Science of Computer Programming 77 (2012) 636–643

Contents lists available at SciVerse ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Regulations and software evolution: An example from the military domain

K. Burak Codur[a,*], Ali H. Dogru[b,1]

[a] Turkish Military Academy, Defense Sciences Institute, 06654 Bakanliklar Ankara, Turkey
[b] Middle East Technical University, Computer Engineering Department, 06531 Ankara, Turkey

ARTICLE INFO

Article history:
Received 19 January 2011
Received in revised form 15 December 2011
Accepted 16 December 2011
Available online 23 December 2011

Keywords:
Software development standard
Software evolution
IEEE/EIA 12207
MIL-STD-498
DOD-STD-2167
MIL-STD-1679

ABSTRACT

In this article, the impact of regulatory changes on software development is assessed in the context of military standards. A previously conducted experiment incorporating three standards is further investigated for this purpose, outlining the characteristics of the evolution in standards and its effects. In addition to this experiment that was designed with projects conducted as graduate class work, a real project from the industry is utilized, to demonstrate the similar effects of the evolution as discovered in the earlier experiment. Finally, the results of the assessment are generalized and a forecast is presented for the next potential regulation change, the IEEE Std 12207-2008.

1. Introduction

The domestic and international markets of information and communication technologies are governed by specific bodies, complying with established rules and mechanisms. Changes in market governance can be explained by power, technology, ideas or domestic politics [3]. Although technology raises the possibility of change, it does not dictate a particular set of changes and it is not the sole enabler of change [3]. Regulations are among the instruments that are used to govern domestic and international markets. A recent example is provided by China, whose policies related to software and standards indicate protectionist tendencies [15].

It is possible to control and direct software development activities utilizing specific regulations, such as software development standards and certification directives. These regulations determine which management practices, which processes and which technologies can be or must be applied during software engineering activities. For example, a regulation may require the use of specific software development process models, such as waterfall, incremental or spiral models, or a regulation like RTCA DO-178B may require the performance of a set of tests on the executable software [14].

A typical example of such regulations is the IEEE/EIA 12207.0-1996 standard. Although this standard was defined by international professional and trade organizations (and thus does not have an obligation initially), government or private bodies can utilize it as a mandatory standard, which is the case with the US Department of Defense (DoD): the IEEE/EIA 12207.0-1996 standard has been the software development standard of the DoD since 1998.

Regulations evolve over time. For the military domain the first software development standard, MIL-STD-1679 (later DOD-STD-1679) was defined in 1978 [9]. It was followed by DOD-STD-2167 (1985) [4], MIL-STD-498 (1994) [10] and

* Corresponding author. Tel.: +90 312 417 51 90x500; fax: +90 3124280930.
E-mail addresses: kbcodur@yahoo.com (K.B. Codur), dogru@ceng.metu.edu.tr (A.H. Dogru).
[1] Tel.: +90 312 210 55 90.

0167-6423/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.scico.2011.12.001

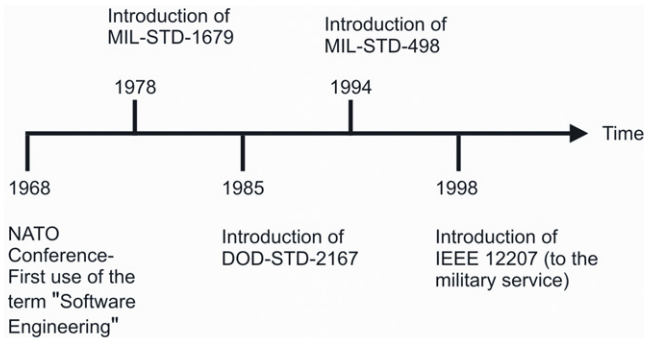
Introduction of MIL-STD-1679
1978
Introduction of MIL-STD-498
1994
Time
1968
NATO Conference- First use of the term "Software Engineering"
1985
Introduction of DOD-STD-2167
1998
Introduction of IEEE 12207 (to the military service)

Fig. 1. Progress of software development standards in the military domain.

