Jumpstarting Application Lifecycle Management: a New Approach with Tool Support

JUNG-SING JWO1, TIEN-SONG HSU2 AND YU CHIN CHENG2

1Department of Computer Science
Tunghai University
Taichung, 40704 Taiwan
2Department of Computer Science and Information Engineering
National Taipei University of Technology
Taipei, 10608 Taiwan

State-of-the-art application lifecycle management (ALM) platforms offer many benefits that enable the software development team to smoothly plan, execute and monitor its development activities. However, to first-time adopters of ALM platforms, the initial adoption presents three challenges that often prevent a successful adoption: to correctly define the team’s current software development activities, to correctly configure the ALM platform accordingly to support these activities, and to ensure that the team follows the decreed discipline throughout the lifecycle. To take on the three challenges effectively and efficiently, we propose a new approach called Rapid Application Lifecycle Management (RALM). RALM features a reference model that is described by a number of activity templates, which allows the adopter to customize, review, discuss, and revise ALM activities in rapid cycles. All customized activity templates are machine-translated to the target ALM platform: activity definitions are translated into process templates and configuration guide and the decreed discipline into engineering practice guidance for the team. An experiment to demonstrate the effectiveness and efficiency of RALM is presented.

Keywords: Application lifecycle management, ALM definition, ALM discipline, ALM implementation, ALM platform

1. INTRODUCTION

Application lifecycle management (ALM) deals with the way a software system or application is conceived, planned, developed, maintained, and decommissioned [1], [2]. Typical lifecycle activities include requirements development and management, project planning, solution development, deployment, issue tracking, and so on [3], [4], [5]. The execution of each of these lifecycle activities involves a multitude of technologies and requires tool supports. To a software development organization, successfully harnessing these technologies and tool supports is a crucial but complicated undertaking. A software development organization will typically have accumulated a number of tools, and these tools must be integrated to support the organization’s process. Aiming at reducing the complexity of integration, comprehensive state-of-the-art application lifecycle management (ALM) platforms such as Microsoft VSTS [6], Borland ALM [7], and IBM Jazz [8] offer convenient features for the organization to define and manage its ALM activities on a unified platform. Once correctly configured, a state-of-the-art ALM platform not only allows the development team to execute and check its daily development activities, but also enables the team to evolve its developmental activities through adjusting and tuning the current ALM platform settings. The latter aspect is especially important for a development team that aggressively seeks opportunities to enhance its capability. For example, once that team has reached a steady state in its current development practices, it can consider adding new practices such as automatic unit and acceptance testing, refactoring, static analysis, and so on, to its current practices. As such new practices are beginning to get exercised, the team will need support from the ALM platform.

While the use of ALM platforms reduces the complexity of integrating development and management tools and allows the team to modify process as required, to the development team, the first hurdle on the way to ALM success lies in correctly defining the ALM activities for the project on hand.

* A preliminary version of this paper was presented in the 9th International Conference on Software Methodologies, Tools and Techniques (SoMeT'10). This research was funded by the National Science Council of Taiwan under grant contracts 99-2221-E-029-017 and 98-2221-E-027-049-MY3, and a grant from MOEA under grant contract 99-EC-17-A-02-S1-135.
and in correctly setting up these activities on an ALM platform. Correctly initiating an ALM platform is challenging in three respects:

1. **Defining ALM activities**: More likely than not, the development organization that the team belongs to will already have some de-facto development process before it considers moving to a state-of-the-art ALM platform. In order to tailor the ALM platform to this de-facto process, a formal or semi-formal definition of its development process is necessary. This can be challenging, especially for the development teams in the small-to-medium size software development organizations, because formally or semi-formally defining the ALM activities and their interrelationships is in itself a complicated undertaking [9], [10].

