*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

# Standardizing the *Software Tag* in Japan for Transparency of Development

Masateru Tsunoda[1], Tomoko Matsumura[1], Hajimu Iida[1], Kozo Kubo[2], Shinji Kusumoto[3], Katsuro Inoue[3], and Ken-ichi Matsumoto[1]

[1] Graduate School of Information Science, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan
{masate-t@is, tomoko-m@is, iida@itc, matumoto@is}naist.jp

[2] Research Center for Advanced Science and Technology, Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan
kubo@rsc.naist.jp

[3] Graduate School of Information Science, Osaka University, 1-5 Yamada-Oka, Suita-shi, Osaka, 565-0871 Japan
{kusumoto, inoue}@ist.osaka-u.ac.jp

**Abstract.** In this paper, we describe the *Software Tag* which makes software development visible to software purchasers (users). A software tag is a partial set of empirical data about a software development project shared between the purchaser and developer. The purchaser uses the software tag to evaluate the software project, allowing them to recognize the quality level of the processes and products involved. With Japanese government support, we have successfully standardized the software tag named *Software Tag Standard* 1.0, and have developed various associated tools for tag data collection and visualization. For its initial evaluation, the software tag has been applied to several projects. This paper also presents various activities aimed at promoting the use of the software tag in Japan and the world.

**Key words:** Information sharing, empirical data, project management, offshore development

## 1 Introduction

Software systems are becoming huge and complex, with our everyday life heavily dependent on such software systems. One of the major concerns of software purchasers (users) in Japan is the quality of the software systems. Japanese society generally demands high-quality software systems with low fault rates and high operability levels.

On the other hand, many software purchasers in Japan are not knowledgeable about the nature of software. It is reported that only 40% of Japanese major companies employ a full-time Chief Information Officer (CIO) and that only 20% of all CIOs are confident of their knowledge about information technologies [10].

Without a sufficient understanding of software quality and software projects, many companies try to purchase software systems from software developers (vendors). This

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

produces a very risky situation. For example, purchasers cannot specify system requirements very well, and they do not oversee the project properly. Such situations often lead to project failures. It is reported that only 31.1% of software projects are recognized as 'successful projects' in Japan [11]. To confront these issues, there is strong demand to provide transparency of software projects to the software purchaser and improve communications between purchaser and developer.

The *Software Tag* is a new scheme to provide information feedback about the project from the developer to the purchaser. It establishes transparency of the software development project by allowing purchasers to view and analyze the elements of the tag. It also provides support for quantitative and qualitative communications between stakeholders. The Software Traceability and Accountability for Global Software Engineering (*StagE*) project [1] is a government-supported project that pursues standardization and promotion of the software tag scheme. In this project, we have defined the detailed structure of the software tag and developed various support tools. The software tag has been applied to real projects of major Japanese organizations. Along with technical development, we have also started various promotion activities, such as formal standardization of the software tag in both domestic and international standards, and exploration of new trade laws for software using the software tag scheme.

An early concept of how software tags could be used for software maintenance was shown in [5]. In this paper, we mainly explain use of the software tag for software development, together with activities and outcomes from the StagE project. In section 2, we describe an overview of the software tag scheme, and in section 3 explain the details of the software tag structure. In section 4, we describe activities of the project. In section 5 we provide some discussion, while in section 6 we outline conclusions and future research topics.

## 2 Overview of the Software Tag Scheme

A software tag is a packaged data set about a software project. It is currently composed of 41 characteristic elements of project data and progress data, as defined in section 3.1. Fig. 1 shows an overview of the software tag scheme.

1. A software purchaser orders development of a software system. The purchaser includes both the final products and the software tag in their requirements.
2. During software development, various kinds of empirical data are created and generated. For example, requirements documents, software design documents, source code, test cases, issue tracking logs, manual documents, review logs, and quality analysis records may be produced. These are collected and archived. Note that we collect not only the final data at the end, but also interim snapshot data during development.
3. The collected data is analyzed for process improvement of the development organization, as is the usual process improvement scheme for software development organizations.

