Journal of Universal Computer Science, vol. 15, no. 17 (2009), 3169-3200
submitted: 4/6/09, accepted: 2/9/09, appeared: 1/11/09 © J.UCS

A SWEBOK-based Viewpoint of the Web Engineering Discipline

Antonio Navarro
(Universidad Complutense de Madrid, Madrid, Spain
anavarro@sip.ucm.es)

Abstract: Despite web engineering being an emerging discipline, there is currently an important array of literature on this subject. The aim of this paper is to provide a software engineering-based view of the web engineering discipline reviewing and classifying a significant part of the software engineering-related literature that makes up its body of knowledge. In order to facilitate the classification of this software engineering literature, this paper categorizes it into knowledge areas, providing a brief analysis of each area. These knowledge areas match the knowledge areas defined in the Guide to the Software Engineering Body of Knowledge (SWEBOK). As an immediate consequence of this paper, a comparison between software engineering and web engineering disciplines arises.

Keywords: Design, Management, Measurement, Web Engineering, Software Engineering Body of Knowledge, Web Engineering Body of Knowledge
Category: D.2

1 Introduction

Web engineering is an emerging discipline that appeared as a result of the importance that the development of web applications has acquired in the last few years [Kappel, 04]. Murugesan et al. [Murugesan, 01b] define this discipline as: "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of web-based applications or the application of engineering to web-based software" [Deshpande, 01; Deshpande, 02]. Thus, this definition of web engineering is very similar to the definition of software engineering provided by the IEEE Std. 610.12-1990: "the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software" [IEEE, 90].

At present, there is a significant amount of literature on web engineering. As web engineering is made up of different disciplines [Barta, 98; Deshpande, 02; Gellersen, 97; Ginige, 01; Murugesan, 01b; White, 96], this literature can be classified according to the different components that make up the web engineering discipline. For example, Ginige and Murugesan [Ginige, 01] identify the following disciplines as constituents of the web engineering discipline: systems analysis and design, software engineering, hypermedia and hypertext engineering, requirements engineering, human-computer interaction, user interface development, information engineering, information indexing and retrieval, testing modeling and simulation, project management and graphic design and presentation.

In order to determine the influence of each constituent discipline, this paper classifies more than seven hundred papers published in the *International Journal of*

Web Engineering and Technology, the *Journal of Web Engineering*, the *International Conference on Web Engineering*, and the Web Engineering Tracks of the *World Wide Web Conference*.

As software engineering seems to be one of the most important constituents of web engineering, this paper provides a classification of part of software engineering-related web engineering literature according to several software engineering knowledge areas. For this classification, web engineering literature outside of the software engineering discipline (e.g. information systems literature) was not considered. Thus, this paper provides a partial vision of the web engineering discipline because it does not analyze the web engineering papers not related to the software engineering discipline. The knowledge areas used to make the classification match the knowledge areas presented in the *Guide to the Software Engineering Body of Knowledge* (SWEBOK) [Abran et al. 2004]. This approach follows the trend proposed by Kappel et al., who suggest the classification of the web engineering discipline according SWEBOK's knowledge areas [Kappel, 04].

The Guide to the SWEBOK is an IEEE-led project that provides an explicit characterization of the boundaries of software engineering [Abran, 04]. Although the SWEBOK is not unanimously recognized as an unquestionable body of knowledge in software engineering [ACM, 00; Saiedian, 02], in practice, it has encountered reasonable acceptance [Callahan, 02; Carrington, 05].

The SWEBOK body of knowledge is subdivided into ten software engineering knowledge areas plus an additional chapter that provides an overview of the knowledge areas of closely related disciplines. The descriptions of knowledge areas are designed to discriminate between the various important concepts, thereby allowing readers to find their way to subjects of interest quickly. SWEBOK knowledge areas are: software requirements, software design, software construction, software testing, software maintenance, software configuration management, software engineering management, software engineering process, software engineering tools and methods, software quality and related disciplines.

The objectives of the Guide to the SWEBOK are: (i) to promote a consistent view of software engineering worldwide; (ii) to clarify the place (and set the boundaries) of software engineering in relation to other disciplines; (iii) to characterize the contents of the software engineering discipline; and (iv) to provide a foundation for curriculum development and for individual certification and licensing material.

The objectives of this paper for the web engineering discipline are not as ambitious as the objectives of the Guide to the SWEBOK for the software engineering discipline, but to some extent, this paper can help to achieve them.

This paper aims to provide a software engineering-based view of part of the web engineering discipline through a set of commented references. Thus, in this paper a person with a background in software engineering can find a guide to a significant part of the web engineering body of knowledge in the context of a software engineering body of knowledge. To obtain an in-depth view of any area, the references specified in the area should be analyzed. To obtain a view midway between the descriptions provided by this paper, and the analysis of a complete knowledge area, valuable books about web engineering discipline can be analyzed [Casteleyn, 09; Kappel, 06; Mendes, 05; Murugesan, 01a; Rossi, 07; Suh, 05]. The main advantage of this paper over these books is its size. These contain hundreds of

pages on the web engineering discipline. On the other hand, this paper provides a software engineering-oriented view of the web engineering discipline in just a few pages. However, the main drawback of this paper in comparison to these books is the less in-depth analysis provided for each knowledge area (although, this paper includes a great amount of references per area). In addition, these books cover some topics beyond software engineering discipline, although, in our opinion, they are also strongly focused on software engineering topics.

Considering the different disciplines that make up the web engineering discipline, and according to Ginige, Murugesan and Pressman [Ginige, 01; Pressman, 04], web engineering can be considered as a new emerging discipline in its own right, rather than subsumed under software engineering. Following this trend, different authors [Kappel, 04; Navarro, 05] suggest that it would be interesting to analyze the need for developing a *Web Engineering Body of Knowledge* (WEBOK), inspired by SWEBOK, but taking those disciplines not included in software engineering. In spite of our previous position [Navarro, 05], this paper does not attempt to state whether web engineering material should be subsumed under bodies of knowledge of every constituent discipline (e.g. the Guide to the SWEBOK in software engineering discipline) or a new guide to the web engineering body of knowledge, encompassing the different disciplines of which it is formed, should be developed. This is a significant decision for the web engineering discipline that must be made in the context of an international committee of researchers rather than by an individual. In any case, this paper could be one of the items that could help to make such a decision.

Thus, if a new web engineering body of knowledge is developed, as long as this body of knowledge includes software engineering knowledge areas, the knowledge areas presented in the Guide to the SWEBOK could be updated and/or removed. On the other hand, if the web engineering references are subsumed in existing bodies of knowledge of every discipline, as long as the Guide to the SWEBOK can subsume the software engineering references of the web engineering discipline, new knowledge areas or topics could appear in this Guide to the SWEBOK. This paper follows a policy of neutrality and the knowledge areas presented in the Guide to the SWEBOK remain unchanged. Therefore, the web engineering literature more closely related to software engineering is classified according to the taxonomy induced by the SWEBOK.

In addition, as the software engineering discipline is a mature discipline compared to web engineering discipline, the references used in the Guide to the SWEBOK are mainly books and these references represent "generally accepted research" [Abran, 04]. On the other hand, this paper includes several works regarding key issues in web engineering, and at present, some of them are still not considered generally accepted research. The selection criterion for the references included in this paper has been kept as simple as possible: (i) where possible, books have been chosen instead of journal papers. These books are flagship references in their area (e.g. [Alur, 03]), or they represent the latest state of the art in some technology (e.g. [Burke, 06]); (ii) where possible, journal papers have been chosen instead of conference papers. These papers have been selected by their suitability to the knowledge area (e.g. [Lowe, 03]) or because their authors are outstanding figures in their area (e.g. [Mendes, 01]); (iii) where no other references were available, conference papers were

chosen. The selection criterion for these papers is the same as the criterion for journal papers.

Thus, after providing a classification of the references, the following sections analyze different SWEBOK knowledge areas. Each section briefly discusses the main concepts of the selected area and provides some representative references in the discipline of web engineering. As in the Guide to the SWEBOK, the references of each knowledge area are classified according to their nature [Abran, 04]: recommended references, list of further readings and list of standards. Likewise in the Guide to the SWEBOK, quotes in square brackets "[ ]" ranging from section 3 to section 7 identify recommended references, while those in parentheses "( )" identify the usual references used to write or justify the text[1] [Abran, 04]. In the case of recommended references, as far as possible, their main contribution to the web engineering domain is made explicit. In addition, as in the case of the SWEBOK, they are listed in matrixes of knowledge areas vs. referenced material [Abran, 04] highlighting their contribution to every knowledge area. Regarding references outside of the list of recommended references, as in the SWEBOK, they are only used as support material [Abran, 04]. Finally, the conclusions are presented.

The work carried out in the development of this paper, and the opinions expressed in it (except those extracted from the referenced material), are the result of the effort of one individual. Therefore, neither IEEE, nor any other entity involved in the development of the SWEBOK, has been involved in the development of this paper.

2 Classification of References

This section classifies more than seven hundred papers published in the *International Journal of Web Engineering and Technology* (up to vol. 5, no.3) [IJWET, 09], the *Journal of Web Engineering* (up to vol. 8, no.4) [JWE, 09], the *International Conference on Web Engineering* (up to ICWE 2009) [ICWE, 2009], and the Web Engineering Tracks of the *World Wide Web Conference* (up to WWW 2009) [WWW, 2009].

The following categories were used to classify the references: accessibility, agents, document and text processing, e-commerce, e-learning, hypermedia and hypertext, human-computer interaction, information systems, programming languages, semantic web, software engineering, web engineering fundamentals, security, and others. These categories were selected according to the disciplines pointed out by [Deshpande, 02; Ginige, 01; Murugesan, 01b] as well as the topics identified in the analyzed conferences and journals. In this paper, these categories are considered as the *constituent disciplines* of the web engineering discipline.

Initially, in order to simplify this categorization, only one discipline was chosen as the key indexing discipline. Full papers, short papers, posters and keynotes were considered. In addition, because the categorization mechanisms used in these conferences and journals were not homogeneous, such mechanisms were unified during the categorization made in this paper. Thus, it is possible that a set of papers

[1] In the remaining sections (i.e. Introduction and Conclusions) the regular reference format of this journal is used.

included in a topic at a conference, may be split into different disciplines during the analysis.

