A Case-study on Using an Automated In-process Software Engineering Measurement and Analysis System in an Industrial Environment

Irina Diana Coman, Alberto Sillitti, Giancarlo Succi
Free University of Bozen-Bolzano
{IrinaDiana.Coman, Alberto.Sillitti, Giancarlo.Succi}@unibz.it

Abstract

Automated systems for measurement and analysis are not adopted on a large scale in companies, despite the opportunities they offer. The fear of the "Big Brother" and the lack of reports giving insights into the real adoption process and concrete usages in industry are barriers to this adoption. We report on a case-study on the adoption and long-term usage (2 years of running system) of such a system in a company focusing on the adoption process and the related challenges we encountered.

1. Introduction

The automation of data collection is a key requirement for the success of software measurement programs [23], [24], [26], [27]. The traditional, manual data collection is time consuming, tedious, error prone and often biased or delayed [1]. Semi-automated data collection (tools such as LEAP [2]) still poses the context switching problem [3]: the semi-automated data collection has negative impacts on the performance of the developers as it requires them to switch continuously between working and recording the work.

To eliminate the context-switching problem, a new generation of tools (such as Hackystat [3] and PROM [4]) focuses on fully automated, non-invasive data collection. Because they allow data collected from on-going projects to be used for improvement of the same project, these tools are also called Automated In-process Software Engineering Measurement and Analysis (AISEMA) systems.

AISEMA systems aim at automatically collecting the data, but also at providing tailored analyses for decision support. They reduce the cost of data collection, as they run in the background and let people focus on their work without any additional workload or distractions. They can collect a large variety of data. Based on these data, they propose: support for process management via software project telemetry [5], assessment of low-level processes such as Test Driven Development (TDD) [6], etc.

Although AISEMA systems have been around already for several years, they are still not adopted on a large scale. Among the commonly cited barriers for industry adoption are the misuse of the data and the privacy issues. Moreover, the very diversity of data collected and analyses proposed can be intimidating as there is a lack of clear understanding on which data would be useful in a real situation.

Case studies focused on the usage of AISEMA systems in industry can help overcome these barriers as they offer concrete examples of benefits, drawbacks, and strategies for overcoming challenges. However, currently, few such reports are available.

In this paper, we present a case study of adoption and usage of an AISEMA system in the software department of a large Italian company. While we do not claim generality of our findings, we consider that they can contribute to further adoption in industry of AISEMA systems by offering insights into the adoption and usage of such systems.

The paper is structured as follows: section 2 discusses the related work; sections 3 and 4 present the settings of the case study and our experience; section 5 discusses the main lessons learnt; finally, section 6 draws the conclusions and identifies directions for future work.

2. Related Work

2.1. Establishing software measurement programs in companies

Most reports on establishing software measurement programs in companies stress the need for automation of data collection as a key requirement for the success of a metrics program [23], [24], [25], [26], [27], [28].

The AISEMA systems started as an attempt to address this need and evolved to automate as much as possible of a software measurement program. Thus, the adoption and usage of AISEMA systems in companies is closely related to establishing a software measurement program.

It is generally agreed that establishing a measurement program is a complicated and risky undertaking with two out of three measurement efforts failing or discontinuing after two years [24]. From the reported experiences with measurement programs [23], [24], [25], [26], [27], [28], [29], we identified several factors agreed upon as success factors for measurement programs:

- **Incremental approach to data being collected**. The establishment of a measurement program should start with the collection of the data that are needed to address the most important goals of the company and that are agreed-upon by all those involved. As the measurement program evolves, more data are collected to address other issues. The incremental approach is considered needed mainly to minimize the burden of data collection.
- **Incremental approach to analyses presented**. It is easier for people to understand first a reduced set of metrics and analyses than to try to handle right from the start a large set. Moreover, as people get familiar with these metrics and benefit from them, they are increasingly able to see and propose new metrics and new analyses for even increased benefits. This is seen mainly as a consequence of the lack of experience of people in measuring and being measured.
- **Automated data collection**. The collection of data should be automated whenever possible. This should increase the validity of data and decrease the overhead of a measurement program, thus decreasing developers' resistance to measurement. Additionally, the capture of the context of the collected data should also be automated.
- **Training**. Adequate training should be provided to the people involved in the measurement program at various levels. Training can range from raising awareness of metrics collected and purpose of measurement to analysis techniques.
- **Developer involvement**. As the developers are usually the ones that collect metrics, their involvement is an important factor to the success of a metrics program. They should have a good understanding of what measures are collected and for what purpose. Additionally, they should be involved in the actual designing of the metrics program.
- **Accuracy of data**. The accuracy of data directly influences the outcome of the measurement program. Thus, accurate data is a prerequisite for the success of any measurement program.
- **Integrity of data and analyses**. The people involved need to have confidence in the integrity of the data and analyses that are presented. Regardless of the actual facts, a perception of "tinkered" data can compromise a measurement program.
- **Usefulness**. The data collected should be useful and its usefulness should be obvious to all participants.
- **Prompt feedback**. All people collecting data need to know how the data are used. This increases the chances that they view positively the measurement program.
- **Dedicated team for measurement**. There should be a team of several individuals that have the explicit responsibility of the measurement program. However, there is not a general agreement on whether this should be the sole activity of the team or not.
- **No usage of data for people assessment.** Data should be used to understand and improve, not to assess people [23].
- **High frequency of data collection.** The frequency with which metrics are collected and made available has positive and significant influence on their use in decision making [28].