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

4. The collected data is used to construct the software tag. Parts of the empirical data are selected and abstracted into the software tag format.
5. The software tag is delivered to the software purchaser periodically during the development and/or finally at the end of the development together with the final software product. The software purchaser evaluates the software development by viewing and analyzing the tag, and accepts the delivered software product.

If a controversy such as a question about the quality of the product occurs between the software purchaser and the developer, the delivered software tag and (if necessary) the empirical data are analyzed, providing a basis for exploring a resolution to the controversy.

The software tag is a key to improving *transparency* of software projects. By examining the software tag, the software purchaser can identify and understand the development process, which has been mostly hidden from the purchaser. The purchaser can evaluate the quality of the processes and products of the project.

For the software developer, the software tag is useful to prove that they have conducted the proper activities in the software project. Also, it can be used to trace the quality of the activities of sub-contractors and sub-sub-contractors... (such contracting chains are very popular in Japan).

This scheme can be very useful for offshore and global development, because transparency and traceability of software development can be established with a fairly low overhead for the developers.

Standardizing the software tag will help to establish a minimum baseline for project quality, and to improve negotiations over software development contracts. Evaluation of software products and projects based on the objective empirical data contained in the software tag will lead to more healthy use of software in society.

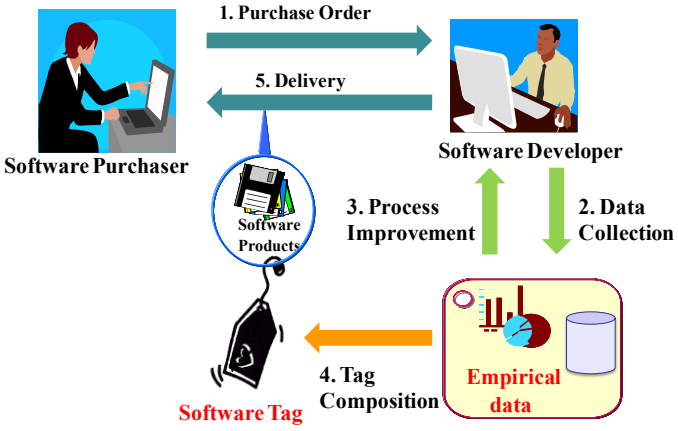


Fig. 1. Overview of Software Tag Scheme

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

# 3 Development of Software Tag Technologies

## 3.1 Software Tag Standard 1.0

We have defined the elements of the software tag as shown in Table 1, named *Software Tag Standard 1.0*. It is composed of 41 tag elements, which are categorized into *project information* and *progress information*. The project information depicts the overall sketch of the project with various basic pieces of information. The progress information provides qualitative and quantitative indices of project achievement with various measures of the development phases. The tag standard provides more precise explanations and example metrics for each tag element which are not presented here.

We divided project information into five categories described below, and settled tag elements for each category (see Table 1).

- basic information of the software project (*Basic Information*)
- information of the system developed by the project (*System Information*)
- information of development framework applied to the project (*Development Information*)
- information of relationships between target project and other projects (*Project Organization*)
- other information (*Others*)

To settle progress information, we referred to ISO/IEC 12207 Software Life Cycle Processes and activities [7], and included process, quality and effort information described below into it.

- information of requirements, design, programming and test for the software (*Requirements*, *Design*, *Programming*, and *Test*)
- information of quality assurance activities on the project (*Quality*)
- information of development effort on the project (*Development Cost*)
- information of project plan and management on the project (*Schedule and Management*)
- other information for the products attached to the software (*Other Products*)

It is not mandatory to use all 41 elements in the software tag in all cases. The purchaser and the developer can negotiate and select elements to use. Also, they can discuss and determine the details of the metrics. For example, #19, Scale of Programming, might be agreed to be measured by lines of code without comments. Based on the software tag standard, the purchaser and the developer should decide followings before using the software tag.