As a result of this classification, software engineering (including analysis and design, requirements, testing and project management) seems to be one of the most relevant constituents of the web engineering discipline. Thus, almost 39% of the published papers were about software engineering topics. Information systems (including indexing and retrieval) were identified as another important constituent of the web engineering discipline. Thus, almost 14% of the published papers were about information systems topics. The percentage of papers about the remaining constituent disciplines range from 1% on programming languages to 10% on semantic web. Finally, almost 11% of the analyzed papers were not classified into any specific discipline.

Discipline	**Number of papers (percentage) considering one main indexing discipline**	**Number of papers (percentage*) considering up to three indexing disciplines**
Software Engineering	279 (38.9%)	329 (45.8%)
Information Systems	98 (13.6%)	211 (29.4%)
Semantic web	68 (9.5%)	82 (11.4%)
HCI	31 (4.3%)	52 (7.2%)
Document and text processing	29 (4.0%)	64 (8.9%)
e-learning	29 (4.0%)	31 (4.3%)
Agents	22 (3.1%)	28 (3.9%)
Hypermedia and hypertext	20 (2.8%)	33 (4.6%)
e-commerce	16 (2.2%)	21 (2.9%)
Security	16 (2.2%)	24 (3.3%)
Accessibility	13 (1.8%)	22 (3.1%)
Web engineering fundamentals	9 (1.3%)	9 (1.3%)
Programming languages	6 (0.8%)	10 (1.4%)
Others	82 (11.4%)	82 (11.4%)
TOTAL	718 (100%)	n/a

*percentage on 718 papers

Table I: Number of papers per constituent discipline of web engineering.

The previous analysis does not attempt to make an exhaustive classification and indexing of web engineering literature. The analysis does not aim to make a precise

ranking of importance of constituent disciplines of web engineering either. Its principal aim is to obtain an approximated view of the importance of every constituent discipline in web engineering. Thus, the analysis clearly indicates that software engineering is a very important part of the web engineering discipline. In addition, the analysis indicates that information systems are another important discipline in web engineering. But of course, these are not the only constituent disciplines in web engineering.

Secondly, a new categorization was made taking into account up to three different classification disciplines. With this new classification, the number of papers of every discipline was increased, but the overall impact of every discipline remained, up to some extent, very similar. Thus, Table I (second column) depicts the number of papers classified under the constituent disciplines, considering one main indexing discipline, and two additional indexing disciplines. Note that the discipline of information systems presents a bigger increase because almost 60 papers about web services that were classified under other main disciplines (agents, e-commerce, e-learning, document and text processing, software engineering and semantic web) have been classified under the information systems discipline after considering it as an additional classification discipline.

Following sections analyze web engineering references in terms of SWEBOK knowledge areas.

3 Software Requirements

The Software Requirements Knowledge Area is concerned with the elicitation, analysis, specification and validation of software requirements. Of the web engineering papers classified under software engineering discipline, 6.81% of these papers were related to software requirements.

The Guide to the SWEBOK identifies seven sub-areas in software requirements: sw. requirements fundamentals, requirements process, requirements elicitation, requirements analysis, requirements specification, requirements validation and practical considerations.

Although requirements may appear to be the same conceptual element and independent of the software domain, according to Escalona and Koch [Escalona 04] some types of requirements differ between software and web engineering. These differences arise due to the special characteristics of the web applications: the presence of different kinds of stakeholders, and the significance of navigational structure, user interface and navigation capabilities in these applications [Escalona 04]. The following sections analyze them.

3.1 Software Requirements Fundamentals

At its most basic, a software requirement is a property which must be exhibited in order to solve some problem in the real world. Software requirements fundamentals is focused on the basis of software requirements: definition of a software requirement, product and process requirements, functional and non-functional requirements, emerging properties, quantifiable requirements and system and software requirements.

This sub-area does not greatly differ between software engineering and web engineering. However, Lowe [Lowe 03] identifies several characteristics of web systems that interfere with different knowledge areas, including software requirements: sophisticated business architecture, distributed nature, visibility of web systems to external stakeholders, uncertainty in the project domain, volatility of customer needs and available technology, short time frames for initial delivery, highly competitiveness, fine-grained evolution and maintenance, increased emphasis on user interface, increased emphasis on quality attributes, open modularized architecture and highly variable customer understanding of the situation.

Moreover, Escalona and Koch [Escalona 04] identify several requirements classified as classic functional and non-functional requirements. Thus, functional requirements are classified as: data requirements, user interface requirements, navigational requirements, personalization requirements and transactional requirements.

3.2 Requirements Process

The requirements process structures the mechanisms to obtain the software requirements. This sub-area is focused on the requirements process itself: process models, process actors, process support and management and process quality and improvement.

Although, the requirements process in web engineering does not differ greatly from software engineering, there are several differences between both disciplines. The main difference is that requirements process models consider the finer-grained classification of requirements that appear in the development of web applications (e.g. navigation) [Escalona 04]. In addition, because web applications are used by the general public, anonymous transient users can be considered as process actors.

In web engineering discipline, the presence of the requirements process varies slightly from one approach to another. For example in the UML-Based Web Engineering approach (Hennicker 01), the requirements process is included in the whole approach. In Plumbing (Navarro 04), the cycle for conceptualization and prototyping is one stage of the process model. In e-Prototyping (Bleek 04) frequent releases of software versions based on short development cycles that help to identify requirements are built. Finally, in Design-driven Requirements Elicitation (Lowe 02) the requirements process is conceived as a part of the design stage. Therefore, in web engineering discipline, there is no a clear consensus regarding to the requirements process.

3.3 Requirements Analysis

Requirements analysis is concerned with the process of analyzing requirements to detect and resolve conflicts between requirements, discover the software boundaries, and elaborate systems requirements to derive software requirements. This sub-area is focused on: requirements classification, conceptual modeling, architectural design and requirements allocation and requirements negotiation.

Regarding requirements analysis, conceptual modeling is one of the most active research areas. In web engineering classic notations intended to characterize

conceptual models (e.g. UML (omg 09)) have been enriched with specific notations that take into account the three dimensions of a web application (Fraternali 99):

- *Conceptual structure* of the contents and their semantic relationships.
- *Navigation* throughout the application content.
- *Presentation* of content and navigation to the user.

In this paper, these notations are analyzed in section 4.5 (software design notations). In any case, Barry (Barry 01) states that new techniques that capture requirements and integrate them within a systems development framework are needed. On the contrary, (Conallen 02) and (Eeles 02) use UML-Web Architecture Extension to deal with requirements analysis stage. Therefore, as in the previous section, there is no a clear consensus about the requirements capture.

3.4 Remaining Sub-areas

We have not found specific references on the rest of the sub-areas (requirements elicitation, requirements specification, requirements validation and practical considerations) beyond [Escalona 04], which mainly compiles classic techniques. However, it is important to point out that according to Escalona and Koch [Escalona 04] more research is needed in this direction. For example, requirements elicitation and practical considerations should take into account that due to the presence of transient anonymous users, several requisites may not be apparent until after the web application is deployed, and may evolve rapidly if the web application is popular.

3.5 Software Requirements. Matrix of Knowledge Area vs. Reference Material

	[Escalona 04]	[Lowe 03]
Software Requirements Fundamentals	*	*
Requirements Process	*	
Requirements Elicitation	*	
Requirements Specification	*	
Requirements Validation	*	
Practical Considerations	*	

3.6 Recommended References for Software Requirements

[Escalona 04] Escalona, M.J., Koch, N. (2004) Requirements Engineering for Web Applications - A Comparative Study. Journal of Web Engineering, 2, 193-212.

[Lowe 03] Lowe, D. (2003) Web system requirements: an overview. Requirements Engineering, 8, 102-113

3.7 List of Further Readings

(Barry 01) Barry, C., Lang. M. (2001) A Survey of Multimedia and Web Development Techniques and Methodology Usage. IEEE MultiMedia, 8, 52-60.

(Bleek 04) Bleek, W.-G., Jeenicke, M., Klischewski, R. (2004) e-Prototyping: Iterative Analysis of Web User Requirements. Journal of Web Engineering, 3, 77-94.

(Conallen 02) Conallen, J. (2002) Building Web Applications with UML. Second Edition. Addison-Wesley Object Technology Series, Boston, MA.

(Eeles 02) Eeles P., Houston K., Kozaczynski, W. (2002) Building J2EE Applications with the Rational Unified Process. The Addison-Wesley Object Technology Series, Boston, MA.
(Fraternali 99) Fraternali, P. (1999) Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. Section 2.2. ACM Computing Surveys, 31, 227-263.
(Hennicker 01) Hennicker, R., Koch, N. (2001) Systematic Design of Web Applications with UML. In Siau, K. and Halpin, T.A. (eds). Unified Modeling Language: Systems Analysis, Design and Development Issues. Idea Group, Hershey, PA.
(Lowe 02) Lowe, D., Eklund, J. (2002) Client Needs and the Design Process in Web Projects. Journal of Web Engineering, 1, 23-36.
(Navarro 04) Navarro, A., Fernández-Manjón, B., Fernández-Valmayor, A., Sierra, J.L. (2004) The PlumbingXJ Approach for Fast Prototyping of Web Applications. Journal of Digital Information, 5, http://jodi.tamu.edu/Articles/v05/i02/Navarro/
(omg 09) Object Management Group (2009) Unified Modeling Language 2.2, http://www.omg.org/spec/UML/2.2/Software design

4 Software Design

Design is defined in the IEEE Std. 610.12-1990 (ieee 90) as both the process of defining the architecture, components, interfaces, and other characteristics of a system or component as well as the result of that process. Of the web engineering papers classified under software engineering discipline, 57% of these papers were related to software design.

The SWEBOK Guide identifies six sub-areas in software design: software design fundamentals, key issues in software design, software structure and architecture, software design quality analysis and evaluation, software design notations, software design strategies and methods.

4.1 Software Design Fundamentals

The concepts, notions, and terminology introduced in this sub-area form an underlying basis for understanding the role and scope of software design. This sub-area focuses on: general design concepts, the context of software design, the software design process, and enabling techniques.

This sub-area does not greatly differ between software engineering and web engineering (the main differences are derived from the characteristics of web applications depicted in section 3.1 -software requirements fundamentals-). Therefore, we have not found specific references in the web engineering discipline.

