Agile Web Development Methodologies: A Survey and Evaluation

Nasrin Ghasempour Maleki and Raman Ramsin

Abstract Dynamic and accessible web systems have gained utmost importance in modern life. Due to the competitive nature of such systems, they need to be superior as to performance, scalability, and security. Web systems typically require short time-to-markets, and it should be possible to easily implement new requirements into working web systems. These ideals have made agile methods especially suitable for developing such systems, as they promote productivity, facilitate continuous interaction with customers, and enhance the flexibility and quality of the software produced. When starting a web development project, selecting the methodology that fits the project situation can be an important factor in the ultimate success of the endeavor. In order to facilitate the selection process, we provide a criteria-based evaluation of fourteen agile web development methodologies. The evaluation results highlight the strengths and weaknesses of the methodologies as to their general processes, modeling languages, agile features, and web development facilities, and can therefore help web developers choose the methodology that best fits their project needs.

Keywords Software development methodology · Agile method · Web system · Web development methodology · Criteria-based evaluation

1 Introduction

Businesses increasingly rely on web systems for maintaining their competitive edge, and the widespread use of these systems has made them indispensable in everyday life. Due to their pivotal role, web systems have to be developed fast, and

N.G. Maleki · R. Ramsin (✉)
Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran
e-mail: ramsin@sharif.edu

N.G. Maleki
e-mail: ghasempourmk@alum.sharif.edu

R. Lee (ed.), *Software Engineering Research, Management and Applications*,
Studies in Computational Intelligence 722, DOI 10.1007/978-3-319-61388-8_1

they should be flexible enough to be easily changed and extended as required; also, special attention should be given to proper requirements engineering and continuous verification/validation of these systems. An important feature of web development projects is their highly dynamic nature, which necessitates constant user feedback. Due to the above characteristics, web development involves much more than mere web "programming": developers have thus realized that using the right software development methodology is essential for successful construction and evolution of web systems.

Agile methodologies are suitable candidates for developing web systems, since they adequately address the specific needs of this context. However, there are many agile web development methodologies to choose from, and choosing the right one can be a serious challenge for web development teams. Making the right choice requires adequate knowledge about the strengths and weaknesses of each methodology; however, development teams should not be expected to acquire this knowledge through hands-on experience with each and every methodology. Fortunately, criteria-based evaluation of methodologies is a proven method for identifying and accentuating the capabilities and limitations of software development methodologies. Several such evaluations have previously been conducted on various types of methodologies [1–3], but the need remains for a comprehensive evaluation of modern agile web development methodologies.

We provide a comprehensive criteria-based evaluation of fourteen prominent agile web development methodologies. Methodologies have been targeted for evaluation based on their popularity and documentation; methodologies that lack proper methodology documentation (on the process, products, and people involved) have not been included. The evaluation criteria have been collected from multiple sources, and have been adapted to the specific characteristics of the agile web development context. Evaluation results clearly show the pros and cons of the methodologies, and can be used by web developers to choose the methodology that fits their needs.

The rest of the paper is structured as follows: Sect. 2 provides a brief overview of the targeted agile web development methodologies; Sect. 3 presents the evaluations criteria; Sect. 4 lists the results of applying the evaluation criteria to the methodologies; and Sect. 5 presents the conclusions and suggests ways for furthering this research.

2 An Overview of Targeted Methodologies

The fourteen agile web development methodologies targeted for evaluation have been briefly introduced throughout the rest of this section.

2.1 MockupDD

MockupDD is an agile model-driven web engineering methodology based on Scrum [4]; its process consists of four phases (Fig. 1):

1. *Mockup Construction*: Requirements are gathered from the collection of stories by customers or final users through using mockups to produce graphical stories.
2. *Mockup Processing*: Important parts of the UI are identified through mapping the basic concepts of mockups to a structural UI meta-model.
3. *Features specification and tags refinement*: Mockups are tagged with labels that represent their semantics. User stories are then adapted with the mockups, and the tags are classified.
4. *Code and Model Generation*: Tags will either be converted into web engineering elements, or be combined to identify more complex design features. After the full definition of tags, an executable version is produced; other models of model-driven web engineering are created based on this version.

2.2 RAMBUS

RAMBUS is an agile methodology loosely based on Scrum [5]; its process consists of three phases (Fig. 2):

1. *Communication*: Communication with users is performed to capture the functional requirements on story cards. To show the behavior of the system, a

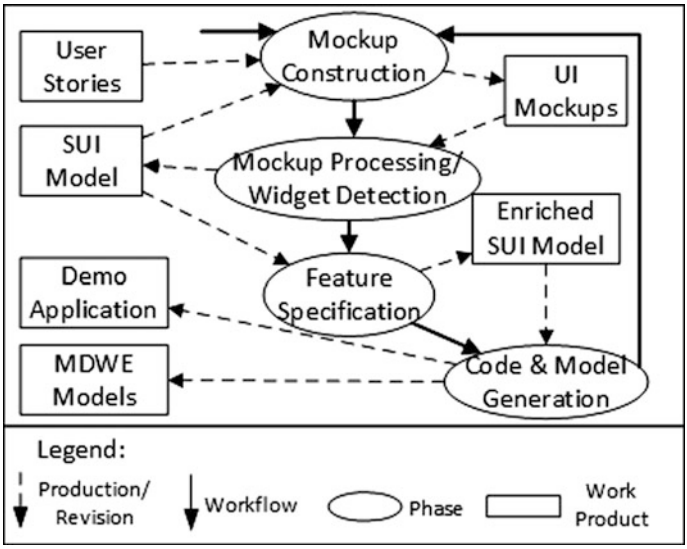

Fig. 1 Process of MockupDD

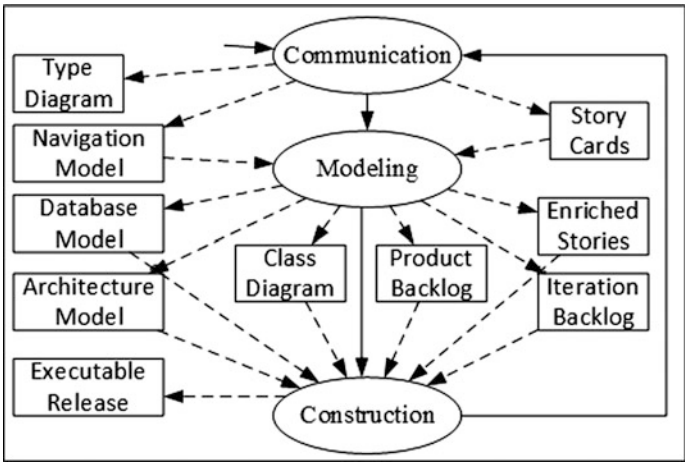

Fig. 2 Process of RAMBUS

navigation model is created. For each story card, user priorities, predicted difficulties in implementing the story, and the relevant items of the navigation model are written on the back of the card. Type diagrams are produced to show the relationships of the elements.

