Research Article

Science, Engineering and Health Studies
https://li01.tci-thaijo.org/index.php/sehs
ISSN (Online): 2630-0087

Toward traceability model between ISO/IEC 29110 artefacts and agile process

Waraporn Jirapanthong

College of Creative Design and Entertainment Technology, Dhurakij Pundit University, Bangkok 10210, Thailand

***Corresponding author**:*
Waraporn Jirapanthong
waraporn.jir@dpu.ac.th

***Received**: 22 February 2023*
***Revised**: 23 May 2023*
***Accepted**: 5 July 2023*
***Published**: 1 December 2023*

***Citation**:*
Jirapanthong, W. (2023). Toward traceability model between ISO/IEC 29110 artefacts and agile process. Science, Engineering and Health Studies, 17, 23040006.

ABSTRACT

This paper presents the efficacy of the traceability model in the development of artefacts related to pre-defined activities in the ISO/IEC 29110 process and ones related to agile process. This efficacy is demonstrated through a study of very small entities (VSEs) in Thailand who applied a development script based on the traceability model for various software projects. Six VSEs, with their respective software development projects, participated in the study with comparable time constraints and budgets. The VSE development teams utilized the traceability model to establish connections between the artifacts of the agile process and the ISO/IEC 29110 standard. The results show the potential benefits for VSEs in the application of the traceability model in such processes and what the teams learned from their participation. Each team's productivity differed only slightly. The average figure of productivity with the traceability model was 1.416. However, the average figure of productivity without the traceability model was 1.3045. It is found the agile process can be facilitated and less effort by applying the traceability model.

Keywords: agile; ISO/IEC 29110; traceability; VSEs; software process

1. INTRODUCTION

Software processes typically include planning, requirements specification, software analysis and design, software development, testing, and deployment. The common challenges during deployment are data management, system integration, inadequate planning, resistance to change, lack of customer engagement, and the rapid changing of digital technology (Barry et al., 2002; Rodríguez et al., 2012). Software deployment often takes months, and project teams may spend considerable time after software deployment in order to provide long term software maintenance and support.

In general, the implementation time of a software project depends on the type, requirement scope, and technologies used for the project. It may take only one to nine months with an average time of 4–12 months. Based on our experience, the implementation time for different software projects varies based on project size. For example, a very small software project with basic features can take less than 2 months, a small software project can take 2–4 months, and an average sized software project can take 4–6 months. However, changes in marketing and customer trends can have a significant impact on software features and requirements which in turn lengthens the time of project development (Gupta et al., 2016; Vellanky, 2007). This is due to the correlation of team size, team effort, implementation time, and team productivity.

According to Barry et al. (2002) and Rodríguez et al. (2012), team size, effort, productivity and project implementation efficiency are the driving factors which determine the meeting of software development deadlines. Furthermore, it is found that meeting customer requirements poses a major challenge to meeting development deadlines.

This is due to potential changes of requirements, which can include redevelopment. Such factors impact project planning including milestones, deliverables, and budget.

Generally, the delivery and deployment of software projects are a complicated endeavor. They require a specific set of tools, technologies, and skills as well as specific configurations for particular systems. Moreover, there is a need for standards and formal processes to support the software development life cycle.

A software process model typically consists of many phases which are limited by resources. An example of a traditional software process model is named waterfall. It is suitable for software projects with stable and complete software requirements requirements. Such development parameters, coupled with the waterfall model, lead to the successful progress of software projects. However, the agile model becomes a more attractive approach when variables are less stable due to team collaboration, customer involvement, and changes in requirements. The agile method more easily meets the needs of project flexibility and rapidly responds to unplanned changes. Scrum, a methodology within the Agile framework, is widely utilized due to its responsiveness and flexibility. The agile framework includes the activities of team collaborations like daily Scrum meetings; of continuous software integration, such as sprint backlogs; and of customer involvement, for instance rapid customer feedback.