2. **Implementing the definitions on an ALM platform**: Commercial and open source ALM platforms require tailoring to suit the team's need. The task of correctly setting up the platform should be easy, but in reality it is not as easily done. Misfits are common: a platform can offer too many unwanted default features while some features critical to the organization’s de-facto process can be missing [11]. To the development team, one of the worse things that can happen is to be forced to take on some unfamiliar technical activities prematurely.

3. **Enforcing ALM discipline**: Even with a correct ALM definition in place and a proper ALM platform setup, whether the defined ALM activities are faithfully executed remains a main area of doubt to many organizations [12], [13]. In particular, aspects related to coordination and cooperation among developers are often viewed as non-technical overhead. Right information at the right time and context is needed [11].

In this paper, we present an instrumented approach to facilitate the tasks of ALM definition, ALM platform setup, and ALM discipline. The approach is called Rapid Application Lifecycle Management, or RALM in short. RALM provides a reference model for ALM against which the software team uses to define its ALM activities. The reference model is recorded in a number of activity templates. When the templates are completed, ALM platform setup is performed by machine-converting the templates to platform-specific process definition files. Where such machine conversion is not possible on the target ALM platform, a step-by-step cookbook for manual setup is produced by the conversion program. Lastly, engineering practice guidance for carrying through the ALM discipline is extracted from the descriptions in the templates and is automatically published on the project portal by the conversion program.

With respect to the three challenges outlined above, RALM achieves the following benefits:

- The activity templates and the reference model facilitate the development team to capture and define its ALM activities and the relationships among the activities using the team’s current process as the ALM baseline.
- By providing a tool to translate the activity templates into platform specific process definition files as well as a configuration cookbook for the chosen ALM platform, the process of initiating ALM platform support is expedited and the problem of ALM platform misfits is effectively resolved.
- The engineering practice guidance is published and maintained on the project portal. The availability of such information assists the team achieves discipline.

The rest of this paper is organized as follows. The characteristics of the state-of-the-art ALM platforms are described in Section 2. The RALM process is presented in Section 3. The tool support for instrumenting ALM platform according to the ALM definitions is presented in Section 4. In Section 5, the benefits for RALM in terms of empowering and time saving in initializing ALM platform are demonstrated experimentally. In section 6, the current studies on adoption ALM solutions are briefly reviewed. Section 7 offers the conclusion and points out the future work.

**2. STATE-OF-THE-ART ALM PLATFORMS**

The development of ALM platforms (e.g., Microsoft VSTS, Borland ALM, and IBM Jazz) has reached a point that they emphasize the customization of processes and tools to suit the development team's need. No longer are teams forced to accept a specific process, especially one that the team is unfamiliar with. As a result, the same ALM platform can support waterfall, unified, agile, or any other home grown software processes [14]. The main features of ALM platforms include configuration and
change management, issue tracking, project management, build management, report management, and so on. As can be seen, a substantial amount of work is involved in setting up the platform for these functions to work correctly.

The state-of-the-art ALM platforms fill the “how-to” gap for organizations that adopt process improvement standards such as CMMI [15], which provides a framework for defining “what” to be done. For example, in configuration management, a support process area in CMMI, a specific practice says that access control should be established. However, until the project has been created and configuration items have been determined, such a practice is at best a declaration; access control setting is only possible after project artifacts have been defined. For this reason, it can be noted that while an ALM platform such as VSTS supports CMMI, customizations such as setting access control rights are still necessary and are performed manually [6]. For this reason, platforms such as VSTS provide built-in definitions, including MSF (Microsoft Solutions Framework) for Agile Software Development and MSF for CMMI Process Improvement, to expedite ALM initiation. Other settings are possible, for example, using process definition files for Scrum [14], [16].