- tag elements to be used
- metrics used for tag elements
- measurement targets of metrics (e.g. whole system, sub systems, or files)
- frequency of measurement

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

– timing of offering the software tag to the purchaser (e.g. every week, every month, or the end of respective process)

**Table 1**. Software Tag Standard 1.0

| Classification | Category | No. | **Tag Element** | Explanation |
|---|---|---|---|---|
| Project Information | Basic Information | 1 | **Project Name** | Unique name of project |
| | | 2 | **Organization** | Information of development organization |
| | | 3 | **Project Information** | Information needed to identify the project characteristics |
| | | 4 | **Customer Information** | Information identifying the purchaser or owner |
| | System Information | 5 | **System Configuration** | Information identifying system configuration to label the type of system |
| | | 6 | **System Scale** | Development system scale |
| | Development Information | 7 | **Development Approach** | Development process type or techniques |
| | | 8 | **Organizational Structure** | Structure of development organization |
| | | 9 | **Project Duration** | Information of development length |
| | Project Organization | 10 | **Super-Project Information** | Name of super project which creates this project |
| | | 11 | **Sub-Project Information** | Name of sub projects which is created by this project |
| | Other | 12 | **Special Notes** | Other necessary or useful data for interpreting or analyzing tag data |
| Progress Information | Requirements | 13 | **User Hearing Information** | Information of user-requirements hearing |
| | | 14 | **Scale** | Amount of requirements |
| | | 15 | **Revisions** | Amount of changed requirement |
| | Design | 16 | **Scale** | Amount of design products |
| | | 17 | **Revisions** | Amount of changed design |
| | | 18 | **Design Coverage by Requirements** | Implementation ratio of design for requirements |
| | Programming | 19 | **Scale** | Amount of programming products |
| | | 20 | **Revisions** | Amount of changed programs |
| | | 21 | **Complexity** | Complexity of programs |
| | Test | 22 | **Scale** | Amount of testing |
| | | 23 | **Revisions** | Amount of changed test |
| | | 24 | **Density** | Ratio of test to system size |
| | | 25 | **Progress Status** | Test progress to plan |
| | Quality | 26 | **Review Status** | Quantity information of review |
| | | 27 | **Review Density** | Ratio of review to system size |

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

**Table 1.** Software Tag Standard 1.0 (*continued*)

| Classification | Category | No. | **Tag Element** | Explanation |
|---|---|---|---|---|
| Progress Information | Quality | 28 | **Review Effectiveness** | Ration of found defects to amount of review |
| | | 29 | **Defect Count** | Number of defects found by test |
| | | 30 | **Fixed Defect Count** | Number of fixed defects |
| | | 31 | **Defect Density** | Ratio of defects to system size |
| | | 32 | **Defect Detection Rate** | Ratio of detected defects to consumed test |
| | | 33 | **Static Check Results** | Report of static checker |
| | Development Cost | 34 | **Overall Cost** | Development and maintenance cost |
| | | 35 | **Productivity** | Ratio of amount of products to overall cost |
| | Schedule and Management | 36 | **Process Management** | Information on management of development process |
| | | 37 | **Purchaser-Developer Meeting Status** | Amount of user-vendor communication |
| | | 38 | **Total Risk Item Count** | Number of risk items in the development |
| | | 39 | **Risk Item Existence Period** | Time length between a risk item creation and deletion |
| | Other Products | 40 | **Scale** | Amount of product metrics not listed above |
| | | 41 | **Revisions** | Amount of change in products not listed above |

In this standard, we have included various kinds of information that are considered important to the purchasers. The overall structure should be simple for the purchaser to understand, so we have tried to keep it as simple as possible. Also, we have tried to keep in mind the balance of the tag elements. This standard does not include tag elements that are computable from other tag elements. There are a number of standards and reports such as SWEBOK, CMMI, ISO/IEC 15939, and reports by the Software Engineering Center in Japan (SEC) which can help interpret the tag elements.