In addition, the process of standardizing software varies depending on the parties involved in the standardization (Humphrey, 2005). The most common and practical are the international ISO/IEC standards. The International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) develop standards for worldwide application. They have also formed the Joint Technical Committee (JCT1) which is divided into subcommittees (SC) and working groups (WG). Each WG is charged with the development of standards in a specialized area. ISO produces two main types of documents: International Standard (IS) and Technical Reports (TR). There are many available standards. As a result, it becomes difficult for development teams to choose one based on practicality. This study focuses on the implementation of the ISO/IEC 29110 standard (International Organization for Standardization, 2015a, 2015b, 2016) in conjunction with the agile process. Although, some authors (Laporte and O'Connor, 2016; Suteeca and Ramingwong, 2016; Alvarez and Hurtado, 2014; Boucher et al., 2012; Laporte et al., 2016; Castillo-Salinas et al., 2020; Buchalcevova, 2021) have researched the implementation of the international standard of software process i.e., ISO/IEC 29110 for software agile processes, issues and impracticality still exist. In particular, there are impracticalities with relation to Thai VSEs and their use of ISO/IEC 29110 for software agile processes. Based on Jirapanthong (2018, 2019), this study focusses on the characteristics of ISO/IEC 29110 standard and the agile process and identifies the key artefacts related to two areas i.e. project management, and system implementation. Based on the study, a traceability model is proposed between artefacts from ISO/IEC 29110 and the agile process. The study was conducted to document the experiences of the approach.

1.1 Agile process

Agile (Schön et al., 2019; Erickson et al., 2005; Dingsøyr et al., 2012) is a project management methodology based on an iterative approach. It focuses on the delivery of product development via self-organizing teams which can respond to change effectively. The agile manifesto consists of 4 core values: i) individuals and interactions over processes and tools, ii) working software over comprehensive documentation, iii) customer collaboration over contract negotiation, and iv) responding to change over following a plan. Moreover, the agile manifesto includes 12 principles: i) customer experience, ii) embracing change, iii) continuous delivery, iv) collaboration, v) transparency, vi) high bandwidth communication, vii) working software, viii) individuals, ix) sustainability, x) just in time, xi) self-organization, and xii) retrospective.

There are two popular frameworks used to implement agile software development; Scrum (Hossain et al., 2009) and Kanban (Anderson, 2010). For the Kanban framework, real-time communication of capacity and full transparency of work is required. Work items are represented visually on a Kanban board, allowing team members to see the state of every piece of work at any time. Work in progress (WIP) sets a limit on the maximum amount of work that can exist in each status of a workflow. Through this visualization, teams can identify inefficiency in their workflow. The key steps for the Scrum framework are as follows:

i) Project planning: The Scrum master and product owner plan the primary list of work that needs to be developed. This list, named the product backlog, encompasses features, requirements, enhancements, and fixes. The product backlog is constantly reprioritized due to changes.

ii) Roadmap planning: The product roadmap is an action plan for how a product will develop over time. The product roadmap provides the framework for a given team's work. It includes a daily action plan for each team. This allows large companies with many agile teams to plan and follow the same product roadmap. The roadmap presents the market segments, product values, and project constraints. All project members are able to access and view the roadmap in order to understand the objectives and timeline of the whole project.

iii) Release planning: A high-level plan for multiple sprints, three to twelve iterations for example, is created during release planning. It is a guideline that reflects expectations about which features will be implemented and when they will be completed. It also serves as a base to monitor progress within the project. Releases can be deliveries done during the project or the final delivery at the end. A Scrum master creates a release plan based on a prioritized and estimated product backlog, the efficiency of teams, and other conditions.

iv) Sprint planning: A sprint plan is a concrete plan for a Scrum team. It defines what exactly will be delivered to the customers and how products can integrate with available releases. The plan provides details of tasks that the software project team must tackle.

v) Sprint execution: Sprint execution is the work the Scrum team performs during each sprint to meet the sprint goal.

vi) Sprint review and retrospective: The sprint review is a meeting at the end of the sprint. The software project team, consisting of the members of the Scrum team, the Scrum master, and the project manager, meet with the product owner and the stakeholders during the sprint and exchange their work experiences. The product owner will update the team regarding what has been done or not. Moreover, they will discuss problems which led to the failure of sprint objectives which should be improved for the next sprint. The sprint retrospective is a meeting that occurs immediately after the sprint review. Team members identify good practices which led to sprint goals being achieved. The sprint retrospective is aimed to improve the team's development process.

The main artefacts from the agile methodology are the burndown chart and the task board. The burndown chart is used to present the team's progress. It shows the outline of user stories in accordance with the schedule. It also conveys the work rate and performance of the team. The chart is kept updated during a sprint. The burndown diagram represents in particular the product and the sprint backlog that will be achieved. The task board is used by each project team member. The board presents tasks and dependencies between tasks and the team members can use the board as a team memo to update the completion of tasks. The board can be used by a project team to assign and track tasks, to share information, and visualize task dependencies and bottlenecks. The flow of individual tasks can also be displayed. This includes tasks that are in progress, finished tasks, and upcoming tasks that may be in a backlog. The tasks can be represented by cards and arranged within a few columns on a board. Currently, there are several software tools to support the task board for agile project development as the tasks can be elaborated into multiple levels. The task board can be used to predict and avoid problems, increase effectiveness, and improve production.

