System Development Methodology Usage in Industry: A Review and Analysis

Audrey S. Griffin
asb10@psu.edu
Department of Business and Economics
Pennsylvania State University - DuBois
DuBois, PA 15801

Alan A. Brandyberry
abrandyb@kent.edu
Department of Management & Information Systems
Graduate School of Management
Kent State University
Kent, Ohio 44242

Abstract

The primary contribution of this research is to point to the need for more comprehensive, holistic research concerning the relative advantage of employing different system development methodologies (SDMs) in different settings. We first use the available literature to suggest a timeline of methodology introduction and discuss the temporal aspects of these introductions through the characteristics of the specific SDMs. Next we summarize and review the studies that are available concerning SDM usage from both public-sector and academic sources. Finally, we discuss the state of research in the area as a whole and suggest directions for future research.

Keywords: System Development Methodology, SDM, Adoption, Literature Review, Timeline

1. INTRODUCTION

Software is fast becoming ubiquitous in business. It is used extensively in the health care industry to operate sophisticated medical devices and equipment, by the communications industry to coordinate vast arrays of networks, by the transportation industry, finance industry, and virtually every other industry. It is difficult to find industries not affected, and sometimes afflicted, by software. It is used for production, service, and support. It is embedded in the appliances we use in our homes, in our automobiles, and in our handheld devices and games. For businesses, the failure of essential software can cause lost revenues, damage reputation of company or brand, and increase liability costs and loss of productivity (McGraw, 2003).

The need for high-quality software is widely recognized. Many factors have been identified that affect software quality, and many different methods have been developed to try to attain a quality software product. These include testing, structured processes, CASE tools, and code generation tools and debuggers. We have also tried to achieve quality by building it into the development process via a methodology.

Problems with system development were first noted in Garmisch, Germany in 1968 at the NATO Software Engineering Conference (Nuar and Randell, 1969). The problems noted then were essentially the same as those encountered today, almost 40 years later. Many system development projects fail, being over-budget and/or late; they may be incompatible with other software or hardware, not meet the original or revised specifications, or have quality issues that hamper use.

Although methodology usage has been highly touted in the academic research arena for many years, it has not been universally embraced by business. IT project failure is still problematic. As recently as 2007, the banking industry reports that applications are still bug-ridden, delivered late, over budget, and unsatisfactory to users (Crosman, 2007). These problems are not unique to banking, and despite the problems, there is still no consensus on how best to develop software applications.

A 1997 survey by Fitzgerald (1997) reveals that 60% of developers responding to the survey used no formal methodology. There are several suggested reasons for this. System development methodologies vary in their complexity and rigor. Highly structured processes tend to have heavy documentation requirements. Methodologies are often process-heavy, and can appear to impede productivity due to their finely detailed requirements. Developers who choose to use no methodology often do so to improve productivity and timeliness of product delivery (Fitzgerald, 1997). Not using a highly structured methodology also allows for increased flexibility, with developers being able to respond more quickly to changing business requirements ('agile' methodology proponents suggest these methodologies do not restrict flexibility). In lieu of using an entire methodology, companies often adapt existing methodologies by using only the high-level definitions as guidelines for the process rather than the full methodology. The benefits to be achieved by the structure and rigor of a full-fledged methodology may not be reflected in the increased development costs.

In 1989, Lyytinen posited that the systems development process would see dramatic changes during the 1990's due to many simultaneous changes affecting the industry (Lyytinen, 1989). Technology changes rapidly, offering new opportunities for firms, but also presenting new challenges. The area of software development is affected by these changes but must also consider users, methods and the organization itself. New application types were proliferating, creating more complex and varied symbols to be stored in the systems. As applications become larger and more complex, changes are also occurring within the environment of the organization that require new skills and new roles. Lyytinen predicted that development will no longer be limited to one methodology but of necessity will be adapted to the needs of the organization.

The balance of the paper is structured as follows. We first use the available literature to suggest a timeline of methodology introduction and discuss the temporal aspects of these introductions through the characteristics of the specific SDMs. Next we summarize and review the studies that are available concerning SDM usage from both public-sector and academic sources. Finally, we discuss the state of research in the area as a whole and suggest directions for future research.

2. METHODOLOGY TIMELINE

Software has had a sketchy history regarding quality. From the 1940's to late 1960's, quality was high (Whittaker and Voas, 2002). Applications were being developed primarily for mainframes, and the compilation environment was not friendly to errors. However, the types of problems being solved were evolving from simple computations to those that were more complex.

Initially, programmers generally worked alone in their own area of expertise on applications that were small by comparison to today's programs. This early method of programming was known as "code and fix" (Boehm, 1988). There were no stages for systems analysis or design, and programmers coded on an ad hoc basis. The "fix" portion became very expensive due to inadequate planning and design (Boehm, 1988). As programs grew larger, more people were involved in the development process, necessitating the implementation and coordination of planning and control processes. Methodologies provide a clear-cut sequence of events to be followed, with deliverables specified by the particular methodology (Boehm, 1988).

By the 1970's, when PC's were introduced, it was easier to compile programs, and developers became lax as they tried to introduce new types of software. This was also the era when software began to develop its reputation as being buggy. Tools and formal methodologies were developed to help developers produce higher quality code. However, both were complex and difficult to use. Differing solutions were and are still being offered, each suggesting that it might be *the* silver bullet; however, according to Frederick Brooks, there *is* no silver bullet (Brooks, 1995).

The computing environment has also grown more complex, with distributed computing, web-based computing and networking, increased miniaturization along with cheap production, multiple processors, and complex applications that are typically completed by teams rather than individuals. These changes create a more labyrinthine development environment that must be carefully managed if quality software is to be achieved.

There has been a great deal of interest in system development methodologies over the years. Many new methodologies have been introduced in attempts to improve processes and developer acceptance. A timeline of selected major system development methodologies is shown in Figure 1 (see Appendix). Dates shown are approximate as some ambiguities exist in establishing start dates.

Of those organizations using a formal methodology, many use methodologies marketed by consulting organizations; others develop their own in-house methodologies, sometimes using another methodology as a template. Occasionally the terms of buyouts or mergers will dictate that a methodology (any) be used, or that a specific methodology is required. Companies contracting with the government are required to have processes in place to ensure repeatability and quality. Furthermore, some organizations use frameworks such as the Rational Unified Process or Capability Maturity Model Integration in conjunction with their system development methodology to provide additional structure and tools to enhance their development process.