The definition process was based on discussions with industry and academic collaborators such as:

**Purchasers**: Tokyo Stock Exchange, Japan Aerospace Exploration Agency, DENSO.
**Developers**: Fujitsu Lab, Hitachi, NEC, SHARP, SRA Key-Tech Lab, Toshiba, NTT Data.
**Others**: Information Technology Promotion Agency, Ministry of Economy, Trade and Industry, Japan (IPA), Nara Institute of Science and Technology, Osaka University.

Thorough the discussion, we recognized that appropriate metrics (tag elements) set and calculation methods of them are different for organizations or projects. Therefore,

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

we made tag elements selective, and on the tag standard, calculation methods of corresponding metrics of tag elements were not included but examples of the metrics are included.

To store, exchange and reuse the software tag, a standard data format is needed. So we settled the draft of the *standard software tag format* which is based on XML format, and are making the tool which converts existing tools' data into standard software tag format data. Software tag support tools explained in the next section treats software tag format data.

### 3.2 Support Tools

We are developing various support tools to promote the software tags scheme. In this paper, we introduce three essential tool prototypes that have been created for planning, collection, and visualization of the software tag.

**Software tag planning tool (*TagPlanner*)**

TagPlanner supports planning software tag data collection. With TagPlanner, users such as project managers can fix tag data definition and its structure easily before starting a software project. Each tag element is connected to a project's task, and users can browse structure of the tasks and details of tag elements by TagPlanner. With TagPlanner, users can see how to collect tag elements. TagPlanner has a typical example of project's tasks and tag metrics, and which can be edited by uses.

Fig. 2 is a screenshot of TagPlanner. Details of functions of each pane are described below.

**Process pane**: Using some process model, this pane exhibits standard process of an organization. In the figure, the process is shown by WBS (work breakdown structure).
**Task pane**: This pane presents tasks selected at the process pane and metrics related to the tasks. When a metric is clicked, measurement method and other information are shown in the detail information pane.
**Detail information pane**: This pane shows how to measure and analysis a metric, person in charge of measurement, and other information. Information on the pane is updated by operating on the task pane or the tag element pane. Frequency of measurement and metrics included in the software tag is settled on the pane.
**Tag element pane**: Tag elements are listed on the pane. Metrics used to compute tag elements are also listed.

The software tag data plan made by TagPlanner is saved as the standard software tag format explained in section 3.1. Software tag plan made by TagPlanner is useful when the purchaser and the developer agree to which tag elements are used. TagPlanner is also useful as the guideline of tag data collection and tag elements selection by referring the typical example of project's tasks and metrics for tag elements.

