Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 27 (2014) 292 – 301

Procedia Computer Science

5th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion, DSAI 2013

Software Process Accessibility in practice: a case study

L. García-Borgoñón[a,b,*], M.A. Barcelona[a,b,*], J.A. García-García[b,**], M.J. Escalona[b,**]

[a] *Instituto Tecnológico de Aragón, c/ María de Luna 7, 50018 Zaragoza, Spain*
[b] *IWT2 Research Group, Universidad de Sevilla, Av. Reina Mercedes s/n, 41012 Sevilla*

Abstract

Software processes are recognized as fundamental assets in development organizations since they support the capability to produce better products. A means for handling the complexity of these processes is through models, and software process modeling languages (SPMLs) are languages to express those processes. Different requirements for SPMLs have been identified by some authors, but accessibility is not one of them. There is little empirical evidence of the use of software processes by people with accessibility difficulties in software organizations. The goal of this case study is to investigate what are the requirements to make software processes become accessible. The subjects are users of a methodology called NDT and its support tools, and who have a kind of disability. The objective is to know the main requirements in order to read and execute software processes and become a process engineer. Conclusions and future work in this field are also presented to improve this area.

Selection and peer-review under responsibility of the Scientific Programme Committee of the 5th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2013).

Keywords: Software Process Modeling; Accessibility; Empirical case study; Qualitative study

1. Introduction

Software systems have become part of our daily life. Developing complex and reliable software applications in the shortest time-to-market with efficiency and effectivity is the most important challenge that software companies are facing everyday. In order to address this challenge, organizations rely on software processes, which are recognized as fundamental assets since they support their capability to produce better products.

A software process is a set of activities, methods and practices used in the production and evolution of software and the associated products [1 2]. Typically processes are often collected on a methodology for the organization, which includes the methods, techniques and support tools and is used for planning, understanding, managing and improving systems and software processes. These processes and methodologies have always been described in appropiate terms

* Corresponding author. Tel.: +34-976-010-000 ; email: laurag@ita.es, mabarcelona@ita.es .
** Corresponding author. Tel.: +34-954-552-852 ; email: julian.garcia@iwt2.org, mjescalona@us.es .

Selection and peer-review under responsibility of the Scientific Programme Committee of the 5th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-exclusion (DSAI 2013).
doi:10.1016/j.procs.2014.02.032

to be used by a developer, but they are often described in manuals or books which project team follow as closely as possible[3].

Software process modeling refers to the activities in creating abstract representations of the methodology and software processes through models, and a software process modeling language is a language used to express process models[4]. Many requirements related to process modeling languages have been identified in the literature[5], such as formality, understandability, expressiveness or usability of processes, but the authors have not treated accessibility as a specific requirement in software process modeling languages and therefore, in software processes and methodologies.

World Wide Web Consortium (W3C)[6] defines Web accessibility as the way that people with different kind of disabilities can use the Web. More specifically, Web accessibility means that people with disabilities can perceive, understand, navigate, and interact with the Web, and that they can contribute to the Web. Web accessibility also benefits others, including older people with changing abilities due to aging. According to this definition, we define accessibility in software process domain as the capacity for people with disabilities to: i) perceive and understand software processes and derived work products; ii) execute the defined processes and finally; iii) design and develop new processes in a software engineering process group.

Upon to the suggestion of the needs for empirical research in software process modeling[7], the purpose of this paper is to analyze which are the accessibility barriers that exist in software processes of an organization through a case study. This study is performed in the context of a methodology and its support tools, named Navigational Development Techniques (NDT)[8]. The empirical data collected and analyzed in this study cover the results of interviews with people with accessibility issues that have been users of NDT.

The remainder of this paper is organized as follows. Section 2 presents the research context of this study and Section 3 summarizes related work on the software process accessibility domain. In Section 4 , the method used for the case study is introduced. Section 5 shows the results of the study, and then, Section 6 offers discussion on these results. Finally, Section 7 provides conclusions and implications for future work.

2. Research Context

The contributors to this research are users of the Navigational Development Techniques (NDT) methodology and its support tools. NDT is a model-driven Web engineering approach. Initially, NDT dealt with the definition of a set of precisely defined metamodels for the requirements and analysis phases. In addition, NDT defines a set of derivation rules, stated with the standard QVT[9], which generates the analysis models from requirements model.