2. *Modeling*: Class and type diagrams are developed/refined iteratively, and a database model is created. User stories are enriched with user acceptance criteria. Reuse options are explored, and nonfunctional requirements are considered in the user stories.
3. *Construction*: Coding and testing are performed, resulting in an executable release. Daily sessions, strict coding standards, test-driven development, continuous integration, and pair programming are the agile practices prescribed by the methodology for this particular phase.

2.3 USABAGILE_Web

USABAGILE_Web is a methodology for designing or reengineering a web system by architectural analysis, creating a UI prototype, and usability testing [6]. Before the main process, three usability assessment activities are performed:

1. *Inspection*: UI structure is inspected to detect usability problems. Typically, a team of 3–5 specialists performs Nielsen analysis. UI functionality is not considered.
2. *Evaluation*: Under the supervision of experts, the usability of web pages is analyzed based on components such as links, forms, and the elements with which the user interacts.

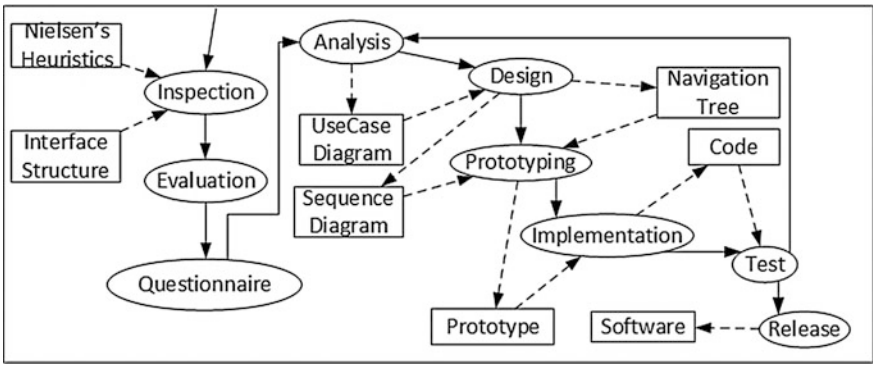

Fig. 3 Process of USABAGILE_Web

3. *Questionnaire*: A questionnaire is used for capturing the wishes and feelings of users after using the UI.

The results of the above activities are documented in a special usability report. The main process uses this usability report as input, and consists of six phases (Fig. 3):

1. *Analysis*: UI behavior is captured in behavioral use case diagrams.
2. *Design*: A summary of the UI structure related to user operations is produced for logical analysis. Navigation features of the UI are shown to the customer, and the feedback is used for analyzing the UI design.
3. *Prototyping*: This phase is integrated with the two previous phases. UI prototypes are created by experts based on analysis and design results.
4. *Implementation*: After UI prototypes are accepted by the customers and experts, the system is implemented.
5. *Test*: A set of potential users are selected (preferably from among those who filled the assessment questionnaire) to test the new UI. New features are implemented as required, and the process is iterated until the product is fully validated by the users.
6. *Release*: The produced/reengineered web system is deployed into the user environment.

2.4 Augmented WebHelix

WebHelix was introduced in 2006 as a spiral lightweight methodology for teaching web development to students [7]. Augmented WebHelix is a practical, business-oriented web development methodology that extends WebHelix with management and Q/A activities [8]; its main process consists of eight phases (Fig. 4):

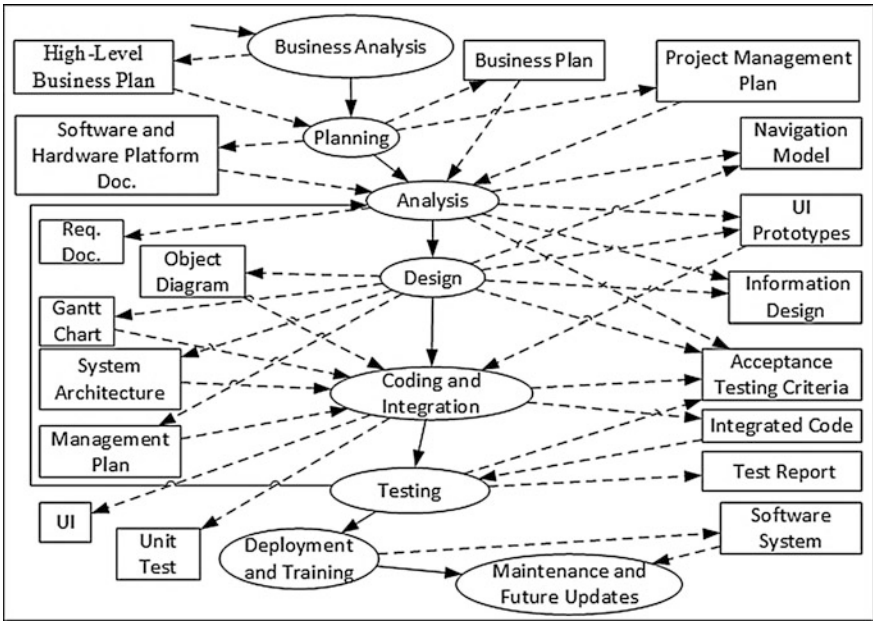

Fig. 4 Process of Augmented WebHelix

1. *Business Analysis*: Spans identifying business processes, identifying real and virtual chains of supply, and providing a high-level business plan.
2. *Planning*: Spans identifying the software and hardware platforms, specifying the project management scheme and the necessary tools and resources, and producing a business plan.
3. *Analysis*: Spans creating or updating the requirements, creating/updating the navigation model, creating UI prototypes, creating/updating the information structure, and identifying criteria for acceptance testing.
4. *Design*: Spans creating or updating a detailed system architecture, updating the system UI and navigation model, creating a system object diagram, creating or updating the system information design, creating or updating the management plan, forming the programming team, creating a Gantt chart, and identifying test criteria for system acceptance.
5. *Coding and Integration*: Spans components selection, implementing the UI, coding, integration, unit testing, code review, and updating the acceptance criteria.
6. *Testing*: Spans web design testing, multimedia testing, and user acceptance testing.
7. *Deployment and Training*: The system is deployed into the network environment, and the users are trained.
8. *Maintenance and Future Updates*: Spans maintaining and updating the system.