As software development has evolved and matured, many methodologies for developing software have been promoted. From the earliest days of software development, when no methodology was used, to today's diverse menu of methodologies and tools, whether methodologies help or hinder has been the subject of much discourse. Proponents of methodological software development generally believe that the structure and rigor of a process promote an end product of higher quality with less risk and fewer surprises. Others believe that system development methodologies are cumbersome and hinder creativity by putting more emphasis on process and procedures.

One of earliest methodologies developed for developing new software systems was the Systems Development Life Cycle (SDLC). It was developed in the late 1960's in response to the need for management control and structure in the system development process (Fitzgerald, 1997). It was an enhancement of an earlier model known as the Stagewise model (Boehm, 1988). The SDLC consists of a series of steps or phases that encompass all of the steps necessary for planning, developing, and implementing a software project. It is commonly referred to as a spiral or waterfall model because the steps are linked, and each builds off of the previous step. The phases are discrete; however it is permissible to regress to an earlier stage if necessary. Each phase has corresponding activities that are performed in that particular step, as well as deliverables or outputs that are required. There are many variations and adaptations of the SDLC as organizations extract the phases and activities that are pertinent to their particular project.

As might be expected, highly structured processes such as the SDLC have problems that are inherent in the process (Hoffer et al., 1996). One such problem with the waterfall model is the assumption that the project can only go through the process once, with user testing and system testing occurring after completion of the project (Brooks, 1995). Although it is permissible to go back to earlier phases, it is actually difficult and costly to do so once deliverables have been produced. Second, the SDLC tends to be a time-consuming and lengthy process. One of the major problems with systems development projects in general is that, not only can requirements not be fully known in advance (McCracken and Jackson, 1982) but, requirements change over time as technology advances and user needs change. Also, users must test the software before requirements can be considered complete (Gordon and Bieman, 1995). If the project is completed, there is a high likelihood that it will not be acceptable to the end users. Implementing changes in design after the design phase is completed increases both the cost of the project and the time to completion.

To address some of the short-comings of the traditional SDLC, many other methodologies have evolved. Structured Analysis and Structured Design (SASD) introduced some of the structure of engineering to programming. It was an attempt to improve the quality of the design phase by dividing projects into smaller, more manageable pieces and using tools such as data flow diagrams and entity-relationship diagrams (Yourdon, 1986; Hoffer et al., 1996). The Spiral Model of the software process was a refinement of the waterfall method, incorporating prototyping into the many stages of development (Boehm, 1988). This risk-driven approach was designed to offer the best features of prior methodologies while introducing quality objectives into the development cycle, thereby enabling developers to eliminate errors early (Boehm, 1988). The Object-Oriented Analysis and Design (OOAD) approach, developed at about the same time, focused on encapsulation and the reuse of code. These approaches gained acceptance because they addressed specific needs identified by project managers and programmers, however they are still essentially structured methodologies (Fitzgerald, 1997; Hoffer et al., 1996).

Involving the User

Although the development process became highly structured, systems continued to typically have lengthy completion times, high development and maintenance costs, loss of interest and morale by project participants, and quality issues, and were still unlikely to meet user needs (Fitzgerald, 1997; Hoffer et al., 1996). One solution was to develop methodologies that incorporate the end user into the development process. Rapid Application Development (RAD) processes such as prototyping strive to incorporate the user's input into the design process. Other methodologies strategically incorporate user input into specific phases of the design process. Joint Application Development (JAD) requires intense user involvement early in the requirements phase (Hoffer et al., 1996). In Europe, Participatory Design (PD) has been widely accepted due to the involvement of the entire user community (Hoffer et al., 1996). Because user acceptance is gained early in the project, the project is more likely to be completed and meet user requirements (Fitzgerald, 1997). While successful at delivering systems that are more compatible with end user expectations, development and maintenance costs remain high primarily due to quality issues.

Software Engineering

To address concerns of poor-quality software, methodologies that incorporate engineering-like specifications, requirements, and processes were developed during the middle to late 1980s. One example is the Cleanroom software engineering process developed at IBM, an incremental development process that uses box structure specification and design, with each increment going through rigorous verification to ensure correctness, followed by statistical quality certification to emphasize defect prevention rather than defect removal (Mills et al., 1987; Hausler et al., 1994). The goal is zero defects (development goal) and failure-free performance (operation goal) (Hausler et al., 1994; Poore, 1999). The Cleanroom process is a life cycle process, covering all aspects from planning through implementation (Hausler et al., 1994).

Agile Methods

The complexity of software engineering's mathematically-based methodologies has led to the development of light-weight methodologies. Several have evolved during the past decade. Agile methods emphasize the generation of an early working product with incremental functionality as opposed to prototyping (Reifer, 2002). Participation by all stakeholders is critical. Agile methods include differing types of practices such as pair programming, team programming, refactoring, RAD, etc. (Reifer, 2002). Benefits touted include improved productivity, reduced costs, shorter time-to-market, and improved quality (Reifer, 2002).

Many agile methodologies have been introduced during the past ten to fifteen years, such as Extreme Programming (XP), Scrum, Dynamic Systems Development Methodology (DSDM), Adaptive Programming, and Crystal Clear. All are intended to address aspects of the system development process that might be lacking or under-developed in other methodologies, or that certain types of applications or development environments require. Additionally, proponents state that these methodologies respond well to inevitable changes in requirements. Incremental design is often a primary tenet, allowing portions of the project to be implemented and working before other portions have even formally been added to the requirements (DeHondt and Brandyberry, 2007). In a 2006 survey of North American and European businesses, Forrester Research found that the use of agile methods increased in 2006 as wider acceptance was attained (Richards, 2006).

Frameworks

Six Sigma was conceived at Motorola during the mid-1980's. Although not technically a system development methodology, it is an approach for managing process and product quality that can be applied to systems development that is reliant on data. Its goal is to attain a quality level of no more than 3.4 defects per million opportunities or better by systematically solving problems. IBM's Rational Unified Process (RUP) had its roots in the 1980s at Rational Software. It is a disciplined, highly customizable process based on the spiral model that utilizes best practices. RUP is an iterative process that emphasizes testing throughout each iteration, thus enabling defects to be identified and fixed early in the process.