3. THE RALM PROCESS

The primary benefit of a comprehensive ALM platform is the provision of all commonly used ALM functionalities (or best practices) on a single platform. However, the numerous interdependent ALM activities also greatly increase the complexity and difficulty of adopting the ALM platform. To avoid being overly general, we propose to look at ALM at the project level. That is, once the project scope, budget and time are known, many activities can be defined down to the specifics, which help eliminating vagueness and avoiding misfits. The RALM process is depicted in Figure 1, where VSTS is used as the target ALM platform although the process is equally applicable to other ALM platforms. RALM process consists of two phases, ALM activity definition and ALM platform initiation:

- In the ALM activity definition phase, the team completes ALM activity templates according to the team’s project requirements represented by the input “My Project ALM Requirements” and creates the output “My Project ALM definition” as shown in Figure 1. The typical project requirements include scope, time, budget, people who are going to work on the project, the development scenario (e.g., waterfall, iterative, or agile), and so on. Furthermore, ALM reference model is used to help review, discuss, and revise the ALM definition in rapid cycles.

- In the ALM platform initiation phase, the ALM activity definitions are implemented on the target ALM platform. RALM converts the customized ALM activity definitions into process definition files through a conversion tool (i.e., ALMTranslator in Figure 1), which generates three types of files: the process definition files to be imported to the target ALM platform, a configuration guide for the target ALM platform, and engineering practice guidance which is published on the project portal.

In the rest of this section, the ALM reference model is explained in detailed.

3.1 The ALM reference model –static model

The ALM reference model, which is shown in Figure 2, is obtained by performing object-oriented analysis on the most commonly used development scenarios, including waterfall, iterative and agile methods. The thirteen objects, or concepts, that are identified in the reference model are grouped into ten different types of templates, which are called activity templates, in RALM:

- **Project definition:** the development team needs to write down the project goal, description and disciplines in this template. The project goal and description provide a clear purpose to help development team on the right direction. The disciplines define some practices which must be followed during this project, for example, programmers need to perform unit testing before they commit codes to the repository.
Artifact definition: the development team defines all the produced artifacts during the project. For each artifact, the development team can define the format as standard of artifact and provide template files. These artifacts need to be managed as the configuration items and committed to the repository.

Role definition: the development team defines the roles and their responsibilities involving the development process. Also, the collaboration of roles and the related artifacts need to be defined in this template. For example, a programmer needs to develop code and perform unit testing according to the design specification; a tester needs to perform the integration and functional testing based on the production code; and so on.

Organization definition: the development team defines the organization structure of the project. The identification, roles, and skills of team member are provided as references for project planning, task dispatching and account management on the ALM platform.

Process definition: the development team defines the development process in this template. The milestones, iterations, and activities of the process are defined clearly for providing a comprehensive instruction to the team members during the project.

Configuration management definition: this template helps the development team identify configuration items, establish baseline, and set up the configuration management system. Also, the artifacts under configuration management are set.

Authorization definition: the development team needs to define the authorization policy for accessing configuration management items according the role playing.

Task dispatching definition: The development team can perform the project planning in this definition. This template can help the development team plan and dispatch the tasks in the project initiation phase. Moreover, some information defined organization definition can provide a basis for performing task dispatching, for example, the tasks with required skills are possible to be assigned to the capable persons.
Software health indicators definition: the software health indicator can provide important information to the development team for recognizing the status of project execution. The development team can define some reports as the software health indicators in this template, such as bug rate report, remaining tasks, team velocity, and so on.

Issue tracking definition: the development team defines the issues for the project, such as requirement changing, bug, process improvement, and so on. Also, the issue tracking processes can differ widely.

The structure of the template is very similar to the one for enterprise consulting [17], which is a form-filled style document. The activity templates allow the development team to specify and review their own ALM definition against the ALM reference model. The templates can help the team iterate the task of defining the ALM activities to satisfaction before implementing on the ALM tool. The definition of above ALM activity templates are designed based on the survey of the state-of-the-art ALM platform. Also, these templates can be extended easily for the project specific needs.

3.2 The ALM reference model – dynamic model

After filling out the ten activity templates, the team can visualize objects in the reference model for the activities performed. The visualization helps the team to review whether an activity is performed as intended. Here, we give an example of how activity of committing code to repository is played out by the objects interacting with each other.