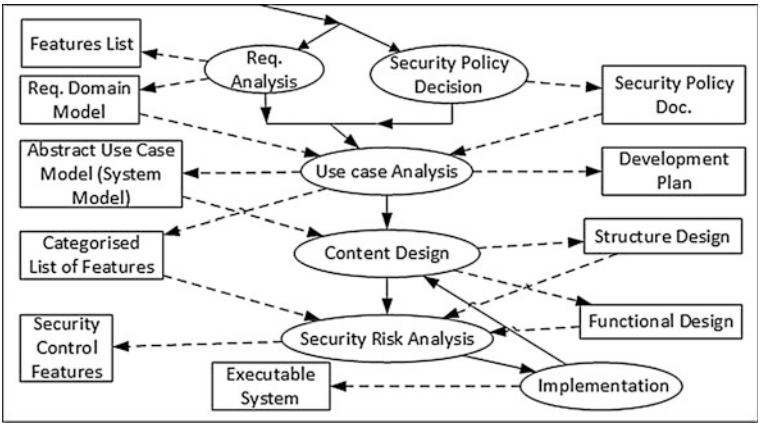

Fig. 5 Process of Secure FDD

2.5 Secure FDD

Secure FDD extends the Feature-Driven Development (FDD) methodology with security analysis and design features in order to develop secure web systems [9]; its process consists of six stages (Fig. 5):

1. *Requirements Analysis*: Security-related needs and expectations of the stakeholders are identified, and security rules are set. A list of features (as defined in FDD) is also produced.
2. *Security Policy Decision*: Policies on how to implement security are specified. These policies help build the web system in a security-conscious manner.
3. *Use Case Analysis*: Features are classified and an overall structural model is produced. Use case analysis is performed for refining the system scope.
4. *Content Design*: A blueprint for implementing the features is produced by conducting structural design (focusing on feature content) and functional design (focusing on the user actions involved in each feature).
5. *Security Risk Analysis*: An iterative-incremental process is performed to determine security control features.
6. *Implementation*: The target system is implemented, with special attention to security features.

2.6 XWebProcess

XWebProcess extends the Extreme Programming (XP) methodology with web development features [10]; its process consists of six stages (Fig. 6):

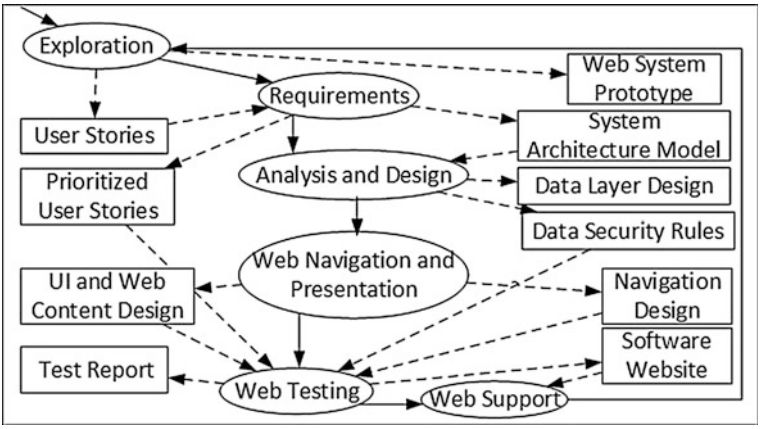

Fig. 6 Process of XWebProcess

1. *Exploration*: High-level requirements are captured in user stories, and the overall system design is determined by prototyping.
2. *Requirements*: The system architecture is defined, with special attention to flexibility, efficiency, and maintainability. User stories are estimated and prioritized, and high-priority stories are selected for development in the next cycle.
3. *Analysis and Design*: The data layer is designed based on the data recovery and security rules added to XP.
4. *Web Navigation and Presentation*: Web content and navigation is designed by using the design practices added to XP. The web system is then implemented.
5. *Web Testing*: Verification and validation are performed, and detected bugs are fixed. Support analysts assist the developers if a special setup configuration (e.g., files, devices, and environment variables) is required for running the tests.
6. *Web Support*: Website components are maintained.

2.7 XP

The XP methodology (in its original, non-extended form) can also be effectively used for developing web systems [11, 12]; its process consists of six phases (Fig. 7):

1. *Exploration*: Activities include team formation, elicitation of high-level requirements (as user stories), and specification of system architecture.
2. *Planning*: User stories are estimated, prioritized, and broken down into development tasks for programmers to complete in 1–3 weeks. A subset of the stories is then selected for implementation in the first release.

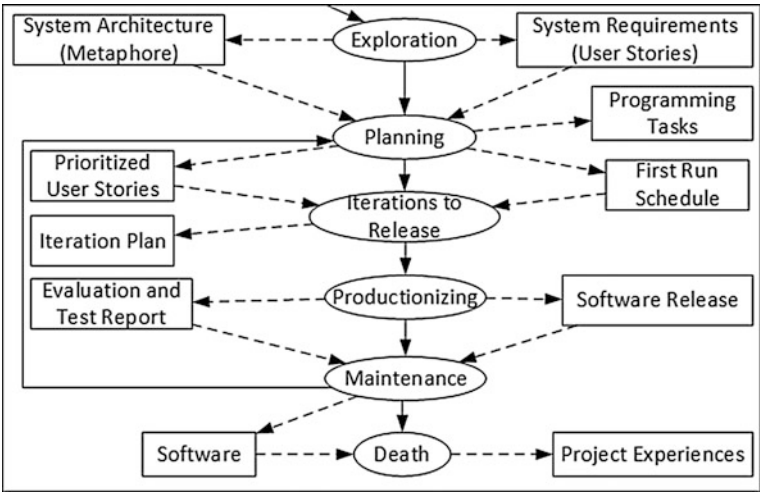

Fig. 7 Process of XP

3. *Iterations to Release*: Analysis, design, coding, testing and integration are performed iteratively in a collective code ownership environment.
4. *Productionizing*: System-wide testing is performed, and the system is deployed into the user environment.
5. *Maintenance*: The remaining user stories are implemented by repeating phases 2, 3 and 4.
6. *Death*: Project review and post-mortem are conducted.