The Software Engineering Institute's Capability Maturity Model (CMM), also introduced in the mid-1980's, asserted that a process of higher maturity leads to increased productivity, reduced cycle time, and fewer defects. It was a five-level framework that enabled an organization to assess its process maturity, and see what is needed to advance to the next level (Humphrey, 1988). The emphasis was on peer review and inspections. Processes should be under statistical control if consistent results are to be achieved. The CMM was replaced with the Capability Maturity Model Integration® in 2002, and is aligned with more modern iterative approaches rather than conventional SDLC methodologies.

There is no methodology that is perfect for every organization or project. Each methodology must deal with variations in people, project, cultures, and changing technologies and designs (Cockburn, 2002). Avison and Fitzgerald (2003) suggest that we are in the "post-methodology" era, and have transitioned from highly structured processes involving phases through the "methodology era" to the current reappraisal of the role and value of methodologies.

While there are many methodologies available, acceptance and use of them has been varied. It is unclear whether they are any more widely used today than they were in the 1990's, despite the benefits attributed to their use. Many studies and surveys have been done from different perspectives. The goals of this study include the consolidation of findings and identification of industry trends. Two types of data are generally available: industry or organization surveys, and academic research. This study analyzes results from both types in separate analyses.

3. INDUSTRY SURVEYS

Although industry surveys exist that target the usage of specific tools or programming languages, relatively few are concerned with development methodologies. The criteria for inclusion in this portion of the study are:

- The survey was conducted by an industry organization or association.
- The survey assessed system development methodology usage.

- The survey reported quantitative data.

A total of five surveys met the criteria. All were conducted between 2001 and 2007 and are listed in Table 1 (see Appendix B). Results from these surveys should be used with discretion. Four of the surveys are designed to assess usage of Agile methodologies, and bias toward the topic is a possibility due to the sponsoring organizations and samples. Therefore results may not be generalizable to the software development industry in general. In addition, only the reported results that are pertinent to this research are considered here.

Only one survey gives a general overview of methodology usage. Entitled Changing Application Development Needs, it was conducted by TechRepublic and sponsored by MKS. TechRepublic (http://techrepublic.com.com/) is a CNET Network web site that has a self-reported community of over four million IT professionals.

TechRepublic members were invited to take the survey in the fall of 2001. There were a total of five hundred fifty completed surveys received from developers or IT Manager/Directors of organizations ranging in size from 100 - 5,000 employees in the U.S., Canada, and Europe.

The results from the TechRepublic, Inc. 2001 (MKS, 2001) survey are reproduced here in Figure 2 (see Appendix A) for reference. It should be noted that at the time of this survey, agile development was in its early days, and was not included. According to the survey, the "Other" category also included those using the Rational Unified Process (RUP), in-house methodologies, and those using no methodology. Respondents were able to choose all methodologies that applied.

The other four industry surveys (Ambler, 2006; Shine, 2003; VersionOne, 2006; VersionOne, 2007) were designed to assess agile methodology usage. Of these, only one survey specified whether respondents had skipped the question, or had chosen "none." It was unclear whether "none" meant no *agile* methodology was used or no methodology was used at all. Results are summarized in Figure 3.

Figure 3: Agile Use

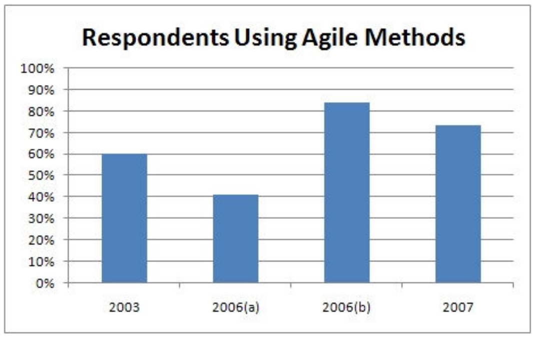
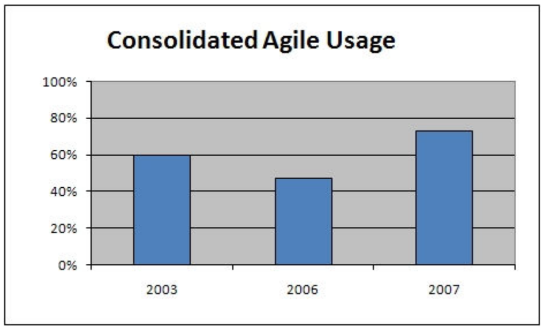


Figure 4: Consolidated Agile Usage

Data from the two 2006 surveys were combined in Figure 4 (see Appendix) to detect



trends. The apparent decrease in 2006 is probably due to the exorbitant number of respondents in Survey 2006(a) that selected "none" as the agile methodology used. Again, it is unclear what "none" represents. While the charts show the use of agile methodologies increasing over time, this could represent some element of bias present in the respondent population.

Survey results were further broken down to look at trends for the specific agile methodologies being used. Of the agile methodologies being used, Figure 5 (see Appendix) shows that there has been a steady decline of Extreme programming (XP) use between 2003 and 2007. The percent of Scrum projects has increased, as has Feature Driven Development (FDD). The largest increase came in the area of custom/hybrid methodology development and usage from zero in 2003 to 32% in 2007, as more companies are customizing their agile methodologies rather than using one that is prescribed. The use of hybrid techniques may help in explaining the decline of methodologies such as XP where the amount of discussion in informal forums and formal press concerning XP does not necessarily seem to support such a drastic decline.

4. ACADEMIC STUDIES

The initial literature search produced a total of thirty-six academic papers dealing with methodology assessment. To be included in this portion of the study, academic research studies meeting the following criteria were used in order to maintain consistency and reliability of data:

- The study used survey research to look at methodology usage.
- Results are quantifiable.

The fourteen methodology studies that survived the specification requirements were conducted between 1986 and 2001.

During the 1986-2001 time period, many changes occurred in software development. The twenty-one studies that were retrieved but not used focused primarily on tool usage or other aspects that included development methods. Case study results were excluded due to their applicability to a single organization. A list of the fourteen studies used is shown in Table 2 (see Appendix B).

One of the difficulties of assessing methodology usage is the large amount of disparity in the studies. This is also the reason that a quantitative meta-analysis of these results was not deemed appropriate. Additionally, the large timeframe involved, where changes in the environment undoubtedly occurred, lends itself more to a descriptive analysis of trends rather than to any attempt to aggregate studies. Reporting of results was found to be occasionally vague and unclear, making it difficult to discern the actual breakdowns of usage versus non-usage. Terminologies also differ as methodologies come and go. The research methodologies used also differed, making it difficult to compare results that were not measuring the same constructs or using similar samples of respondents.