Subsequently, the methodology was improved and nowadays, NDT defines a set of metamodels for every phase of the lifecycle of software development: the feasibility study phase, the requirements phase, the analysis phase, the design phase, the implementation phase, the testing phase, and finally, the maintenance phase. Besides, it states new transformation rules to systematically generate models (these new models are known as basic models). These transformations are identified by the stereotype <<QVTTransformation>>. Figure 1 shows the first part of the NDT lifecycle.

After carrying out these transformations systematic, NDT allows analysts make transformations in order to enrich and complete this basic model. Transformations are represented in Figure 1 through the stereotype <<NDTSupport>>. NDT controls these transformations by means of a set of defined rules and heuristics, to ensure consistency between requirements and analysis models.

An important number of companies in Spain, such as Airbus Military, Icosis, Everis, Sadiel and Fujitsu among others, and some institutions such as the Andalusian Regional Government, and other, work with NDT and the associated tools for software development. This is possible due to the fact that NDT is completely supported by a set of tools, grouped in the NDT-Suite[10]. NDT-Suite works on/with a UML-based profile.

The real life uncovers many problems that should not take place, despite the application of methodologies. In many cases the exact application of methodological phases ends up being a mere formality, whereas in other cases, projects framed in a methodology change or patch code. This fact causes inconsistency between the documentation and the final system.

In the last years, NDT has evolved again and now, in order to offer a suitable and a global solution for the real application of NDT, a global framework named NDTQ-Framework was developed. NDTQ-Framework comprises a set of processes involving development processes, management processes, quality processes, testing processes and

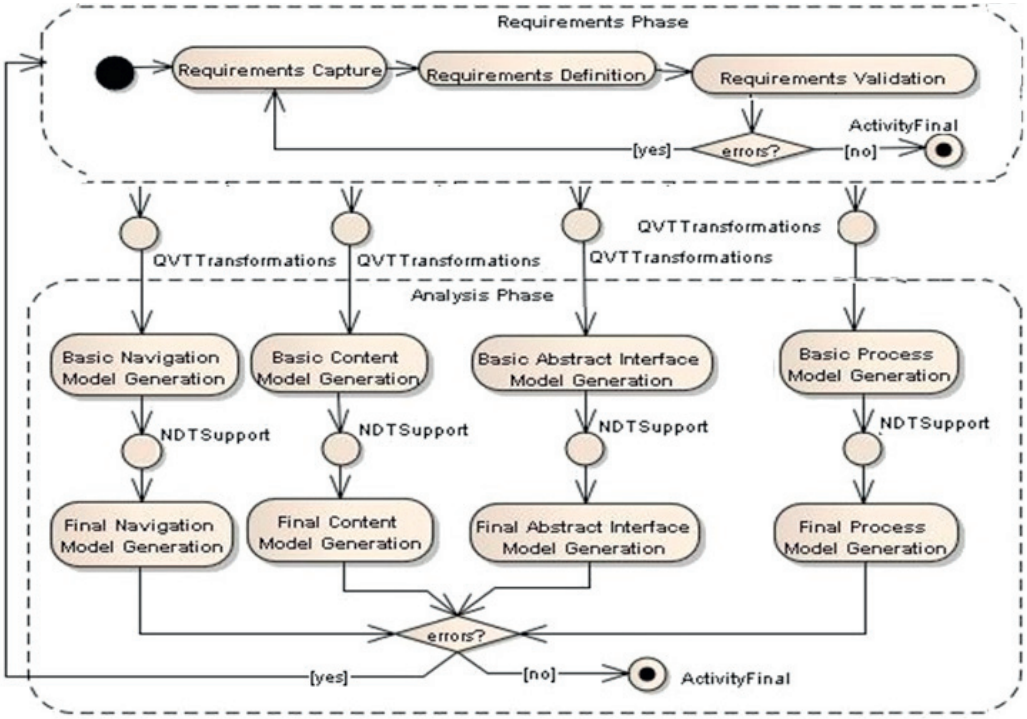

Fig. 1. First part of the NDT sequential lifecycle.

security processes. This environment is based on different reference models like CMMi (Capability Maturity Model Integration)[2] and ITIL (Information Technology Infrastructure Library)[11], and its application in real projects are certificated under different standards such as ISO 27001, ISO 9001:2008, UNE EN 16602 and ISO 14000.