2.8 UML-Based Agile Method

The agile web development method proposed by Lee et al. involves modeling activities using an extension of UML [13]. The produced UML model for the web application consists of a navigation model, a components communication model, a conceptual model, and an architectural model. Developers can thus take advantage of both model-based and test-based development. Its cyclic process consists of two phases (Fig. 8):

1. *Analysis*: Requirements analysis is performed, and the conceptual model is produced (as a class diagram). An architecture is also defined (as a component diagram).
2. *Construction*: This phase consists of two sub-phases: Build and Sophistication. During Build, developers iteratively select a subset of the requirements and

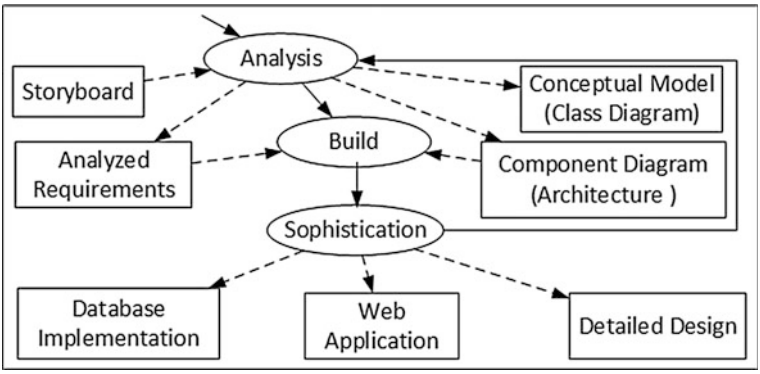

Fig. 8 Process of UML-Based Agile Method

build a storyboard that depicts how the requirements are realized through user interactions. Sophistication involves detailed design and implementation.

The implemented components are integrated with existing subsystems, and integration/regression tests are applied. The cycle is repeated until all the requirements are satisfied.

2.9 Crystal Orange Web

Crystal Orange Web is a variant of Cockburn's Crystal Orange methodology specifically designed for ongoing web development projects [14]. It stresses the importance of collaboration among the developers, and makes extensive use of agile practices. Instead of providing a specific lifecycle, the methodology prescribes five agile conventions: "Regular Heartbeat with Learning", "Basic Process", "Maximum Progress, Minimum Distractions", "Maximally Defect Free", and "A Community Aligned in Conversation"; these conventions facilitate the development and constant evolution of a web system over an extended period of time.

2.10 S-Scrum

S-Scrum is a variant of Scrum aimed at developing secure web systems [15]. The objective is to provide critical security web services and perform security analysis and design during early stages of Scrum. The methodology accommodates changing requirements; moreover, if the changed requirement is a critical security requirement, the current sprint is cut short and a new sprint is started in order to implement the changed requirement. The process of S-Scrum is analogous to the

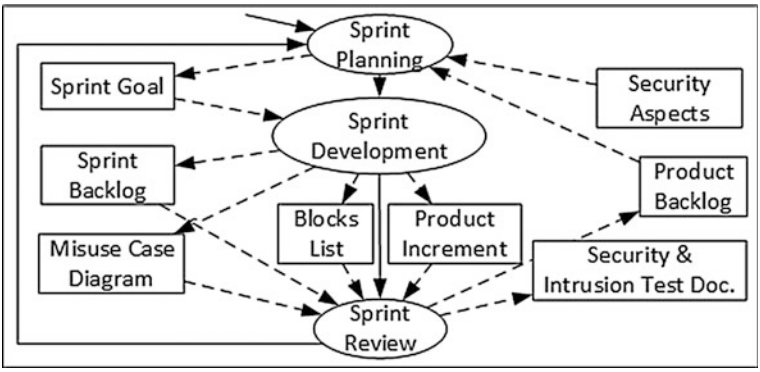

Fig. 9 Process of S-Scrum

original Scrum process; however, special attention is given to developing and applying security and intrusion tests, and producing a misuse case diagram (Fig. 9).

2.11 Scrum for CMMI Level 2

Salinas et al. have proposed an extended variant of Scrum to accommodate CMMI-Level 2 in the context of web development [16]. The methodology claims to have achieved this by adding a time-boxed "Sprint 0" at the beginning of the Scrum process (Fig. 10). "Sprint 0" deals with quality assurance, project data management, and project evaluation. After "Sprint 0", the original Scrum process is enacted along with the proposed extensions: project data is collected during Scrum meetings, and project reports are produced at the end of each sprint.