Several studies reported system development methodology usage vs. non-usage. In a survey of members of the St. Louis Chapter of Association of Systems Managers, Sumner and Sitek (1986) received forty-five usable responses (26%) from thirty-eight of the 172 companies surveyed by mail. Members of this association are primarily project managers and systems analysts. They report that 38 of the 45 respondents use a system development methodology. There was no mention that multiple responses from one firm were in any way adjusted to firm-level results that would make interpretation consistent with other studies. Of those respondents using a methodology, nineteen were developed in-house, seventeen were purchased, and two indicated that their systems were both purchased and developed in-house.

Gordon et al. (1987) surveyed top computer executives in 990 U.S. firms to ascertain what processes were being used to develop computer-based information systems. Their results were based upon 97 (9.8%) returned questionnaires. The focus was on the systems development life cycle (SDLC) approach of system development. They found that respondents used a total of thirty-six different approaches to develop systems that were used alone or in combination with the SDLC approach. These included traditional/classical, structured, automated, prototyping and information center.

In a study that looked at the relationship between system development and maintenance, Dekleva (1992) surveyed each of the Fortune 500 companies in 1985. There were 112 (22.4%) respondents. Thirty-five of the returned questionnaires were excluded from further study because either no methodology was used (4), the methodology was unknown (25), or there were conflicting responses (6). Those using methodologies were classified according to the methodology used: traditional SDLC, software engineering, information engineering, prototyping, computer-aided software engineering, and other. A specific breakdown of the number of respondents in each category was not given. Systems were then further categorized as being modern or traditional, with 44 being considered traditional and 32 as modern.

Ward et al. (1996) conducted a survey of senior IS/IT managers and business managers within the Times Top 100 plus 150 additional large companies within the U.K. to evaluate the realization of IS/IT benefits. Of the 250 surveys mailed, sixty responses (24%) were received. Results indicate that 52% (31) use a methodology for system development and 15% (9) use no methodology in any of the three areas. Methodology usage for project management and investment appraisal were also assessed in this survey. Respondents were able to select as many categories as were applicable.

In a survey mailed to 776 named individuals in different U. K. organizations, Fitzgerald (1998) found that methodology usage is more likely to occur in larger organizations. Of the 162 responses received (21%), 60% indicated that no system development methodology is used. The remaining 65 companies use methodologies classified as commercial (14%), internal based on commercial (12%) or internal not based on commercial (14%).

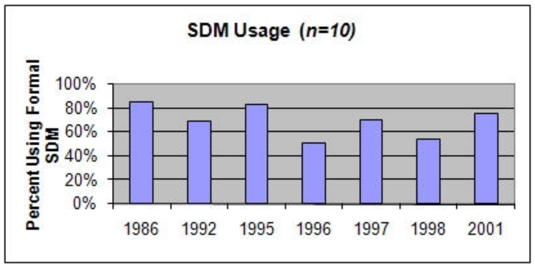
Roberts et al. (1998) queried sixty-one companies in the U.S. and Canada in an exploratory study to determine factors that impact the implementation of system development methodologies. The survey was sent to 329 contact persons in 61 different companies. A total of 192 completed responses (58.35%) were received, with eight being the maximum received from one company. For participation, respondents were required to have been using a system development methodology for at least two years. Exploratory factor analysis revealed factors that companies considering the implementation of a development methodology should consider. Companies in the study were using Ernst & Young Navigator, Anderson Consulting METHOD/1, TI Information Engineering, KW Information Engineering, JMA Information Engineering, custom methodologies, and other unspecified methodologies. Although numbers of responses for each methodology were reported, it was not clear what the per-company breakdown was due to the allowance of multiple respondents per company.

Software development methods in Brunei Darussalam were studied by Rahim et al. (1998) to explore usage in both public and private sector firms in Asia. The survey was sent to IS managers and executives from 100 organizations. Each organization received one survey. Thirty-six responses were received. It was revealed that a total of nine methodologies are used by the 24 respondents using methodologies, while twelve firms indicated that no methodology was used. A Likert scale (0-4) was used, and respondents were able to indicate the extent to which each methodology was used in their company. Results indicate that in-house developed methodologies and SSADM were the more frequently used, and that prototyping was used on an occasional basis. Other methodologies used included Object Oriented Design, Information Engineering, Structured Design by Yourdon, Structured Design by DeMarco, System Requirements Engineering, and Jackson Systems Design. They found differences between public and private organizations, the nature of the business sector, and mature and novice organizations.

The 2000 study by Khalifa and Verner targeted eighty-two senior software developers from Hong Kong and Australia. They evaluated the extent of usage of the waterfall and prototyping methodologies. It was not specified whether the respondents were known or required to be using a methodology in order to participate, or whether any of the respondents did not use any methodology. Of the eighty-two respondents, 42% used the waterfall method only, 8% used prototyping only, and 44.7% used both waterfall and prototyping (Khalifa and Verner, 2000).

Of the ten studies reporting usage vs. non-usage of system development methodologies, half focus on the U.K. Two of the surveys are based on Asian companies, one is from Finland, one from the U.S., and one uses Fortune 500 companies not otherwise specified. Figure 6 shows the overall trend in general methodology usage. While no overall trend is apparent due to the diverse geographic locations being studied and relatively small sample sizes, it appears that more than 50% of companies developing software use some system development methodology.

Figure 6: Methodology Usage



Since there were only two surveys each from the U.S. and Asia, there was not enough data to establish trends. However, the two U.S. surveys were consistent, averaging 86%, although they were from the mid-1980's. The two Asia surveys from 1998 were also consistent with each other, averaging 68% of companies using a system development methodology. Although there were five studies that assessed methodology usage in the U.K., there was again no obvious trend. This was probably due to differences in the research methodologies applied.

The remaining six studies looked at the specific methodologies being used by companies to develop software and are assessed next. A survey by Hardy et al. (1995) looked at methodology usage and customization in the U.K. Five hundred ten companies were randomly selected from categories determined by categorizing a list of graduate jobs and courses. The response rate was 20%. Of that, although only 18% reported that they were using no methodology, only 44% reported using a formal structured system development methodology. Most used an in-house methodology which meant it was developed in-house, or a collection of tools being used. Twenty-four percent reported using SSADM. Others used included Yourdon, JSD, OOD, Formal specification, and IE.