In any case, in web engineering, unlike general purpose software applications, it is possible to find web applications without business logic, which could be named *websites* [Conallen 02][Shklar 03](Winckler 03). These websites can be developed by people without computer science skills using visual editors (see section 7.1 -software engineering tools-). Obviously, these websites are out of the scope of most of the software engineering and web engineering design techniques.

4.2 Key Issues in Software Design

A number of key issues must be dealt with when designing software. In particular, this sub-area focuses on concurrency, control and handling of events, distribution of

components, error and exception handling and fault tolerance, interaction and presentation, and data persistence. These topics take on a special relevance in web engineering discipline, since these applications exhibit the characteristics enumerated in section 3.1 (software requirements fundamentals).

Concurrency focuses on how to decompose the software into processes, tasks and threads and how to deal with related efficiency, atomicity, synchronization, and scheduling issues. This is a key issue in web engineering because web applications are inherently concurrent. To guarantee the correct execution of concurrent code in business tier, the presence of stateless objects can be the easiest solution [Alur 03]. Regarding integration and data tier some type of data locking should be used to guarantee the data consistence. This locking can be imposed over the data tier (e.g. relational databases locking [Fowler 02]), or using object-oriented covers [Fowler 02](Keith 06). In any case, the database concept of *transaction* becomes very important in web engineering applications (Bernstein 97)(Little 04). In addition, in web engineering, designs have to take into account those problems derived from the unpredictable rise in the number of users in the applications, as a result, for example, of a slashdot phenomenon (WikiPedia 09) or a Denial of Service (DOS)/Distributed DOS (DDOS) attack [Steel 06].

Control and handling of events focuses on how to organize data and control flow and how to handle reactive and temporal events. If an anchor is considered as any device able to start an HTTP request [Shklar 03] (e.g. an HTML anchor or a submit button of a web form), in web engineering there is a key event produced in the presentation tier that must be handled: the anchor selection. In static applications, the anchor selection is equivalent to the request of content to a web server [Shklar 03]. If some computational process is attached to this anchor on the client side (e.g. using JavaScript (Flanagan 06)), the response to the anchor selection can include any computational behavior. In the same way, any computational process can be invoked on the server side using server-side technologies that provide support to the business tier elements (e.g. Java servlets (Hall 03)). Because HTTP is a stateless protocol [Shklar 03], these technologies use the concept of user *session* [Conallen 02] to keep the user case state when clients are browsing. In addition, in web applications, the control flow is enhanced with the routing of client requests [Shklar 03] through the components of the web application (static and/or dynamic [Conallen 02]).

Distribution of components focuses on how to distribute the software throughout the hardware, how the components communicate and how middleware can be used to deal with heterogeneous software. The distribution of components has found one of its most important fields of application in web development. Thus, the design of well-defined architectures where every component has specific responsibilities assigned is paramount in the design of web architectures [Alur 03][Fowler 02][Steel 06]. Finally, in recent years, the concept of *web service* [Skonnard 02] has been one of the most widely used terms in the web community. Web services dramatically enhance software reusability using new communication methods [Newcomer 02]. In addition, web services can be considered as a way of implementing a *Service-Oriented Architecture* (SOA) (Erl 05)[Mahmud 05].

Error and exception handling and fault tolerance focus on how to prevent and tolerate faults and deal with exceptional conditions. This topic affects every tier of a web application. In web engineering this topic is more complicated than in software

engineering due to the networked nature of the applications (Schmidt 00)[Steel 06]. In addition, in web applications and SOA applications, exception handling is affected by privacy, security, and dynamic integration (Erl 05).

Interaction and presentation focus on how to structure and organize the interactions with users and the presentation of information. This topic mainly affects the presentation tier (Navarro 08), and it is a key issue in web engineering, which has common boundaries with other areas such as Human Computer Interaction (McCracken 03). Presentation is so important in web engineering that design notations explicitly take it into account as is described in section 4.5 (software design notations). In addition, usability becomes a key issue in web applications (Nielsen 06). Recently, the classic interaction pattern between browsers and servers [Shklar 03] has been enhanced with the AJAX approach, which enables asynchronous communication with the server (Asleson 05).

Finally, *data persistence* focuses on how long-lived data must be handled. In web engineering, data presented to the user can include semantic metadata [Berners-Lee 01] to enhance its retrieval and accuracy. Data persistence, mainly affects the integration and data tier. Web applications deal with data that are inherently persistent. In static applications without business logic, the data presented to the user is in a persistent format that is retrieved by the web server (e.g. HTML (w3c 99) or XML (w3c 04) format). In dynamic applications, where there is a significant business logic or dynamic data, the information managed by the web application, generally has to be stored in a persistent format (e.g. relational databases [Fowler 02] or other type of object-oriented covers [Fowler 02](Keith 06)). In addition, in contrast to traditional software application, a new type of persistence appears in web application: the session persistence. Sessions support different data needed during browsing (remember than HTTP is a stateless protocol). According to Fowler [Fowler 02] there are three ways to implement the session state: client session state (e.g. cookies), server session state, and database session state. Regarding transient data elements, in web applications these elements can last either throughout one HTTP request, throughout an entire session, or the lifetime of the software component thread. They can also be scoped to be available only to the current request execution thread, other threads associated with the current session, all threads in the server context, or all threads in the entire server [Alur 03][Conallen 02][Fowler 02].

4.3 Software Structure and Architecture

In its strictest sense, a software architecture is a description of the subsystems and components of a software system, and the relationships between them. This sub-area focuses on architectural structures and viewpoints, design patterns and families of programs and frameworks.

An important source for web patterns [Alur 03], which conforms to Christodoulou's tiers (Christodoulou 01), has its root in the J2EE platform (Mukhar 05), although most of these patterns can be used in other platforms. According to this categorization, web patterns can be classified in terms of the tier in which they are used:

- *Presentation tier patterns*, focused on the presentation logic required to service the clients that access the systems.

- *Business tier patterns*, focused on the business services required by the application clients.
- *Integration tier patterns*, focused on communication with external resources and systems.

Fowler [Fowler 02] is another important source of design patterns that identifies similar tiers. Steel [Steel 06] focuses on security patterns only.

Regarding families of programs and frameworks, there is a significant array of these frameworks/technologies in the web engineering community. For example, Active Server Pages (ASPs) (Esposito 08b), Java Server Pages (JSPs) and servlets (Hall 03), or PHP: Hypertext Preprocessor (Lerdorf 06) are some of the more relevant technologies that can be used to extend web servers with general-purpose computational processes [Shklar 03]. These technologies can have additional frameworks that help to implement some functionality, such as the implementation of a model-view-controller architecture (e.g. Struts (Carnell 04)), the implementation of the user interface (e.g. Java Server Faces -JSF- (Geary 04)), or the implementation of a persistence framework (e.g. Hibernate (Bauer 06)). In addition, there are technologies specifically designed to deal with the implementation of distributed components such as Enterprise Java Beans (Burke 06) or Microsoft .NET (Esposito 08a).

The above-mentioned technologies are server-based technologies (i.e. their code is executed in the server). As regards client-based technologies (i.e. their code is executed in the client), JavaScript (Flanagan 06) and Java Applets (Cowell 00) are two of the most famous client-side technologies. Recently, JavaScript has been enhanced with AJAX (Asleson 05). At present, most technologies have incorporated AJAX philosophy (Darie 06)(Jacoby 06)(Woolston 06).

Finally, because architectural structures and viewpoints talks about views of software design (e.g. logical view vs. physical view, behavioural vs. data view) and architectural styles (e.g. model-view-controller, three tier systems, etc.), there are no great differences in this topic between software engineering and web engineering beyond the underlying technological details.

4.4 Software Design Quality Analysis and Evaluation

This sub-area includes a number of quality and evaluation topics specifically related to software design. Most of them are covered in a general manner in the Software Quality knowledge area. This sub-area focuses on: quality attributes, quality analysis and evaluation techniques and measures.

Typical software engineering quality attributes (e.g. maintainability, correctness, etc.) are also applicable to the web engineering discipline. Regarding web applications, Offutt [Offutt 02] identifies reliability, usability and security as the three most important quality criteria for web software success. Other additional quality criteria identified by Offutt are: availability, scalability, maintainability, and time to market. In addition, accessibility is a key issue in the design of web applications [Brewer 04](Burks 06). Note that, except for usability and accessibility, which mainly affect the presentation tier, the remaining quality attributes affect every tier of a web application.

Regarding measures, those collected in SWEBOK do not take into account the specific characteristics of web applications (e.g. navigation). As there is no widely

used design notation in web engineering, available measures for design quality are specific to design notations (Abrahão 02)(Lanzi 04).

Quality analysis and evaluation techniques in web engineering do not differ greatly with regard to software engineering. Of course, web engineering discipline includes some specific issues of this discipline (e.g. special measures have been defined to evaluate the quality of a hypertext (Dhyani 02)).

Finally, the IEEE Std. 2001-2002 IEEE Recommended Practice for the Internet - Web Site Engineering, Web Site Management, and Web Site Life Cycle (ieee 2002) comprises a set of good practices to enhance the overall quality of the design of a web application.

4.5 Software Design Notations

There are many notations and languages to represent software design artifacts. The SWEBOK Guide splits them into structural descriptions (static view) and behavioral descriptions (dynamic view).

Although this classification is still applicable to web engineering, and almost every design notation mentioned in the Guide to the SWEBOK can be used to characterize some aspect of web applications, in practice most of the web engineering design notations take into consideration three main components in web design (Koch 03) (as identified in section 3.3 -requirements analysis-):

- *Conceptual model*. It expresses the main conceptual elements in web applications and their semantic relationships. There are three groups of notations focusing on: (i) classes (in an object-oriented sense) and their relationships (Baresi 00)[Conallen 99](De Troyer 98)(Gómez 01)[Hennicker 01](Schwabe 01); (ii) entities (in a relational sense) and their relationships [Ceri 00](Garzotto 93)[Isakowitz 95](Thalheim 01); and (iii) elements (or nodes in a hypertext sense) and their relationships (Montero 04)[Navarro 07].
- *Navigation model*. It expresses how the elements of the conceptual model are finally related in terms of links, and how these links are accessed by the user. Those notations that use classes or entity diagrams at the conceptual level use access primitives describing the navigation in terms of elements (considering the relationships between classes or entities, of course). On the other hand, those notations that use elements at the conceptual level, group them into elements that are simultaneously browsed.
- *Presentation model*. It expresses the final appearance of the application. Here the approaches differ from more evolved ones where the static and dynamic behavior of the user interface is described (e.g. [Hennicker 01](Schwabe 01)) to others where this component is less evolved (e.g. (Garzotto 93)[Isakowitz 95]).