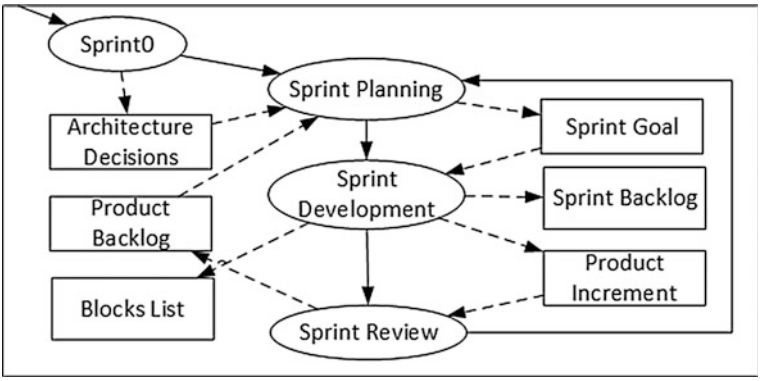

Fig. 10 Process of Scrum for CMMI Level 2

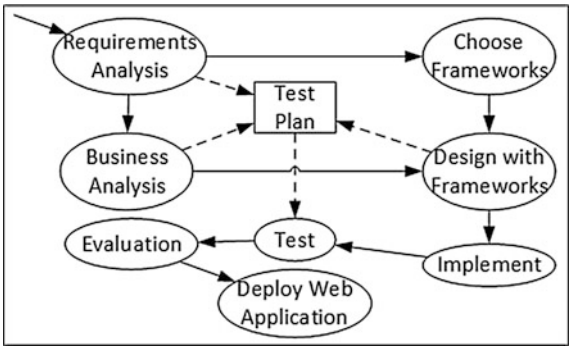

Fig. 11 Process of AWDWF

2.12 AWDWF

As the name suggests, AWDWF (Agile Web Development with Web Framework) is the result of integrating Web Framework features with the Agile Web Development process, with the specific aim of achieving fast response to requests and quick adaptation to change [17]; an analysis conducted on web development with AWDWF has shown that productivity and quality are improved. The process of the methodology consists of eight phases (Fig. 11): Requirements Analysis, Business Analysis, Choose Frameworks, Design with Frameworks, Implement, Test, Evaluation, and Deploy Web Application. The Web Framework provides a simple MVC-based programming model that can shorten the development cycle.

2.13 AWE

The iterative process of the Agile Web Engineering (AWE) methodology [18] consists of six stages (Fig. 12): Business Analysis, Requirements Analysis, Design,

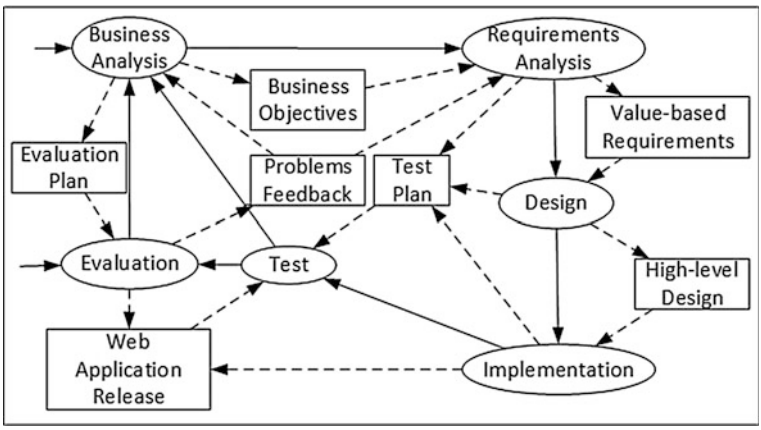

Fig. 12 Process of AWE

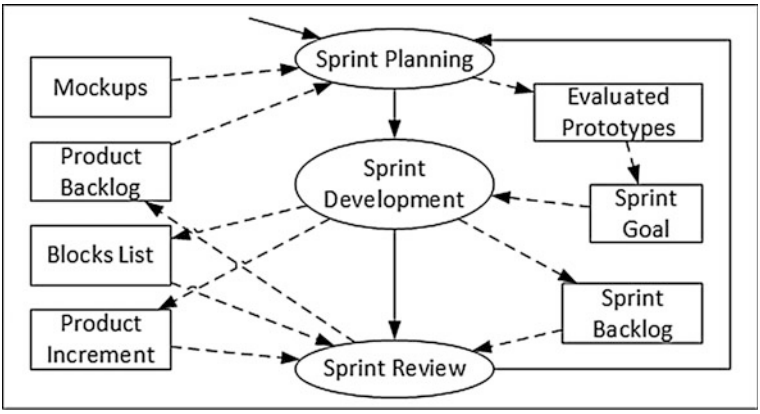

Fig. 13 Process of MDE-Scrum

Implementation, Test, and Evaluation. During the Design stage, a high-level implementation is produced that addresses all architectural issues. This version is evolved into a release of the system during Implementation and Test. Evaluation involves design-independent appraisal by the ambassador user and developers.

2.14 MDE-Scrum

This mockup-based methodology combines Model-Driven Engineering (MDE) with Scrum [19]. At the start of the process (Fig. 13), requirements are captured in user stories and mockups of the system are designed. User-approved mockups are converted into annotated UML models of the desired system. A functional prototype of the system is then generated based on these models, and is converted into an executable release. Major conversions are automated through the MockupToME tool. The Scrum process facilitates the conversion process by focusing the effort on specific high-priority features, and by supporting user-centered refinement of mockups and models.

3 Evaluation Criteria

The evaluation criteria were collected from various sources, including [1–3, 20]. They have been grouped based on the methodology feature that they evaluate. There are four groups of criteria: *modeling language*, *process*, *agility*, and *web-based features;* the criteria belonging to these categories are described in Tables 1, 2, 3 and 4, respectively.

Table 1 General criteria for evaluating methodologies—Modeling language group [1]

Name	Type	Possible values
Support for specific modeling language	SC	1: Not prescribed/enforced; 2: Prescribed; 3: Enforced
Simplicity to learn and use	SM	Yes/No
Expressiveness of modeling language	SM	Yes/No
Support for complexity management	SM	Yes/No

Table 2 General criteria for evaluating methodologies—Process group [1]