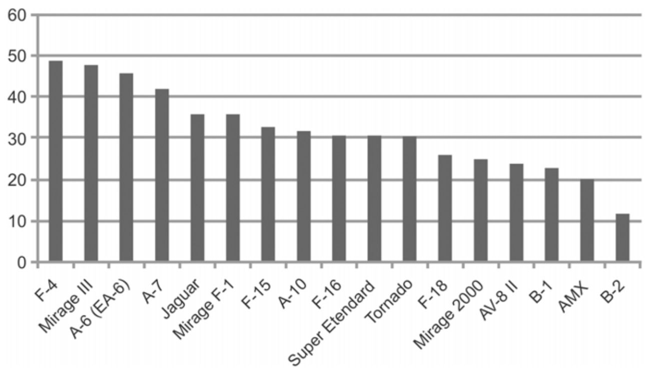

Fig. 2. In service history (in years) of jet aircrafts developed by NATO members.

IEEE/EIA 12207.0-1996 (1998) [5]. Fig. 1 depicts the chronology of the introduction of these standards. A review of these regulations' progress can be the starting point for understanding of their impact on software evolution.

However, although it may be possible to perform qualitative assessments on the progress of such regulations, finding empirical data and making quantitative analysis of the progress is not possible due to the nature of the subject: as regulations change over time, so do the requirements of the projects, the hardware and operating systems that run the software, the software engineering methodologies, the programming languages and the software engineers. Thus it is not possible to isolate the effects of regulatory changes from other factors. In a hypothetical world it might be possible to carry out two instances of a project with exactly the same requirements, software development environment and project team, but in accordance to different regulations; however such experimentation is impossible in the real world. Therefore, in order to perform quantitative analyses about the progress of regulations, special experiments and case studies have been conducted and are investigated in this article.

The military domain provides a field appropriate to research on the progress of regulations. This domain includes the whole spectrum of applications, but it is a monopsony, i.e., a market with a single acquirer: the armed forces. Thus the military domain is a closed and controlled domain. In this particular case, software development standards are the tools for regulation. Another relevant property of the military domain is the longer utilization time of the assets. For example, as depicted in Fig. 2, the average period of utilization of a military combat aircraft extends to thirty years. The software that operates on these aircrafts has a similar lifespan. Aircrafts in civil usage probably have similar utilization potential, but a ten-year old airliner is considered old and a twenty year-old ex-airliner, later adapted as a cargo aircraft, may be considered dangerous to fly. Consequently, a military aircraft may be developed in compliance with one standard and later may have to adapt to a newer standard. This long term utilization of military hardware increases the probability of one or more regulation changes in the life-time of the software. Therefore the connection between the lifespan of the software and the evolution of software development standards is strong in the military domain. Utilizing these unique properties of the military domain, discussions in the leading workshops related to this field of interest, such as the ERCIM Workshop on Software *Evolution* (*EVOL*) and the International Workshop on Principles of Software *Evolution* (IWPSE), can be investigated [2]. Such discussions demonstrate that the effects of regulations on software evolution have been neglected aspects of related research. This paper builds on these discussions, enhancing the research presented at IWPSE-EVOL 2009 with a case study and a projection onto the next standard.

Unlike previous research, this paper underlines the software engineering aspects of the standards' progress, which can be classified as an evolution. Previously, Moore and Rada covered the progress of standards, and stated that "if the 1970s and 1980s were a period of differentiation in life-cycle standards, the 1990s are a period of consolidation" [11]. McDonald recently summarized the history of military software development standards, emphasizing the struggle between DoD and contractors [8]. These studies supplement our research as qualitative analyses on the subject.

In this article, the results of research into the effects of software development standards on software evolution in the military domain are presented. The research can be summarized as three steps:

1. Further interpreting the outcome of our previously conducted experiment [2].
2. Applying the presumed adaptation steps as discovered in the previous experiment, to another case study conducted by Royce [13], and
3. Forecasting the next adaptation expected to render the standard effective in the near future.