Figure 3 shows the affected objects in the reference model in gray. For the activity of committing code to repository, seven objects are related to the activity, including Project (MyProject), Repository, Artifact (Code), Organization, Person, Role (Developer) and AccessControl. A developer needs to commit codes to the code repository after adding new functionalities or fixing bugs. In Figure 4, a developer commits the changed code to the repository (steps 1 to 2). Since each developer can only access code that is allocated to them, the repository checks the access control right by querying the ALM Platform (steps 3 to 7). If cleared for access right, code is committed (steps 8 to 9), otherwise, the commit action is denied (step 10).
4. THE DESIGN OF ALMTRANSLATOR

The definition-to-settings conversion tool, ALMTranslator, is designed for generating three types of platform-specific ALM files: process definition, configuration guide, and engineering practice guidance. The target platform is the Microsoft Visual Studio Team System 2005, and the team can define the ALM definitions by using Microsoft Office 2003 Excel which helps the team customize their ALM definitions more conveniently. The process of definition-to-settings conversion in ALMTranslator is described in the following steps:

1. Loading the activity templates: The team's customized ALM activity template is loaded into ALMTranslator.
2. Transforming to a XML-based model: The team's ALM definitions are transformed as the
internal representation. Based on this model, the team’s ALM definitions can be converted to platform-specific ALM files.

3. Performing XML-based model verification: This step ensures that the model is correct and sufficient to generate the platform-specific ALM files; otherwise go back to revise the team’s ALM definitions until the verification passes.

4. Loading ALM platform (in this case, the VSTS2005) configuration: The platform-specific ALM translation rules are loaded into the ALMTranslator. The translation rules include file structure, format, environment, the process definition, and so on.

5. Generating process definition files, configuration guide and engineering practice guidance: In the final step, the process related elements of the internal representation are parsed. The VSTS2005 process template architecture is referred to generate the process definition files. Then, the configuration guide is created by extracting the configuration parameters. Also, the engineering practice guidance is produced according to the ALM definitions.

Figure 5 shows the design class diagram of ALMTranslator. The ALM activity templates are parsed into the XML-based model by the ALMParser. The internal representation is manipulated by the ALM2VSTS for generating the platform-specific process definition files and the configuration guide. ALM2VSTS also needs to get the configurations of target platform through the specific ConfigLoader (VSTSConfigLoader in this case). Finally, the PracticeGenerator generates the platform-independent engineering practice guidance.

5. EXPERIMENTAL RESULT

The goal of the experiment is to evaluate the effectiveness the RALM method in two respects: (1) the extent to which RALM enables the developer in defining and initiating ALM activities on the ALM platform correctly and (2) the time saving that RALM’s tool support achieves in so doing. The Microsoft VSTS2005 is chosen as the ALM platform. In the experiment, four groups of subjects (whose backgrounds are described in Section 5.1) were asked to define the ALM activities and to set up VSTS2005, accordingly, for a multi-site project of developing a balanced scorecard (BSC) application [18]. The experiment was designed and conducted following the guidelines in [19], [20].

5.1 Subjects and Preliminary Training

Table 1 summarizes backgrounds and groupings of the 12 subjects who volunteered to participate
in the experiment. The participants were graduate students who have successfully passed the graduate-level courses of Software Engineering and Software Lifecycle Management offered by the Department of Computer Science and Information Engineering of the National Taipei University of Technology [21], [22]. In the former course, the participants had acquired working knowledge in lifecycle activities including requirements, design, testing, and so on; in the latter course, the students received training on various lifecycle models and tool supports for lifecycle activities, and were required to complete a term project of setting up the Scrum process for a Scrum team on IBM Jazz or Microsoft VSTS for a data synchronization and backup application. In Section 5.4, it is confirmed that such backgrounds gave the participant sufficient working knowledge to complete the tasks involved in the experiment.