A New Zealand survey in 1996 assessed the relationship between the methodology used and the demographics of the organization (Pastor Urban and Whiddett, 1996). Responses were received from 353 of the 563 organizations surveyed, the highest response rate of any of the studies looked at in this analysis. The population was representative of the population by New Zealand economic sectors, and evenly represented large, medium, and small companies. Forty-seven percent of the companies were using structured methodologies such as SSADM, structured, and Yourdon. This type of methodology appears to be most suited for large organizations. The authors reported that most of the methodologies are customized. The remaining 53% used a variety of methods including object-oriented (15%), prototyping (35%), Multiview, and Socio-technical. Medium and small organizations were more likely to use prototyping.

Chatzoglou (1997) surveyed people involved in U.K projects that were divided into three strata - academics, software houses and consultancies, and industry. Seventy-two (38.92%) responses of the 182 surveys sent out were included in the study. Sixty-nine percent of the projects used methodologies in various stages of the development process. It was reported that 31% used no methodology. For those projects using methodologies, 23% each used SSADM and in-house methodologies. Only 8% used prototyping. Twenty-seven percent of projects used "other" unspecified methodologies. The author concluded that when using a methodology for the entire development process including requirements capture and analysis, the time to completion, effort and cost of development, and number of people involved is reduced.

There were two studies included from 1998. Iivari and Maansaari (1998) sent questionnaires to primarily IS managers of eighty-seven organizations identified as using CASE tools. Of the 420 questionnaires mailed, there were 63 (15%) received from 44 companies. Although most of the questions were in reference to CASE tool usage, respondents were asked a few questions about the development methodology used on a 5-point Likert scale. The majority used object-oriented approaches (39%), followed by SA/SD at 23%. Other methodologies used included IE, JSD, in-house developed methodologies, other, and "anonymous." In the "anonymous" category respondents listed techniques used rather than a formal method. Twelve respondents did not answer this question. 34% of respondents indicated they used a commercial methodology, and 34% used an in-house methodology primarily adapted from a standard method.

In the second study from 1998, an assessment of software engineering practices in Singapore was conducted to determine the extent of methodology usage (Poo and Chung, 1998). Of the 240 organizations surveyed, fifty-four valid responses were received. Sixty-eight percent reported using a methodology, with the remaining 30 percent using none. One response did not have an answer to this question. Most of the respondents used the sequential/waterfall model (37%). Slightly less than seventeen percent reported using the incremental model, and nine percent used rapid prototyping. Eleven percent of the respondents were not sure what methodology was used.

In a survey designed primarily to look at multimedia and web development techniques, Barry and Lang (2001) queried the top 1,000 companies in general industry in the U.K. regarding their development methods. They received 98 responses (10%). Sixty-five of the respondents answered the question regarding methodology usage, with one quarter of them indicating that they use no methodology. Seventy-five percent of the rest use an in-house methodology, 16.9% use SSADM, and 13.8% use RAD. Respondents could choose more than one methodology. Respondents felt that methodologies are too cumbersome. Many tend to use older tools and approaches rather than newer ones.

Due to inconsistencies in research methodologies and reporting, it is difficult to obtain longitudinal data and consolidate the data. There appear to be regional differences also, but the samples are too small to make that distinction. Half of the studies here are from the U.K. None of the six studies specifying methodology usage were from the United States. Figure 7 (Appendix A) shows overall methodology usage in the six studies that gave a breakdown. Figure 8 (Appendix A) shows the methodologies identified in each of the six studies. The data used to generate those two charts is shown in Table 3 (see Appendix B).

5. DISCUSSION

In addition to reviewing and summarizing the existing research available concerning the usage of SDMs, one of the primary contributions of this effort is to point to the lack of good systematic research related to SDM usage. At any given point of time it is difficult to judge what the dominant methodologies are as well as the timelines concerning their diffusion and abandonment in practice. The relatively simple question of "who is using what, for what, and why?" needs to be answered before questions that are even more meaningful to practice can be posed. If SDM usage research is to yield benefits then it must provide insights into what methodologies are best applied in what scenarios. It is conceptually appealing to assume that the choice of a particular SDM for a particular project is likely to have a significant impact on the likelihood for success of that project. The relative advantage of employing a specific SDM is likely related to a number of interrelated determining factors such as project characteristics, development team characteristics, corporate culture, societal cultural differences, individual developer characteristics, and others. Research that can add to the body of knowledge concerning these complex relationships is difficult to operationalize but would yield significant advances if successfully undertaken.

No obvious trends are seen in looking at system development methodology adoption from 1986 - 2001. This could be due to the lack of surveys that focused solely on methodology usage, or the wide disparity in geographical locations surveyed. Discrepancies in survey methods also made it difficult to compare results. It was clear that many companies are still using older, more familiar methodologies. Some companies use more than one, fitting the methodology to the project or the team. In studies using Likert scale responses, many organizations indicate that they are using more than one methodology in varying degrees. This information is not possible to ascertain in the other studies.

If there is any trend to be discerned, it is that in-house methodologies, whether developed completely in-house or customized from other methodologies, seem to be increasing (Griffin, 2008). As more software tools are available to developers, the inflexibility of a process-intensive methodology becomes less desirable and is often seen as unnecessary. The choice of development methodology is a function of the user, the organizational environment, and what developers like and are familiar with. If one were to ask a member of the general population, "What is the right or best language to speak?" the answer would probably be "There is no 'right' or 'best' language." People speak a language that they know and that is common to their environment. Variation is the rule rather than the exception. Similarly, the question of "Which is the right or best system development methodology to use?" will probably generate the same response, as developers view it similarly. One size does not fit all, and the increase in methodology customization seems to support this.

REFERENCES

Ambler, S. (2006) "Survey Says: Agile Works in Practice." Dr. Dobb's Journal. http:/www.ddj.com/dept/architect/191800169.

Avison, D. E. and Fitzgerald, G. (2003). "Where now for development methodologies?" Communications of the ACM, 46(1):79-82.

Barry, C. and Lang, M. (2001). "A survey of multimedia and web development techniques and methodology usage." IEEE MultiMedia, 8(2):52-60.

Boehm, B. W. (1988). "A Spiral Model of Software Development." IEEE Computer, 5:61-72.

Brooks, Jr., F. P. (1995). The Mythical Man-Month: Essays on Software Engineering 25th Anniversary Edition. Addison-Wesley, Reading, Massachusetts, 2nd edition.