Section 2 introduces our previous experiment [2] and discusses its outcome. We conducted this experiment in an academic setting, due to the difficulty of obtaining data from real projects conducted in the military industry. Although the experiment proved very useful in assessing the main characteristics of the evolution of military standards, information about its repercussions in real industry would greatly enhance this research. To conduct a real industry analysis would require information about projects that have been developed more than once, complying with different standards. This is almost impossible. Fortunately, there exists a previously published case study by Royce [13] which exposed related information for one standard (naturally enough, the standard their project complied with). Section 3 describes the adaption steps that would be necessary if this project had to comply with the next standard that was introduced after its delivery. Finally, Section 4 includes comments about the new standard, the IEEE Std 12207-2008 [6]. It is observed that there exists a convergence of software development standards between military and civilian domains. This convergence establishes a basis to generalize the evolution observed in the military domain to a wider perspective.

2. Progress of software development standards in the military domain

This section introduces our preliminary work that was mostly conducted before, and that provided the basic information we use in the investigation of the progress in standards and their influence on software development. As stated in the previous section, four software development standards have been used in the military domain: MIL-STD-1679 (1978–1985), DOD-STD-2167 (1985–1994), MIL-STD-498 (1994–1998), and IEEE/EIA 12207.0-1996 (1998–present). These standards mainly regulate the development phase of software, determining how software development is planned, the requirements that are defined and the software that is designed, coded and delivered to the customer. They also define a set of required documents and provide the templates for these documents. The earlier research on development standards [2] compared these standards using two methods: *content analysis*[2] and *software development experiment.*

The content analysis study compared the standards by referring to their texts. Process models for each standard were generated utilizing the Business Process Model Notation (BPMN, http://www.bpmn.org/). Also, an ontology for standards was created, enabling the researchers to compare the essential and necessary elements such as the definitions, requirements and references, of the standards [2].

In our software development experiment, the same experimental projects were implemented in compliance with each of these standards and adequate data were collected for comparison [2]. The comparison was conducted in the context of an academic course, with student groups acting as project teams. Each project team performed a project (which consisted of planning, requirement analysis and preliminary design phases) in compliance with MIL-STD-1679, DOD-STD-2167 and MIL-STD-498 and each project was performed once (by different project groups) in accordance to the standards. Data such as schedule, effort and lines of code estimates and number of requirements were collected for each project implementation. The data were gathered and the results were statistically analyzed. As stated in the introduction section, since it is not possible to find a real life example in which the same project is implemented according to two or more different military standards, this experiment is unique.

Results of the research provided a detailed comparison between the standards and underlined their progress. Among the major findings, the following are listed and briefly explained below [2]:

- *Change in the subjects focused on*: Standards may have specific foci. For example, MIL-STD-1679 focuses on coding, whereas MIL-STD-498 focuses on safety, security and privacy requirements. These specific focus points may have a significant impact on software with a lifecycle that spans through the jurisdictions of multiple standards. Change in the focus may require modification of software and addition of new functionalities or the performance of specific analyses.
- *Increase in the number of requirements*: Our experimental findings demonstrate that the number of defined software requirements increased as standards progressed. In the experiment [2] the number of requirements defined by project groups increased as the standards progressed (i.e., as the groups started to work with newer standards). The requirements were counted by the authors of this article. Sentences with the "shall" modal verb were taken as requirements sentences. Counting of modal verbs has been employed by previous research, such as Automated Requirements Measurement Tool

[2] Content analysis is a method frequently used for social research, also known as document analysis. The following sources can be referred to for this application of content analysis: "Kalof L., Dan A., Dietz T. Essentials of Social Research, Open University Press, McGraw-Hill, 2008" and "Henn M., Weinstein M., Foard N. A Short Introduction to Social Research, SAGE Publications, 2006".