<table>
<thead>
<tr>
<th>Group</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Control group</td>
<td>Control group</td>
<td>Experimental group</td>
<td>Experimental group</td>
</tr>
<tr>
<td>Numbers of subjects</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Experience with VSTS2005</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Using the RALM method</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The participants were organized into four groups of three persons each. Participants in groups A and C had no prior experience with VSTS2005, while those in groups B and D had up to one year's experience with VSTS2005. Groups A and B did not apply the RALM method and served as the control groups; groups C and D applied the RALM method and served as the experimental groups.

The participants were given a two-day training workshop. The first day covered the objective and procedures of the experiment, rules of timer recording, a review on ALM activities, and an introduction to VSTS2005. The latter focused on how to customize the VSTS process templates for project management and reporting, issue tracking, test management, build management, and so on. A number of process template customization exercises were completed in the workshop by the participants to ensure that they understand the material covered.

The second day training was attended only by groups C and D for learning the RALM method and the use of ALMTranslator.

5.2 Project and Process Specifics

The multi-site BSC development project is adapted from a real industrial development case provided by an enterprise solution provider. At the time the project was completed, the enterprise solution provider had completed its CMMI ML3 certification. The waterfall process was used. The project duration is three months and involves 11 developers at headquarter and an offshore location. Each developer has a different access control right to the code repository. The engineering practices adopted include requirements management, configuration management, change management and issue tracking. The reports include bug rates, remaining works, and burn-down charts. All of these ALM requirements can be implemented on VSTS2005.

The project is staffed with members from five departments across two sites. The project manager (PM) \( m_1 \) from the product department (code named P) is responsible for requirement development, project monitoring and product release. The quality assurance engineer (QA) \( q_1 \) from the quality assurance department (code named Q) is responsible testing and integrating the product. The development department, which has branches both at headquarter (code named H) and at an offshore location (code named O), is responsible for the analysis, design, implementation, and integration. After the initial requirements are listed, the project manager \( m_1 \) has proposed the following development roles: system analyst (SA) \( a_1 \), system designer (SD) \( d_1 \), and programmers (PR) \( p_1, p_2, \) and \( p_3 \) with \( H \) and programmers \( p_4, p_5, \) and \( p_6 \) with \( O \). Responsibilities of post-release services fall with service engineer \( s_1 \) of customer service department (code named S).

Initial analysis by \( m_1 \) and \( a_1 \) has determined that BSC will constitute a common component library \( C \) and five modules \( N_1, N_2, N_3, N_4, \) and \( N_5 \). All development works are done at headquarter \( H \).
except the implementation of N3, N4, and N5, which are done at the offshore location O. The artifacts of the BSC projects include requirements, system analysis, data dictionary, data models, source code, compiled executable, unit test code and test reports.

The waterfall process is defined as follows. First, \( m_1 \) provides requirements specification, which is analyzed by \( a_1 \) to produce analysis documents for C, N1, N2, N3, N4, and N5. The design of C is then completed by \( d_1 \) and then implemented by \( p_1 \). During the implementation work of C, \( d_1 \) completes the design of N1, N2, N3, N4, and N5. When C is completed, implementation work of N1, N2 are performed by \( p_2 \) and \( p_3 \), respectively, and implementation work of N3, N4, and N5 are performed by \( p_4 \), \( p_5 \), and \( p_6 \), respectively. When all implementation work, including unit testing, is completed, \( q_1 \) performs integration testing and produces testing reports. Any errors are reported. Once an error is confirmed, a person is assigned to fix it, both of which are decided by project manager \( m_1 \). The process is repeated until the work is finished. After that, \( m_1 \) takes care of the release of the BSC application. All bugs and issues (e.g., new requirements) reported by the users and customers are handled by \( s_1 \).