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

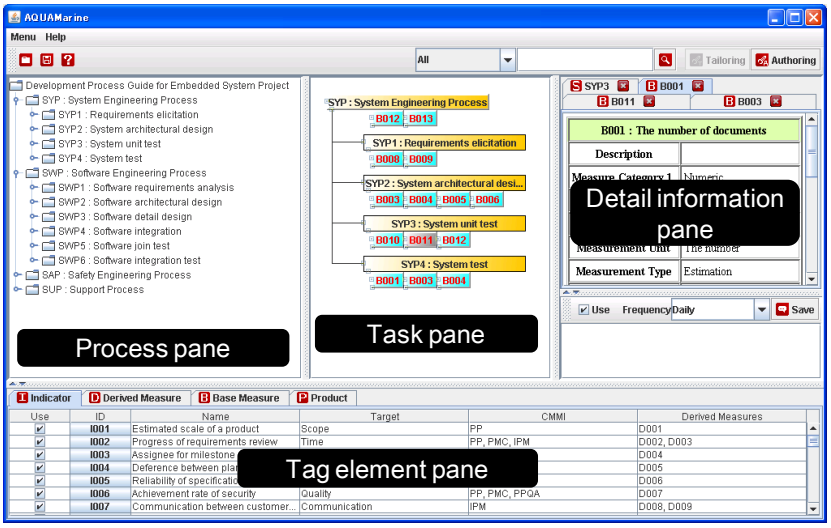


Fig. 2. TagPlanner Screenshot

**Software tag data collection tool (*CollectTag*)**

CollectTag supports collection of empirical data from software projects and creation of a software tag. CollectTag uses a wizard as a user interface, allowing the user (developer) to easily input the necessary data for the software tag. For each project, the purchasers and developers determine the metrics for the tag elements. To provide generality, we implemented CollectTag as a translator that converts a set of empirical data provided by the developer into the standard software tag format. That is, the developer periodically inputs values for each tag element, and then CollectTag outputs a software tag.

First, a user selects a tag element, and settles a metric for the element. For example, when [Programming]-[Scale] (#19) is selected, the user can select [Lines of code] or [Function point] (Fig. 3).

Next, the user selects data input method (Fig. 4). To reduce the effort of data input, CollectTag provides automatic data collection mechanisms for 11 of the tag elements in the progress information, if the target project uses common software development tools for configuration management and bug tracking. For example, LOC (#19: Scale) and CK (#21: Complexity) metrics [3] can be automatically collected and calculated from configuration management tools such as CVS or Subversion. When data is input manually, empirical data should be related to the data.

Finally, CollectTag generates the software tag elements in standard software tag format. This makes it easy to provide the output to other visualization and analysis tools for further processing.

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

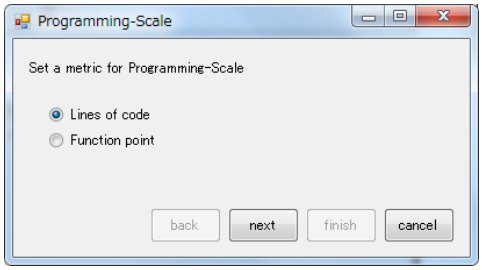


Fig. 3. Selecting a Metric (CollectTag)

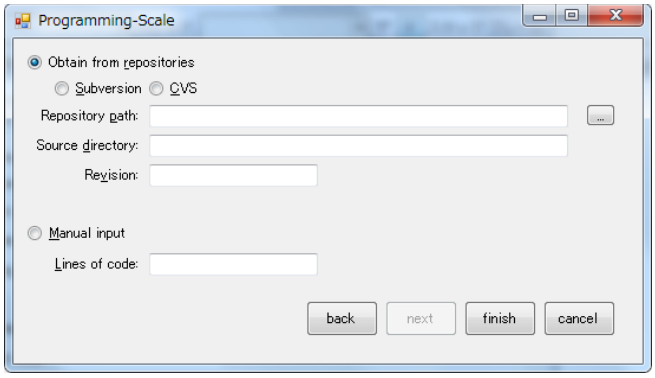


Fig. 4. Selecting an Input Method (CollectTag)

**Software tag visualization tool (*TagReplayer*)**

TagReplayer provides fundamental features for integrated visualization of various historical data included in standard software tag format data. TagReplayer employs the metaphor of video player manipulation for its user interface so that users can replay the progress of the project just like watching video on TV. Users can also instantly recall the details of any points along the timeline based on the software tag. TagReplayer aligns progress information from the software tag as a series of events.

Fig. 5 is a screenshot of TagReplayer. Details of TagReplayer are explained below.

**Time bar**: The time bar indicates a point of time which TagReplayer replays a project. By moving slider, replay goes to a certain point of time. Replay interval can be changed, and stopping, fast forward or fast rewind of replay are also available.

**Event list view**: On the event list view, tag data is listed by time series. A user knows events happened on a certain day from the view.