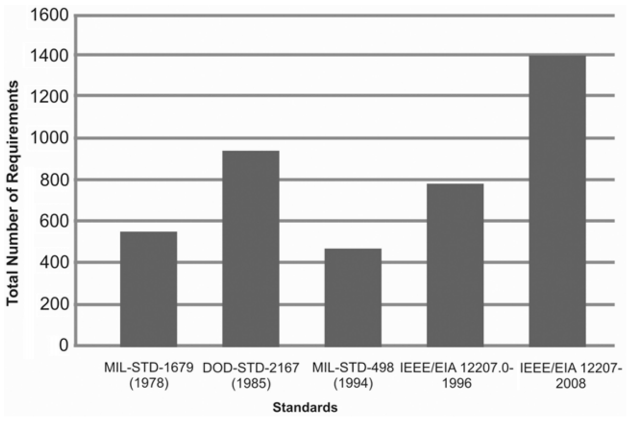

Fig. 3. Number of regulatory requirements defined in the texts of software development standards.

of NASA [12] and by Valeri and Eiche [16]. Although there is a debate on which modal verb to use [7], utilization of the "shall" modal verb to phrase the requirements is mentioned in the standards' explanations. In requirements counting, sentences with multiple requirements were especially considered: Alexander and Stevens [1] state that each requirement sentence should include only one requirement, but the standards do not always comply with this rule. Therefore, in order to get an exact count of requirements, the authors counted separately all the requirements that were included in the sentences. Another indicator of this phenomenon is the number of regulatory requirements defined by the standards. An exact count of sentences with "shall" modal verb provides the number of these, as depicted in Fig. 3 (sentences containing multiple requirements, connected with "and" or commas, count more than once). Here it should be noted that the scopes of IEEE/EIA 12207.0-1996 and IEEE Std 12207 2008 are larger than the military defined standards. Nevertheless, in recent standards, starting with MIL-STD-498 (1994), an increase in the number of requirements is noticeable.[3]

- *Increase in the estimates performed for planning:* In the software development experiment, the estimates made by project groups for effort and project duration increased as the standards progressed (i.e., as the groups started to work with newer standards) [2].
- *Expansion of the scope*: As the software engineering principles and techniques evolve, new concepts are continuously introduced and explored. As a consequence, software development standards necessarily include the majority of these updated notions, principles and techniques. Accordingly, the scope of the standards broadens and the standards contain more concepts as they progress. However as demonstrated in Table 1, some concepts such as "document templates" may, rarely , disappear from the standards. The occasions of such disappearances are very few and can be neglected in the assessment made for the "expansion of the scope".
- *More detailed processes:* When software development standards for the military domain are modeled using BPMN, a repetitive pattern of frequent acquirer audits becomes noticeable [2]. Such activities are counted as "control steps", and are reflected in Fig. 4 along with the number of required deliverable documents for each standard. As Fig. 4 indicates, there has been an increase in both the number of control steps and the number of documents.
- *Increase in the variety of software development models:* In parallel with the expansion of the scope, the standards refer to a variety of software development models. In the earliest standard (MIL-STD-1679), utilization of the waterfall model was necessary. As standards progressed, iterative and incremental models were also included. In our experiment we observed that project groups utilized newer models in their implementations. Based on this observation, we can provisionally assume that variation of software development models in the software development standards will be well received by developers.

This list of findings shows that if evolution of software spreads over a jurisdiction of multiple standards, then we would have to manage this evolution through the differences in the standards. For example, we may have to manage the more detailed software development processes or the increase in the number of requirements.

In this section, we presented the results of our experiment conducted before [2] and also discussed its outcomes. In the next section, a previously conducted military software development project [13] will be used to simulate the effects of these findings.