1.2 ISO/IEC 29110

ISO/IEC 29110 (International Organization for Standardization, 2015a, 2015b, 2016) is aimed at supporting SMEs (Small and Medium Enterprises) in the assessment and improvement of their software processes. The key objective of ISO/IEC 29110 is to establish a standard which provides a set of software profiles for supporting SMEs. The standard includes complete and extensive processes which are separated into four profile levels: entry profile, basic profile, intermediate profile, and advanced profile. The basic profile is applicable for SMEs to improve their working process. In particular, it is suitable for very small units including no more than 25 staff members per project team. According to this profile, there are two defined processes in a software development project. Each process includes defined tasks and work products. The defined processes are project management (PM) and software implementation (SI). The activities of the project management process are project planning, project plan execution, project assessment and control, and project closure. The activities of software implementation process are software implementation initiation, software requirements analysis, software architectural and detailed design, software construction, software integration and tests, and product delivery. There are a total 50 tasks under all activities which involve 22 defined work products. Some tasks are related to the creation of new work products and some are related to updating existing ones. As shown in Figure 1, the project management process includes four activities and customer involvement. The implementation process is then linked to the consequence of the project management process. The standard does not define the specific methods of software development, it only provides the basic profile of software development activities.

Though the standard aims to provide a set of best practices and guidelines for improving the quality of the software process, the standard does not prescribe a particular software development methodology. In practice, software development teams in small companies have some difficulties. For example, they may have an issue selecting software tools and development methods which conform with the standard. Some studies (Laporte and O'Connor, 2016; Suteeca and Ramingwong, 2016; Alvarez and Hurtado, 2014; Boucher et al., 2012; Laporte et al., 2016; Castillo-Salinas et al., 2020; Buchalcevova, 2021) present case studies on the compliance of software process standards. They also describe the pros and cons for the use of standards. In Thailand, there are a large number of very small-sized software companies. They apply different software methods and tools to implement software projects. They also experience the difficulties of following the standard. According to studies conducted on the software development of Thai organizations (Waewseangsang and Khongmalai, 2014; Mongkolnam et al., 2009; Suwanya and Kurutach, 2008), most Thai organizations do not focus on the standard of ISO/IEC 29110.

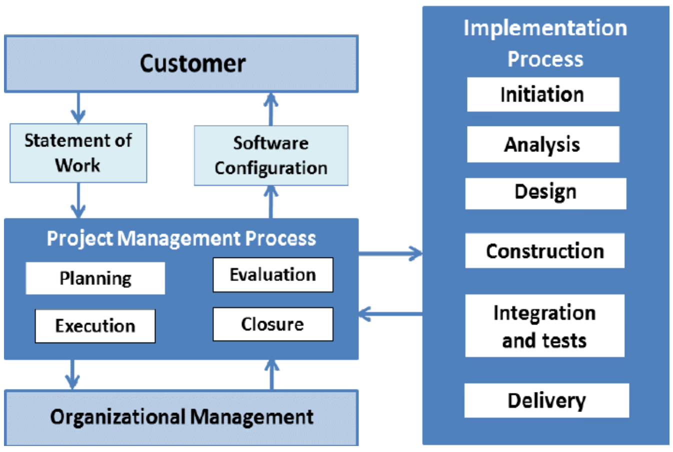

Figure 1. ISO/IEC 29110 basic profile processes and activities (from www.iso.org/standard/51154.html)

2. MATERIALS AND METHODS

2.1 ISO/IEC 29110 activities and artefacts

According to ISO/IEC 29110, the artefacts (work products) are created, updated, and used in pre-defined activities. As shown in Table 1, artefacts created during an activity can be related to other activities. For example, project plans, meeting records, verification results, and project repositories are created during project planning. The project plan is updated during project execution and applied during software implementation initiation. Some artefacts are produced periodically (verification results, validation results, and meeting records), some are produced conditionally (change requests) and some created and updated in many activities (project repository and traceability record). Software artefacts (requirements specification, software design, software, software components, test cases, and reports) are produced and updated during the activities of the software implementation process. Some artefacts (change request, correction register, and progress status record) are collected during the software process, some are used as input for other activities, and others (maintenance documentation, software configuration, verification results, and acceptant record) are completed at the end of process.