Due to the close relation between adoption of AISEMA systems and establishment of a software measurement program, the success factors for the two are also likely to be related. Several of the above success factors are met by intrinsic properties of AISEMA systems: automated data collection, accuracy of data and high frequency of data collection. In section 5, we relate our lessons learnt on the adoption of an AISEMA system to the above success factors.

2.2 Existing automated measurement systems

Most of the existing automated measurement systems belong to one of the following categories:

- **Timesheet software systems**. Examples of such systems are MetriQ[1] Rescue Time[2], SLife[3] TimeSnapper[4], Web TimeSheet[5] or TrackTime[6]. These systems are mainly concerned with recording the total time spent on tasks or general activities such as playing music or navigating the Internet. Their primary intended usage is for reporting and generating

[1] http://www.metriq.biz
[2] http://rescuetime.com
[3] http://www.slifelabs.com
[4] http://www.timesnapper.com
[5] http://www.replicon.com
[6] http://www.mamooba.com/TrackTime

invoices. They are not specifically tailored for software development as they do not go into the details of specific development activities.

- **Automated in-process software engineering measurement and analysis (AISEMA) systems**. Examples of such systems are Hackystat [3], PROM [4], 6th Sense Analytics [7], EPM [8], ECG [9], and SUMS [10]. AISEMA systems are specifically tailored for software development and they usually collect code metrics as well as process data. They are non-intrusive and propose a large variety of analyses and reports.

AISEMA systems adopt mainly two types of approaches to data collection: generic (the data collection is independent from the tools that the developers use) and specific (the data collection depends on the tools that the developers use).

Hackystat [3] collects data through "sensors", one for each software tool from which data are collected. Currently, there are sensors for several IDEs, testing frameworks, build tools, configuration management systems, static analysis tools, issue tracking systems, and word processing tools. The tool also collects size metrics for various programming languages.

PRO Metrics (PROM) [4] collects both specific and generic data. Currently, it has plug-ins for several IDEs, word processing tools, email clients, and issue tracking systems. PROM collects also object-oriented and procedural metrics for various programming languages (e.g., C/C++, Java, C#, etc.).

6th Sense Analytics is based on Hackystat sensors and focuses mainly on 3 measures: Active Time, Churn (lines of code added or deleted), and Flow Time. EPM focuses on integrating data from issue tracking systems, code repositories, and email servers. ECG is still a prototype aimed at studying programmer behavior and identifying various types of episodes (such as copy-paste-change). SUMS focuses on collecting data about development for high performance computing.

2.3 Reported usages of AISEMA systems

Existing reports on the usage of automated measurement systems are set mainly in laboratory environments and provide examples of usage. Such studies are a good starting point for understanding a specific automated measurement system. However, they do not uncover challenges that are specific to real-world settings and to specific problems.

The studies on Hackystat report on its usage during the development of the system itself [5], during an university course [11], and in a laboratory experiment performed to validate automated detection of TDD [6].

Ohira *et al.* [8] reported on the usage of EPM during the development of another of their projects. Nystrom *et al.* [10] present some preliminary analyses done with SUMS on data collected during a workshop, and during a laboratory experiment with undergraduate students.

Previous studies involving PROM report on analyses performed with the collected data, rather than focusing on the adoption process. Rossi *et al.* [12] use data collected with PROM in a public administration to study the transition from Microsoft Office to OpenOffice.org. Coman *et al.* use data collected with PROM in a small company to explore an approach to enhance the system with automated inference of higher-level information [13] or to evaluate claimed benefits of PP (Pair-Programming: two developers working at the same computer) [18]. Moser *et al.* use data collected with PROM in small companies to analyze the effect of refactoring on maintenance [19], quality [20], effort [21], and reuse [22].

3. Research settings

This work is the first analysis of the adoption and usage for an extensive period (33 months) of an AISEMA system in an industry environment. The system used is PROM [4].

3.1 Data

In the company under study, the system collected mainly three types of data: code metrics, issue tracking data, and developer activity information.

The code metrics consisted in the object-oriented metrics set proposed by Chidamber and Kemerer [17]. A component of the system connected to the code repository and computed these metrics on a daily basis. Another component connected to the issue tracking system and retrieved information on the status of all the issues on a daily basis.

The developer activity information consisted mainly in "events" of interaction between developer and computer. Generic data consisted in the application in use and the title of the focused window. Specific data consisted in detailed information from the IDEs. All the data had granularity of 1 second. Additionally, developers could also manually introduce data regarding non-computer activities or PP sessions.

3.3 Team and company

The team is part of the IT department of a large Italian company. The team works mainly on developing and maintaining custom software that is needed by other departments of the company. Due to the sensitive nature of the data presented here, the company remains anonymous.