[3] The variety of requirements in the standards increased as their scope expanded and their processes became more detailed. However, it is possible to observe with the introduction of MIL-STD-498 a sharp decrease in some categories of requirements due to the maturing of the software engineering field, rendering those requirements trivial. For example, for coding and programming, there are 83 requirements in MIL-STD-1679, 49 in DOD-STD-2167 and 19 in MIL-STD-498 and this decrease is compatible with the advances in software development tools and software engineering. There are also similar decreases for other requirement types.

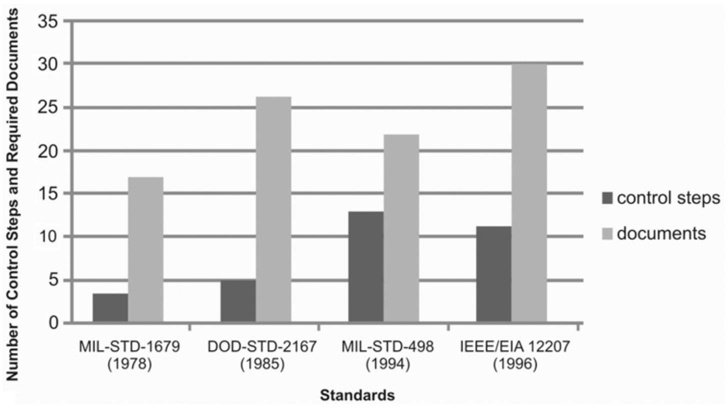

Fig. 4. Number of control steps and number of required deliverable documents for each standard.

Table 1
Coverage of some software engineering concepts by standards.

Concept	MIL-STD-1679	DOD-STD-2167	MIL-STD-498	IEEE/EIA 12207
Undelivered software	X	$\surd$	$\surd$	$\surd$
Incremental and evolutionary software development	X	$\surd$	$\surd$	$\surd$
Tailoring	X	$\surd$	$\surd$	$\surd$
Reuse	X	$\surd$	$\surd$	$\surd$
Independence in tests	X	$\surd$	$\surd$	$\surd$
Traceability	X	$\surd$	$\surd$	$\surd$
Versioning	X	$\surd$	$\surd$	$\surd$
Software transition	$\approx$	$\approx$	$\surd$	$\surd$
Testability	X	$\approx$	$\surd$	$\surd$
Validation	X	X	X	$\surd$
Service	X	X	$\surd$	$\surd$
Stress tests	$\surd$	X	X	$\surd$
User training	X	$\approx$	$\surd$	$\surd$
Document templates	$\surd$	$\surd$	$\surd$	X

('X' indicates no coverage, '$\approx$' indicates partial coverage, "$\surd$" indicates full coverage).
Software Engineering concepts were extracted from the texts of the standards during content analysis.

3. A case study of a development standard transition

This section presents our application of the findings about the update from one standard to a newer one, onto a previously conducted case study. Data about the software aspects of military projects are hard to find because of the secret nature of the projects. On the other hand, examples from military projects would best suit the research presented in the previous section. Fortunately, a detailed case study about a military software development project conducted at TRW has been presented by Walker Royce in the book "Software Project Management" [13]. This is the sole case study openly available that includes both qualitative and quantitative data about the software engineering aspects of a military project.

The project described in the case study is a "Command Center Processing and Display System-Replacement" (CCPDS-R) project. This project is presented as an example of successful project management and software implementation. In the project, software was designed for a ballistic missile warning system. Core of CCPDS-R consisted of 355,000 source lines of code and 6 Computer Software Configuration Items (CSCIs): Network Architecture Services, System Services, Display Coordination, Test and Simulation, Common Mission Processing, and Common Communications.

The project was conducted in 1987 through 1994. As a consequence of this time frame, CCPDS-R was developed in accordance with DOD-STD-2167. In 1994, when the project was completed, MIL-STD-498 was introduced. In this section, we seek to answer the following question: What would be the actions necessary to revise CCPDS-R so that it would be compatible with MIL-STD-498? The case study could not, for obvious reasons, include the answer to this question in its original text. Here, the answer will be sought by analyzing the details of the case study using the results of the research detailed in the previous section. Essentially, then, this section simulates the findings of the previous section on this case study.