Table 1. ISO/IEC 29110 activities and related artefacts

Activities	Artefact created	Artefact updated	Artefact as input
Project planning	Project plan Meeting record Verification results Project repository		State of work
Project plan execution	Project repository backup Meeting record Progress status record	Project repository Project plan Progress status record	Change request Correction register
Software implementation initiation			Project plan
Software requirements analysis	Change request Requirements specification Verification results Software user documentation	Validation request	Project Repository
Software architectural and detailed design	Software design Change request Test cases and test procedures Verification results Traceability record		Requirements specification Product repository
Software construction	Software components Test report Product operation guide Software configuration Software	Traceability record Test cases and test procedures Software user documentation	Software design Project repository
Project assessment and control	Change request Correction register		Progress status record Project plan
Project delivery	Maintenance documentation Verification results	Software configuration	Project repository
Project closure	Acceptance record	Software configuration	Project plan

2.2 Framework of agile process with ISO/IEC 29110 standard

This study's approach focuses on the agile Scrum process. As shown in Table 2, artefacts created during an activity can be related to other activities. For example, project plan and product backlog are created during project planning. The project backlog is then used during roadmap and release planning and updated during sprint planning due to a change or feedback. Feedback is reviewed and used for subsequent activities. According to the agile process, some activities occur iteratively. Some artefacts are continuously updated like the burndown chart: some are produced periodically such as daily meeting records and release; and some are produced conditionally through feedback and change.

As shown in Figure 2, some artefacts are created during an activity and repeatedly used for subsequent activities. The activities of agile process are iteratively performed until the product is complete.

Table 2. Activities and artefacts in agile Scrum process

Activities	Artefact created	Artefact updated	Artefact as input
Project planning	Project plan Product backlog		User stories
Roadmap planning	Features Roadmap		Project plan Product backlog
Release planning	Burndown chart Release plan	Roadmap Features	Product backlog Feedback
Sprint planning	Spring backlog	Product backlog Burndown chart	
Sprint execution	Daily meeting record Release		Feedback
Sprint review	Review record	Product backlog Burndown chart	Feedback Release plan

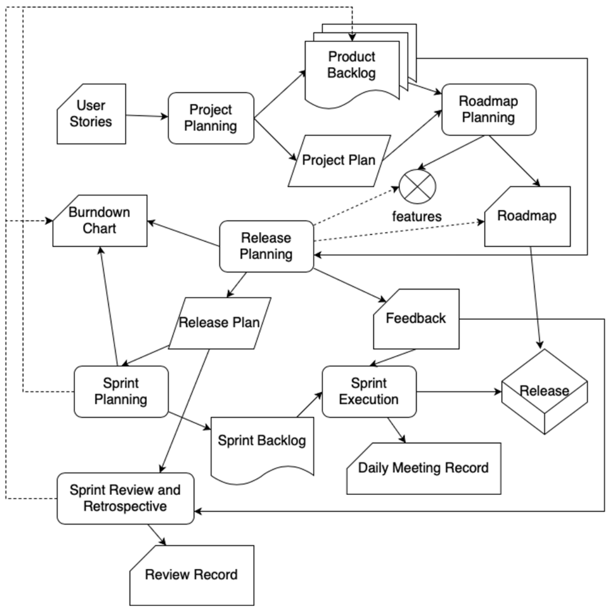

Figure 2. Agile process with related artefacts

The mapping model of agile process with ISO/IEC 29110 standard are identified in Table 3. It shows the relationship between artefacts produced during sprint execution and ISO29110 work products. For example, a work product, labeled acceptance record, produced during the ISO29110 activity of project closure is related to release and to the final product produced during the agile activity of sprint review. Similarly, progress status record created and updated during the ISO29110 activities (project plan execution, and project assessment and control) is related to the burndown chart created during the agile activity (sprint review, sprint planning, and release planning).

Table 3. The traceability model of related artefacts in ISO/IEC 29110 and agile Scrum activities

ISO 29110 work products [activities in ISO/IEC 29110]	Related artefacts in agile approach [activities in agile approach]
Acceptance record [project closure]	Release, Final product [sprint review]
Change request [project assessment and control], [software requirements analysis], [software architectural and detailed design]	Sprint backlog [Sprint review], [release planning]
Correction register [project assessment and control]	Meeting record [daily meeting]
Meeting record [project plan execution]	Meeting record [sprint meeting], [daily meeting]
Progress status record [project plan execution], [project assessment and control]	Burndown chart [sprint review], [sprint planning], [release planning]
Project Plan [Project planning], [Project plan execution]	Product backlog, Sprint backlog [Project planning], [Roadmap planning], [Release planning], [Sprint planning]

Table 3. (Continued)