The software team is composed of several regular developers, one senior developer acting as the leader of the team and one manager of the IT department. This manager is responsible for the IT department in front of a higher-level, non-IT manager. During our study, the team has grown from 10 to 20 regular developers.

The regular developers in the team are all Italian, between 35 and 40 years old. There is only one female in the team. They all hold university degrees in computer related fields and have from 10 to 15 years of programming experience.

The team works on several projects, mainly in C#. They are an Agile team [15] inside a non-agile organization. They use a customized version of Extreme Programming [14]. In particular, they use weekly iterations, user stories, pair-programming, and test-driven development.

The team works in a single large room where each member has his own personal space. Therefore, informal communication between the developers can occur easily.

3.4 Goals of the introduction of PROM

The manager and leader of the team were the initial promoters of the adoption of an automated measurement system. They had two major goals:

- improve the process of the team.
- have a more objective and quantifiable way of presenting the activities of the team to the upper-level, non IT management of the company. Since the company is not an IT producer, the IT department has often been perceived as a “cost” for the company

During the adoption and usage of the system, other goals became clear. These are presented in section 4.6.

Among the existing AISEMA systems, only Hackystat and PROM collected the large variety of data that was needed to address the goals of the company. Due to lack of space, this paper does not include a comparison of the two systems, but such a comparison can be found in [4].

The main reasons for which the company chose PROM are: availability of experts throughout the adoption process, flexibility to use already existing data, and possibility to extend data collection to address specific goals of the company.

3.5 Adoption process

For the adoption of the system in the company, we designed a plan in 5 steps, taking into account the known factors of success for establishment of measurement programs. The 5 steps of this plan are:

1. **Planning**. During this phase, we plan to refine the goals and to identify the required measurements. To accomplish this, we use the Goal Question Metric [16]. In this stage, the participants from the company are the leader and manager of the team as well as the higher-level manager.
2. **Training**. During this stage, we try to gain the support of the developers for using the proposed system. To accomplish this, we prepare tutorial movies showing the installation and usage of the system and we give several presentations on the system to the developers and their leader. The presentations cover the usage of the tool as well as detailed explanations on the data collected. Following the presentations, there are open discussions between us and the developers. All open issues raised during these discussions are addressed by the next round of presentations.
3. **Pilot deployment**. During this stage, we test the introduction of the system and the accuracy of data collection. The system is installed on the machines of only two developers. To assess the accuracy of the data, the two developers receive via email a daily summary of their own data. They provide us with feedback signaling any problems or suggestions regarding the data or their experience with the tool.
4. **Deployment**. The goal is to complete the deployment of the tool on all developers’ machines and to ensure the accuracy of the data collection. To check the accuracy of the data, the developers receive via email a daily summary of their own data. We receive their feedback on the data and on other issues related to the system.
5. **Usage**. In this stage, the system is fully functional collecting accurate data and providing value to the company via the requested reports and analyses.

4. Experience

As there are no established criteria for assessing the success of the adoption of an AISEMA system, it is hard to evaluate to which extent the adoption was successful in the company under study. Considering the two criteria for success of measurement programs (usage in decision making and impact on organization performance [29]), at this stage there are at least three signs of success. The analyses showing the total time spent waiting for the compilation of big projects triggered the (long waited for) higher-level management approval of new hardware. The analyses showing the time spent on various activities led the team manager to take into consideration dedicating one person to answer emails from users in order to decrease the high amounts of time that email was taking from all developers. Although at this stage there was no formal evaluation of the impact on organization performance, the AISEMA system was perceived by developers as having a positive impact on their performance especially because of automating the time consuming task of filling out the weekly activity reports required by the accounting department.

Given the above positive signs and the continued usage of the AISEMA system, we consider the adoption process in this case as successful. However, the phases of the adoption process itself sometimes took place differently from our initial plan. We start this section with a presentation and discussion of the different phases of the adoption process as they took place in the company under study. Then, we present the actual goals as they emerged during and after the usage of the system. Finally, we describe our experiences regarding the critical issue of data accuracy.

4.1 Planning

The initial planning phase took place before the two years of AISEMA system usage and was longer than expected (9 months). This was partially due to the busy calendars and different roles and backgrounds of the people involved (the higher-level manager, the team manager and the team leader).

The definition of the GQM and the identification of the means to collect the derived metrics required a significant amount of effort. The participants had to come to an agreement regarding the aspects to be used for assessing and improving the performance of the department. Our task was to try to understand all points of view and to propose a solution that would address them and would still be technically possible. At the end of this phase, we produced a formal document detailing the GQM and the mechanism for collecting all required metrics using PROM. The actual GQM is outside the scope of this paper and is not presented here due to lack of space.

Suitable bridges have been developed for PROM to collect server-side data from their corporate repositories of issues and version control systems. Altogether, the planning phase played different roles for those involved. It allowed us to get an initial, although limited, understanding of the environment and to get a perception of what would bring value to the company. It allowed the two managers and the team leader to clarify their goals and to get an initial concrete understanding of how the AISEMA system can actually help them achieve their goals. Finally, it gave us all a concrete, although preliminary, common understanding and starting point.