The case study mentions its software development standard, DOD-STD-2167, under the subtitle "DOD-STD-2167A Artifacts." As the title suggests, the documentation aspects of the standard are presented and the tailoring that has to be performed to comply with DOD-STD-2167 is described.

Among the major differences between DOD-STD-2167 and MIL-STD-498 is the special focus of MIL-STD-498 on security, safety, and privacy requirements. Other major differences between those two standards are listed in Table 2. Table 3 lists the documentation differences.

Table 2
Comparison of DOD-STD-2167 and MIL-STD-498.

Subject	DOD-STD-2167	MIL-STD-498
Systems Engineering	Systems engineering documents concerning system requirements and system design are mentioned. The software development team reviews these documents and where these documents have not been prepared, the team prepares them.	Systems engineering activities are a part of the process and the standard defines participation of the software development team in these activities.
Transition of software/user training	Transition of software is not defined as an explicit step of the process. Support issues are addressed throughout the development.	Transition of software is a part of the process. Along the preparations for support, software developer is also tasked with installation of software at user sites.
References	Normative references to other standards are given.	No normative references are included.
Development	Application of a top-down approach for software development is an obligation. A waiver is required where this requirement is not implemented.	Tailoring is possible.

Table 3
Comparison of documentation required by DOD-STD-2167 and MIL-STD-498.

Document group	DOD-STD-2167	MIL-STD-498
Systems engineering documents	Operational concept document	Operation concept description
	System/segment specification	System/subsystem specification
		System/subsystem design description
Planning documents	Software development plan	Software development plan
	Software test plan	Software test plan
	Software configuration management plan	(included in software development plan)
	Software quality evaluation plan	(included in software development plan)
		Software installation plan
		Software transition plan
Requirements specifications	Interface requirements specification	Interface requirements specification
	Software requirements specification	Software requirements specification
Design specifications	Software top level design document, software detailed design document	Software design description
	Interface design document	Interface design description
	Data base design document	Database design description
Test description and report documents	Software test description, software test procedure	Software test description
	Software test report	Software test report
Software product documents	Software standards and procedures manual, software programmer's manual	Computer programming manual
	Software product specification	Software product specification
	Version description document	Software version description
Operator manuals	Computer system operator's manual	Computer operation manual
	Software user's manual	Software user manual
	Computer system diagnostic manual	
	Firmware support manual	Firmware support manual
		Software input/output manual
	Computer resources integrated support document	
		Software center operator manual
Trouble reports and change documents	Engineering change proposal	
	Specification change notice	

As the aim is to adopt ready- to-use software to a new standard, development processes are beyond the scope of this analysis. Thus, although explicit inclusion of systems engineering in the process defined by MIL-STD-498 is an important difference, it will not be included in this discussion. Therefore, for the specific purposes of this study, the following three major differences are taken into consideration for the adaptation of CCPDS-R to MIL-STD-498:

1. The specific focus of MIL-STD-498 on security, safety and privacy requirements.
2. A more detailed process for software transition , and
3. Additional documentation.

MIL-STD-498 requires identification of security, safety, and privacy critical CSCIs. If there are such CSCIs, an assurance strategy must be defined, recorded and satisfied. In the case of CCPDS-R, its finished CSCIs would be reviewed to assess their conformity with these requirements and additional documentation would have to be prepared. For example the Common Mission Processing CSCI that contains missile warning algorithms is a critical CSCI from the security and safety points of view. As described by Royce, this CSCI has 15 000 source lines of code (4.2% of the total), none of which is automatically generated. Also, in its original form, a total of 268 tests were implemented for the verification of this CSCI (12% of total number of the verification tests). The cost percentage of the Common Mission Processing CSCI is 4%.

As MIL-STD-498 defines detailed process and documentation for software transition, some additional documentation work for CCPDS-R would again be needed. This would also cover major documentation differences between the two standards, such as the ''Software Installation Plan'' and ''Software Transition Plan''. These two documents would be additional to the 37 documents prepared for the project.