ISO 29110 Work products [activities in ISO/IEC 29110]	Related artefacts in agile approach [activities in agile approach]
Project repository [project planning], [project plan execution]	Repository [project planning], [roadmap planning], [Release planning], [roadmap creating]
Project repository backup [project plan execution]	Backup [sprint planning]
Requirements specification [software requirements analysis]	User stories [roadmap planning], [release planning], [sprint planning]
Software [software integration and test]	Release [sprint execution]
Software components [software construction]	Release [sprint execution]
Software configuration [product delivery], [project closure]	Software configuration [sprint execution]
Software design [software architectural and detailed design]	Sprint backlog [sprint execution]
Software user documentation [software requirements analysis], [software integration and test]	Release [sprint execution]
Statement of work, test cases and test procedures [software architectural and detailed design], [software integration and test]	User stories, unit and integration test result [sprint planning]
Test report [software integration and test]	Unit and integration test result [sprint execution]
Traceability record [software architectural and detailed design], [software construction], [software integration and test]	Roadmap, user stories, product backlog, release [sprint planning]
Validation results [software requirements analysis]	Feedback [daily meeting], [sprint execution]
Verification results [project planning, software requirements analysis], [software architectural and detailed design]	Feedback, meeting record [project meeting], [sprint meeting], [sprint review]
Product operation guide [software integration and test]	-, [sprint execution]
Maintenance documentation [product delivery]	-, [sprint execution]

2.3 Statistical analysis

The research objectives focused on the challenges of implementing the ISO/IEC 29110 standard with the agile process. The research encompasses two parts. The first part is to study an ecosystem aligned with a software lifecycle. The factors related to drive the process in the ecosystem are identified in the study. The factors are then compiled in the form of evaluation measures. The metrics include the policies and practices used by the software team leaders. The second part is to evaluate the overall performance of the ecosystem. The scope of performance is quantified using success factors. The scores of the factors or sub-factors are accumulated. The research also compares such factors with other ecosystems at similar lifecycle phases.

For the first part, there are the following steps:

1. Software projects and teams

VSEs in Thailand (Bangkok) contributed to this research. The VSEs aimed to improve their software process compliance with the ISO/IEC 29110 standard proposed in this study. The study identified the practices of the standard and how to meet the success of software development. The team's knowledge, distributed resources, and deployed software processes were recorded. The study comprised six software project teams in which members had equal skills. The project teams were assigned different projects and asked to drive the software process in alignment with agile methodology.

The six software projects were from different VSEs with different characteristics, i.e., the level of maturity, the professionals, the professional community and society, and the corresponding business area. Basically, the software teams had a working experience and knowledge of software engineering. They had experience developing software systems and delivering software products; however, their software project was driven by various constraints and factors, e.g., customer requirements, business strategies, and digital enhancement. The teams did not fully apply the agile framework during development due to such constraints and factors. Their project progress was then not predictable or repeatable; nor were good practices identified. All six software projects were established in similar timelines. Each team member was assigned and responsible for tasks which were driven by the agile framework e.g., system analyst, project manager, and software developer. All projects had the limitation of time, budget, and resources. In particular, the projects had to be completed within 4 months and include 2.5 person-months. Moreover, the teams were requested to implement the projects under the agile framework and be compliant with the ISO/IEC 29110 standard.

2. The traceability model of related artefacts in ISO/IEC 29110 and agile Scrum activities

Development teams were encouraged to discipline themselves with the ISO/IEC 29110 standard and apply the traceability model, as described in the previous section, during the agile process.

One of the contributions was to identify both explicit and implicit relationships between software artefacts in the agile process. Templates for ISO/IEC 29110 work products were also provided. In particular, the ISO/IEC 29110 toolkit, which includes a set of work product templates was prepared and utilized by development teams.

The second part contained, the following steps:

1. Execute experimental plan.

In this step, each team developed their own software project under planned budgets and constraints. The profile of each team and project is shown in Tables 4. The VSEs had different business goals and objectives. Basically, the range of software projects covered in this study were different. The number of team developers refers to the human resources of each team and to those who had similar skills. The team's work experience number shows the average working years of each team in the software engineering field.

However, our approach is to perform process improvement and to assist teams in their alignment with project management methodologies to overcome the differences of software projects and enable them to apply the standard. As discussed with the project teams, the key factors in helping teams successfully transition to agile Scrum method were identified:

a) Every team needs members who contribute individually as experts, with light supervision and guidance. Smaller lean teams or start-ups are suitable for this work environment.

b) Each team must set expectations and train their team members to ensure that they understand how agile scum works, and what benefits the standard will have.

c) The VSEs should prototype the implementation of the standard and the agile Scrum methodology. They should not manage too many large changes or implement changes too quickly. The successful process can then be deployed to the rest of the company's projects.