At present, and to some extent, most of these design notations have incorporated the ability to describe general-purpose business logic explicitly [Brambilla 06][Koch 04](Rossi 03). Even, some of them have been updated to take into account the characteristics of rich internet applications (Brambilla 08)(Meliá 08)(Preciado 08)(Rossi 08).

4.6 Software Design Strategies and Methods

According to the SWEBOK Guide, there are various general strategies to help guide the design process. In contrast with general strategies, methods are more specific as they generally suggest and provide a set of notations to be used with the method, a description of the process to be used when following the method and a set of guidelines for using the method. This sub-area focuses on: general strategies, function-oriented design, object-oriented design, data-structure centered design, component-based design and other methods.

The general strategies used in SWEBOK can also be applied in web engineering: stepwise refinement, patterns, incremental approach, etc.

Regarding specific methods, the new domain forces the application of new methods, which can be classified in terms of the design notation (section 4.5 -software design notations-) that they use: class-oriented, entity-oriented or element-oriented.

4.7 Software Design. Matrix of Knowledge Area vs. Reference Material

	Software Design Fundamentals	Key Issues in Software Design	Software Structure and Architecture	Software Design Quality Analysis and Evaluation	Software Design Notations
[Alur 03]		*	*		
[Berners-Lee 01]		*			
[Brambilla 06]					*
[Breuer 04]				*	
[Ceri 00]					*
[Conallen 99]					*
[Conallen 02]	Chapter 1	Chapter 1, Chapter 2			
[Fowler 02]		*	*		
[Hennicker 01]					*
[Isakowitz 95]					*
[Koch 04]					*
[Mahmud 05]		*			
[Navarro 07]					*
[Newcomer 02]		*			
[Offutt 02]				*	
[Shklar 03]	Chapter 1	*	*		
[Skonnard 02]		*			
[Steel 06]		*	*		

4.8 Recommended References for Software Design

[Alur 03] Alur, D., Crupi, J., Malks, D. (2003) Core J2EE Patterns. Best Practices and Design Strategies. Sun Microsystems Press – Prentice Hall, Upper Saddle River, NJ..

[Berners-Lee 01] Berners-Lee, T., Hendler H., Lassila, O. (2001) The Semantic Web. Scientific American, May 2001.

[Brambilla 06] Brambilla, M., Ceri, S., Fraternali, P. Manolescu, I. (2006) Process Modeling in Web Applications. ACM Transactions on Software Engineering and Methodology, 15, 360-409.

[Brewer 04] Brewer, J. (2004) Web Accessibility Highlights and Trends. Proceedings of 2004 International Cross-Disciplinary Workshop on Web Accessibility (W4A), New York, NY, May 17-22, pp. 51-56. ACM Press, New York, NY.

[Ceri 00] Ceri, S., Fraternali, F., Bongio, A. (2000) Web Modeling Language (WebML): a modeling language for designing Web sites. Computer Networks, 33, 137-157.

[Conallen 99] Conallen, J. (1999) Modeling Web Application Architectures with UML. Communications of the ACM, 42, 63-70.

[Conallen 02] Conallen, J. (2002) Building Web Applications with UML. Second Edition. Addison-Wesley Object Technology Series, Boston, MA.

[Fowler 02] Fowler, M. (2002) Patterns of Enterprise Application Architecture. Addison-Wesley Professional, Boston, MA.

[Hennicker 01] Hennicker, R., Koch, N. (2001) Systematic Design of Web Applications with UML. In Siau, K., and Halpin, T.A. (eds) Unified Modeling Language: Systems Analysis, Design and Development Issues, Idea Group, Hershey, PA.

[Isakowitz 95] Isakowitz, T., Stohr, E.A, Balasubramanian, P. (1995) RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM, 38, 34-44.

[Koch 04] Koch, N., Kraus, A., Cachero, C. and Meliá, S. (2004) Integration of Business Processes in Web Application Models. Journal of Web Engineering, 3, 22-44.

[Mahmud 05] Mahmud, Q.H. (2005) Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI). http://java.sun.com/developer/technicalArticles/WebServices/soa/index.html

[Navarro 07] Navarro, A., Fernández-Valmayor, A. (2007) Conceptualization of Hybrid Websites. Internet Research, 17, 207-228.

[Newcomer 02] Newcomer E. (2002) Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley Professional, Boston, MA.

[Offutt 02] Offutt, J. (2002) Quality attributes of web software applications. IEEE Software, 37, 25-32.

[Shklar 03] Shklar, L., Rosen, R. (2003) Web Application Architecture: Principles, Protocols and Practices. John Wiley & Sons Inc., Chichester.

[Skonnard 02] Skonnard, A. (2002) The Birth of Web Services. http://msdn.microsoft.com/webservices/webservices/understanding/webservicebasics/default.aspx

[Steel 06] Steel, C., Nagappan, R., Lai, R. (2006) Core Security Patterns. Best Practices and Strategies for J2EE, Web Services, and Identity Management. Prentice Hall, Upper Saddle River, NJ.

4.9 List of further readings

(Abrahão 02) Abrahão, S.M., Olsina, L., Pastor, O. (2002) Towards the Quality Evaluation of Functional Aspects of Operative Web Applications. Revised Papers of Advanced Conceptual Modeling Techniques, ER 2002 Workshops, Tampere, October 2002, LNCS 2503, pp. 325-338. Springer-Verlag, Berlin.

(Aslesson 05) Aslesson, R., Schutta, N.T. (2005) Foundations of AJAX. Apress, Berkeley, CA.

(Baresi 00) Baresi, L., Garzotto, F., Paolini, P. (2000) From Web Sites to Web Applications: New Issues for Conceptual Modeling. Proceedings of Conceptual Modeling for E-Business and the Web, ER 2000 Workshops, Salt Lake City, October 2000, LNCS 1921, pp. 89-100. Springer-Verlag, Berlin,

(Bauer 06) Bauer, C., King, K. (2006) Hibernate in Action. Manning Publications, Greenwich, CT.

(Bernstein 97) Bernstein, P.A., Newcomer, E. (1997) Principles of Transaction Processing. Morgan Kaufmann, Silicon Valley, CA.

(Brambilla 08) Brambilla, M., Preciado, J.C., Linaje, M., Sanchez-Figueroa, F. Business Process-Based Conceptual Design of Rich Internet Applications. Proceedings of ICWE 2008, pp. 155-161.

(Burke 06) Burke, B., Monson-Haefel, R. (2006) Enterprise JavaBeans 3.0. O'Reilly Media Inc., Sebastopol, CA.

(Burks 06) Burks, M.R., et al. (2006) Web Accessibility: Web Standards and Regulatory Compliance. Friends of ED, Berkeley, CA.

(Carnell 04) Carnell, J., Harrop, R. (2004) Pro Jakarta Struts, Second Edition. Apress, Berkeley, CA.

(Christodoulou 01) Christodoulou, S.P., Zafiris, P.A., Papatheodorou, T.S. (2001) Web Engineering: the Developer's View and a Practitioner's approach. Proceedings Web Engineering 2000. Springer-Verlag, Berlin.

(Cowell 00) Cowell, J. (2000) Essential Java 2 fast: How to Develop Applications and Applets with Java 2. Springer, Berlin.

(Darie 06) Darie, C. et al. (2006) AJAX And PHP: Building Responsive Web Applications. Packt Publishing, Birmingham.

(De Troyer 98) De Troyer, O.M.F., Leune, C.J. (1998) WSDM: A User Centered Design Method for Web Sites. Computer Networks, 30, 85-94.

(Dhyani 02) Dhyani, Keong Ng, W, Bhowmick, S.S. (2002) A Survey of Web Metrics. ACM Computing Surveys, 34, 469-503.

(Erl 05) Erl, T. (2005) Service-Oriented Architecture (SOA): Concepts, Technology, and Design. Prentice-Hall PTR, Upper Saddle River, NJ.

(Esposito 08a) Esposito, D., Saltarello, A. Microsoft .NET: Architecting Applications for the Enterprise. Microsoft Press, 2008.

(Esposito 08b) Esposito, D. Programming Microsoft ASP.NET 3.5. Microsoft Press, 2008.

(Flanagan 06) Flanagan, D. (2006) JavaScript: The Definitive Guide. Fifth edition. O'Reilly Media Inc., Sebastopol, CA.

(Garzotto 93) Garzotto, F., Paolini, P., Schwabe, D. (1993) HDM - A Model-Based Approach to Hypertext Application Design. ACM Transactions on Information Systems, 11, 1-26.

(Geary 04) Geary, D., Horstmann, C. (2004) Core Java Server Faces. Prentice Hall PTR, Upper Saddle River, CA.

(Gómez 01) Gómez, J., Cachero, C., Pastor, O. (2001) Conceptual Modeling of Device-Independent Web Applications. IEEE MultiMedia, 8, 26-39.

(Hall 03) Hall, M, Brown, L. (2003) Core Servlets and Java Server Pages, Vol. 1: Core Technologies, Second Edition. Prentice Hall PTR, Upper Saddle River, CA.

(ieee 90) IEEE Std. 610.12-1990. 1990. IEEE standard glossary of software engineering terminology. The Institute of Electrical and Electronics Engineers, New York, NY.

(Jacoby 06) Jacobi, J., Fallows, J.R. (2006) Pro JSF and AJAX. Building Rich Internet Components. Apress, Berkeley, CA.

(Keith 06) Keith, M., Schincariol, M. (2006) Pro EJB 3: Java Persistence API (Pro). Apress, Berkeley, CA.

(Koch 03) Koch, N., Kraus, A. (2003) Towards a Common Metamodel for the Development of Web Applications. Proceedings of the Third Conference on Web Engineering, Oviedo, July 2003, LNCS 2722, pp. 497-506. Springer-Verlag, Berlin.