3. Related Work

In this section we present related work in the field of accessibility and software process. Firstly, we found several authors that exposed the importance of facilitating process understandability, as a key to improve communication between stakeholders, and therefore they have included understandability as a requirement in the field of software process engineering, but accessibility was not analyzed.

Arlow and Neustadt[12] argued that not all stakeholders are able to understand a process model which is shown by UML or other graphical interface. They presented some problems regarding visual models: i) it is necessary to know how a modeling tool operates to deep into the model; ii) HTML-based export models are often difficult to read and navigate, reducing their practical use; iii) it can be difficult to determine where to start reading, either when reading the model in a modelling tool; iv) the trivialisation of business requirements by visual modelling, the way the authors called the difficuly to find a business requirement when it is expressed in a visual notation and not in a concise way.

Nicolás et al.[13] presented a systematic review of the literature related to the generation of textual requirements specifications from software engineering models. They argued that both researchers and practitioners can benefit from an improvement in the readability of software engineering models, making these models available to a wider spectrum of stakeholders and thus improving their usability and facilitating their validation.

Bendraou[14] considered understandability as a crucial requirement for Software Process Modeling Languages (SPMLs), because SPMLs and process models cannot be used if they cannot be understood. He also considered semantic richness (the SPML ability to express what is actually performed during software development process), executability and tooling support. He also reviewed[5] requirements related to SPML and showed they varied from facilitating human understanding, to analyzing processes, or to providing an automated execution support.

Secondly, after the publication of the Web Content Accessibility Guidelines (WCAG)[15], there were plenty of papers focused on how people with disabilities can use the Web. If existing software tools use web for creating, reading and executing software process, most of work done in the field of Web Accessibility Initiative (WAI)[16] could be applied in the software process domain. Unfortunately, most of existing software process tools are not Web-Based, specially for process editing, but it could be a way to scan through the improvement in these support tools.

Finally, we found a set of papers that deep into the relationship between accessibility and software process, by adding some practices or user-centered methodologies in order to achieve accessible products. Although they were efforts to achieve process-driven accessibility, they did not focus on getting the processes accessible.

Seffah et al.[17] proposed the integration of the overall system development process and the User-Centered Development process for designing the interactive components. Alarcón et al.[18] argued the use of techniques of accessibility in software process engineering could be reflect in benefits of time, cost and quality. Haesen et al.[19] proposed a conceptual process framework for Multi-disciplinary User-centred Software Engineering (UCSE) processes.

In summary, software processes cannot be used if they cannot be understood, so that understandability is a crucial point to take into account in order to achieve a successful implementation of software process engineering in practice. Although accessibility is a clear barrier to achieve process understandability, we have not found any work in the literature that seeks to address this problem.

4. Research Method

Case Study has been the method chosen to achieve the goal described in Section 1. A case study is an empirical method aimed at investigating contemporary phenomena in their context[20,21]. It is a research method whose key characteristics are that i) it is flexible, coping with the complex and dynamic characteristics of software engineering; ii) its conclusions are based on qualitative, collected from multiple sources in a planned and consistent manner; and iii) it adds to existing knowledge by being based on previously established theory or by building theory[22]. To elaborate this work, the guidelines proposed by Runeson[23], for conducting and reporting case study research in Software Engineering, have been followed.

These guidelines establish that there are five major process steps to conduct a case study: *Case study design*, where objectives are defined and the case study is planned, including research questions and subjects selection, *Preparation for data collection*, where procedures and protocols for data collection are defined, *Collecting evidence*, *Analysis of collected data* and *Reporting*. Due to it is a flexible design strategy, there is a significant amount of iteration over the steps. The data collection and analysis may be conducted incrementally. However, the case study should have specific objectives set out from the beginning.

The activities included in each step will be described in detail in the following sections.

4.1. Research Questions

Software processes are important assets into a company since that these processes gather organization culture, practices and work methods. They must be accessible for everyone in the organization development teams and thus, the research question (RQ) that we proposed was the following one: "What are the main requirements to make software processes become accessible?".