To facilitate artifact access control and version control, developers can only access configuration items that are allocated to them. The project directory structure is shown in Figure 6. The common library and module names are used as the directory names. The directory “Test” is used by QA \( q_1 \) as the test and integration work area. The directory “Release” is used by PM \( m_1 \) for release. Further, version control is facilitated by stipulating that new version is established upon artifact check-in and old versions must be available. A check-in procedure is competed only if tasks, artifacts, or bug information are correctly entered for bidirectional traceability.

Changes and events that occur during the lifecycle phases are captured as issues. The following types of issues are defined: requirements issue, work dispatch issue, bug issue, new version issue, and other issue. Actions taken against an issue are based on issue state. For example, the states of requirements issues are as follows: (1) the roles PM, SA, SD, and PR are permitted to propose new requirements or changes to existing requirements. (2) The proposed changes are reviewed by the PM, who decides the necessity of the proposed changes and consults with the SA and SD on the analysis on change impact. Once accepted, the issue is set up with deadlines and assigned. On the other hand, the proposed change can be rejected or shelved. (3) PM effects any modifications for configuration management to reflect the proposed change. (4) PM determines issue closure or abandonment.

Lastly, a project portal is maintained for project BSC. The portal provides three kinds of information to the team members: project management information, engineering practice guidance including responsibilities for each role, upstream and downstream collaborations within the waterfall model, and discipline.

**5.3 Procedure and Execution**

The experiment was carried out by following a pre-defined procedure and was proctored by the experiment administrator, a role played by one of the authors.

Subjects in experimental groups C and D followed the RALM method as described in Figure 1, while subjects in control groups A and B were asked to follow the process as described in Figure 7.
Note that the process in Figure 1 and Figure 7 are basically the same except in the use of RALM. All subjects perform their tasks on a centralized Microsoft Team Foundation Server.

Subjects were given time to read the project ALM requirement documents. Their understanding was checked by the experiment administrator through individual interviews. Afterwards, on his or her own, each of the subjects spent two or more days to perform the tasks of Figure 1 (for the experimental group) and Figure 7 (for the control group) in three sessions: session 1 for ALM definitions (of the ALM activity definition phase), session 2 for VSTS configuration (of the ALM platform initialization phase), session 3 for publishing engineering practice guidance (of the ALM platform initialization phase). Subjects were required to finish a session before moving on to the next session. Note that:

1. In session 1, the subjects in groups C and D (the experimental groups) were required to fill the ten ALM activity templates described in Section 3.1. The subjects in the groups A and B (the control groups) wrote down the ALM definitions in their own formats. The results were checked by the experiment administrator.

2. In session 2, the experimental groups C and D used the ALMTranslator to translate the templates completed in session 1 into the VSTS process template files and a configuration guide to help them initialize the BSC project on VSTS2005. In contrast, the control groups A and B manually customized the VSTS process template and configure the VSTS2005 settings to fit their own ALM definitions.

3. Lastly, in session 3, the experimental groups C and D only were required to set up the portal and publish the engineering practice guidance on the portal, while the control groups A and B wrote up the engineering practice guidance as well as turn them into web pages.

![Diagram](image_url)

**Fig. 7.** The process followed by the control group.

### 5.4 Results

Table 2 shows the four groups' average time spent in completing each session. With the exception of group A, which only completed session 1 (ALM definitions), all the other groups were able to complete all the three sessions. From Table 2, the following observations can be made:

- All groups were able to complete ALM definitions for the BSC project. This confirmed that the participants were indeed about equal in their software engineering and software lifecycle management knowledge and had the working knowledge to complete the session 1 task. Note that, however, the groups C and D’s use of RALM templates was able to achieve a time saving of about 30% when compared to groups A and B.
Despite completing the first-day training on VSTS2005, the subjects in group A were unable to complete, within the given amount of time, the task of correctly initializing the project by customizing the process templates and manually performed the remaining settings that cannot be achieved by the former on VSTS2005. This shows that the task of implementing the ALM definitions completed in session 1 on VSTS2005 is still a challenging. In particular, it is observed that, the subjects of group A were frequently unable to locate the right instructions to configure the VSTS2005; such frustrations had prevented them from completing session 2 and session 3. In contrast, the subjects of group C, who are equally unfamiliar with VSTS2005 as the group A subjects, were able to complete the task of implementing ALM definitions on VSTS2005 by applying RALM with ALMTranslator. This shows that RALM can empower developers unfamiliar with the ALM platform the capability to perform the setup correctly.