According to the key factors identified, a prototype plan was created with six teams and projects. As shown in Table 4, team members acting as experts for each contribution were organized and project prototypes were created.

Table 4. Developer team A's profile

Team	**A**	**B**	**C**	**D**	**E**	**F**
VSE's business goals and objectives	Be an academic institute; offer academic services	Be a government sector; support information systems within the organization	Start up; innovative and research	Start up; be funded by NIA	A software house; web application development	A software house; web application development
Domain of software product	Online learning system	Incident management system	Logistic management system for coffee shops	Online learning system	Health information system for a hospital	Information system for a HR activities
No. of team developers (persons)	4	3	2	6	3	2
No. of team's working experience (years)	6	7	2	2	8	2
Time budget	2 months	3 months	3 months	4 months	2 months	6 months
Number of stakeholders (party)	3 (users, product owner, developer team)	2 (users, developer team); note that the team is the product owner themselves.	2 (users, developer team); note that the team is the product owner themselves.	5 (users, product owner, funder, domain expertise, developer team); note that the project is funded by NIA	3 (users, product owner, developer team)	5 (users, product owner, developer team)
Number of staff in each role of the project	Project manager & Scrum master: 1	Project manager & Scrum master: 1	Scrum master & requirements engineer: 1	Project manager: 1	Project manager & Scrum master: 1	Project Manager & Scrum Master: 1
	Requirements engineer: 1	Requirements engineer & software designer & software developer & software tester: 2	Project manager & software designer & software developer & software tester: 1	Scrum master & requirements engineer: 1	requirements engineer & software designer & software developer: 1	requirements engineer & software designer & software developer: 1
	Software designer & software developer & software tester: 2			Software designer: 1		
				Software developer & software tester: 3		

Quantifying performance with success factors and metric identified factors and metrics for evaluating the results of the software projects. The performance of the software projects is quantified and scaled in terms of success factors. The success factor scores represent performance, human resource management, team experience, team engagement, and agile experience.

However, this research focused on the study of implementing the ISO/IEC 29110 standard effectively and practically. The research metrics were time and effort spent to satisfy user requirements. In particular, the productivity of each project team was measured. The parameters of software projects such as project size, requirements, stability, time, budget, resources, and team experiences were all recorded (Rodríguez et al., 2012; Kitchenham and Mendes, 2004; Wagner and Ruhe, 2018). In this research, the variables were as follows: a) the development teams' business goals and objectives and b) the domain of software products. A list of tasks which can be defined by day (8 working hours) were defined. The variables, i.e., task phase, individual profile, or project state were defined. In this research, a task refers to a job that requires time to be delivered in order to achieve the task's objectives and the default time was set to five hours. Otherwise, tasks were considered delivered within a day (set to eight hours). We then identified the productivity of each team was identified using Equation 1.

$$\text{Productivity} = \frac{\text{Planned Effort} \div 5}{\text{Actual Effort}} \times 8 \qquad (1)$$

Whereas planned effort for each task was accumulatively measured and compared with actual effort for each task, the planned effort was estimated by working hours for each task. The milestones and project phases were also estimated and related to the tasks in the project plan. The actual effort was progressively measured during implementing tasks.

Six VSEs agreed to implement the ISO/IEC 29110 standard. The software projects of each team were considered to be of the same size, but in different areas. The number of people in the development teams are relatively equal (2–6 people). However, the business goals and objectives of each VSE were different. Table 4 shows the characteristics of the VSEs.

3. RESULTS AND DISCUSSION

According to the research, the software teams created the software products. Each team was asked to perform software development activities under the agile process in accordance with the ISO/IEC 29110 standard. The scenario was created based on the typical environment of a VSE ecosystem. Some activities involved many types of documents. Therefore, the teams were asked to follow a development script. The script encompassed the core activities of the ISO/IEC 29110 standard and agile process. The software teams participating in the research were asked to perform the following tasks during the development script: i) application of software tools to create related documents; and ii) enabling traceability between artefacts.

Table 5. Software project measurement

Team	LOC	FP	Planned effort (days)	Actual effort (days)	Productivity
Team A	61591	12	90	118	1.22
Team B	14471	4	90	92	1.57
Team C	8326	3	90	104	1.38
Team D	77543	16	90	125.2	1.176
Team E	35120	5	90	94	1.53
Team F	13580	4	90	89	1.62