(Lanzi 04) Lanzi, P.L., Matera, M., Maurino, A. (2004) A Framework for Exploiting Conceptual Modeling in the Evaluation of Web Application Quality. Proceedings of The Fourth International Conference on Web Engineering, Munich, July 2004, LNCS 3140, pp. 50-54. Springer-Verlag, Berlin.

(Lerdorf 06) Lerdorf, R., Tatroe, K., MacIntyre, P. (2006) Programming PHP. O'Reilly Media Inc., Sebastopol, CA.

(Little 04) Little, M., Maron, J., Pavlik, G. (2004) Java Transaction Processing. Design and Implementation. Prentice Hall PTR, Upper Saddle River, NJ.

(McCracken 03) McCracken, D., Wolfe, R.J., Spool J.M. (2003) User-Centered Website Development: A Human-Computer Interaction Approach. Prentice Hall PTR, Upper Saddle River, NJ.

(Meliá 08) Meliá, S., Gómez, J., Pérez, S., Díaz, O. A Model-Driven Development for GWT-Based Rich Internet Applications with OOH4RIA. Proceedings of ICWE 2008, pp. 13-23.

(Montero 04) Montero, S., Díaz, P., Aedo, I. (2004) AriadneTool: A Design Toolkit for Hypermedia Applications. Journal of Digital Information, 5, http://jodi.tamu.edu/Articles/v05/i02/Montero/

(Mukhar 05) Mukhar, K. et al. (2005) Beginning Java EE 5: From Novice to Professional. Apress, Berkeley, CA.

(Navarro 08) Navarro, A., Fernández-Valmayor, A., Fernández-Manjón, B., Sierra, J.L. (2008). Characterizing Navigation Maps for Web Application with the NMM Approach. Science of Computer Programming, 71, 1, 1-16.

(Nielsen 06) Nielsen, J., Loranger, H. (2005) Prioritizing web usability. New Riders Press, Berkeley, CA.

(Preciado 08) Preciado, J.C.; Linaje, M.; Morales-Chaparro, R.; Sanchez-Figueroa, F.; Gefei Zhang; Kroiss, C.; Koch, N. Designing Rich Internet Applications Combining UWE and RUX-Method. Proceedings of ICWE 2008, pp. 148-154.

(Rossi 03) Rossi, G., Schmid, H.A., Lyardet, F. (2003) Customizing Business Processes in Web Applications. Proceedings of EC-Web 2003, pp. 359-368.

(Rossi 08) Rossi, G., Urbieta, M., Ginzburg, J., Distante, D., Garrido, A. Refactoring to Rich Internet Applications. A Model-Driven Approach. Proceedings of ICWE 2008, pp. 1-12.

(Schmidt 00) Schmidt, D., Stal, M., Rohnert, H., Buschmann, F. (2000) Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects. John Wiley & Sons Inc., Chichester.

(Schwabe 01) Schwabe, D., Esmeraldo, L., Rossi, G., Lyardet, F. (2001) Engineering Web Applications for Reuse. IEEE MultiMedia, 8, 20-31.

(Thalheim 01) Thalheim, B., Düsterhöft, A. (2001) SiteLang: Conceptual Modeling of Internet Sites. Proceedings of Conceptual Modeling - ER 2001, 20th International Conference on Conceptual Modeling, Yokohama, November 2001, LNCS 2224, pp. 179-192. Springer-Verlag, Berlin.

(Wikipedia 09) Wikipedia, Slashdot, (2009) http://en.wikipedia.org/wiki/Slashdot

(Winckler 03) Winckler, P. Palanque, P. (2003) StateWebCharts: A Formal Description Technique Dedicated to Navigation Modeling of Web Applications. Proceedings of Interactive Systems. Design, Specification, and Verification, 10th International Workshop, DSV-IS 2003, pp. 61-76. http://virtual.inesc.pt/dsvis03/papers/27.pdf

(Woolston 06) Woolston, D. (2006) Pro AJAX and the .NET 2.0 Platform. Apress, Berkeley, CA.

(w3c 99) World Wide Web Consortium. HTML 4.01 Specification (1999) http://www.w3.org/TR/html4/

(w3c 04) World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Third Edition) (2004) http://www.w3.org/TR/2004/REC-xml-20040204/

4.10 List of standards

(ieee 2002) IEEE Std. 2001-2002 (2002). IEEE Recommended Practice for the Internet - Web Site Engineering, Web Site Management, and Web Site Life Cycle (2002). The Institute of Electrical and Electronics Engineers, New York, NY.

5 Software Testing

Software testing consists of the dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior. Of the web engineering papers classified under software engineering discipline, 6.45% of these papers were related to software testing.

The SWEBOK Guide identifies five sub-areas in this knowledge area: software testing fundamentals, test levels, test techniques, test related measures, test process.

5.1 Test Levels

Test levels focus on both the target of the test and the objectives of testing. The target of the test identifies three test stages: unit, integration, and system testing. In web engineering applications, these stages are also applicable, but they must take into account the issues identified in section 4.2 (key issues in software design) (Ash 03).

The objectives of testing identify the properties being tested. Classic software engineering objectives of testing (e.g. functional, installation, regression, performance, stress, etc.), can also be applied to web software (Ash 03)(Menasce 02)[Nguyen03](Subraya 06). In addition, Nguyen [Nguyen 03] identifies additional types of tests: user interface, server-side, database, help, and security testing. Ash (Ash 03) identifies two additional types of test: client-side and server-side testing. Finally, security is an essential issue in web applications [Andrews 06](Splaine 02). Andrews [Andrews 06] identifies different concerns in the security of web applications: attacks to the client, state-based attacks, attacks to the user-supplied input data, language-based attacks, attacks to the server, authentication, privacy, and web services-related attacks.

5.2 Remaining sub-areas

Regarding the remaining sub-areas (software testing fundamentals, test techniques, test related measures and test process), in our opinion, there are no significant differences between these areas in software engineering and web engineering disciplines. For example, [Hao 06] includes an analysis about web software testing methods, and Alalfi [Alalfi 07] analyzes different modeling methods used in website verification and testing. Of course, web applications require special testing tools, which are depicted in section 7.1 (software engineering tools).

5.3 Software Testing. Matrix of Knowledge Area vs. Reference Material

	[Alalfi 07]	[Andrews 06]	[Hao 06]	[Nguyen 03]
Test Levels		*		*
Test Techniques	*		*	

5.4 Recommended References for Software Testing

[Alalfi 07] Alalfi, M.H., Cordy, J.R., Dean, T.R. (2007) A Survey of Analysis Models and Methods in Website Verification and Testing. Proceedings of International Conference on Web Engineering 2007, Como, July 16-20, pp. 306-311.

[Andrews 06] Andrews, M., Whittaker, J.A. (2006) How to Break Web Software: Functional and Security Testing of Web Applications and Web Services. Addison-Wesley Professional, Boston, MA.

[Hao 06] Hao, J., Mendes, E. (2006) Usage-based statistical testing of web applications. Proceedings of International Conference on Web Engineering 2006. Menlo Park, CA, July 12-14, pp. 17-24. ACM Press, New York, NY.

[Nguyen 03] Nguyen, H.Q., Johnson, R., Hackett, M. (2003) Testing Applications on the Web: Test Planning for Mobile and Internet-based Systems. John Wiley & Sons Inc., Chichester.

5.5 List of Further Readings

(Ash 03) Ash, L. (2003) The Web Testing Companion: The Insider's Guide to Efficient and Effective Tests. John Wiley & Sons Inc., Chichester.

(Menasce 02) Menasce, D.A. (2002) Load Testing of Websites. IEEE Internet Computing, 6, 2002.

(Splaine 02) Splaine, S. (2002) Testing Web Security: Assessing the Security of Web Sites and Applications. John Wiley & Sons Inc., Chichester.

(Subraya 06) Subraya, B.M (ed) Integrated Approach to Web Performance Testing: A Practitioner's Guide. IRM Press US, Hershey, PA.

6 Software Engineering Process

The Software Engineering Process knowledge area is concerned with the definition, implementation, assessment, measurement, management, change and improvement of the software life cycle processes themselves. Of the web engineering papers classified under software engineering discipline, 21.50% of these papers were related to software engineering process.

The SWEBOK Guide identifies four sub-areas in this knowledge area: process implementation and change, process definition, process assessment, process and product measurement.

6.1 Process Definition

A process definition can be a procedure, a policy, or a standard. Software life cycle processes are defined for a number of reasons that include increasing the quality of the product, facilitating human understanding and communication, supporting process improvement, supporting process management, providing automated process guidance and providing automated execution support. This sub-area focuses on software life cycle models and processes, notations for process definitions, process adaptation and automation.

According to the SWEBOK Guide, Software life cycle models serve as a high-level definition of the phases that occur during development. In this sub-area, we have identified several approaches. Hall and Lowe present the classic process models (e.g. spiral) adapted for web development [Hall 98]. As in the case of general-purpose

software, waterfall development presents more problems than prototyping and incremental development. Eeeles et al. [Eeeles 02] adapt The Unified Software Development Process (Jacobson 99) using the ideas put forward by Conallen in his UML-Web Application Extension (Conallen 02). Thus, a web version of the classic UML-based process model is obtained. Fraternali, Ginige, and Lowe define process models with two development loops focused on conceptualization/prototyping and design/development [Fraternali 99][Wills 98][2]. These process models are very similar to those depicted in [Hall 98], but they separate the loop focused on conceptualization from the loop focused on design. Díaz (Díaz 01) and Navarro (Navarro 04) present two iterative process models (similar to the one proposed by [Fraternali 99]), while others present models not explicitly iterative (De Troyer 98)[Isakowitz 95][Koch 01](Schwabe 95). Finally, Maurer, Martel and McDonald use lightweight techniques focused on coding and testing [Maurer 02](McDonald 03). These processes are the web version of their Extreme Programming-based counterparts in software engineering (Beck 00). As in software engineering, the use of one model or another is determined by several factors such as the nature of the project, or the customer's needs (Pressman 04).

Regarding remaining topics (software life cycle models, software life cycle processes, notations for software definitions, process adaptation and automation), in our opinion, they can be directly assimilated into web engineering.