Name	Type	Possible values
Coverage of generic lifecycle	SC	D: Definition; C: Construction; M: Maintenance
Support for seamless transition between phases	SC	1: No; 2: Potentially; 3: Yes
Support for smooth transition between phases	SC	1: No; 2: Potentially; 3: Yes
Type of lifecycle	D	Waterfall (W.), Iterative-Incremental (I.-I.), etc.
Attention to design activities	SM	Yes/No
Potential of integration with other methodologies	SC	Integration strategy: 1: Not required; 2: Required but not provided; 3: Provided
Adequacy of products	SC	Relevant products in: 1: No phases; 2: Some phases; 3: All phases
Consistency of products	SC	1: Products overlap; 2: Products do not overlap
Support for modeling different views in products	SC	S: Structural; F: Functional; B: Behavioral
Support for modeling different granularity levels in products	SC	S: System; P: Package; C: Component; O: Object; D: Domain; SD: Sub-Domain; PR: Product; F: Features
Support for modeling different abstraction levels in products	SC	A: Analysis; D: Design; I: Implementation
Testability of products	SC	1: Not addressed; 2: Partial; 3: High
Tangibility of products (to customer and/or development team)	SC	1: None tangible; 2: Some not tangible to team members; 3: Some not tangible to customer; 4: All tangible
Traceability of products to requirements	SM	Yes/No
Definition of roles	SC	1: Roles not defined; 2: Roles defined, but without responsibilities; 3: Both roles and responsibilities defined
Required team knowledge/experience	SM	Yes/No
Support for team motivation mechanisms	SM	Yes/No
Expressiveness of process	SC	1: No; 2: To some extent; 3: Yes

(continued)

Table 2 (continued)

Name	Type	Possible values
Completeness of process definition	SC	L: Lifecycle; A: Activities; TP: Techniques/Practices; R: Roles; P: Products; U: Umbrella Activities; RL: Rules; ML: Modeling Language
Rationality and consistency of activities	SC	1: Problems in consistency and rationality; 2: Problems in consistency; 3: Problems in rationality; 4: No problems
Support for complexity management in process	SM	Yes/No
Attention to detail in process definition	SC	Details provided for: 1: No phases; 2: Some of the phases and internal tasks; 3: All phases and internal tasks
Definition of phase inputs and outputs (I/O)	SC	1: I/O not defined; 2: I/O defined implicitly; 3: I/O explicitly defined for all phases
Availability of documentation on process	SM	Yes/No
Tool support for process	SM	Yes/No
Ease of use of process	SC	1: Weak; 2: Average; 3: Good
Availability of experience reports of practical use	SM	Yes/No
Configurability of process	SC	1: No; 2: Possible, but not addressed explicitly; 3: Explicitly addressed
Flexibility of process	SC	1: No; 2: Possible, but not addressed explicitly; 3: Explicitly addressed
Specification of criticality level addressed by process	SC	1: Defined explicitly; 2: Not defined explicitly, but can be inferred; 3: Not defined and cannot be inferred
Platform-adaptivity of process	SM	Yes/No
Support for formalism	SM	Yes/No
Support for scalability	SC	1: Small; 2: Medium; 3: Large
Support for modularity	SM	Yes/No
Support for requirements elicitation	SC (D)	M: Uses conventional methods (description); D: Uses a specific method (description); N: No certain way
Support for requirements specification	D	
Support for requirements-based process	SM	Yes/No
Support for requirements prioritization	SM	Yes/No
Need for observation of specific constraints/assumptions	SC	1: Constraints/Assumptions exist 2: Constraints/Assumptions prescribed 3: No constraints/assumptions

Table 3 Criteria related to agility characteristics [1]

Name	Type	Possible values
Support for early and continuous delivery of working software	SC	1: Neither early nor continuous; 2: Continuous but not early; 3: Early and continuous
Support for active user involvement	SM	Yes/No
Support for continuous customer feedback	SM	Yes/No
Support for self-organizing teams	SC	1: Not discussed; 2: Addressed; 3: Ignored
Support for face-to-face conversation	SM	Yes/No
Support for velocity monitoring and control	SM	Yes/No
Attention to team behavior/efficiency	SM	Yes/No
Task assignment method	D	Voluntary sign up, Team-assigned, Manager-assigned, etc.
Support for continuous integration	SM	Yes/No
Modeling coverage	SM	Yes/No
Support for standards	SM	Yes/No
Support for iterative-incremental process	SM	Yes/No
Support for agile techniques	SM	Yes/No
Support for requirements flexibility	SM	Yes/No
Support for rapid production of artifacts	SC	1: No; 2: To some extent; 3: Yes
Support for lean development (through short time spans, and the use of tools)	SM	Yes/No
Support for learning (from previous iterations/projects)	SC	1: Not addressed; 2: Addressed implicitly; 3: Addressed explicitly
Provision of feedback by process	SM	Yes/No

The evaluation framework has been validated according to the four meta-criteria defined in [21]; validation shows that the proposed criteria are *general* enough to be applied to all agile web development methodologies, *precise* enough to help identify their similarities and differences, *comprehensive* enough to cover their important characteristics, and *balanced* in covering the major types of features in a methodology (Technical, Managerial, and Usage). The criteria's definition conforms to the Feature Analysis approach [22], in that they are of three types (based on their results): *Simple* (SM: Yes/No results), *Scale* (SC: results are discrete levels), and *Descriptive* (D: results are narrative statements).

Table 4 Criteria related to key features of web-based systems [2, 3]

Name	Type	Possible values
New or extended methodology	SM	1: New; 2: Extended (methodology)
Support for specification of technical web characteristics	SM	Yes/No
Support for architectural web design	SM	Yes/No
Support for early UI design	SM	Yes/No
Support for web-based security	SM	Yes (How?)/No
Support for rapid web development	SM	Yes (How?)/No
Support for web usability	SM	Yes (How?)/No
Addressed level of web criticality	SM	1: Low, 2: Medium, 3: High
Support for web reliability	SM	Yes (How?)/No
Support for web flexibility	SM	Yes (How?)/No
Attention to web design aspects (logic, content, navigation, UI)	SC	1: Logic; 2: Content; 3: Navigation; 4: UI; 5: Not addressed
Support for tuning the development speed based on process feedback	SC	1: No; 2: Some recommendations given; 3: Fully supported
Specification of web-related products and roles	SC	1: Only type defined; 2: Names and some recommendations given for products; 3: Fully defined

4 Results of Evaluation

The results of evaluating the targeted web development methodologies are presented in Tables 5, 6, 7 and 8, based on the type of criteria used for evaluation. If a methodology cannot be evaluated according to a certain criterion, the result has been marked with a '–'. The results clearly highlight the strengths and weaknesses of each methodology, and can be used for selecting and/or improving the methodologies.