4.2 Training

The installation and usage of the client-side of the system has been on a voluntary basis. Each developer decided individually whether to install and use it.

During the training phase we tried to gain the developers' support for using the system. We started with presentations on the installation and usage of the system, followed by detailed explanations on the data collected and its usage. The presentations were followed by open discussions between us and the developers. During these discussions, developers were free to ask any questions and express any doubts about the system. After each discussion we analyzed the main concerns and identified ways to address them.

We made clear from the beginning the measures taken to ensure privacy and to eliminate data misuse: the data collection could be stopped temporarily or definitively, completely or only for some applications, at any moment, without needed explanations; the analyses based on personal data were available only to the one collecting it while the others (including managers) could see only analyses of aggregates of all data. However, this did not seem to address all concerns of developers. Following discussions with developers, we enhanced the system to allow the users to see their own data previously to sending it to the central server and delete it if desired.

Developers wanted to have a deep understanding of the inner functioning of the system. They wanted to know not only what data are collected, but also how they are collected, how they are handled and stored, how does the system handle special situations such as idle time, interruptions, or pair programming. Moreover, they wanted direct, unmediated (read-only) access to the database where data were finally stored.

Although it was a long, resource consuming phase, we consider that the training was crucial in gaining support and active participation from developers. The time between the presentations was also important as the developers did not feel under time pressure to take a decision. By getting all their questions answered and seeing that the issues they raise are addressed, the developers gained confidence in the system and at the end of this phase all of them agreed to use it.

4.3 Pilot deployment

The pilot deployment consisted in deploying the system on two developer machines. The system started collecting data immediately and the developers received every morning an email report containing the data collected the previous day. The report contained a breakdown of the time recorded on the various applications and on the various files or methods. We asked developers for feedback to ensure that the data collected were accurate.

During this phase, we uncovered only minor issues, mainly requests for improving the format of the daily report. We addressed these requests. As we did not uncover any other major data issues, we ended this phase after two weeks.

4.4 Deployment

Table 1. Improvement requests gathered in deployment phase.

Feature Description	Feature Type
Add a short summary at the beginning of the report. The details should follow.	Presentation
Show in the report the intervals of the day when data was recorded, rather than the total time recorded.	Presentation
Group the files and applications shown in the report by categories corresponding to higher-level activities.	Presentation
Recognize some frequently used external editors when used from within the IDE.	Internal
Collect and report also data on resource usage, namely the time required for compilation of various projects.	Internal
Automate time consuming tasks such as filling out weekly forms on developers' activity.	Internal
Allow regular expressions to define applications for which data are not collected.	Internal
Collect automatically data on meetings and appointments set in the Calendar in Outlook.	Internal
Allow manual entering of data corresponding to work on tasks.	Internal
Support also team-programming and integrate it with the issue tracking system.	Internal

Given the positive experience of the pilot deployment, we expected that the deployment phase would also be short and without major problems. In reality, it was very long (almost 9 months) and many issues surfaced. There were three main causes for this:

- The increased number of users and their increased experience with the system triggered a large number of requests for improvement, regarding both the presentation and the internal features of the system (Table 1). The longer usage revealed also several special cases that had not been previously taken into consideration. For example, while working remotely or on a virtual machine, the system recorded only a long "black-box" event of type remote or virtual respectively. Where possible, the system had to be installed also on the virtual machine or on the remotely accessed machine to gather the data at the same level of detail as on the physical machine of the developer.
- The changes to the process of the team (such as the introduction of extensive usage of pair- and team-programming) triggered new extensions to the system. The initial support for pair-programming needed to be extended with support for team-programming (more than 2 developers working together on a task). The support was also extended to retrieve the issues from the issue tracking system and to store, at the end of a session, the amount of work done together with a link to the user selected issue.
- Changes to the environment (such as newly installed extensions to the IDE or new company security policies) affected the already installed and working system. In some cases, there was a need of redeployment of some of the components.

To clearly understand and address more complex requests such as grouping of files and applications by activities, we took an iterative approach. We proposed a solution, showed it to the developers, and then used their feedback to propose an improved solution. Although effective, this approach also extended the duration of the deployment phase.

4.5 Usage

This phase started when the system started providing value to the company. The first delivered value was a reliable daily and weekly report of activity delivered to each developer. This was in fact the upgraded version of the daily report that served initially for checking data accuracy.

The upgraded daily report contained a summary of time spent in solo and pair-programming (together with the names of partners), the time spent on pair- or team-programming (broken down by task and partners), and the compilation time broken down by project. The weekly report contained the same information, but aggregated for each day of the past week. The daily report contained additionally detailed information on the time spent on various files and applications.

In its new format, the developers considered the reports as reliable, objective summaries of the work done during the previous day or week. They relied on the reports for the weekly stand-up meetings and for filling out the weekly activity reports required by the accounting department of the company.

The next step was to upgrade the system to provide also directly the activity report in the form required by the accountancy department.