The basic simulation of transition from DOD-STD-2167 to MIL-STD-498 performed using this case study demonstrates that, even for a successfully finished project, there is some transition work to perform when regulations change. Therefore software practitioners should monitor changes in regulations and be prepared to tackle transitions.

4. The next frontier: IEEE 12207 2008

After investigating the evolution in software development based on available standards and case studies, the next step is to speculate about similar developments for a recently introduced standard. In 2008, a new version of IEEE/EIA 12207.0-1996 was introduced as ''IEEE Std 12207-2008.'' In addition to the inclusion of modern software engineering concepts and practices, the aim of this standard is to cover the convergence aspects of software engineering and systems engineering (along the ISO/IEC 15288 standard). However, being so recent this new standard has not yet lead to experimental work. Therefore we can only present our views on the possible effects on the military software development domain as a result of studying the standard alone.

The progress of software development standards seems to continue with IEEE Std 12207-2008. The concepts mentioned in Section 2 — expansion of scope, increase in the number of regulatory requirements and a more detailed process – are all applicable to IEEE Std 12207-2008. Other major changes in IEEE 12207 2008 are discussed below:

- *Ease of adaptation and tailoring:* As mentioned in Section 1.3 of the standard, IEEE Std 12207-2008 ''does not detail the life cycle processes in terms of methods or procedures'', ''does not detail documentation in terms of name, format, explicit content and recording media'', ''does not prescribe a specific system or software life cycle model, development methodology, method, model or technique'' and ''is not intended to be in conflict with any organization's policies, procedures, and standards or with any national laws and regulations''. These explanations have the potential to ease the process of adoption of this standard. Another major enabler for adaptation or tailoring is the lack of document templates in IEEE Std 12207-2008. Software developers will probably not need to prepare new documents or revise already prepared documents.
- *Addition of new processes:* In compliance with the finding about the expansion of scope (mentioned in Section 2), IEEE Std 12207-2008 introduces new processes such as ''Organizational Project-Enabling Processes'' and ''Software Reuse Processes.'' An organization wishing to comply with IEEE Std 12207-2008 but maintaining software that was developed before the introduction of this standard, will probably have to perform transitional work.
- *Roles for the acquirer:* In the acquisition process, IEEE/EIA 12207.0-1996 lists tasks of the acquirer with the ''will'' modal verb, whereas IEEE Std 12207-2008 uses the ''shall'' modal verb. Thus tasks assigned to the acquirer are described with a stronger emphasis. This is a major difference in the progress of software development standards and expands the jurisdiction of the standard to include the acquirer, as well as the more traditional target, the developer.

According to our findings summarized in Section 2, practitioners should expect the continuation of general trends such as expansion of the scope, increase in the number of requirements and a more detailed process. With IEEE Std 12207-2008, differently from previous regulation changes, acquirers should be ready to accept some further obligations. Finally, it will be appropriate to assume that a similar trend may continue with some differences taking place smoothly, while some new activities have already appeared.

5. Conclusions

This research analyzes and presents the outcome of case studies for investigating the evolution of software development standards. It demonstrated that the process defined by the standards became more detailed, the number of requirements increased, the scope was expanded and the variety of software development models increased. These led to an increase in the estimations performed during the software planning phase.

Regulations are an integral part of software engineering. Similar to the evolution of software, regulations also evolve in time. Thus, evolution of regulations is intricately intertwined with software evolution. It can be concluded that no earthshaking expectations surface related with the evolution of software due to the evolution of standards. The result of this research can suggest a predictable adaptation activity based on observed trends in the development of the standards, such as increased requirements, prediction and documentation expectations. These expectations will be valid in cases where a given development extends in time such that one standard has to be considered in the beginning of the project and another at the end.