6.2 Process and Product Measurement

The term process measurement means that quantitative information about the process is collected, analyzed, and interpreted. Measurement is used to identify the strengths and weaknesses of processes, and to evaluate processes after they have been implemented and/or changed. Software product measurement notably includes the measurement of product size, product structure, and product quality.

According to Mendes et al. [Mendes 01], by using measurement principles to evaluate the quality and development of existing web applications, we can obtain feedback that will help us understand, control, improve, and make predictions about these products and their development processes.

There are several approaches in the literature regarding product measurement. Some approaches focus on the development of measures[3] and tools for the evaluation of web applications [Chang 02][Ivory 02][Mendes 01][Olsina 02b]. Other approaches are centered on measures classification. For example, Dhyani et al. [Dhyani 02] identify several web measures grouped by: (i) web graph properties; (ii) usage characterization; (iii) web page similarity; (iv) web page search and retrieval; and (v) information theoretic. Mendes et al. (Mendes 05) provide a taxonomy of hypermedia and web application size measures. Another identification of web measures is made by Calero et al. (Calero 04). Finally, Olsina et al. (Olsina 02a) propose the development of a repository for web measures, providing a conceptual model for the domain of measures.

---

[2] [Wills 98] is used as reference because no other sources that described the Ginige-Lowe process model were found.

[3] *Measure* or *metric* is sometimes used without distinction [Abran et al. 2004]. This paper uses the term *measures* in accordance with the SWEBOK.

6.3 Remaining sub-areas

Regarding the remaining sub-areas (i.e. process implementation and change, and process assessment), in our opinion, there are no significant differences between these areas in software engineering and web engineering disciplines.

6.4 Software Engineering Process. Matrix of Knowledge Area vs. Reference Material

	[Chang 02]	[Dhyani 02]	[Eeles 02]	[Fraternali 99]	[Hall 98]	[Isakowitz 95]	[Ivory 02]	[Koch 01]	[Maurer 02]	[Mendes 01]	[Olsina 02b]	[Wills 98]
Process Definition			*	Sect. 2.1	Chap. 7-8	*		*	*			*
Process and Product Measurement	*	*					*			*	*	

6.5 Recommended References for Software Engineering Process

[Chang 02] Chang, W.-K., Hon, S.-K. (2002) Assessing the Quality of Web-Based Applications via Navigational Structures. IEEE MultiMedia, 9, 22-30.

[Dhyani 02] Dhyani, Keong Ng, W., Bhowmick, S.S. (2002) A survey of Web metrics. ACM Computing Surveys, 34, 469-503.

[Eeles 02] Eeles P., Houston K., Kozaczynski, W. (2002) Building J2EE Applications with the Rational Unified Process. The Addison-Wesley Object Technology Series, Boston, MA.

[Fraternali 99] Fraternali, P. (1999) Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. ACM Computing Surveys, 31, 227-263.

[Hall 98] Hall, W., Lowe, D. (1998) Hypermedia and the Web: An Engineering Approach. John Wiley and Sons Inc., Chichester.

[Isakowitz 95] Isakowitz, T., Stohr, E.A, Balasubramanian, P. (1995) RMM: A Methodology for Structured Hypermedia Design. Communications of the ACM, 38, 34-44.

[Ivory 02] Ivory, M.Y. Hearst, M.A. (2002) Improving Web Site Design. IEEE Internet Computing, 6, 56-63.

[Koch 01] Koch, N., Kraus, A., Hennicker R. (2001) The Authoring Process of the UML-based Web Engineering Approach. Proceedings of First International Workshop on Web-oriented Software Technology (IWWOST01), Valencia, 18-20 June. http://www.dsic.upv.es/~west/iwwost01/

[Maurer 02] Maurer, F., Martel, S. (2002) Extreme Programming: Rapid Development for Web-Based Applications. IEEE Internet Computing, 6, 86-90.

[Mendes 01] Mendes, E., Mosley, N., Counsell. S. (2001) Web Metrics-Estimating Design and Authoring Effort. IEEE MultiMedia, 8, 50-57.

[Olsina 02b] Olsina, L., Rossi, G. (2002) Measuring Web Application Quality with WebQEM. IEEE MultiMedia, 9, 20-29.

[Wills 98] Wills, G.B., Crowder, R.M., Heath, I. Hall, W. (1998) Industrial Hypermedia Design. University of Southampton.M98-2, 4.

6.6 List of Further Readings

(Beck 00) Beck, K. (2000) Extreme Programming Explained: Embrace Change. Addison Wesley Longman, Boston, MA.

(Calero 04) Calero, C., Ruiz, J., Piattini, M. (2004) A Web Metrics Survey Using WQM. Proceedings of The Fourth International Conference on Web Engineering, Munich, July 2004, LNCS 3140, pp. 147-160. Springer-Verlag, Berlin.

(Conallen 02) Conallen, J. (2002) Building Web Applications with UML, 2nd Edition. The Addison-Wesley Object Technology Series, Boston, MA.

(De Troyer 98) De Troyer, O.M.F., Leune, C.J. (1998) WSDM: A User Centered Design Method for Web Sites. Computer Networks, 30, 85-94.

(Díaz 01) Díaz, P., Aedo, I., Montero, S. (2001) Ariadne, a Development Method for Hypermedia. Proceedings of Database and Expert Systems Applications, 12th International Conference, Munich, September 2001, LNCS 2113, pp. 764-774. Springer-Verlag, Berlin.

(Jacobson 99) Jacobson, I., Booch, G., Rumbaugh, J. (1999) The Unified Software Development Process. The Addison-Wesley Object Technology Series, Boston, MA.

(McDonald 03) McDonald, A., Welland, R. (2003) Agile Web Engineering (AWE) Process: Multidisciplinary Stakeholders and Team Communication. Proceedings of the Third Conference on Web Engineering, Oviedo, July 2003, LNCS 2722, pp. 515-518. Springer-Verlag, Berlin.

(Mendes 05) Mendes, E., Counsell, S., Mosley, N. (2005) Towards a Taxonomy of Hypermedia and Web Application Size Metrics. Proceedings of The Fifth International Conference on Web Engineering, Sydney, July-August 2005, LNCS 3579, pp. 110-123. Springer-Verlag, Berlin.

(Navarro 04) Navarro, A. Fernández-Valmayor, A., Fernández-Manjón, B., Sierra, J.L. (2004) Conceptualization, Prototyping and Process of Hypermedia Applications. International Journal of Software Engineering and Knowledge Engineering, 14, 565-602.

(Olsina 02a) Olsina, L., Lafuente, G., Pastor, O. (2002) Towards a Reusable Repository for Web Metrics. Journal of Web Engineering, 1, 61-73.

(Pressman 04) Pressman, R.S. (2004) Software Engineering: A Practitioner's Approach. 6th edition. McGraw-Hill, New York, NY.

(Schwabe 95) Schwabe, D., Rossi, G. (1995) The Object-Oriented Hypermedia Design Model. Communications of the ACM, 38, 45-46.

7 Software Engineering Tools and Methods

Software development tools are computer-based tools that are intended to assist the software life cycle processes. Software engineering methods impose structure on the software engineering activity with the aim of making the activity systematic and ultimately more likely to be successful. Of the web engineering papers classified under software engineering discipline, 5.02% of these papers were related to software engineering tools and methods.

The SWEBOK Guide identifies two sub-areas in this knowledge area: software engineering tools and software engineering methods.

7.1 Software Engineering Tools

Software engineering tools allow repetitive, well-defined actions to be automated, reducing the cognitive load on their user who is then free to concentrate on the creative aspects of the process. This sub-area focuses on tools for dealing with

software requirements, software design, software construction, software testing, software maintenance, software configuration management, software engineering management, the software engineering process, software quality and miscellaneous.

Software design tools is one of the more prolific segments in web engineering. Most design notations have a specific tool that facilitates the development of visual models and, in some cases, the development of a prototype or the final application (Gómez 04)(Knapp 04)(Montero 04)(Paiano 04)(Thalheim 04)(WebRatio 09). Furthermore, commercial UML CASE tools nowadays incorporate web facilities to make web modeling easier (Borland 09)(ibm 09c).

Software construction tools is another segment where web engineering tools have had great success. Fraternali [Fraternali 99] classifies them into six categories: (i) visual editors and site managers (e.g. (Adobe 09b)(Microsoft 09)(NetObjects 09)); (ii) web-enabled hypermedia authoring tools (e.g. (fg 09)); (iii) web-DBPL integrators (e.g. (Adobe 09a)(Oracle 09b)(Sun 09)); (iv) web form editors, report, writers, and database publishing wizards (e.g. (Apple 09)(ibm 09d)(Recrystallize 09)); (v) multiparadigm tools (e.g. (ibm 09a)); and (vi) model-driven application generators (e.g. (Oracle 09a)).

Software testing tools have found great applicability in the web engineering discipline [Ash 03][Nguyeng 03]. These tools vary from those based on model-driven testing (Baresi 05) to those focused on the container testing of web portal applications (Xiong 05). An informal classification of these tools can be found in (Hower 09): load and performance test tools (e.g. (hp 09a)), link checkers (e.g. (w3c 09a)), HTML validators (e.g. (w3c 09b)), web functional regression tools (e.g. (ibm 09b)), web site security (e.g. (McAfee 09)), external site monitoring services (e.g. (webmetrics 09)), web site management tools (e.g. (hp 09b)(Vanderdonckt 01)), log analysis tools (httpd 09) (e.g. (Webtrax 09)), and other tools (e.g. (Eclipse 09)). Finally, (Rica 06) provides a similar classification of web testing tools.

7.2 Software Engineering Methods

Software engineering methods usually provide a notation and vocabulary, procedures for performing identifiable tasks, and guidelines for checking both the process and the product. This sub-area focuses on: heuristic methods, formal methods, and prototyping methods.

These topics are not disjointed because they represent different concerns. In our opinion, most of these methods have been incorporated in web engineering, as described in sections 4.5 (software design notations) and 6.1 (process definition).

7.3 Software Engineering Tools and Methods. Matrix of Knowledge Area vs. Reference Material

	[Ash 03]	[Fraternali 99]	[Nguyen 03]
Software Engineering Tools	Appendix J	Sections 3-9	Chapter 21

7.4 Recommended References for Software Engineering Tools and Methods

[Ash 03] Ash, L. (2003) The Web Testing Companion: The Insider's Guide to Efficient and Effective Tests. John Wiley & Sons Inc., Chichester.