Though both familiar with the VSTS, it was found that, on average, the subjects in group D spent 45% less time than subjects of group B by applying the RALM method. It had been observed that the ALM activity templates, the process template files and the configuration guide generated by ALMTranslator had helped the subjects in group D to concentrate on the primary ALM activities and without worrying about the detailed configuration of VSTS2005.

Even more interestingly, in terms of the time spent in correctly setting up the ALM platform, subjects of group C (who were not familiar with VSTS2005 but used RALM) spent 33.5% less time than subjects of group B (who were familiar with VSTS2005 but did not use RALM). Thus, to group C subjects, the ALM activity templates and the ALMTranslator had helped just as in the case of group D.

With the use of the RALM method, the gap of prior familiarity with VSTS2005 had been narrowed: overall, group C (who were not familiar with VSTS2005) only took 21% more time than group D (who were familiar with VSTS2005).

### Table 2 Experimental results on the BSC project.

<table>
<thead>
<tr>
<th>Group</th>
<th>Phase</th>
<th>Session</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALM activity</td>
<td>1. ALM Definitions</td>
<td>426</td>
<td>434</td>
<td>304</td>
<td>285</td>
</tr>
<tr>
<td></td>
<td>definitions</td>
<td>2. VSTS</td>
<td>N/A</td>
<td>355</td>
<td>255</td>
<td>178</td>
</tr>
<tr>
<td></td>
<td>ALM platform</td>
<td>3. Publishing</td>
<td>N/A</td>
<td>94</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>initialization</td>
<td>engineering</td>
<td>N/A</td>
<td>94</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>practice guidance</td>
<td>N/A</td>
<td>94</td>
<td>28</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td><strong>Total</strong></td>
<td>N/A</td>
<td>883</td>
<td>587</td>
<td>485</td>
<td></td>
</tr>
</tbody>
</table>

Unit: minutes

5.5 Threats to Validity

The participants of this experiment are limited to the students who have already attended the Software Engineering and Software Lifecycle Management classes. While the selection criterion ensures the participants have the required background for the experiment, it also limits the possibility of finding more participants. Thus, the average time recorded is likely to suffer from the problems common with a small sample.

Whether subjects are familiar with VSTS2005 is checked by the experiment administrator asking subjects a few questions about VSTS2005. Thus, it is difficult to really evaluate the participants' working knowledge with VSTS2005. This may have affected the outcome of the experiment, for example, it is possible the group C is only slightly worse than group D because some of the group C participants actually know about VSTS2005.

Lastly, it was not possible to fit the whole experiment within one day. Although there were no obvious motives for the participants to do so, some participants may have worked between the sessions to resolve difficulties encountered in the previous sessions. Time spent in this way is unaccountable and can affect the results of this experiment.
6. RELATED WORKS

There are several researches related to adopting ALM solutions. Kääriäinen J. et al. defined an ALM framework for documenting and finding improvement for the company's ALM solutions. The development of the framework has been presented in [23] and [24]. The principal elements of the ALM framework were the following: creation, management, traceability, and reporting of lifecycle artifacts, process, tool integration, and so on. Pirklbauer G. et al. identified key problem areas typically addressed by ALM and provided guidance on how to develop an ALM strategy for the software development organizations [25].