Aggregated versions of these reports, showing the usage of various software tools, the time needed for compiling the projects, and the time spent by the team on various tasks were available for the leader and manager of the team. They offered them an overview of the activities and of the resource usages.

The system also provided value to the company by showing the evolution of the metrics agreed upon during the planning phase. Metrics were gradually added as some of them required integrating data that the company was already collecting by other means or extending the system to collect additional data.

Once most of these metrics were available, the manager and the team leader realized that there are too many metrics to be evaluated at a glance. Therefore, they would need a reduced set of metrics (from 5 to 8) that would inform them on the status of the projects and of the process. However, this reduced set should contain enough information to warn them of potential problems. In case of such warnings, they would rely on the detailed set of metrics to better understand and localize the problem and to take decisions.

4.6 Goals of PROM usage

During the actual usage of the system, we received extensive feedback from the developers, the team leader and the manager of the team. It soon became obvious that the different people involved had in fact different goals for using the AISEMA system, depending on their role in the team.

The developers wanted to automate time consuming tasks such as filling out time sheets at the end of each week. They also were interested in having a reliable, objective view of self-performance.

The team leader needed to evaluate the status of the process and of various projects at any time. He also wanted to drive the improvement of his team based on objective measurements. He needed global evaluations of the process and of each project, and detailed evaluations of the aspects that needed improvement.

The manager had two main goals: to evaluate the work product and to make the IT department’s effort visible to the non IT, higher-level, manager. To do so, he needed an adequate, tangible, objective way of evaluating the quality of the software produced and the performance of the IT department. The evaluation of quality and performance had also to be global rather than focusing on a single specific aspect.

Although the goals were different, all people involved had a common requirement about the metrics presented: brevity. They needed to be able to understand at a glance all the metrics corresponding to a goal. Thus, even when wanting a global evaluation that takes all aspects into account, they preferred a short, aggregated result to a long, detailed one. This aspect seemed to be crucial for metrics usage. In cases where the aggregation was not obvious, they preferred to give up some precision by ignoring some aspects.

4.7 Data accuracy

The data accuracy proved to be an issue that needed constant monitoring during all the usage of the system. Even though an initial assessment ensured the accuracy of the data collection, the experience showed that the situation can alter at any time. The main reasons for the alteration of the data accuracy were the following:

- Software failures. The existing software systems are fragile. The repeated crashes of the IDE on some of the developers’ machines (while they were testing problematic pieces of code) triggered the deactivation of the plug-in for data collection.
- Changes to the software systems. The upgrades of various tools interfered sometimes with the data collection. Some upgrades of the IDE or the installation of new plug-ins for the IDE disabled the plug-in for data collection. The change of the security policy of the company also affected temporarily the transfer of the data to the central server.
- Hardware failures. Although infrequent, it happened that some data were lost due to the hard-drive failure. The data temporarily stored locally on a developer’s machine have been corrupted and were lost.

In most of the cases, a simple reinstallation or restart of the components affected was enough to solve the problem. However, precious time was lost in recognizing that there is a problem and in identifying its source.

5. Lessons learnt

The experience gained can be summarized in 6 lessons learnt that are presented in detail in the following paragraphs. Table 2 relates our experience to the factors of success reported in the literature (discussed previously in section 2.1) for establishment of traditional measurement programs which do not use AISEMA systems. We added to the existing factors, three factors specific for AISEMA systems.

Table 2. Success factors in establishing traditional measurement programs and in the adoption of AISEMA systems.

Factors of success	**Traditional measurement programs**	**AISEMA systems**
Incremental approach to data being collected	Needed to minimize burden of data collection.	*Not needed, as the collection of additional data does not increase the burden on developers.*
Incremental approach to analyses presented	Needed to let people accommodate.	Needed to let people accommodate.
Automated data collection	Most programs use manually or partially automated data collection.	It is *intrinsically ensured.*
Training	People need training on purpose of measurement, metrics collected, and *analysis techniques.*	People need training on purpose of measurement, metrics collected, *storage*, and *inner functioning of the AISEMA system.*
Developer involvement	Developers should be involved in the design of the metrics program.	Developers should be involved in the design of the metrics program and of the *functionalities of the system.*
Accuracy of data	Accuracy can be ensured through *data collection discipline* and *constant checking.*	Accuracy can be ensured through *self-healing* and *self-monitoring capabilities of the AISEMA system.*
Integrity of data and analyses	Confidence in the integrity of data and analysis *cannot be automatically ensured.*	Confidence in the integrity of data *can be automatically ensured* by giving developers access to the database and knowledge on how data are collected, accessed and treated.
Usefulness	The metrics collected should be useful.	The metrics collected should be useful.
Prompt feedback	*Cannot be automatically ensured.*	*Can be automatically ensured* through daily and/or weekly reports.
Dedicated team for measurement	There is a need for *a team with the explicit responsibility* for the measurement program.	*Initially an experienced user of the system is needed* to design the analyses; *once designed, the analyses are automated.*
No usage of data for people assessment	*Cannot be automatically ensured.*	Can be *partially ensured automatically* through data privacy and control over one's own data.
High frequency of data collection	It requires automated data collection or *increased costs due to more work.*	It is *intrinsically ensured.*
Simplicity, clarity, and brevity in presentation	No reports available.	*The presentation of data has to be simple, clear and short (5-8 metrics) summarizing the status and warning about potential problems.*
Different data aggregations	No reports available.	*Different aggregations of metrics are needed for different goals.*
Integration of data	No reports available.	*The data from external sources should be integrated.*