Chatzoglou, P. D. (1997). "Use of methodologies: an empirical analysis of their impact on the economics of the development process." European Journal of Information Systems, 6:256-270.

Cockburn, A. (2002). Agile Software Development. The Agile Software Development Series. Addison-Wesley.

Crosman, P. (2007). "Wall street firms turn to best practices models to speed up software delivery." Wall Street \& Technology: http://www.wallstreetandtech.com/showArticle.jhtml?articleID=199601961.

DeHondt, G. & Brandyberry, A. (2007). "Programming in the eXtreme: Critical characteristics of Agile implementations." e-Informatica Software Engineering Journal. 1(1):43-58.

Dekleva, S. M. (1992). "The influence of the information systems development approach on maintenance." MIS Quarterly, 16(3):355-372.

Fitzgerald, B. (1997). "The use of systems development methodologies in practice: a field study." Information Systems Journal, 7:201-212.

Fitzgerald, B. (1998). "An empirical investigation into the adoption of systems development methodologies." Information and Management, 34(6):317-328.

Gordon, C. L., Necco, C. R., and Tsai, N. W. (1987). "Toward a standard systems development life cycle." J. Syst. Manage., 38(8):24-27.

Gordon, V. S. and Bieman, J. M. (1995). "Rapid Prototyping: Lessons Learned." IEEE Software, 12(1):85-95.

Griffin, A. S. (2008). Examining the Decision Process and Outcomes of System Development Methodology Adoption. (PhD dissertation, Kent State University, 2008).

Hardy, C. J., Thompson, J. B., and Edwards, H. M. (1995). "The use, limitations, and customization of structured systems development methods in the united kingdom." Information and Software Technology, 37(9):467-477.

Hausler, P. A., Linger, R. C., and Trammell, C. J. (1994). "Adopting Cleanroom software engineering with a phased approach." IBM Systems Journal, 33(1):89-109.

Hoffer, J. A., George, J. F., and Valacich, J. S. (1996). Modern Systems Analysis and Design. The Benjamin/Cummings Publishing Company, Inc.

Humphrey, W. S. (1988). "Characterizing the software process: A maturity framework." IEEE Software, 5(3):73-79.

Iivari, J. and Maansaari, J. (1998). "The usage of systems development methods: are we stuck to old practices?" Information and software technology, 40:501-510.

Khalifa, M. and Verner, J. M. (2000). "Drivers for software development method usage." IEEE Transactions on engineering management, 47(3):360-369.

Lyytinen, K. (1989). "New challenges of systems development: a vision of the 90's." SIGMIS Database, 20(3):1-12.

McCracken, D. D. and Jackson, M. A. (1982). "Life Cycle Concept Considered Harmful." ACM Software Engineering Notes, 7(2):29-32.

McGraw, G. (2003). "Making Essential Software Work." Technical report, Cigital, Dulles, Virginia. http://www.cigital.com.

Mills, H. D., Dyer, M., and Linger, R. C. (1987). "Cleanroom software engineering." IEEE Software, 4(5):19-24.

MKS: Changing Application Development Needs, a TechRepublic Survey, Sponsored by MKS. (2001). Retrieved July 5, 2006 from http://www.mks.com.

Nuar, P. and Randell, B., editors (1969). Software Engineering. NATO Science Committe.

Poo, D. C. C. and Chung, M. K. (1998). "Software engineering practices in singapore." The Journal of Systems and Software, 41(1):3-15.

Poore, J. H. (1999). Cleanroom Software Engineering: A Reader, Chapter 4 - The Cleanroom approach to six sigma: Combining information, pages 71-82. NCC Blackwell.

Rahim, M., Seyal, A. H., and Rahman, M. N. A. (1998). "Use of software systems development methods: An empirical study in brunei darussalam." Information & Software Technology, 39:949-963.

Reifer, D. J. (2002). "How good are agile methodologies?" IEEE Software, pages 16-18. July/August.

Richards, K. (2006). "Early mainstream: Agile develops in the enterprise." Application Development Trends, page 2.

Roberts, T., Gibson, M., Fields, K., and Rainer, R. (1998). "Factors that impact implementing a system development methodology." IEEE Transactions on Software Engineering, 24(8):640-649.

Shine Technologies: Agile Methodologies Survey Results. (2003). Retrieved from http://www.shinetech.com/download/attachments/98/ShineTechAgileSurvey2003-01-17.pdf.

VersionOne Survey: "The State of Agile Development," (2006). Retrieved from http://www.versionone.com/surveyresults.asp.

VersionOne 2nd Annual Survey: "The State of Agile Development," (2007). Retrieved from http://www.versionone.com/pdf/StateOfAgileDevelopmet2_FullDataReport.pdf.

Sumner, M. and Sitek, J. (1986). "Are structured methods for systems analysis and design being used?" Journal of Systems Management, 37(6):18-23.

Urban, J. L. and Whiddett, R. J. (1996). "The relationship between systems development methodologies and organisational demographics: A survey of new zealand organisations." In Information Systems Conference of New Zealand, 1996. Proceedings, page 179, Palmerston North, New Zealand. ISBN: 0-8186-7710-4.

Ward, J., Taylor, P., and Bond, P. (1996). "Evaluation and realisation of is/it benefits: An empirical study of current practice." European Journal of Information Systems, 4(2):214-225.

Whittaker, J. A. and Voas, J. M. (2002). "50 Years of Software: Key Principles for Quality." IT Pro, pages 28-35. Nov/Dec 2002. Publication of IEEE.

Yourdon, E. (1986). "What Ever Happened to Structured Analysis." Datamation, pages 133-138.

