# A Framework for Performance and Collaborative Trade-off Analysis in Software Products

Henok Laike
HiLCoE, Software Engineering Programme, Ethiopia
henymen12@yahoo.com

Mesfin Kifle
HiLCoE, Ethiopia
Department of Computer Science, Addis Ababa University, Ethiopia
kiflemestir95@gmail.com

**Abstract**

With the increase in size and complexity, the need for software performance and balanced trade-offs are becoming very intensive. Furthermore, software products with poor performance or an unbalanced trade-offs have a diverse effect on businesses, stakeholder relationships, budget, schedule, etc. This paper proposes a three layered framework for performance and collaborative trade-off analysis aimed at primarily incorporating performance and trade-off considerations at every step of the development lifecycle that can be adopted to different process models, lifecycles, application domains and development methodologies. Additionally, a selection/decision support model is proposed to assist the framework in order to conduct a collaborative trade-off analysis and evaluation. The design of the framework and selection model follows a layered approach derived from a preliminary model and generic process outlines with entities that are integrated. The feasibility and effectiveness of the framework is simulated and evaluated by focusing on the core part of the framework. The findings of the implementation were concrete which assures the ability of the approach to address issues related with performance and trade-offs (PTOs) of software products.

*Keywords*: Performance; Trade-offs; Quality Attributes; Selection/Decision Model; Development Life Cycle

## 1. Introduction

Computer systems are used in many critical applications where a failure in meeting not only functional requirements but also non-functional requirements can have serious consequences including loss of lives or property [1, 2]. A successful software product can be put as one that efficiently balances functional as well as quality attributes so as the intended users of the system will find it useful without any demerit [3, 4]. The issues of software PTOs are very critical that the areas have brought a significant loss in revenue of the software industry in the developed countries. For instance, in Ethiopia where the software industry is at its crawling stage and where several complex mega projects are underway as part of the growth and transformation plan, the need for a reliable software product even in terms of performance and trade-offs is unquestionable since such huge projects highly rely on software. Moreover, due to the country's financial limitation, the software products must be produced within the allocated time frame, budget and most of all meeting the desired functionalities [5]. Poor performance problems in software products in the US cost the industry millions of dollars annually in lost revenue, decreased productivity, increased development and hardware costs, and damaged customer relations [6].

PTO problems can occur nearly on every software product, either on a newly-developed system or a legacy application and the issue highly affects project budget, productivity and customer relations [6]. Although there are various methods for analyzing PTOs, the analysis has typically been performed in isolation [7]. Thus, ad-hoc techniques are employed which might result in negative consequences [8]. Unfortunately, tuning such problems require quite a lot amount of resources since the cost to fix an error when going along the development stages increments dramatically [9].

Achievement of performance and balanced tradeoffs depends not only on the overall software architecture but also on other developmental stages such as code-level practices which includes language choice, detailed design, algorithms, data structures, testing, and so forth [7]. Thus, in order to produce efficient software, especially in terms of PTOs, analysis of PTOs has to be incorporated in all the software development stages such as specification, development, validation and evolution. Besides, to assess to what extent the quality attributes of a software product have been met, software architecture evaluation needs to be conducted at various phases of the software development life cycle [10, 11, 12, 13]. On the other hand, trade-offs are made if a system is to be made. However, what is less understood is the possibility for making an informed and optimal trade-offs during a system development process [7].

This paper will enable to explicitly look into selected tools and methods used to analyze performance and trade-offs in development stages, process models, methodologies and application domains and it will also enable to quantitatively analyze current trends of sample software development firms towards the issues. It presents a framework that incorporates performance and collaborative trade-offs analysis through the use of a selection/decision model that will be integrated into software development stages of requirements, implementation, validation and deployment to enable software products be dependable in the areas of PTOs. To investigate the research problem deeply, a mixed method design, which is a procedure for collecting, analyzing and mixing both quantitative and qualitative data at some stage of the research process within a single study is used [14, 15].

## 2. Related Work

The research work in [16] presents a framework for SPE of client/server systems which are part of a real SPE study carried out for downsizing a mainframe based system - Recruitment & Training System (RTS) to a Client/Server (C/S) environment which is based on a customized language – Clisspe which is based on Queuing Network Model (QNM) and Software Performance Engineering (SPE). The research considers the integration of performance prediction and assessment into all phases of the software development life cycle (SDLC) unlike the common development approach and uses a performance model which is a set of processes to be integrated at each level of SDLC – Requirement Analysis, System Design, Program Design, Program Coding and System Testing.