Table 5 Results of evaluation based on general methodology evaluation criteria—Modeling language

Criterion	Targeted methodologies													
	MockupDD	RAMBUS	USABAGILE_Web	WebHelix	S-FDD	XWeb-Process	XP	UML-AW	Crystal Orange Web	S-Scrum	Scrum-CMMI	AWDWF	AWE	Scrum-MDE
Support for specific modeling language	1	1	3	1	3	1	1	3	1	1	1	1	1	3
Simplicity to learn and use	–	–	Y	–	Y	–	–	Y	–	–	–	–	–	Y
Expressiveness of modeling language	–	–	Y	–	Y	–	–	Y	–	–	–	–	–	Y
Support for complexity management	–	–	Y	–	Y	–	–	N	–	–	–	–	–	Y

Table 6 Results of evaluation based on general methodology evaluation criteria—Process

Criterion	Targeted methodologies													
	MockupDD	RAMBUS	USABAGILE_Web	WebHelix	S-FDD	XWeb-Process	XP	UML-AW	Crystal Orange Web	S-Scrum	Scrum-CMMI	AWDWF	AWE	Scrum-MDE
Coverage of generic lifecycle	D C M	D C M	D C M	D C M	D C M	D C M	D C M	D C M	–	D C M	D C M	D C M	D C M	D C M
Support for seamless transition	3	2	2	1	3	1	1	3	–	1	1	1	1	1
Support for smooth transition	3	3	3	3	3	3	3	3	–	3	3	3	3	3
Type of lifecycle	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.	I.-I.
Attention to design activities	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Potential of integration	1	1	1	1	1	1	1	1	2	1	1	1	1	1
Adequacy of products	3	3	3	3	3	3	3	3	–	3	3	1	3	3
Consistency of products	2	2	2	2	2	2	2	2	–	2	2	2	2	2
Support for modeling different views	S F B	S F B	S F B	S B	S F B	S	S	S F B	S	S F	–	S F B	S	S F B
Support for modeling different granularity levels	P C D	P C D O	P D	P D	S C O D F	O	O	P C O D	–	P D	P D	P D	P	P C D
Support for modeling different abstraction levels	D I	A D I	A D I	D I	A D I	D I	D I	A D I	–	A D I	A D I	A D I	I	D I
Testability of products	3	3	3	3	3	3	3	3	–	3	3	2	3	3
Tangibility of products	4	4	4	4	4	4	4	4	–	4	4	1	4	4
Traceability to requirements	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y	Y	Y
Definition of roles	1	3	1	1	3	3	3	3	3	3	3	2	3	3
Team knowledge/experience	N	Y	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Support for team motivation	1	1	1	1	1	2	2	1	2	1	1	1	1	1
Expressiveness of process	3	2	3	3	3	3	3	3	3	3	3	3	3	2

(continued)

Table 6 (continued)

Criterion	Targeted methodologies													
	MockupDD	RAMBUS	USABAGILE_Web	WebHelix	S-FDD	XWeb-Process	XP	UML-AW	Crystal Orange Web	S-Scrum	Scrum-CMMI	AWDWF	AWE	Scrum-MDE
Completeness of definition	L A P U	L A P U R RL	L A P U	L A U	L A P U R ML	L A P U R TP	L A P U R TP	L A P U R RL ML	L A U R RL	L A P U TP R RL	L A P U R RL	L A R	L A P U R TP	L A P U
Rationality and consistency	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Complexity management	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y	Y	Y
Attention to detail in process	3	2	3	2	3	3	3	4	2	3	3	3	1	2
Definition of phase I/O	3	3	2	2	2	3	3	3	1	3	3	1	3	2
Availability of documentation	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	N
Tool support for process	Y	Y	N	N	Y	Y	Y	Y	–	N	N	N	N	Y
Ease of use of process	2	3	3	3	3	3	3	3	–	3	3	3	3	3
Availability of reports	Y	N	N	N	Y	Y	Y	Y	–	Y	Y	Y	Y	Y
Configurability of process	1	1	1	1	3	1	1	1	3	3	3	3	1	1
Flexibility of process	3	3	3	3	3	3	3	1	3	3	3	3	3	1
Criticality level	2	2	2	2	1	2	2	2	1	1	2	2	2	2
Platform-adaptivity of process	N	N	N	Y	N	N	N	Y	–	N	N	Y	N	N
Support for formalism	N	N	N	N	N	N	N	N	–	N	N	N	N	N
Scalability	2	2	3	3	3	2	2	2	2	3	3	2	3	2
Modularity	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y	Y	Y
Requirements elicitation	D: User Story	D: User Story	M: By Usability Experts	M: Reqs. Doc.	D: Features	D: User Story	D: User Story	M: User Reqs.	N	D: User Story	N	M: Reqs. Doc.	N	D: User Story

(continued)

Table 6 (continued)

Criterion	Targeted methodologies													
	MockupDD	RAMBUS	USABAGILE_Web	WebHelix	S-FDD	XWeb-Process	XP	UML-AW	Crystal Orange Web	S-Scrum	Scrum-CMMI	AWDWF	AWE	Scrum-MDE
Requirements specification	Mockups	Product Backlog	Prototype	Reqs. Doc.	Features	User Story	User Story	Story-board	–	Product Backlog	Product Backlog	Prototype	–	User Story
Requirements-based process	Y	Y	Y	Y	Y	Y	Y	Y	–	Y	Y	Y	Y	Y
Requirements prioritization	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Constraints/assumptions	1	1	3	1	1	2	2	1	1	2	3	3	2	1

Table 7 Results of evaluation based on agility characteristics

Criterion	Targeted methodologies													
	MockupDD	RAMBUS	USABAGILE_Web	WebHelix	S-FDD	XWebProcess	XP	UML-AW	Crystal Orange Web	S-Scrum	Scrum-CMMI	AWDWF	AWE	Scrum-MDE
Early and continuous delivery	3	3	3	3	2	3	3	2	3	3	3	3	3	3
Active user involvement	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Continuous customer feedback	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Self-organizing teams	2	1	2	1	3	3	3	3	1	3	3	1	1	3
Face-to-face conversation	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Velocity monitoring and control	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
Team behavior/efficiency	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Task assignment method	Team	–	–	–	Manager	Team	Team	Team	–	Team	Team	–	–	Team
Continuous integration	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Modeling coverage	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y
Standards	N	Y	N	N	N	Y	Y	N	N	N	Y	N	N	N
Iterative-Incremental process	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Agile techniques	N	N	N	N	N	Y	Y	N	N	Y	Y	N	N	N
Flexibility	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Rapid production of artifacts	2	3	2	2	3	3	3	2	3	3	3	3	3	3
Leanness	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Learning	2	3	3	2	3	3	3	3	3	3	3	3	3	3
Feedback by process	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Table 8 Results of evaluation based on key features of web-based systems