[Fraternali 99] Fraternali, P. (1999) Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. ACM Computing Surveys, 31, 227-263.

[Nguyen 03] Nguyen, H.Q., Johnson, R., Hackett, M. (2003) Testing Applications on the Web: Test Planning for Mobile and Internet-based Systems. John Wiley & Sons Inc., Chichester.

7.5 List of Further Readings

(Adobe 09a) Adobe Macromedia ColdFusion (2009) http://www.adobe.com/products/coldfusion/

(Adobe 09b) Adobe Macromedia Dreamweaver (2009) http://www.adobe.com/products/dreamweaver/

(Apple 09) Apple WebObjects (2009) http://www.apple.com/webobjects/

(Baresi 05) Baresi, L., Fraternali, P., Tisi, M. Morasca, S. (2005) Towards Model-Driven Testing of a Web Application Generator. Proceedings of The Fifth International Conference on Web Engineering, Sydney, July-August 2005, LNCS 3579, pp. 75-86. Springer-Verlag, Berlin.

(Borland 09) Borland Together (2009) http://www.borland.com/us/products/together/index.html

(Eclipse 09) Eclipse Test and Performance Tools Platform Project (TPTP) (2009) http://www.eclipse.org/tptp/index.html

(fg 09) Formula Graphics Multimedia (2009) http://www.formulagraphics.com/

(Gómez 04) Gómez, J. (2004) Model-Driven Web Development with VisualWADE. Proceedings of The Fourth International Conference on Web Engineering, Munich, July 2004, LNCS 3140, pp. 611-612. Springer-Verlag, Berlin.

(Hower 09) Hower, R. (2009) Software Q.A./ Test Resource Center, Web Site Test Tools and Site Management Tools, http://www.softwareqatest.com/qatweb1.html

(hp 09a) HP httperf (2009) http://www.hpl.hp.com/research/linux/httperf/

(hp 09b) HP OpenView (2009) http://www.openview.hp.com/

(httpd 09) HTTPD 09 Log Analyzers (2009) http://www.uu.se/Software/Analyzers/

(ibm 09a) IBM Lotus Domino Designer (2009) http://www-142.ibm.com/software/sw-lotus/products/product4.nsf/wdocs/dominodesignerhome (ibm 09b) IBM Rational Functional Tester (2009) http://www-306.ibm.com/software/awdtools/tester/functional/index.html

(ibm 09c) IBM Rational Software Architect (2009) http://www-306.ibm.com/software/awdtools/architect/swarchitect/index.html

(ibm 09d) IBM Rational Web Developer for WebSphere Software (2009) http://www-306.ibm.com/software/awdtools/developer/web/index.html

(Knapp 04) Knapp, A., Koch, N., Zhang, G. (2004) Modeling the Structure of Web Applications with ArgoUWE. Proceedings of The Fourth International Conference on Web Engineering, Munich, July 2004, LNCS 3140, pp. 615-616. Springer-Verlag, Berlin.

(McAfee 09) McAfee Foundstone Enterprise (2009) http://www.mcafee.com/us/enterprise/products/vulnerability_management/foundstone_enterprise.html

(Microsoft 09) Microsoft Office FrontPage (2009) http://www.microsoft.com/products/info/default.aspx?view=22

(Montero 04) Montero, S., Díaz, P., Aedo, I. (2004) AriadneTool: A Design Toolkit for Hypermedia Applications. Journal of Digital Information, 5, http://jodi.tamu.edu/Articles/v05/i02/Montero/

(NetObjects 09) NetObjects Fusion (2009) http://www.netobjects.com/

(Oracle 09a) Oracle Designer (2009) http://www.oracle.com/technology/products/designer/index.html

(Oracle 09b) Oracle PL/SQL Web Toolkit (2009) http://www.oracle.com/technology/software/products/intermedia/htdocs/descriptions/web_access.html
(Paiano 04) Paiano, R., Pandurino, A. (2004) WAPS: Web Application Prototyping System. Proceedings of The Fourth International Conference on Web Engineering, Munich, July 2004, LNCS 3140, pp. 256-260. Springer-Verlag, Berlin.
(Recrystallize 09) Recrystallize Software Crystal Reports (2009) http://www.recrystallize.com/merchant/crystal-reports/crystal-reports-11.htm
(Rica 06) Ricca, F., Tonnela, P. (2006) Detecting anomaly and failure in Web applications. IEEE MultiMedia, 13, 44-51.
(Sun 09) Sun JDBC (2009) http://java.sun.com/javase/technologies/database/index.jsp
(Thalheim 04) Thalheim, B., Schewe, K.-D., Romalis, I., Raak, T., Fiedler, G. (2004) Website Modeling and Website Generation. Proceedings of The Fourth International Conference on Web Engineering, Munich, July 2004, LNCS 3140, pp. 577-578. Springer-Verlag, Berlin.
(Vanderdonckt 01) Vanderdonckt, J., Bouillon, L., Souchon, N. (2001) Flexible Reverse Engineering of Web Pages with VAQUISTA. Proceedings of the Eighth Working Conference on Reverse Engineering, Stuttgart, November 2001, pp. 241-248.
(webmetrics 09) WEBMETRICS AppMonitor (2009) http://www.webmetrics.com/applicationmonitoring.html
(WebRatio 09) WebRatio Website (2009) http://www.webratio.com/
(Webtrax 09) Webtrax (2009) http://www.multicians.org/thvv/webtrax-help.html
(w3c 09a) World Wide Web Consortium Link Checker (2009) http://validator.w3.org/checklink
(w3c 09b) World Wide Web Consortium Markup Validation Service (2009) http://validator.w3.org/
(Xiong 05) Xiong, W., Bajwa, H., Maurer, F. (2005) WIT: A Framework for In-container Testing of Web- Portal Applications. Proceedings of The Fifth International Conference on Web Engineering, Sydney, July-August 2005, LNCS 3579, pp. 87-97. Springer-Verlag, Berlin.

8 Other Knowledge Areas

As previously described, there are still SWEBOK knowledge areas to be analyzed in relation to web engineering: software construction, software maintenance, software configuration management, software engineering management and software quality, as well as disciplines related to software engineering.

As far as we have seen in literature there are no important differences between these knowledge areas in software engineering and web engineering. Thus, the percentage of web engineering papers classified under these knowledge areas is almost negligible. Of course, in software construction, standards in construction is related to section 4.2 (key issues in software design) and 4.3 (software structure and architecture), but there are no noticeable differences in the remaining topics (note that quality measurement has been analyzed in section 6.2 -process and product measurement-). In the interest of conciseness, and as we have referenced some of the more relevant standards for web development in section 4.3 (software structure and architecture), we do not make an in-depth analysis of the software construction knowledge area.

9 Conclusions

This section presents the conclusions grouped into different topics: (i) web engineering; (ii) software engineering and web engineering; (iii) development or updates of bodies of knowledge; and finally (iv) overall conclusions.

9.1 Conclusions on Web Engineering

First of all, we should conclude that web engineering is a heterogeneous discipline. As Deshpande and Hansen state: "Web engineering is a discipline among disciplines, cutting across computer science, information systems, and software engineering, as well as benefiting from several non-information technology specializations" [Deshpande, 01]. The papers analyzed during the development of the work presented in this paper endorse this claim.

However, this heterogeneity does not imply that the work regarding web engineering cannot be classified in existing disciplines. In this way, almost 89% of the papers analyzed could be categorized under the following disciplines: accessibility, agents, document and text processing, e-commerce, e-learning, hypermedia and hypertext, human-computer interaction, information systems, programming languages, semantic web, software engineering, web engineering fundamentals and security.

In this classification, 16% of the papers analyzed where related to the topic of web services. To define web services as a previously unidentified constituent discipline of web engineering was considered but web services appear in the context of agents, e-commerce, e-learning, document and text processing, information systems, software engineering and semantic web. Therefore, these disciplines should be defined as different constituents of web services. Thus, to avoid this double classification, in this paper the web services-related papers were classified under the constituent disciplines of web engineering.

Regarding the 11% of unclassified papers, in this category there are papers belonging to heterogeneous categories such as multimedia issues, web server performance analysis, domain-specific web applications, or domain-specific issues. The small number of papers in every tentative category, the heterogeneity of their nature, and the lack of any work regarding the need for recognizing them as belonging to a new specific category, has led us to consider them as unclassified.

9.2 Conclusions on Software Engineering and Web Engineering

On analyzing the literature on software-engineering related papers of the web engineering discipline we can conclude that software engineering and web engineering are different disciplines with a non-empty intersection.

This intersection is non-empty because almost 46% of the papers analyzed were related to the software engineering discipline. As this paper describes, the knowledge areas of software requirements, design, testing, process and tools as well as software construction have undergone a major evolution in the web engineering discipline in relation to their software engineering counterparts. In particular, the design knowledge area presents a significant evolution in the web engineering discipline. Possibly, the need for more specific techniques in the web engineering discipline has

promoted the evolution of these knowledge areas from the software engineering discipline. Specifically, the main changes in web engineering per knowledge area are:

- Software requirements:
  - Several characteristics of web applications that interfere with different knowledge areas are identified in web engineering (section 3.1).
  - New types of requirements arise in web engineering (section 3.1).
  - There is no clear consensus regarding the requirement process in web engineering (section 3.2).
  - There is no clear consensus regarding the requirements capture in web engineering (section 3.3).
- Software design:
  - Simple websites and complex web applications are distinguished in web engineering (section 4.1).
  - The software engineering key issues in software design take on a great relevance in web engineering. Therefore, these issues are specialized and developed further in web engineering (section 4.2).
  - New design patterns arise in web engineering. These patterns are also applicable to non web-based software (section 4.3).
  - New programming frameworks have been developed in web engineering to make the development of web applications easier (section 4.3).
  - New quality attributes arise in web engineering (section 4.4).
  - Accessibility is a key issue in the design of web applications (section 4.4).
  - Specialized design notations are needed in order to characterize the complete design of web applications (section 4.5).
- Software testing:
  - New types of tests arise in web engineering (section 5.1).
  - Security, and its testing, become a key issue in web engineering (section 5.1).
- Software engineering process:
  - Specialized process models are defined in web engineering (section 6.1).
  - New specific product measures are defined in web engineering (section 6.2).
- Software engineering tools and methods:
  - New design, development and testing tools are developed in web engineering (section 7.1).