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

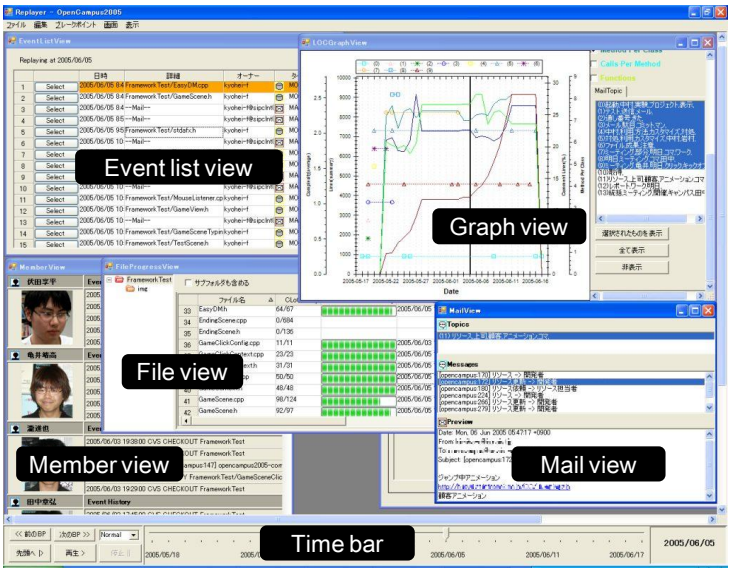


Fig. 5. TagReplayer Screenshot

**Graph view**: The graph view shows transition of lines of code (LOC) during a project using line graph. Moreover, topics made from natural text mining and clustering of mail archive are overlaid on the chart, and it helps understanding what was happened on the project.
**Member view**: The member view displays assigned tasks and completed task history of each project member. A user also sees workload of each member from the view.
**File view**: The file view indicates completion rate of each file at time point of replaying. The completion rate is computed using cumulative changed LOC at the end of the project. Using the view, a user recognizes dilatory files.
**Mail view**: The mail view shows detail of topics shown in the graph view. The view includes topics list, mail subject list, and a mail body.

Also, TagReplayer has the breakpoint function which stops replay if a certain condition is true, and the function which shows source code at time point of replaying.

To confirm effectiveness of TagReplayer, software development process in which some students engaged was replayed to subjects who did not engaged in the project, using TagReplayer. As a result, subjects recognized stagnant period and the reason of stagnant. This experience shows that TagReplayer is very useful for postmortem project reviews.

### 3.3 Applications of the Software Tag

We present here three example cases of application of the software tags scheme to real software projects.

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

- A course registration system for a university with 26K LOC in Java was developed for five months by a medium-sized software company in Japan. 32 elements of tag data were collected, and we analyzed data to know the project status. Comparing test density (#24) and defect density (#31) with the publicly available benchmark values from the Software Engineering Center in Japan (SEC), they are lower than the benchmark value. It means that using the software tag, the purchasers and developers see the probability of insufficient testing density. Also, we analyzed transition of total amount of source code (#19; programming scale), modified amount of source code (#20; programming revisions), and static check results (#33). They varied at a certain day during test phase, and from a commit comment on a SCM, refactoring was done on the day. So, such tag elements are useful to know characteristic event in the project.
- A medium-sized stock exchange system for a stock market was enhanced by a Japanese major software development company for more than two years. Projects data is offered from the company to us, and we made the software tags from it. We speculated project status based on the analysis of them, and interviewed the company to confirm actual status of the projects. According to the results, we concluded that comparing tag elements related to the requirements phase such as requirements revisions (#15), number of defects about design (#29; defect count), and number of review (#26; review status) between functions, the purchaser and developer were able to identify problems with the requirements completeness caused by frequent changes.
- A Japanese software development company ordered several small-sized projects such as development of a project management support system from various offshore companies in China and Korea. Three finished projects data is offered from the company to us. We analyzed it and interviewed the company to confirmed actual status of the projects. As a result, we concluded that although remotely located from each other, the purchasers and developers could understand the progress of the specifications comparing tag elements such as the number of review (#26; review status), the number of user hearing (#13; user hearing information), and number of defects about design (#29; defect count) between functions.

## 4. Activities for Promotion and Diffusion

The StagE project is also actively promoting and diffusing the software tags scheme in industry as follows.