In addition, many documents were created such as software design documents, software specifications, source code, and team management. The number of each type of documents was measured. For example, the number of defects from the review of design and code artefacts, the number of pages or lines of code from the coding phase, the number of operational configurations from the testing phase, and the entities of team size and average team experience were collected. The size measurement of software products involved two aspects: (a) line of code (LOC) and (b) function point (FP). As shown in Table 5, the number of LOC created by each team were 61,591, 14,471, 8,326, 77,543, 35,120 and 13,580 respectively. Although the teams followed the development script based on the agile process, they implemented production phases on different platforms e.g., the web, Android, and iOS. However, the numbers of each software product's FPs were different depending on the requirements. Moreover, the quality of software products were measured in terms of maintainability attributes. The attributes were coding effort, design effort, percentage of modules changes, classes changes, and classes added. Table 5 shows measured the productivity of each team. As described in the previous section, the productivity was calculated based on planned effort, actual effort, and an 8-hour work day. The figures of productivity show each team's measurement of effort spent per 8 man-hour day.

Additionally, the productivity of project teams which employed traditional software processes without the traceability model is shown in Table 6. The figures present productivity measurements that were accumulated based on previous software projects with similar sized requirements and time budgets. The figures show that the productivity of the same team utilizing the traceability model is higher than the one without the traceability model.

Table 6. Software project measurement

Team	Productivity with the traceability model	Productivity without the traceability model
Team A	1.22	1.10
Team B	1.57	1.45
Team C	1.38	1.311
Team D	1.176	1.09
Team E	1.53	1.28
Team F	1.62	1.596

4. CONCLUSION

Six development teams applied the traceability model for the software development process. The process encompasses agile activities and is compliant with the ISO/IEC 29110 standard. Each team's productivity differed only slightly. The average figure of productivity was 1.699. All teams delivered their software products successfully. A retrospective review concludes that individual team risk factors were identified and appropriately addressed. Artefacts were created and traceable in support of their respective software lifecycle. The mean time to implement changes on software products was also found, for example Team C spent 15.5 days to tackle all changes. Impacts on their time budget and related resources were also measured. An average time of implementing changes in each team was calculated. We have recovered the work effort can be decreased by applying the traceability model. The agile Scrum process can be facilitated by having traceability. The impact of changes can be fully addressed.

Additionally, at the beginning, organizations saw big differences or gaps between their plans and their actuals; however, as teams matured these gaps reduced. After the project, each team needed to look back and revise the problems and incidents occurring during the project development. The cause of the incidents needed to be identified and analysed. The causes could be factors from inside or outside the team. There was a correlation between productivity and errors. The experiments show that not only is productivity high in software project development, but so is defect density. This may indicate that the planned effort for quality assurance activities is not sufficient. Ideally, teams need to balance between two attributes, productivity and quality. In practice, teams may need to trade-off between degrees of productivity and the quality of the software product development. Since the actual effort spent during defect removal activities can be painful and costly, it can lead to a high cost for software deployment. It is noted that process improvement in design and code review can increase the focus on quality during the development lifecycle. Furthermore, the early detection of issues and defects can lead to high defect removal efficiency and requires less effort. The cost for defect removal at the early stage of software development is less than at the later stages of development.

According to the agile framework, workflow needs to be monitored and the burndown chart needs to be updated. Project attributes need to be integrated progressively. Some tasks or resource may be replanned during release or sprint planning, some backlog may be re-prioritized due to later project attributes. For future work, verification metrics will be investigated to study different activities and frameworks of software development and deployment. Another goal is to define the effective and efficient improvement of software processes.

ACKNOWLEDGMENT

This research wouldn't have been possible without the VSEs which contributed in this research to promote and support ISO 29110 certification.

REFERENCES

Alvarez, J. J., and Hurtado, J. A. (2014). Implementing the software requirements engineering practices of the ISO 29110-5-1-1 standard with the unified process. In *Proceeding of the 2014 9th Computing Colombian Conference (9CCC),* pp. 175–183. Pereira, Colombia.

Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*, Washington DC: Blue Hole Press.

Barry, E. J., Mukhopadhyay, T., and Slaughter, S. A. (2002). Software project duration and effort: An empirical study. *Information Technology and Management*, 3(1), 113–136.

Boucher, Q., Perrouin, G., Deprez, J.-C, and Heymans, P. (2012). Towards configurable ISO/IEC 29110-compliant software development processes for very small entities. In *Systems, Software and Services Process Improvement: Communications in Computer and Information Science, Vol. 301* (Winkler, D., O'Connor, R. V., and Messnarz, R., Eds.), pp. 169–180. Berlin: Springer.

Buchalcevova, A. (2021). Towards higher software quality in very small entities: ISO/IEC 29110 software basic profile mapping to testing standards. *International Journal of Information Technologies and Systems Approach*, 14(1), 79–96.