L1. It is important to gain support from developers

An usually cited barrier in the adoption of automated measurement systems in companies is the lack of cooperation from developers. This is usually attributed to fears of data misuse or the fear of the "Big Brother". Our experience shows however that these barriers can be overcome by following five main steps:

- **Ensure data privacy.** Nobody has access to personal data of somebody else. Everybody has access to her own personal data and to aggregated data of all the team.
- **Give the developers the access to all the information they require about the system and its usage.** Answer developers all questions they have about the usage or inner workings of the system. Offer them time to accommodate with the system and organize several open discussions to identify and address new issues as they appear.
- **Offer the developers a fair choice on whether to use the system or not.** This requires management support. Make clear from the beginning to the developers that they have a choice on whether to use the system or not. After the training phase, ask them about their option. Respect their decision.
- **Give developers full control over their own data.** Give the developers the possibility of interrupting or stopping the data collection at any time and for any interval of time. Allow them to specify applications from which data would not be collected. Allow them also to see and delete the data collected before sending them to the server. Our experience shows that the developers use very little these facilities. The mere presence of these facilities seems to be a needed reassurance that the privacy rights are respected.
- **Take into account developers' suggestions regarding the usage of the system.** During all the phases of the adoption process, receive feedback from developers and act towards addressing the issues raised. Sometimes, this might imply a considerable supplementary amount of work as new features are added to the system. However, our experience shows that such improvements can also bring value to the company. For instance, the automation of the weekly reports for the accounting department also brought value to the company as the developers gained precious time to devote to their tasks.

L2. Patience and commitment are needed until the system produces value to the company.

The adoption of the AISEMA system in the company under study had a very long set-up phase (planning, training, pilot deployment, and deployment). This was an initial investment from the company, as the system was not yet delivering any value while consuming resources. The company and team's patience and commitment to adopting the system was rewarded only after this phase when the system started incrementally to deliver value to the company. Currently, it is still in use and the value delivered is increasing as more usages are taken into consideration.

L3. Presentation is as important as accurate data collection.

The deployment phase was so long, partially due to many requests to improve the presentation of the data collected. The data collected did not bring real value until the developers, the team leader, and the managers started using it. To use it, they basically needed simplicity, brevity and clarity in presentation, even at the expense of precision. Figure 1 gives an example.

L4. There should be different aggregated views on the data rather than hierarchical break down.

Our initial approach was that the developers have access to their individual, detailed data, the leader to a first level of aggregation over the data of the whole team and the manager to an even higher level of aggregation of the data. However, it soon became apparent that in fact this hierarchical view on the data was not appropriate. What was needed was a set of different, parallel views on the data.

Each view should address the goals of one role in the team. The views cannot be aggregations of those of other roles, as the goals are not necessarily in an ascending order of abstraction, but simply differently oriented. For instance, the developers received daily reports of their activity on different parts of the code, while the leader preferred summarizing views of metrics connected to increasing the external or internal quality of the code.

The big challenge here is in designing these views. Our experience shows that most of the goals require a global view that takes into consideration several aspects. Considering the goal of the manager to evaluate the work product, there were more than 30 metrics defining external and internal product quality. On the other hand, the people want between 5 and 8 metrics at most, preferably without losing accuracy. Thus, the challenge consists in finding a sound way to aggregate the many metrics into a few numbers that would give the general picture.

Designing a sound aggregation of software metrics is not obvious and it is still an open issue. Such an aggregation should accept as input metrics with different measurement scales. It should have an adequate level of sensitivity to warn about potential problems in various areas. It should also be robust to the addition of new metrics. Such new metrics should not modify the result, but rather increase its precision.

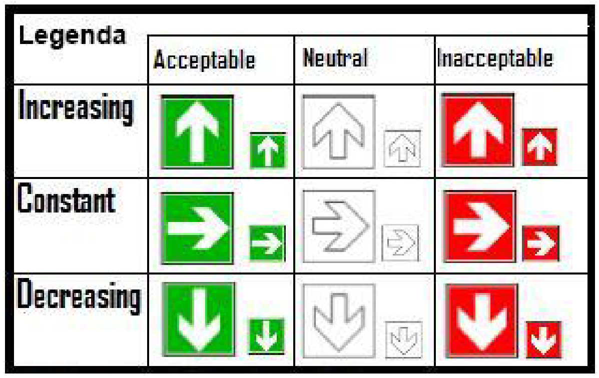

Figure 1. Metrics presentation. Big arrows show the trend over the past 8 weeks while small ones show the trend over the last week.