Domains determine the applicable regulations. The military domain differs from other domains because of its monopsony property, the rigor in the application of regulations and its scope of usage (the military domain includes the whole spectrum of software applications). Thus it is a suitable domain to study the evolution of regulations. However, although the unique properties of the military domain make it suitable to study the evolution of software development standards, there are major obstacles to studying the phenomenon, such as finding empirical data. This article described the evolution of software development standards in the military domain utilizing previously presented research in the form of content analysis and the unique software development experimentation. The consequences of this evolution for software practitioners are identified. Most of the findings about the characteristics of the evolution were obtained during the analysis of the results conducted in the previous experiment [2]. Also it is now clear that these findings can be applied for adapting an already completed project to a newer standard: the adaptation process can be prescribed through pre-determined activity steps. Such an activity is performed in this research, on the outcome of another previously conducted case.

The next standard, IEEE Std 12207-2008 is compatible with parts of this evolution. These parts are more detailed processes, with the addition of new processes and an increase in the number of regulatory requirements. In addition, IEEE Std 12207-2008 seems to be more flexible and adoptable, and it explicitly assigns some tasks to the acquirer.

As the effects of regulation changes on the evolution of software are being realized, the interest will probably shift to the management of this phenomenon. Quantitative data about the level of effort that is needed to execute the transitions between regulations will be sought. Our experiment with military software development standards did not yield a significant pattern related to the working hours of the project teams. Nevertheless level of effort is a parameter to be considered in future work in this field.

The military and civilian domains have mutually affected each other in the past. Thus, it can be claimed that evolution of software development standards in the military and civilian domains have much in common. In fact the change of regulation institution in the military domain (from DoD to EIA and IEEE) demonstrates the convergence between military and civilian domains. The history of the evolution of regulations, as presented here, provides clues to the way ahead for practitioners.

References

[1] I. Alexander, R. Stevens, Writing Better Requirements, Addison-Wesley, 2002.
[2] K.B. Codur, A.H. Dogru, Evolution of software development standards in the military domain and effects on software applications, in: Proc. IWPSE-EVOL'09, Amsterdam, The Netherlands, August 24–25, 2009, pp. 41–45.
[3] P.F. Cowhey, J.D. Aronson, Transforming Global Information and Communications Markets, The MIT Press, 2009.
[4] DOD-STD-2167, Defense system software development, Military Standard, US Department of Defense, 06/04/1985.
[5] IEEE/EIA 12207.0-1996, Industry implementation of international standard ISO/IEC 12207:1995, The Institute of Electrical and Electronics Engineers, Inc., 1996.
[6] IEEE Std 12207 2008, Systems and software engineering—Software life cycle processes, The Institute of Electrical and Electronics Engineers, Inc., 2008.
[7] J.E. Kasser, The first requirements elucidator demonstration (FRED) tool, Systems Engineering 7 (3) (2004) 243–256.
[8] C. McDonald, From, Art form to engineering discipline?: A history of US Military software development standards, 1974–1998, IEEE Annals of the History of Computing 32 (4) (2010) 32–47.
[9] MIL-STD-1679, Software development, Military Standard, US Department of Defense, 01/11/1978.
[10] MIL-STD-498, Software development and documentation, Military Standard, US Department of Defense, 05/11/1994.
[11] J.W. Moore, R. Rada, Organizational badge collecting, Communications of the ACM 39 (8) (1996) 17–21.
[12] L. Rosenberg, T. Hammer, J. Shaw, Software metrics and reliability, in: 9th International Symposium on Software Reliability Engineering, Germany, November 1998.
[13] W. Royce, Software Project Management A Unified Framework, Addison Wesley, 1998.
[14] RTCA/DO-178, Software considerations in airborne systems and equipment certification, RTCA, 1992.
[15] D. Strok, Enter the protectionist dragon? IEEE Software (March–April) (2005) 83–87.
[16] R. Valerdi, B. Eiche, On counting requirements, in: Proceedings CSER 2005, March 23–25, Hoboken, NJ, USA.