In order to concrete the scope of the case study, the general question was reformulated into three more specific questions, which guided this explotarory research work:

- RQ1. What are the main acccesibility requirements in order to:
  - read software processes and work products?
  - execute software processes and use support tools?
  - be a proccess engineer and define software processes?
- RQ2. According to WCAG 2.0[15], what are the three most important and the three less important requirements for the software process engineering domain?
- RQ3. What should be the main future changes to be developed to improve the accessibility of software processes?.

4.2. Data Collection Procedure

As previously mentioned, the research participants for this study are people who have used the NDT methodology and its support tools and have some accessibility difficulties. They are computer engineers, and have been used in practical NDT from 2 to 5 years as users of NDT or members of the NDT process definition group.

Semi-structured interviews were conducted with questions on the accessibility in NDT Suite and NDTQ-Framework. The interview was designed and conducted by a researcher who was not subject to the case study. Each interview lasted between 10 to 15 min and was recorded in full as a document file. The text and conclusions were sent to interviewees for comments and corrections. The full list of questions is provided in Appendix A.

4.3. Analysis Procedure

The number of responses in this study is too low to generate any statistics. The results are, however, important when interpreted in the context of case studies, especially since there have been few empirical studies performed on this field. Thus, a qualitative data analysis method was used. The basic objective of the analysis is to derive conclusions from the data, keeping a clear chain of evidence. The chain of evidence means that a reader should be able to follow the derivation of results and conclussions from the collected data[21].

There are two different parts of data analysis of qualitative data, hypotheses generating techniques and hypotheses confirmation techniques[24]. The first one is intended to find hypotheses from the data and the second one can be used to confirm that a hypothesis is really true.

Then we will present the hypotheses we had before conducting the interviews. In order to specify accessibility requirements, we have classified them into these elements: i) process description; ii) process workflow and; iii) derived work products and support tools.

Process description includes the textual definition of roles, tasks and work products. This topic is sometimes embedded in the methodology as a set of processes or procedures, and is recorded in what the organizations call quality manual. Process workflow covers the sequence of tasks to achieve a goal and is usually described graphically by workflows, business processes diagrams or UML Activity Diagrams. Derived work products means the set of documents, artifacs, deliverables derived from the execution of tasks. Some of them are generated manually from stakeholders but others are created by using support tools which are included in this group. Table 1 shows our hypotheses regarding main accessibility requirements for every element. Although the importance of compliance varies depending on the role (read, execute or software process definition), all of them should be considered to achieve accessibility in the field of software process engineering. When we refered to WAI-compliant, it means that taking into account elements of Web Accessibility[16], a way of translate a workflow sequence and UML diagrams into text should be investigated. Human-Computer Interface (HCI) of support tools should follow the recommendations of accessibility as if they were a website.

Table 1. Hypotheses regarding main Accessibility Requirements

Element	Requirement	To Read	To Execute	To Define
Process description	Text Alternatives	X		
	Adaptability	X		
Process workflow	Navigability	X	X	
	WAI-compliant workflow sequence	X	X	X
Word products and support tools	WAI-compliant UML diagrams			X
	WAI-compliant HCI of support tools		X	X
All	Keyboard Accessible	X	X	X

According to WCAG 2.0[15], in our opinion the three most important requirements to be considered to achieve software process accessibility should be: i) 1.1 Text Alternatives; ii) 2.1 Keyboard Accessible and; iii) 2.4 Navigable. The three less important requirements should be: i) 1.2 Time-based Media; ii) 2.2 Enough Time and; iii) Input Assistance. Regarding main future changes we consider the following:

- Process description should be exported to WCAG 2.0[15] websites.

- Support tools should offer a web-based Human-Computer Interface.
- A way of translate workflows and UML diagrams into accessible text should be investigated.
- Software Process Modeling Languages should include accessibility fields.

5. Results

In this Section the main results derived from the interviews are shown. Firstly, a brief description of interviewees is presented in Table 2. Then, accessibility problems reported by interviewees when using NDT are shown in Table 3. Considering WCAG 2.0[15], the three most and less important requirements in the software process domain are presented in Table 4. Finally, Table 5 shows the main future changes proposed by interviewees.

To facilitate information understandability Tables show the responses grouped by type of disability: physical (persons 1 and 3) and visual (persons 2 and 4).