Appendix A - Figures

Figure 1: Timeline of methodology development

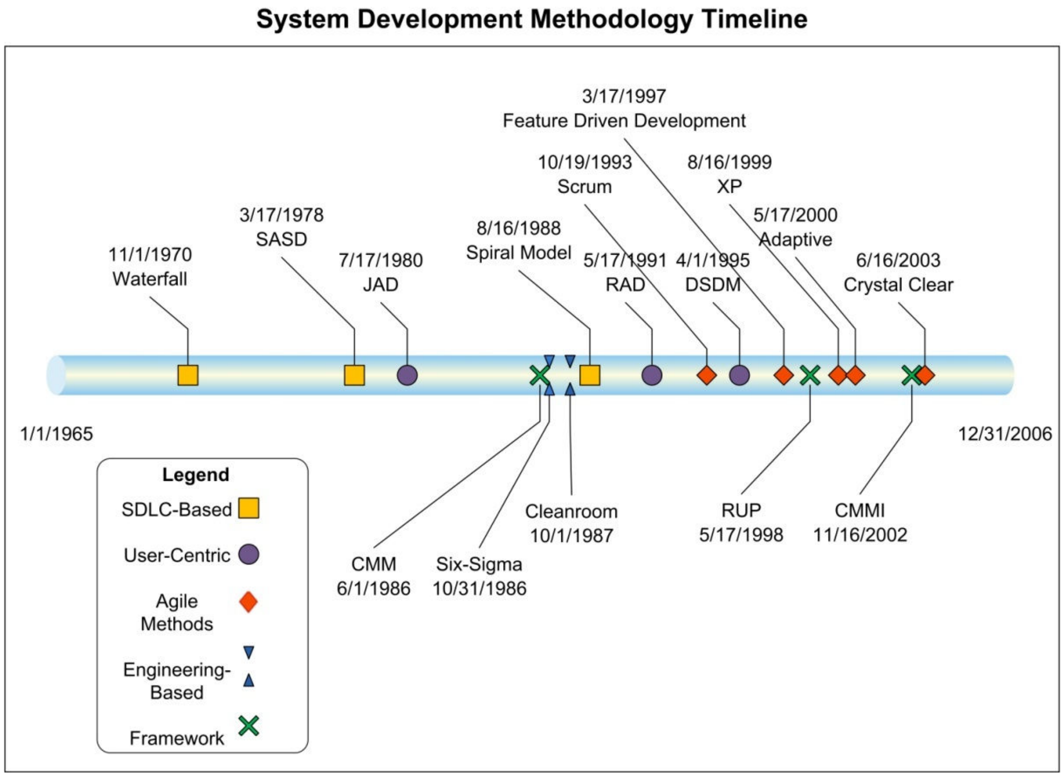


Figure 2: TechRepublic Survey Results (MKS, 2001)

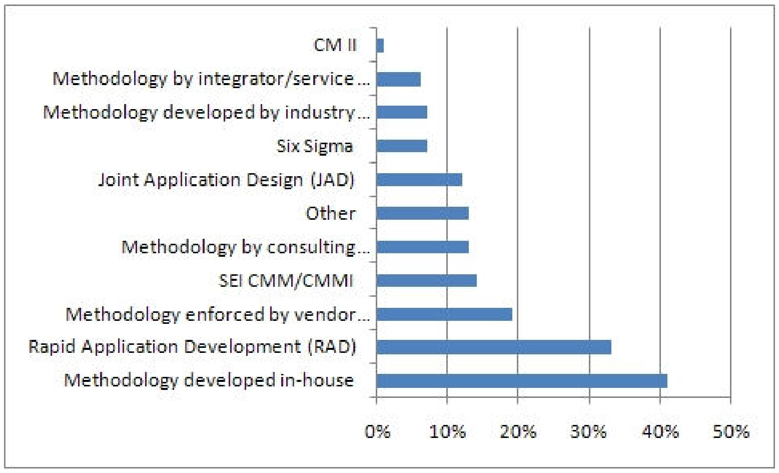


Figure 5: Agile Methods Used

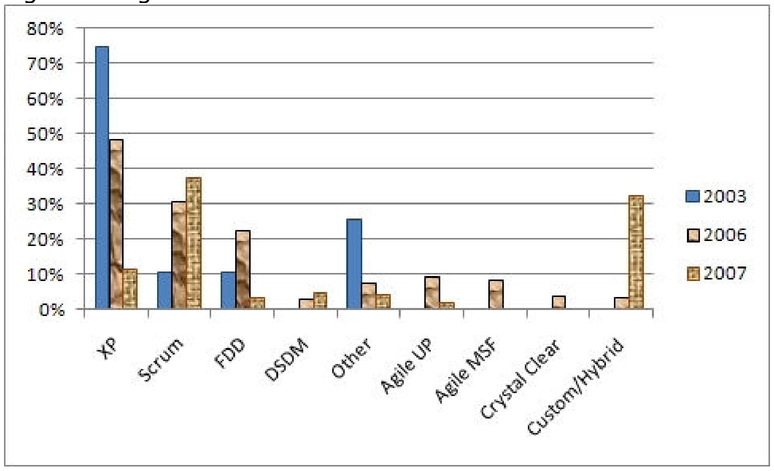


Figure 7: Methodologies Being Used

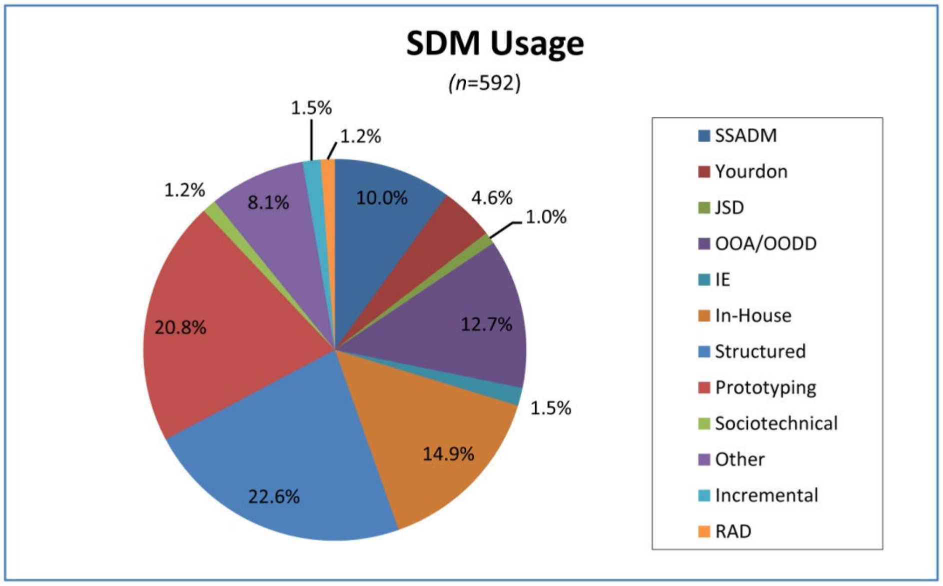
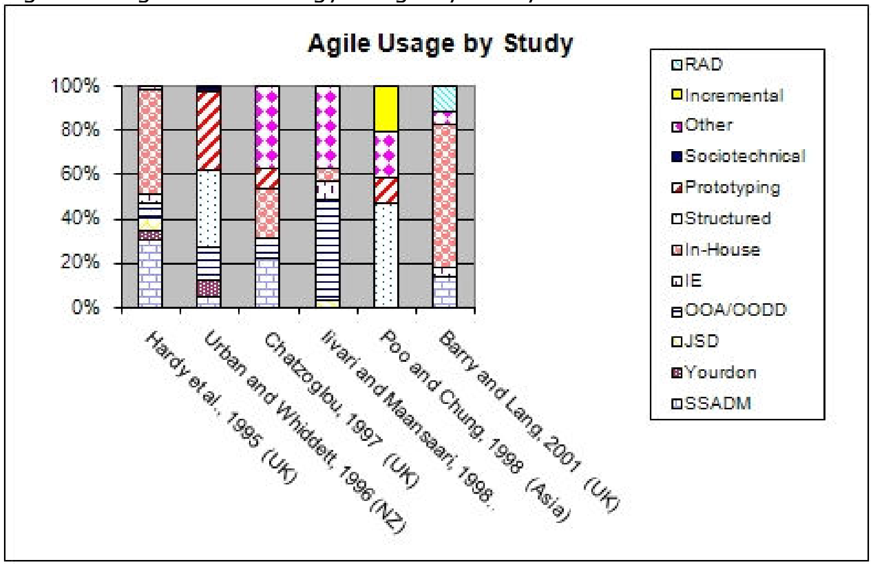