Since each team has its own needs, the challenge is to find suitable tool implementation of ALM for complicated, real-life situations. This had been reported even for teams that belong to the same organization, where several case studies involved using VSTS as the platform for running Scrum. Moore et al. suggested making modifications to the process template or even creating new process template from scratch [26]. Medina-Dominguez et al. proposed the adaptation of process template based on the project patterns and a model to support process improvement by using patterns in a TFS environment [27]. In these works, the tailoring of the ALM platform was performed manually. RALM can obviously help by proving a more streamlined initiation.

Lastly, Goth classified the existing ALM tools into four categories: full life-cycle management, targeted planning applications, consulting component and ALM plug-in [28]. According to this classification, RALM can be seen as a combination of consulting component and ALM plug-in.

7. CONCLUSION AND FUTURE WORK

For a software development team that is initiating application lifecycle management on a state-of-the-art ALM platform, the three main challenges are to define the ALM activities and the relationships among them according to the team’s current practices, to correctly implement the ALM definitions on the selected ALM platform, and to make sure that the team following the decreed ALM disciplines. Unless all three challenges are met effectively and efficiently, the software development team can all but abandon its effort in adopting ALM due to the initial difficulties. To meet the three challenges effectively and efficiently, RALM, a new method for jumpstarting ALM practices on the state-of-the-art platform such as the Microsoft VSTS, has been proposed. On being effective, RALM provides a collection of ALM activity templates and a reference model that enable the development team to define and verify its ALM requirements. On being efficient, the activity templates are written in spreadsheet and machine-translated into platform-specific process definitions and additional configuration guide for features that must be configured manually. The combined use of templates, machine-translation and configuration guide enables the team to define, review, implement, receive feedback, and revise ALM activities in rapid cycles, thereby mitigating the risk of team getting bogged down in the ALM initiation phase. As demonstrated by the results of the experiment, the adoption of RALM enables developers unfamiliar with VSTS2005 to perform the setup correctly. Also, RALM expedites the task of ALM initiation. Among the developers unfamiliar with VSTS2005, those using RALM spend 33.5% less time than those without using RALM. Finally, among the developers familiar with VSTS2005, those using RALM spend 45% less time than those without using RALM.

The present work only covers the task of ALM initiation. Although this is an important first hurdle for the development team to cross, a further issue is to support the development team in evolving the way that life cycle management is conducted, e.g., by adopting new engineering practices such as test driven design, continuous integration, and so on. On the state-of-the-art ALM platforms, this is achieved through manually adjusting and tuning the existing platform settings. It would be interesting if such adjustments can be achieved by adding and modifying the templates. Currently, this is not yet possible since the ongoing project with the original settings may suffer project data loss if new settings are machine-implemented on the ALM platform. Lastly, although the current ALMTranslator only covers the Microsoft VSTS, it is easy to extending the ALMTranslator to other ALM platforms such as the IBM Jazz and the Borland ALM.
REFERENCES

8. IBM Jazz, available: http://jazz.net/.


Jung-Sing Jwo (周忠信) received the M.S. and the Ph.D. degree from the University of Oklahoma, both in Computer Science. He is currently an Associate Professor at the Department of Computer Science, Tunghai University, Taichung, Taiwan, where his research interests include software engineering and enterprise computing. He is a member of the Software Engineering Association of Taiwan (SEAT).

Tien-Song Hsu (徐天送) is a Ph.D. student at the Department of Computer Science and Information Engineering, the National Taipei University of Technology, Taiwan, where he received the M.S. degree. His research interests include software engineering and cloud computing. Mr. Hsu is a student member of the Software Engineering Association of Taiwan (SEAT).

Yu Chin Cheng (鄭有進) received the M.S.E. degree from the Johns Hopkins University in 1990 and the Ph.D. degree from the University of Oklahoma, in 1993, both in Computer Science. He is currently a Professor at the Department of Computer Science and Information Engineering of the National Taipei University of Technology, Taiwan. Dr. Cheng teaches and conducts research in object-oriented systems, software lifecycle management, and software architecture. He is a member of the IEEE Computer Society, the ACM, and the Software Engineering Association of Taiwan (SEAT).