Table 2. General description of interviewees

	Person 1	**Person 2**	**Person 3**	**Person 4**
Accessibility Problem				
Visual		X		X
Physical	X		X	
Auditory	X			
NDT Experience				
Years using NDT	3	2	5	2
Read processes	X	X	X	X
Execute processes	X	X	X	X
Define processes		X	X	X

Table 3. Main accessibility problems reported by interviewees when using NDT

Person	**Id**	**Accessibility problems when using NDT**
Person 1		No problems at all
Person 3		No significant problems
	PB1	It may be interesting to improve the user experience when users navigate through the GUI tools
Person 2	PB2	Process navigability
	PB3	How to get the result after NDT process testing
Person 4	PB4	The main problem for visual impaired with NDT and software development are graphical models
	PB5	Sometimes people with visual problems can not see the whole model or have problems following relationships
	PB6	It is too difficult to represent a UML diagram or BPMN diagram in clear text, speech or braille

6. Discussion

According to the qualitative data analysis method used, in this Section hypotheses confirmation techniques[24] are discussed from the results obtained from interviews. Firstly, the coverage we have had with interviewees is shown.Then, we discuss every Research Question.

6.1. Research Coverage

As it was exposed in Section 4.3, the number of data retrieved in this study was too low to generate any statistics but, in our opinion, results are very relevant for an empirical case study in a concrete domain.

Table 2 summarizes that we have interviewed 4 people, two with a physical disability and two with impaired vision, thus covering the two kinds of major problems in the field of accessibility. Because only one of them had hearing

Table 4. WCAG 2.0 importance in software process domain

Problem	Person 1	Person 3	Person 2	Person 4
1.1 Text Alternatives			+	+
1.2 Time-based Media		-	-	-
1.3 Adaptable			+	+
1.4 Distinguishable	+	-		
2.1 Keyboard Accessible	+	+		
2.2 Enough Time	-		-	-
2.3 Seizures				
2.4 Navigable	+	-		
3.1 Readable		+	+	
3.2 Predictable	-		-	-
3.3 Input Assistance	-	+		
4.1 Compatible				+

Table 5. Main future changes proposed by interviewees

Person	Id	Main future changes proposed
Person 1	FC1	Improve HCI of software tools to be the most intuitive possible
Person 3	FC2	Improve graphical canvas management by using keyboard shortcuts
Person 2	FC3	Allow user to configure accessibility in the software process domain according to his needs
Person 4	FC4	Software processes should be delivered in a more dynamic format (not doc or pdf).
	FC5	Alternatives to UML are needed (textual or hierarchicals)

problems we have decided to remove the barrier of the study. They have been used in practical NDT from 2 to 5 years and they have played the role as an user (to read and execute) and a member of the process group.

As a conclusion, although the sample is not large we think that it completely covers the types of problems that can help to validate the hypotheses and to answer Research Questions.

6.2. RQ1. What are the accessibility requirements

Table 6 compares initial hyphoteses to data extracted from interviews regarding main accessibility requirements. We argued that the best way to elicit the needs was through asking the major problems encountered and future changes suggested by interviewees, as shown in Appendix A.

For this reason, the Table 6 shows the relationship between: the requirements shown in 1; the problems reported from people and detailed in Table 3; the main future changes requested and shown in Table 5. We can conclude that all our initial thoughts have been confirmed.

Table 6. Initial hyphotesis vs interviewees: Main Accessibility Requirements

Element	Hypothesis	Id PB	Id FC
Process description	Text Alternatives		FC4
	Adaptability		FC4
Process workflow	Navigability	PB2	
	WAI-compliant workflow sequence	PB4, PB6	
Word products and support tools	WAI-compliant UML diagrams	PB5, PB6	FC5
	WAI-compliant HCI of support tools		FC1
All	Keyboard Accessible		FC2

6.3. RQ2. Three most and less important WCAG requirements in software process engineering domain

Table 7 shows the comparison of our initial hyphotesis and interviewees regarging WCAG 2.0 importance in software process domain. We can conclude that the most important are: i) 1.1 Text Alternatives; ii) 2.1 Keyboard Accessible and; iii) 1.3 Adaptable. The less important are: i) 1.2 Time-based Media; ii) 2.2 Enough Time and; iii) 3.2 Predictable.

In view of the results we can see that although there is some general agreement grouping by type of disability, there are nuances that come from the degree of disability or the specific role of each person.