Castillo-Salinas, L., Sanchez-Gordon, S., Villarroel-Ramos, J., and Sánchez-Gordón, M. (2020). Evaluation of the implementation of a subset of ISO/IEC 29110 software implementation process in four teams of undergraduate students of Ecuador. *An empirical software engineering experiment. Computer Standards & Interfaces*, 70, 103430.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N. B. (2012). A decade of agile methodologies: towards explaining agile software development. *Journal of Systems and Software,* 85(6), 1213–1221.

Erickson, J., Lyytinen, K., and Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management*, 16(4), 88–100.

Gupta, V., Dutta, K., and Chauhan, D. S. (2016). Mass market development strategies of software industries: Case study based research. *Perspectives in Science*, 8, 96–100.

Hossain, E., Ali Babar, M., and Paik, H. (2009). Using Scrum in global software development: A systematic literature review. In *Proceeding of the 2009 4th IEEE International Conference on Global Software Engineering (ICGSE 2009)*, pp. 175–184. Limerick, Ireland.

Humphrey, W. (2005). *PSP: A Self-Improvement Process for Software Engineers*, Boston: Addison Wesley.

International Organization for Standardization. (2015a). Software engineering—lifecycle profiles for very small entities (VSEs)—part 2–1: Framework and taxonomy, *ISO/IEC 29110*, 2(1), 32.

International Organization for Standardization. (2015b). Software engineering—lifecycle profiles for very small entities (VSEs)—part 3: Assessment guide, *ISO/IEC TR 29110*, 3(1), 6.

International Organization for Standardization. (2016). Software engineering—lifecycle profiles for very small entities (VSEs)—part 1: Overview, *ISO/IEC TR 29110*, 1, 23.

Jirapanthong, W. (2018). The study of international standard implementation for supporting software process in startup business enterprise. In *Proceeding of the 8th International Workshop on Computer Science and Engineering (WCSE)*, pp. 735–739. Bangkok, Thailand.

Jirapanthong, W. (2019). Experience in applying of ISO 29110 to agile software development. *Journal of Information Science and Technology*, 9(1), 63–70.

Kitchenham, B., and Mendes, E. (2004). Software productivity measurement using multiple size measures. *IEEE Transactions on Software Engineering*, 30(12), 1023–1035.

Laporte, C. Y., and O'Connor, R. V. (2016). Implementing process improvement in very small enterprises with ISO/IEC 29110: A multiple case study analysis. In *Proceeding of the 10th International Conference on the Quality of Information and Communications Technology (QUATIC 2016),* pp. 125–130. Lisbon, Portugal.

Laporte, C. Y., O'Connor, R. V., and Paucar, L. H. G. (2016). The Implementation of ISO/IEC 29110 software engineering standards and guides in very small entities. *Evaluation of Novel Approaches to Software Engineering: Communications in Computer and Information Science, Vol. 599* (Maciaszek, L. A. and Filipe, J., Eds.), pp. 162–179. Cham: Springer.

Mongkolnam, P., Silparcha, U., Waraporn, N., and Vanijja, V. (2009). A push for software process improvement in Thailand. In *Proceeding of the 2009 16th Asia-Pacific Software Engineering Conference*, pp. 475–481. Penang, Malaysia.

Rodríguez, D., Sicilia, M. A., García, E., and Harrison, R. (2012). Empirical findings on team size and productivity in software development. *Journal of Systems and Software*, 85(3), 562–570.

Schön, E.-M, Sedeño, J., Mejías, M., Thomaschewski, J., and Escalona, M. J. (2019). A metamodel for agile requirements engineering. *Journal of Computer and Communications*, 7(2), 85–106.

Suteeca, K., and Ramingwong, S. (2016). A framework to apply ISO/IEC29110 on Scrum. In *Proceeding of the 2016 International Computer Science and Engineering Conference (ICSEC),* pp. 1–5. Chiang Mai, Thailand.

Suwanya, S., and Kurutach, W. (2008). An analysis of software process improvement for sustainable development in Thailand. In *Proceeding of the 8th IEEE International Conference on Computer and Information Technology (CIT)*, pp. 724–729. Sydney, NSW, Australia.

Vellanky, P. N. (2007). *Software Maintenance – A Management Perspective (Issues, Tools, Techniques, and Trends)*, Boca Raton, FL: Universal-Publishers.

Waewseangsang, P., and Khongmalai, O. (2014). Risk management in Best practice: Case study CMMI in software industry in Thailand. *KMUTT Research and Development Journal,* 37(1), 133–141. [In Thai]

Wagner, S., and Ruhe, M. (2018). Systematic review of productivity factors in software development. In *Proceeding of the 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008)*, pp. 1–6. Beijing, China.