To provide the company with value as soon as possible, during this case-study, we took a different approach. Together with the managers, we selected, for each goal, a maximum of 8 metrics (losing thus some precision) and we tried to convey as much information as possible through the data presentation by means of size, color and shape. Thus, we showed for each metric the trends over various time intervals by means of a big and a small arrow, on green, red or white background (Figure 1). The direction of the arrow shows the trend, the size of the arrow shows the time interval, and the color of the background helps spotting problems fast.

L5. The system should be able to integrate data from various sources.

Relying only on the data that the system itself collects limits the value that the system can bring to the company. The company was actually collecting lots of data (for instance data about nightly builds) even before adopting the AISEMA system. These data were used only a couple of times per year for company review. Moreover, they were used mainly for justifying the process rather than for steering it. The little usage of the data made the developers consider data collection as yet another bureaucratic burden.

In the case under study, the AISEMA system was easily extendable to use also the existing data from external sources. This enriched the analyses that the system offers to its users. Importing these data from external sources ensured a smooth adoption, without requiring the team to change its existing process. While it is clear that no system will ever collect all possible data in all specific situations, it should be flexible enough to adapt with little effort and to make use of existing accurate data regardless of how they are actually collected.

Additionally, an AISEMA system obviously does not automatically collect data regarding non-computer related activities of developers (such as meetings or phone calls). In our case, this problem was addressed in two ways:

- the AISEMA system collected also data from the electronic calendars of users regarding scheduled events (such as meetings).
- users could manually input data regarding other activities such as phone calls or unplanned meetings.

L6. The system should be self-monitoring and self-healing.

Our experience showed that data accuracy should not be taken for granted even though it has been ensured by an initial assessment (section 4.7). Changes in the environment might affect the quality of the data collected even in a subtle way that is not producing a visible error (for instance data with corrupted values but intact structure). The measurement system has to constantly monitor itself and the data collected in order to identify as soon as possible potential problems.

When possible, the system should also be able to perform self-healing. For instance, a plug-in that was disabled due to repeated crashes of the instrumented IDE should be re-enabled. However, when self-healing is not possible, a warning of the potential problem and suggestions about possible causes are still very helpful. From our experience, these two phases consume the most time and energy. Once the problem is noticed and its cause identified, the solution is usually simple.

In the case of the system we used, the failure detection mechanisms are in general located at each of the components. However, this was not enough, as in some cases the components were silently disabled. Moreover, in the case of a lack of connection (which might however be legitimate) the failure of a local component is not visible to the server.

6. Conclusions and future work

In this paper, we presented a case study on the adoption and long term usage (nine months of planning followed by two years and on-going usage) of an AISEMA system in the software department of a company. Our findings have to be considered taking into account the characteristics of the company and of the system under study. By offering insights into the benefits and challenges of the adoption and usage of an AISEMA system in industry, we hope that this report can help other companies to consider their adoption of a measurement system. The lessons learnt can also guide improvement of AISEMA systems as they show existing challenges in usage and propose solutions.

Our experience shows that the company planning to adopt an AISEMA system should be willing to accept a long initial set-up phase. At the end of this phase, the system starts delivering value to the company.

Contrary to the existing perception that developers are against AISEMA systems, we found out that their cooperation can be easily gained by following 5 steps: ensuring data privacy, detailed information on the system prior to usage, a free choice of usage, full control to developers over data collection, and taking into account developers' suggestions. The full control over the data collected is rarely used in practice, but its presence is a needed reassurance to the developers.

The presentation of data and analyses is just as important as their accuracy and integrity. While lots of data are required to have an accurate view on processes or products, the data should be aggregated to provide people with summarizing views. The AISEMA systems should provide views that show the status at a glance and warn when there are problems. The views do not represent a hierarchy of aggregations of data built one on top of another. Instead, they should be parallel aggregations of the same data that address the different goals of the different roles in the team.

The aggregation of the data is still an open issue. We consider as an initial list of requirements of a sound aggregation the following: ability to deal with data on different scales, robustness to addition of new metrics, adequate level of sensitivity. We consider this list as a very preliminary and incomplete one and we envisage future work that explores the issue of useful aggregations of software engineering data.

The AISEMA systems should be enhanced with self-monitoring and self-healing capabilities. They should be able to cope with changes in the environment where data are collected (software and hardware changes or failures). They should also constantly monitor the quality of the data collected and warn about potential problems. We are currently working on an approach to detect potential problems of various components on client machines, by performing continuous checks of data consistency on the server.

7. References

[1] Johnson, P. M., and Disney, A., *A Critical Analysis of PSP Data Quality: Results from a Case Study.* Journal of Empirical Software Engineering, Vol. 4(4), 1999.

[2] Moore, C. A., *Project LEAP: Personal Process Improvement for the Differently Disciplined,* ICSE, 1999.

[3] Johnson, P. M., Kou, H., Agustin, J., Chan, C., Moore, C., Miglani, J., Zhen, S., and Doane, W. E. J., *Beyond the Personal Software Process: Metrics Collection and Analysis for the Differently Disciplined*. ICSE, 2003.