Table 7. Initial hyphotesis vs interviewees: WCAG 2.0 importance in software process domain

Problem	Hypothesis	Physical	Visual	Conclusion
1.1 Text Alternatives	+		++	+
1.2 Time-based Media	-	–	–	-
1.3 Adaptable			++	+
1.4 Distinguishable		+-		
2.1 Keyboard Accessible	+	++		+
2.2 Enough Time	-	-	–	-
2.3 Seizures				
2.4 Navigable	+	+-		
3.1 Readable		+	+	
3.2 Predictable		-	–	-
3.3 Input Assistance	-	-+		
4.1 Compatible			+	

6.4. RQ3. Main future changes to be developed to improve the accessibility of software processes

Our initial hypotheses raised in relation to major future changes to be developed to improve the accessibility of software processes have been validated and fully agree with the views of those interviewed. In our proposals should be added the use of keyboard shortcuts to manage graphical tools. Table 8 shows the relationship between our hypotheses presented in Section 4.3 and data obtained in the interviews and detailed in the Table 5.

Table 8. Main Future Changes Hypotheses Validation

Hypothesis	Id FC
Support tools should offer a web-based Human-Computer Interface.	FC1
Software Process Modeling Languages should include accessibility fields.	FC3
Process description should be exported to WCAG 2.0[15] websites.	FC4
A way of translate workflows and UML diagrams into accessible text should be investigated.	FC5

7. Conclusions and Future Work

This paper has presented a study that analyses the suitability of a methodological solution for software process management to user with different kind of disability. The paper is focused on analysis this suitability in users that have used or using NDT and its support tools. All of them are computer engineers and it is very important to conclude that this study presents results oriented to people with a good knowledge about software process. For this aim, an interview was developed and we ask each of them to answer without know the rest of participant. We counted with four people with physical, hearing and visual disabilities. This number seems small but it offers a suitable set to value NDT and its tools suitability because the subject of the study is not frequent and it is very restrictive: computer engineers with experience in NDT and with a grade of disability upper than 50%.

The main conclusion obtained is that NDT and NDT tools do not present special constraints for our subjects under study. They refers the same limitations than other computer engineers in the use of software processes and tools: to understand requirements, to assure the quality of models, etc. In fact, with a suitable hardware adapted to their disability they do not find special limitations in their use.

As a future work, we want to increase this study with other computer engineers with other kinds of disability in order to assure our results. Besides, another important line is to replicate this study under the point of view of the final user, and not under computer engineers. In this work we only consider computer engineers that use software processes but these software processes generate results and deliverable that have to be validate with final users and non-computer engineers (for instance, a business expert). Thus, we want to analyses how suitable are our software processes deliverable for final users with different kinds of disability.

Acknowledgements

This work has been supported by the projects TEMPROS (TIN2010-20057-C03-02) and Test4DBS (TIN2010-20057-C03-01) of Ministerio de Educación y Ciencia, Spain, and the NDTQ-Framework project (TIC-5789) of the Junta de Andalucía, Spain.

Appendix A. Interview Template

1. Description of the accessibility problems that manifest the interviewee.
2. Description of experience of using NDT framework.
3. Description of accessibility problems when using NDT framework.
4. In your opinion, according to WCAG 2.0, what are the three most important and the three less important requirements when using NDT. Plese consider the quick reference of WCAG 2.0[15] and the WCAG checklist[25] as follows:

    - 1.1 Text Alternatives: Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.
    - 1.2 Time-based Media: Provide alternatives for time-based media.
    - 1.3 Adaptable: Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
    - 1.4 Distinguishable: Make it easier for users to see and hear content including separating foreground from background.
    - 2.1 Keyboard Accessible: Make all functionality available from a keyboard.
    - 2.2 Enough Time: Provide users enough time to read and use content.
    - 2.3 Seizures: Do not design content in a way that is known to cause seizures.
    - 2.4 Navigable: Provide ways to help users navigate, find content, and determine where they are.
    - 3.1 Readable: Make text content readable and understandable.
    - 3.2 Predictable: Make Web pages appear and operate in predictable ways.
    - 3.3 Input Assistance: Help users avoid and correct mistakes.
    - 4.1 Compatible: Maximize compatibility with current and future user agents, including assistive technologies.

5. In your opinion, what should be the main future changes to be developed to improve the accessibility of software processes?.