The research in [17] aims to create a SPE Center of Excellence (COE) by focusing on SDLC stages to be integrated with SPE phases. The COE has a responsibility to ensure the achievement of performance engineering effectively in each phase of SDLC. Some tasks of the COE in each stage include capturing and prioritizing performance requirements using a template, implementing design patterns, review of application architecture and design alternatives, providing performance testing framework and reusable library of scripts and developing capacity planning models, technology and hardware, and maintaining performance log analyzers.

The study in [18] proposes to integrate performance analysis in the early phases of Model Driven Development (MDD) for Software Product Lines (SPL) which is illustrated by an e-commerce case study. It involves two phases of model transformations by adding performance annotations to a Unified Modeling Language (UML) model. A first model transformation realized in the Atlas Transformation Language (ATL) derives the UML model for specific product with concrete MARTE performance annotations from the SPL model. A second transformation generates a Layered Queuing Network performance model for the given product by applying an existing transformation approach named PUMA (Performance by Unified Model Analysis).

The research work in [19] presents a collaborative architectural design process between stakeholders to make it global and distributed by providing Architectural Knowledge (AK) to make design

decisions by mainly focusing on high level design stages. The architecting process from an AK perspective has fundamental components such as Requirement Engineering (RE), scope problem space, propose solution, evaluate solution & choose one and evaluate architecture & modify the architecture description. Furthermore, in order to support the collaborative architecting process, the research implemented a Knowledge Architect (KA) which is a tool suite for creating, using, translating, sharing, and managing AK.

The work in [20] presents a tool set called ATAM Collaborative Environment (ACE) which is Web-based application that supports the entire ATAM process where the current prototype focuses on the brainstorming and voting process. The primary purpose is for advancing the traditional ATAM and to make the analysis and evaluation process collaborative rather than focusing on the analysis process and outcomes.

The research work in [21] presents a framework to unify concepts of software engineering process derived from the IEEE standards and to create a generic development life cycles. It does not consider the inclusion of performance and trade-off analysis into the development stages but the generic stages are essential components of this research to be aligned with the proposed framework and selection/decision model.

In general, the research works reviewed have provided concrete information to base the fundamental concepts of this research work. To mention some, the works have shown: approaches to adopt performance considerations as a model, a generic development stage as part of a software engineering model and framework, approaches to integrate performance and trade-offs in development stages and collaborative ways to perform software architecture designs. As a shortcoming, most of the related works did not focus on performance and trade-off analysis and evaluation. There were no selection methodologies employed for collaborative trade-off analysis. The findings were not supported by an empirical study and prototype implementation and the works were limited in focusing only on specific application domains and architectures and organization based systems. In contrast, this paper undergoes a systematic approach to address the shortcomings that were identified and also used the relevant approaches, concepts, processes, models and frameworks.

## 3. The Proposed Solution

### *3.1 Framework for Performance and Trade-off Analysis*

Initially, a preliminary software development process model is organized to serve as a roadmap of the overall process of design and implementation of the framework and model comprised of five interconnected components: Platform, Process Model, Methodology, Application Domain and Life Cycles. Though the interaction and interdependence among the components differ for various software products, a typical software development process will incorporate the function of either all or some of these components.

Since the primary purpose of this research is to integrate PTO analysis into the software development stages, a matrix based analysis is conducted to depict the resemblance among different process models' life cycles. The matrix shows how some of the common process models incorporate these fundamental stages and describes how other development stages are related [17, 22, 23, 24, 25]. Here, the difference among each of the process models in terms of their context, implementation and iterations are also noted. Hence, a generic development life cycle adopted from [21] is chosen for the implementation of the framework and model having stages of Requirement, Design, Construction, Verification and Deployment.

The design of the framework follows a layered approach which simplifies the overall design complexity by allowing each component to be decomposed into manageable entities and enables the layers to be integrated easily. The framework is composed of three layers which are transformed into a structured set of goals to be achieved in the interpretation and application of the framework. Figure 1 shows the pre-analysis framework.