[4] Sillitti, A., Janes, A. Succi, G., and Vernazza, T., *Collecting, Integrating and Analyzing Software Metrics and Personal Software Process Data* EUROMICRO, 2003.

[5] Johnson, P. M., Kou, H., Paulding, M., Zhang, Q., Kagawa, A., and Yamashita, T., *Improving Software Development Management through Software Project Telemetry*. IEEE Software, August 2005.

[6] Kou, H. and Johnson, P. M., *Automated Recognition of Low-level Process: A Pilot Validation Study of Zorro for Test-driven Development*. Intl. Workshop on Software Process, 2006.

[7] Burnell, G. *6th Sense Analytics*. Homepage, http://www.6thsenseanalytics.com

[8] Ohira, M., Yokomori, R., Sakai, M., Matsumoto, K., Inoue, K., and Torii, K. *Empirical Project Monitor: A Tool for Mining Multiple Project Data*. Workshop on Mining Software Repositories, 2004.

[9] Schlesinger, F. and Jekutsch, S., 2006. *ElectroCodeoGram: An Environment for Studying Programming*. Workshop on Ethnographies of Code, 2006.

[10] Nystrom, N. A., Urbanic, J., and Savinell, C. 2005. *Understanding Productivity through Non-intrusive Instrumentation and Statistical Learning.* 2nd Workshop on Productivity and Performance in High-End Computing, 2005

[11] Johnson, P. M., Kou, H., Agustin, J. M., Zhang, Q., Kagawa, A. and Yamashita, T. *Practical Automated Process and Product Metric Collection and Analysis in a Classroom setting: Lessons Learned from Hackystat-UH*. Intl. Symposium on Empirical Software Engineering, 2004.

[12] Rossi, B., Scotto, M., Sillitti, A. and Succi, G. *An Empirical Study on the Migration to OpenOffice.org in the Public Administration*. Special Issue on Web-based, Community Driven Open Source Systems of the Intl. Journal of Information Technology and Web Engineering, 2006.

[13] Coman, I. D. and Sillitti, A. *An Empirical Exploratory Study on Inferring Developers' Activities from Low-Level Data*. Intl. Conference on Software Engineering and Knowledge Engineering, 2007.

[14] Beck, K. and Andres, C. 2005. *Extreme Programming Explained: Embrace Change*. Addison-Wesley, 2005

[15] Cockburn, A. 2006. *Agile Software Development: The Cooperative Game*, 2nd ed., 2006.

[16] Basili, V.R., and Weiss, D.M. 1984. *A Methodology for Collecting Valid Software Engineering Data*. IEEE Transactions on Software Engineering, Vol. 10(3), 1984.

[17] Chidamber, S. R. and Kemerer, C. F. *A Metrics Suite for Object Oriented Design*. IEEE Transactions on Software Engineering, Vol. 20(6), 1994.

[18] Coman, I. D., Sillitti, A. and Succi, G. *Investigating the Usefulness of Pair-Programming in a Mature Agile Team*. Intl. Conference on Agile Processes and eXtreme Programming in Software Engineering, 2008.

[19] Moser R., Pedrycz W., Sillitti A. and Succi G. *A model to identify refactoring effort during maintenance by mining source code repositories*. Intl. Conf. on Product Focused Software Process Improvement, 2008.

[20] Moser R., Abrahamsson P., Pedrycz W., Sillitti A., and Succi G., *A case study on the impact of refactoring on quality and productivity in an agile team.* IFIP Central and East European Conference on Software Engineering Techniques, 2007.

[21] Abrahamsson P., Moser R., Pedrycz W., Sillitti A., Succi G., *Effort Prediction in Iterative Software Development Processes – Incremental Versus Global Prediction Models.* Intl. Symposium on Empirical Software Engineering and Measurement, 2007.

[22] Moser R., Sillitti A., Abrahamsson P., and Succi G., *Does refactoring improve reusability?*. Intl. Conf. on Software Reuse, 2006.

[23] Pfleeger, S. L., *Lessons Learned in Building a Corporate Metrics Program*, IEEE Software, May 1993.

[24] Daskalantonakis, M., *A Practical View of Software Measurement and Implementation Experiences Within Motorola*, IEEE Transactions on Software Engineering, Vol. 18, 1992.

[25] Offen, R. J. and Jeffery, R., *Establishing Software Measurement Programs*, IEEE Software, pp. 45-53, Mar./Apr. 1997.

[26] Hall, T. and Fenton, N. *Implementing Effective Software Metrics Programs,* IEEE Software, Mar./Apr. 1997.

[27] Iversen, J. and Mathiassen, L., *Lessons from Implementing a Software Metrics Program*. Hawaii Intl. Conf. on System Sciences, 2000.

[28] Gopal, A., Krishnan, M. S., Mukhopadhyay, T., and Goldenson, D. R., *Measurement Programs in Software Development: Determinants of Success*. IEEE Transactions on Software Engineering, Vol. 28(9), 2002.

[29] Berry, M., and Jeffery R., *An Instrument for Assessing Software Measurement Programs.* Empirical Software Engineering, Vol. 5, 2000.