**International/Domestic Standardization**

Interviews with several Japanese software purchasers and developers, along with offshore software developers for Japanese companies in some countries, convinced us that most software purchasers and developers would strongly demand that the software tag and tools should be international and/or domestic technical standards in software engineering. To support this, we are now serving ISO as a committee member of the working group on process assessment, ISO/ITC JTC1/SC7/WG10. We are also work-

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

ing with some software tool developers to construct a de facto standard software project management system that includes the tag support tools mentioned in Sec. 3.2.

**International Collaboration**
Offshore software development is one of the most useful application areas of the software tag scheme. To encourage and accelerate international collaboration to have various kinds of case studies and experiments of the software tag in offshore software development, we established Asia-Pacific Software Engineering Research Network (APSERN) in 2008 with software engineering researchers in NICTA (National ICT Australia), ISCAS (The Institute of Software, Chinese Academy of Sciences), and so on.

**Professional discussion of Legal issues**
In the case of a legal dispute between software purchasers and developers, the software tag can clarify their liabilities and has the potential to help resolve such legal issues in software development. The StagE project has a committee examining the legal issues of software development. Members of this committee include lawyers, patent attorneys, and software engineers. The committee has interviewed many software developers in Japan and China to compile data about troubles between software purchasers and developers. It also distributed questionnaires to more than a hundred software developers in Japan to analyze the trends of such troubles. The software tag provides an opportunity for collaboration between software engineering and software trade law.

## 5. Discussion

Discussions about the software tag scheme are described below.

- There are metrics repositories aimed at improving and benchmarking development organizations [6], along with some software measurement paradigms [2], [4], [8], [9]. Also, there are projects which involve some companies and are now coping with establishment of software quality [12] [13]. However, the software tag provides a unique approach to involve software purchasers in the quality improvement framework by providing development transparency. As far as we know, there is no similar approach presented in the technical literature.
- We believe that the benefits of the software tag scheme for software purchasers will be substantial because the development processes and the developed products become more visible and understandable. However, purchasers will need to collaborate more closely with developers, providing effort and enthusiasm to create successful projects.
- We have presented the first standard of the software tag with 41 elements. In some sense, these are very basic data for indicating development quality, and they may be insufficient to perform detailed analysis. However, as a standard used for various software development projects, the set should be minimal and low cost. As presented in Sec. 3.1 and 3.2, our tag standard 1.0 is a lightweight set with low collec-

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

tion and assembly cost. It is important to continue practical applications of the software tag, and to get feedback for further improvement of the standard.

- Some developers disclose information whose role is similar to the software tag with the progress report meeting. The progress report meeting and software tag bring similar effects. Metrics used in the activity may be similar to tag elements, so tag elements are considered to be not uncommon. Still the software tag is effective to propagate such a good practice involved the purchaser.
- The role of the software tag is similar to the medical checkup and financial statements. On the medical checkup, many bodily data are collected and evaluated, and the results are shown to the person to see his/her health condition. Financial statements which are used for settlement of accounts indicate amount and flow of capital by various money amounts. They are disclosed investors and business partners to exhibit soundness of the company. In the similar way, the software tag discloses various project and product metrics to the purchaser to signify soundness of process and products.
- Although there may be the risk of tampering software tag data, it is difficult to tamper several tag elements with keeping consistency through some versions of software tags. On the other hand, it is not difficult to rebuild the software tag from stored source data. Hence, tampering software tag data would not get along. It may be a good way that a third party stores source data of the software tag, and verify correctness of the software tag when conflict occurs. Also, the third party may analyze data to guarantee independence of the evaluation and reliability of the results.
- Software tag standard 1.0 does not include concrete metrics. However, it would be not easy to settle metrics for many tag elements from scratch. A catalog of typical metrics set for software tag elements should be made to support planning software tag in the future. The catalog will be organized by the purpose, and explain how to collect and analyze tag elements. The catalog will be browsed on TagPlanner (see Sec. 3.2). With analyze methods and benchmarks on the catalogue, purchasers csn confirm validity of evaluation of software tags to some extent.