Criterion	Targeted methodologies													
	Mockup-DD	RAMBUS	USABAGILE_Web	WebHelix	S-FDD	XWeb-Process	XP	UML-AW	Crystal Orange Web	S-Scrum	Scrum-CMMI	AWDWF	AWE	Scrum-MDE
New or extended methodology	1	1	2	1	2	2	1	2	1	2	2	2	1	2
Specification of technical web characteristics	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
Architectural web design	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Early UI design	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Web-based security	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Rapid web development	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Web usability	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Addressed level of web criticality	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Web reliability	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Web flexibility	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Attention to web design aspects	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 4	1 2 3 4	1 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4	1 2 3 4
Tunability of development speed	2	3	2	2	2	3	3	2	3	2	2	2	2	2
Specification of web-related products and roles	2	3	1	2	3	3	3	3	3	3	3	1	3	3

5 Conclusions and Future Work

By evaluating the targeted methodologies, their individual strengths and weaknesses are highlighted. However, apart from these evaluations, some general observations can also be made: it can be observed that some of the targeted methodologies pay special attention to web security issues, a feature that is increasingly considered as essential in modern web systems; it can also be observed that Scrum variants seem to fully cover the different web development contexts that are commonly encountered.

As future work, we intend to propose a comprehensive agile web development methodology that addresses the weaknesses of existing methodologies while making use of their strengths. Another strand of research can focus on using the evaluation results for extending existing methodologies so that their shortcomings are properly addressed.

References

1. Farahani, F.F., Ramsin, R.: Methodologies for agile product line engineering: a survey and evaluation. In: Proceedings of the International Conference on Intelligent Software Methodologies, Tools and Techniques (SOMET'14), pp. 545–564 (2014)
2. Babanezhad, R., Bibalan, Y.M., Ramsin, R.: Process patterns for web engineering. In: Proceedings of the Computer Software and Applications Conference (COMPSAC'10), pp. 477–486 (2010)
3. Kaur, S., Singh, H.: Quality metrics for agile web engineering based on GQM approach. VSRD-IJCSIT **2**(6), 454–461 (2012)
4. Rivero, J.M., et al.: Mockup-driven development: providing agile support for model-driven web engineering. Inf. Softw. Technol. **56**(6), 670–687 (2014)
5. Pereira, V., Francisco, A.: Introducing a new agile development for web applications using a groupware as example. Commun. Comput. Inf. Sci. **165**, 144–160 (2011)
6. Benigni, G., Gervasi, O., Passeri, F.L., Kim, T.: USABAGILE_Web: a web agile usability approach for web site design. Lect. Notes Comput. Sci. (LNCS) **6017**, 422–431 (2010)
7. Whitson, G.: WebHelix: another web engineering process. J. Comput. Sci. Coll. **21**(5), 21–27 (2006)
8. Subramanian, N., Whitson, G.: Augmented WebHelix: a practical process for web engineering. In: Software Engineering for Modern Web Applications: Methodologies and Technologies, pp. 25–27. IGI Global (2008)
9. Ge, X., et al.: Agile development of secure web applications. In: Proceedings of the International Conference on Web Engineering (ICWE'06), pp. 305–312 (2006)
10. Sampaio, A., Vasconcelos, A., Sampaio, P.R.F.: Design and empirical evaluation of an agile web engineering process. In: Proceedings of the Brazilian Symposium on Software Engineering (SBES'04), pp. 194–209 (2004)
11. Maurer, F., Martel, S.: Extreme programming: rapid development for web-based applications. Internet Comput. **6**(1), 86–91 (2002)
12. Ambler, S.W.: AM Throughout the XP Lifecycle. http://www.agilemodeling.com/essays/agileModelingXPLifecycle.htm (2002)

13. Lee, W., et al.: Agile development of web application by supporting process execution and extended UML model. In: Proceedings of the Asia-Pacific Software Engineering Conference (APSEC'05), pp. 93–200 (2005)
14. Cockburn, A.: Agile Software Development: The Cooperative Game. Addison-Wesley (2002)
15. Mougouei, D., Fazlida, N., Sani, M., Almasi, M.M.: S-Scrum: a secure methodology for agile development of web services. WCSIT J. **3**(1), 15–19 (2013)
16. Salinas, C.J.T., Escalona, M.J., Mejías, M.: A scrum-based approach to CMMI maturity level 2 in web development environments. In: Proceedings of the International Conference on Information Integration and Web-based Applications and Services (IIWAS'12), pp. 282–285 (2012)
17. Hu, R., Wang, Z., Hu, J., Xu, J., Xie, J.: Agile web development with web framework. In: Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08), pp. 1–4 (2008)
18. McDonald, A., Welland, R.: Agile Web Engineering (AWE) process: multidisciplinary stakeholders and team communication. In: Proceedings of the International Conference on Web Engineering (ICWE'03), pp. 515–518 (2003)
19. Basso, F.P., Pillat, R.M., Roos-Frantz, F., Frantz, R.Z.: Study on combining model-driven engineering and Scrum to produce web information systems. In: Proceedings of the International Conference on Enterprise Information Systems (ICEIS'14), pp. 137–144 (2014)
20. Hesari, S., Mashayekhi, H., Ramsin, R.: Towards a general framework for evaluating software development methodologies. In: Proceedings of the Computer Software and Applications Conference (COMPSAC'10), pp. 208–217 (2010)
21. Karam, G.M., Casselman, R.S.: A cataloging framework for software development methods. Computer **26**(2), 34–44 (1993)
22. Kitchenham, B., Linkman, S., Law, D.: DESMET: a methodology for evaluating software engineering methods and tools. Comput. Control Eng. J. **8**(3), 120–126 (1997)