Figure 8: Agile Methodology Usage by Study

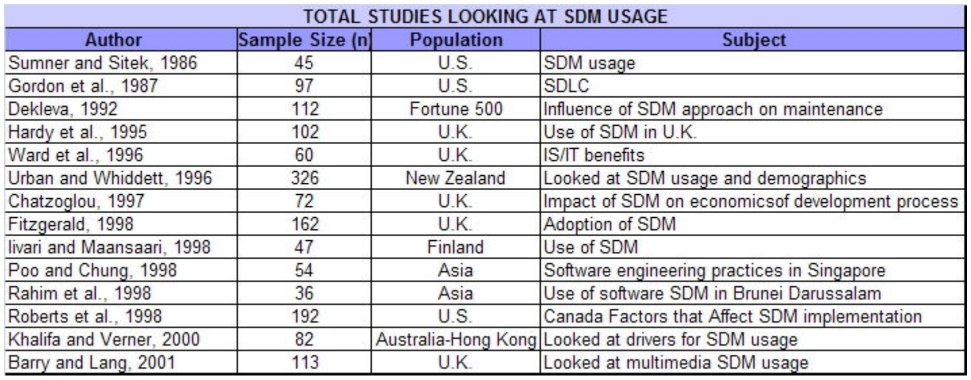


Appendix B – Tables

Table 1: Industry Surveys

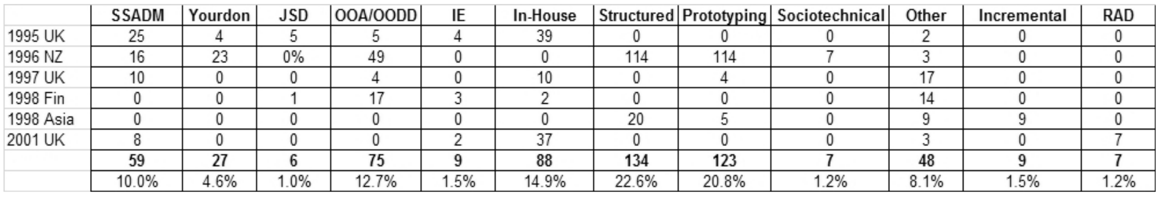
Date	Title	Sample (*n*)	Publisher	Sponsor
2001	Changing Application Development Needs	550	TechRepublic, Inc.	MKS
2003	Agile Methodologies	131	Shine Technologies PtY Ltd.	Shine Technologies PtY Ltd.
2006	Survey Says: Agile works in practice	4,232	Dr. Dobb's Journal	Scott Ambler/Dr. Dobb's Journal
2006	Survey on Agile Adoption	722	VersionOne	Agile Alliance
2007	The State of Agile Development	1,681	VersionOne	APLN

Table 2: Studies looking at SDM Usage



TOTAL STUDIES LOOKING AT SDM USAGE			
Author	Sample Size (n	Population	Subject
Sumner and Sitek, 1986	45	U.S.	SDM usage
Gordon et al., 1987	97	U.S.	SDLC
Dekleva, 1992	112	Fortune 500	Influence of SDM approach on maintenance
Hardy et al., 1995	102	U.K.	Use of SDM in U.K.
Ward et al., 1996	60	U.K.	IS/IT benefits
Urban and Whiddett, 1996	326	New Zealand	Looked at SDM usage and demographics
Chatzoglou, 1997	72	U.K.	Impact of SDM on economicsof development process
Fitzgerald, 1998	162	U.K.	Adoption of SDM
Iivari and Maansaari, 1998	47	Finland	Use of SDM
Poo and Chung, 1998	54	Asia	Software engineering practices in Singapore
Rahim et al., 1998	36	Asia	Use of software SDM in Brunei Darussalam
Roberts et al., 1998	192	U.S.	Canada Factors that Affect SDM implementation
Khalifa and Verner, 2000	82	Australia-Hong Kong	Looked at drivers for SDM usage
Barry and Lang, 2001	113	U.K.	Looked at multimedia SDM usage

Table 3: Data for Methodology Studies



	SSADM	Yourdon	JSD	OOA/OODD	IE	In-House	Structured	Prototyping	Sociotechnical	Other	Incremental	RAD
1995 UK	25	4	5	5	4	39	0	0	0	2	0	0
1996 NZ	16	23	0%	49	0	0	114	114	7	3	0	0
1997 UK	10	0	0	4	0	10	0	4	0	17	0	0
1998 Fin	0	0	1	17	3	2	0	0	0	14	0	0
1998 Asia	0	0	0	0	0	0	20	5	0	9	9	0
2001 UK	8	0	0	0	2	37	0	0	0	3	0	7
	59	**27**	**6**	**75**	**9**	**88**	**134**	**123**	**7**	**48**	**9**	**7**
	10.0%	4.6%	1.0%	12.7%	1.5%	14.9%	22.6%	20.8%	1.2%	8.1%	1.5%	1.2%