## 6. Conclusions

We have introduced our activities for standardization of the software tag in Japan. The software tag contains software development data, and it brings purchasers transparency of software development. We identified 41 items for seeing software process and products, and defined them as the standard tag element set. To support software tag scheme, we made tools for planning, collecting, and analyzing tag data. From three example cases of application of the software tags scheme, it is expected that the software tag scheme is useful to find problems of requirement analysis or to grasp progress of offshore software development.

Through these activities, the concept of the software tag is becoming well understood in Japan. From discussion with purchasers and developers, we think that their interest toward development data sharing gets higher than before, and they seem to be realizing how to use the software tag in detail. It appears that for purchasers and de-

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

velopers who have software development management skill, the software tag scheme has small disadvantage but has possibility of big advantage.

Our future work will focus on making international/domestic standards of the software tag. With such standardization, the software tag is expected to be used in various software industries, where we think it will strongly promote participation and understanding of software development by purchasers. Also, to reduce adaptation cost of the software tag, we will delivery software tag support tools and the software tag guidebook which explains how to use the software tag. That would accelerate incorporating the software tag scheme in the industrial practices. Moreover, we will make a template of the contract document, considering software tag and legal issues of software development.

## Acknowledgments

This work is being conducted as a part of the StagE project, The Development of Next-Generation IT Infrastructure, supported by the Ministry of Education, Culture, Sports, Science and Technology. We are grateful to the members of the StagE project, especially to Michael Barker, Akito Monden, Makoto Matsushita, and Shuji Morisaki.

## References

1. Barker, M., Matsumoto, M., and Inoue, K.: Putting a TAG on Software: Purchaser-Centered Software Engineering. In Handbook of Research on Software Engineering and Productivity Technologies: Implications of Globalization, Ramachandran, M., and Carvalho R. A., Eds. Information Science Reference, Hershey, PA (2009) 38-48
2. Basili, V.R., and Rombach, H.D.: The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Trans. Software Eng. 14, 6 (1988) 758-773
3. Chidamber, S. R., and Kemerer, C. F.: A Metrics Suite for Object Oriented Design. IEEE Trans. Software Eng. 20, 6 (1994) 476-493
4. Chirinos, L., Losavio, F., and Bøegh, J.: Characterizing a data model for software measurement. Journal of Systems and Software, 74, 2 (2005) 207-226
5. Inoue, K.: Software Tag for Traceability and Transparency of Maintenance. In Proceedings of 24th IEEE International Conference on Software Maintenance ICSM 2008 (2008) 476-477
6. International Software Benchmarking Standards Group (ISBSG): ISBSG Estimating: Benchmarking and research suite (2004)
7. ISO/IEC 12207: Systems and software engineering - Software life cycle processes. International Organization for Standardization (2008)
8. ISO/IEC 15939: Software engineering - Software measurement process framework. International Organization for Standardization (2002)
9. Kitchenham, B. A. Hughes, R. T., and Linkman, S. G.: Modeling Software Measurement Data. IEEE Trans. Software Eng. 27, 9 (2001) 788-804
10. Nikkei Business Publications, Inc.: Survey Report on IT Investment and CTO in Japan. Nikkei Information Strategy. March (in Japanese) (2008)

*M. Ali Babar, M. Vierimaa, and M. Oivo (Eds.), International Conference on Product Focused Software Development and Process Improvement (PROFES 2010), LNCS 6156, pp.220-233, June 2010 (Limerick, Ireland)*

11. Nikkei Business Publications, Inc.: Second Survey of Japanese Software Projects. Nikkei Computer. Dec. 1, 36-53 (in Japanese) (2008)
12. The Consortium for IT Software Quality: CISQ – The Global Standard for IT Software Quality. http://www.it-cisq.org/
13. The Quamoco Consortium: Quamoco – The Benchmark for Software Quality. https://quamoco.in.tum.de/wordpress/?lang=en