Of course, there are web-specific works that can hardly be classified under the software engineering discipline. In fact, due to the multidisciplinary nature of the web engineering discipline, almost 54% of the papers analyzed were not related to software engineering. Thus, we can find papers on accessibility, agents, document and text processing, e-commerce, e-learning, hypermedia and hypertext, human-computer interaction, information systems, programming languages, semantic web, web engineering fundamentals, security, and others, which are not related to the software engineering discipline.

On the other hand, we can also find software engineering papers not related to the web engineering domain. It is another matter whether those papers can be applied to web engineering projects or not. For example, a software engineering book about software project management can hardly be classified under the web engineering discipline, notwithstanding project management can be very useful in web engineering projects. This feature is not specific to web engineering. For example, project management can also be very useful during the development of information systems, but project management is not classified under the information systems discipline either. Thus, the knowledge areas of software maintenance, configuration management, engineering management and quality have suffered a minor evolution in the web engineering discipline in relation to their software engineering counterparts. The minor evolution of these knowledge areas on the web engineering discipline can be explained by the fact that: (i) these are generic knowledge areas directly applicable to almost every type of software; or (ii) more research in the web engineering discipline is needed in these areas.

9.3 Conclusions on the Development or Updating of Bodies of Knowledge

The development of a body of knowledge in an engineering discipline is a complex issue that has to be carefully analyzed.

Once the different disciplines that make up the web engineering discipline have been identified, there are no particular problems with classifying the software engineering-related references in the web engineering discipline under the SWEBOK's knowledge areas. Thus, most topics in every knowledge area can be maintained, and the need for other topics should be carefully analyzed.

On the other hand, due to the presence of constituent disciplines of web engineering different from software engineering, there are knowledge areas that could be updated in the Guide to the SWEBOK. For example, the software design knowledge area could contain a sub-area on the web information system design. Including web information systems as another branch of web engineering discipline, independent of software engineering, may be another choice.

Thus, two approaches emerge: (i) considering bodies of knowledge of every constituent discipline of web engineering and enriching them with the web engineering specific literature and/or specific knowledge areas; or (ii) creating a specific web engineering body of knowledge comprising the constituent disciplines of web engineering identified in this paper as its main knowledge areas. Of course, other knowledge areas could be added if needed.

The choice between both of these approaches is an important decision that could have significant consequences in both web engineering and its constituent disciplines. Therefore, in our opinion, this work could be addressed by an international working group of multidisciplinary researchers. Indeed, this paper could be one of the items that could help make such a decision. In any case, if a new web engineering body of knowledge is made and this new body of knowledge makes references to other disciplines (e.g. software engineering or information systems), readers should have some knowledge of these disciplines in order to understand the web engineering discipline.

Regarding the software engineering discipline, in our opinion, it is a fact that if this discipline is required to be able to characterize the design and development of

present multi-tier, distributed, and service-oriented web applications, several web engineering references have to be included in the Guide to the SWEBOK in order to update it. In this case, the significant amount of references in some knowledge areas (e.g. software design) makes it impossible to stay within the limitation of the Guide to the SWEBOK of five hundred pages of reference material per knowledge area. Otherwise, if software engineering is required to remain a generic discipline, independent of any deployment platform, the web engineering community should analyze the advantages of developing a web engineering body of knowledge.

9.4 Overall Conclusions and Future Work

In conclusion, we should state that web engineering is a multidisciplinary discipline strongly influenced by software engineering. Both disciplines are different, but have a non-empty intersection. Thus, this paper tries to characterize the nature of this non-empty intersection, reviewing a significant part of the web engineering literature related to software engineering. This literature has been classified according to the knowledge areas defined in the Guide to the SWEBOK, and after analyzing this classification, we can conclude that software engineering is one of the main constituent disciplines of web engineering.

Regarding the remaining constituent disciplines, as well as the unclassified material, similar work to the one presented in this paper could be done in order to characterize the remaining web engineering literature. Thus, the web engineering discipline could be entirely characterized by its discrete constituent disciplines. After the comparison of web engineering versus its constituent disciplines is made, the decision to develop a web engineering body of knowledge could take into account important background work that may help to make such a decision.

Acknowledgements

El Ministerio de Educación y Ciencia (TIN2008-06708-C03-01/TSI, TIN2009-14317-C03-01/TSI), *La Comunidad Autónoma de Madrid* (4155/2005) and *La Universidad Complutense de Madrid* (*Group* 921340) have supported this work.

References

[Abran, 04] Abran, J.W. Moore, P. Bourque, and R. Dupuis.: Guide to the Software Engineering Book of Knowledge: 2004 Edition-SWEBOK. http://www.swebok.org 2004.

[Alur, 03] Alur, D., Crupi, J., and Malks, D.: Core J2EE Patterns. Best Practices and Design Strategies. Sun Microsystems Press – Prentice Hall, Upper Saddle River, NJ, 2003.

[ACM, 00] Association for Computing Machinery. A Summary of the ACM Position on Software Engineering as a Licensed Engineering Profession, 2000, http://www.acm.org/serving/se_policy/selep_main.html.

[Barta, 98] Barta, R.A., and Schranz, M.W. (1998). JESSICA: an Object-Oriented Hypermedia Publishing Processor. *Computer Networks*, 30: 281-290.

[Burke, 06] Burke, B., and Monson-Haefel, R. Enterprise JavaBeans 3.0. O'Reilly Media Inc., Sebastopol, CA, 2006.

[Callahan, 02] Callahan, D., and Pedigo, B. (2002). Educating Experienced IT Professionals by Addressing Industry's Needs. *IEEE Software* 19: 57-62.

[Carrington, 05] Carrington, D., Strooper, P., Newby, S., and Stevenson, T. (2005). An Industry/University Collaboration to Upgrade Software Engineering Knowledge and Skills in Industry. *Journal of Systems and Software* 75: 29-39.

[Casteleyn, 09] Casteleyn, S., Daniel, F., Dolog, P., Matera, M. Engineering Web Applications. Springer, 2009.

[Deshpande, 01] Deshpande, Y., And Hansen, S. (2001). Web Engineering: Creating a Discipline among Disciplines. *IEEE MultiMedia* 8: 82-87.

[Deshpande, 02] Deshpande, Y. Murugesan, S., Ginige, A., Hansen, S., Schwabe, D., Gaedke, M., and White, B. (2002). Web Engineering. *Journal of Web Engineering* 1: 3-17.

[Gellersen, 97] Gellersen, H.-W., Wicke, R., and Gaedke, M. (1997). WebComposition: An Object-Oriented Support System for the Web Engineering Lifecycle. *Computer Networks* 29: 1429-1437.

[Ginige, 01] Ginige, A., and Murugesan, S. (2001). Guest Editors' Introduction: The Essence of Web Engineering-Managing the Diversity and Complexity of Web Application Development. *IEEE MultiMedia* 8: 22-25.

[IEEE, 90] IEEE Std. 610.12-1990. IEEE standard glossary of software engineering terminology. The Institute of Electrical and Electronics Engineers, New York, NY, 1990

[ICWE, 09] International Conference on Web Engineering 2009, http://icwe2009.webengineering.org/

[IJWET, 09] International Journal of Web Engineering and Technology, http://www.inderscience.com/browse/index.php?journalID=48#board

[JWE, 09] Journal of Web Engineering, http://www.rintonpress.com/journals/jwe/

[Kappel, 04] Kappel, G., Michlmayr, E., Pröll, B., Reich, S., and Retschitzegger, W. 2004. Web Engineering, Old Wine in New Bottles? In International Conference on Web Engineering 2004 (ICWE 2004), LNCS 3140, 6-12.

[Kappel, 06] Kappel, G., Pröll, B., Reich, S., and Retschitzegger W. (Eds.). Web Engineering: The Discipline of Systematic Development. John Wiley & Sons Inc., Chichester, 2006.

[Lowe, 03] Lowe, D. (2003). Web system requirements: an overview. *Requirements Engineering* 8: 102-113.

[Mendes, 01] Mendes, E., Mosley, N., and Counsell, S. (2001). Web Metrics-Estimating Design and Authoring Effort. *IEEE MultiMedia* 8: 50-57.

[Mendes, 05] Mendes, E., and Mosley, N. (Eds.). Web engineering. Springer-Verlag, Berlin, 2005.

[Murugesan, 01a] Murugesan, S., and Deshpande, Y. (Eds.) Web Engineering: Managing Diversity and Complexity of Web Application Development. LNCS 2016, Springer-Verlag, Berlin, 2001.

[Murugesan, 01b] Murugesan, S., Deshpande, Y., Hansen, S., and Ginige, A. Web Engineering: A New Discipline for Development of Web-Based Systems. In Web Engineering: Managing

Diversity and Complexity of Web Application Development, S. Murugesan, and Y. Deshpande, Eds. LNCS 2016, Springer-Verlag, Berlin, 2001.

[Navarro, 05] Navarro, A., Sierra, J.L., Fernández-Valmayor, A., and Fernández-Manjón, B. A First Step Towards the Web Engineering Body of Knowledge. In Proceedings of Fifth International Conference on Web Engineering, LNCS 3579, 2005, 585-587.

[Pressman, 04] Pressman, R.S. Software Engineering: A Practitioner's Approach. 6th edition. McGraw-Hill, New York, NY, 2004

[Rossi, 07] Rossi, G., Pastor, O., Schwabe, d. and Olsina, L. (Eds.) Web Engineering: Modelling and Implementing Web Applications. Springer. 2007.

[Saiedian, 02] Saiedian, H., Bagert, D.J., and Mead, N.R. (2002). Software Engineering Programs: Dispelling the Myths and Misconceptions. *IEEE Software* 19: 35-41.

[Suh, 05] Suh, W. (Ed.). Web Engineering Principles and Techniques. Idea Group Publishing, Hershey, PA, 2005.

[White, 96] White, B. Web Document Engineering. Stanford Linear Accelerator Publication, SLAC-PUB. 7150, 1996.

[WWW, 09] World Wide Web Conference 2009, http://www2009.org/