References

1. W. S. Humphrey, Managing the Software Process (Hardcover), Addison-Wesley Professional, 1989.
2. M. B. Chrissis, M. Konrad, S. Shrum, CMMI for Development: Guidelines for Process Integration and Product Improvement, Pearson Education, 2011.
3. B. Henderson-Sellers, C. Gonzalez-Perez, A comparison of four process metamodels and the creation of a new generic standard, Information and Software Technology 47 (1) (2005) 49–65.
4. K. Z. Zamli, P. A. Lee, Taxonomy of process modeling languages, in: Computer Systems and Applications, ACS/IEEE International Conference on. 2001, IEEE, 2001, pp. 435–437.
5. R. Bendraou, J.-M. Jézéquel, M.-P. Gervais, X. Blanc, A comparison of six uml-based languages for software process modeling, Software Engineering, IEEE Transactions on 36 (5) (2010) 662–675.
6. W3C, http://www.w3.org/WAI/intro/accessibility.php, Introduction to Web Accessibility, last accessed 07-2013.
7. X. Bai, H. Zhang, L. Huang, Empirical research in software process modeling: A systematic literature review, in: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, IEEE, 2011, pp. 339–342.
8. M. J. Escalona, G. Aragon, NDT. A Model-Driven Approach for Web Requirements, IEEE Transactions on Software Engineering 34 (3) (2008) 377–390. doi:http://doi.ieeecomputersociety.org/10.1109/TSE.2008.27.
9. OMG, http://www.omg.org/spec/QVT/1.1/, QVT, Meta Object Facility (MOF) 2.0 Query/View/Transformation (2011).
10. J. A. García-García, M. A. Ortega, L. García-Borgoñon, M. J. Escalona, NDT-Suite: A Model-Based Suite for the Application of NDT, in: ICWE, 2012, pp. 469–472.
11. A. de Jong, A. Kolthof, Fundamentos de ITIL, Vol. 3, Van Haren Publishing, 2008.
12. J. Arlow, I. Neustadt, Enterprise Patterns and MDA: Building better software with archetype patterns and UML, Addison-Wesley Professional, 2004.
13. J. Nicolás, A. Toval, On the generation of requirements specifications from software engineering models: A systematic literature review, Information and Software Technology 51 (9) (2009) 1291–1307.
14. R. Bendraou, Uml4spm: Un langage de modélisation de procédés de développement logiciel exécutable et orienté modèle, Ph.D. thesis, Université Pierre et Marie Curie (UPMC), type : Thèse de Doctorat – Soutenue le : 2007-09-06 – Dirigée par : Gervais, Marie-Pierre – Encadrée par : BLANC Xavier (2007).
15. W3C, http://www.w3.org/WAI/WCAG20/quickref, Web Content Accessibility Guidelines Quick Reference, last accessed 07-2013.
16. W3C, http://www.w3.org/WAI/, Web Accessibility Initiative, last accessed 07-2013.
17. A. Seffah, E. Metzker, The obstacles and myths of usability and software engineering, Communications of the ACM 47 (12) (2004) 71–76.
18. H. F. Alarcón, A. M. Hurtado, C. Pardo, C. A. Collazos, F. J. Pino, Integración de técnicas de usabilidad y accesibilidad en el proceso de desarrollo de software de las mipymes, Avances en sistemas e informatica 4 (3) (2007) 149–156.
19. M. Haesen, K. Coninx, J. Van den Bergh, K. Luyten, Muicser: A process framework for multi-disciplinary user-centred software engineering processes, in: Engineering Interactive Systems, Springer, 2008, pp. 150–165.
20. C. Robson, Real world research: A resource for social scientists and practitioner-researchers, Vol. 2, Blackwell Oxford, 2002.
21. R. K. Yin, Case study research: Design and methods, Vol. 5, Sage, 2009.
22. C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, A. Wessln, Experimentation in software engineering, Springer Publishing Company, Incorporated, 2012.
23. P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering 14 (2) (2009) 131–164.
24. C. B. Seaman, Qualitative methods in empirical studies of software engineering, Software Engineering, IEEE Transactions on 25 (4) (1999) 557–572.
25. W3C, http://www.w3.org/WAI/GL/2005/06/checklist-proto.html/, Web Content Accessibility Guidelines 2.0 Checklist, last accessed 07-2013.