoring of several alphas for one project, in our solution, we provide a unified platform for multiple different projects. In proof-of-concept scenario, we illustrated the existence of different projects using different software development process methodologies in the same platform (as shown in Figure 10). This kind of a view would provide a unified view of multiple projects in the same organization.

As a result, we saw that Essence language and kernel are useful to establish a common terminology (dimensions, alphas to be monitored, work items to completed) and visualization for monitoring various software development projects. In addition, having a tool gathering data from various tools and allowing manual data entry on project status is a promising medium to monitor multiple projects in a unified way.

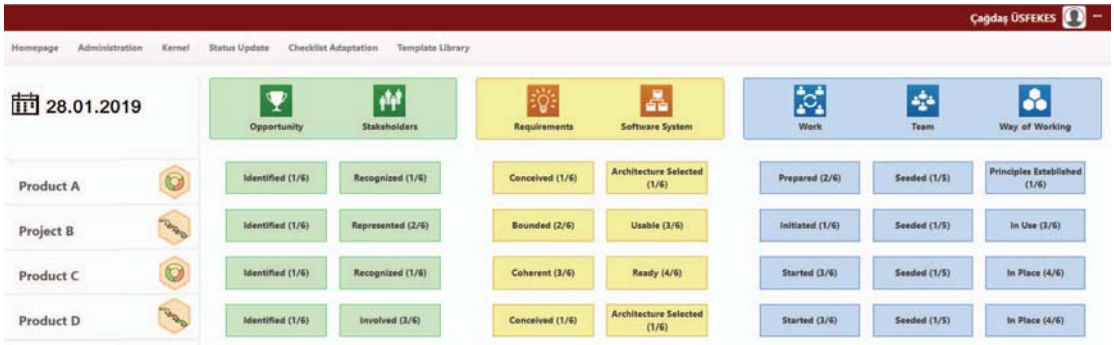

Fig. 10. Main dashboard of the project monitoring tool

VI. Related Work

SEMAT Accelerator (SematAcc) is a tool developed based on Essence kernel [10]. This tool enables the teaching, adoption, and research of Essence in controlled experiments and case studies [10]. The tool allows its users to mark the checklist items of each alpha state as completed. It visualizes the state of each alpha using a spider plot and the progress in each area of concern via a bar chart.

A dashboard application based on Essence framework aims to document the project progress, determine and assess the current project status, and support a team in planning next steps [11]. This tool displays the checklists for each alpha state and allows its users to mark each checklist item as completed. It has a main dashboard that shows a summary view of all alpha states and visualizes the overall progress by using different colors for completed, uncompleted, and optional states.

Our tool provides all of the functionalities provided by these two tools. In addition, our tool is and can be integrated with various ALM tools. Such integrations enable more accurate data collection for project progress with much less effort. Moreover, the project progress provided by our tool would be more up-to-date since automatic data collection can be triggered very frequently without any human intervention.

VII. Conclusion and Future Work

With the recent increase of new DevOps toolchains and new development process methodologies, hybrid tools and processes are a reality of our life in software development. To manage these types of scenarios, we proposed a conceptual solution based on the Essence framework to effectively unify data coming from various tools and teams. Based on this approach, we further developed tool support for our approach to monitor projects at an organizational level. This tool is integrated with various ALM tools to gather data on the statuses of various activities. Moreover, stakeholders can manually update the status of an activity, if a tool is not used for an activity. Since all the data gathered from other tools and stakeholders are mapped to the language and the kernel of Essence framework, it is possible to provide a unified view of projects independent from variations in methods and tools used in these projects.

Our future work includes validating this conceptual solution in a real-life scenario. We plan to add some functionalities for monitoring projects at a more fine-grained level using “sub-alpha” concept in Essence framework. We are also planning to differentiate incomplete checklist items from the ones that are behind schedule.

References

[1] M. Kuhrmann *et al.*, “Hybrid software and system development in practice: waterfall, scrum, and beyond,” in *Proceedings of the 2017 International Conference on Software and System Process - ICSSP 2017*, 2017.

[2] M. Kersten, “A cambrian explosion of DevOps tools,” *IEEE Softw.*, 2018.

[3] E. Tüzün, B. Tekinerdogan, Y. Macit, and K. İnce, “Adopting Integrated Application Lifecycle Management within a Large-Scale Software Company: An Action Research Approach,” *J. Syst. Softw.*, 2018.

[4] Object Management Group, “Essence - Kernel and Language for Software Engineering Methods,” *OMG*, no. Version 1.1, 2015.

[5] Project Management Institute, *Project Management Body of Knowledge: A Guide to the Project Management Body of Knowledge*. 2017.

[6] K. Beck *et al.*, “Manifesto for Agile Software Development,” *The Agile Alliance*, 2001.

[7] I. Jacobson, P.-W. Ng, P. E. McMahon, I. Spence, and S. Lidman, *The Essence of Software Engineering*. 2013.

[8] G. Giray, E. Tüzün, B. Tekinerdogan, and Y. Macit, “Systematic approach for mapping software development methods to the essence framework,” in *Proceedings - 5th International Workshop on Theory-Oriented Software Engineering, TOSE 2016*, 2016.

[9] K. Schwaber and J. Sutherland, “The scrum guide,” *Scrum. org, Oct.*, vol. 2, no. July, p. 17, 2011.

[10] D. Graziotin and P. Abrahamsson, A Web-based modeling tool for the SEMAT Essence theory of software engineering. Journal of Open Research Software 1:e4, 2013.

[11] S. Brandt, M. Striewe, F. Beck and M. Goedicke, "A Dashboard for Visualizing Software Engineering Processes Based on ESSENCE," 2017 IEEE Working Conference on Software Visualization (VISSOFT), Shanghai, 2017, pp. 134-138.