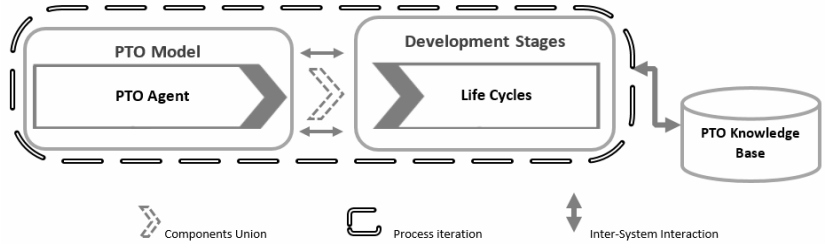


*Figure 1:* Pre-analysis Framework of the PTO Analysis Framework

The PTO Model represents a major component of the framework which is the PTO agent that contains all the vital attributes of PTO requirements to be directly plugged into the development stages. It basically contains needs or demands that should be met at each level of the development process. There are five PTO agents for each stage of development life cycle as shown in Figure 2.

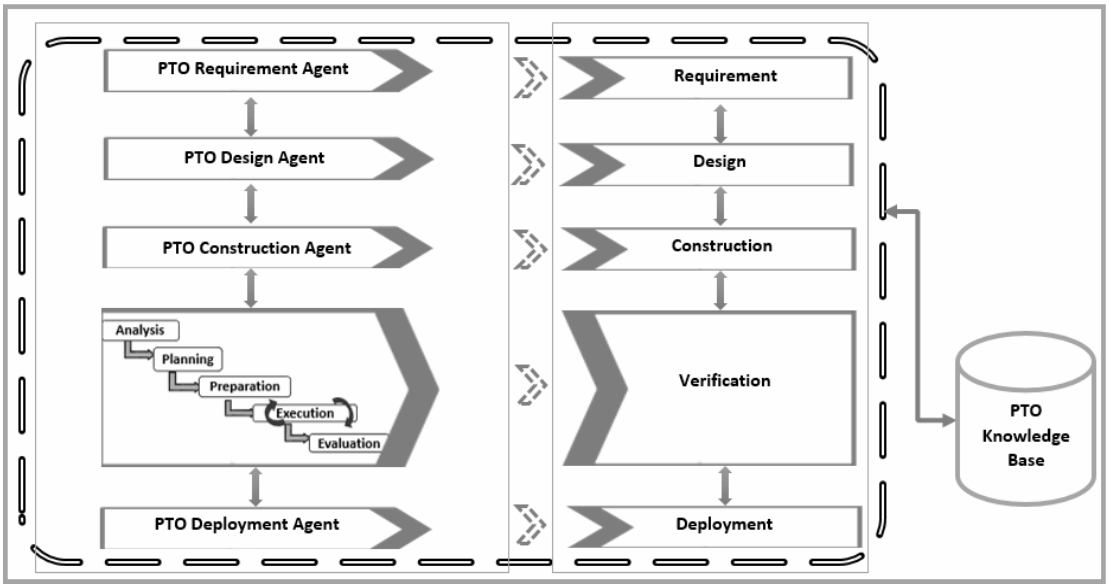


*Figure 2:* The PTO Analysis Framework

Each of the PTO agents has a distinct responsibility to be embedded onto the development stages. The roles of the PTO agents are shown in Table 1. Out of these agents, the PTO Verification Agent has another set of sub stages that are shown in Table 2.

*Table 1:* Summarized Roles of PTO Agents

| *PTO Requirement Agent* | *PTO Design Agent* | *PTO Construction Agent* | *PTO Verification Agent* | *PTO Deployment Agent* |
|---|---|---|---|---|
| Gathers PTO requirements, Aligning PTO requirements with business requirements, Requirement Verification (scrutinizing against actual system needs & application circumstances), Setting performance | Integration of PTO considerations, Implementation of the selection/decision model, Conducting Architecture evaluation, Incorporating performance tactics, Verification: | Verification of requirements from previous agents, PTO based code-level adjustments (classes, modules, libraries, variables, testing cases), Verification and validation, and | Overall assessment of PTO requirements, Involves analysis, planning &preparation, execution and evaluation, Verification and validation, PTO | Verification of the overall PTO, Assessment plans, PTO checkups and confirmation, and Documentation |

| | | | | |
|---|---|---|---|---|
| targets/metrics (response time, throughput, memory, bandwidth etc.), Prioritization, Communication and Documentation | validating/refining design and requirement phase needs, Arch. Tactic planning, prediction & assessment and Documentation | Documentation | tuning process, Communication, and Documentation | |

*Table 2:* The PTO Verification Agent

| *Analysis* | *Planning & Preparation* | *Execution* | *Evaluation* |
|---|---|---|---|
| Devising a strategy and proof of concepts on the tools that are suitable to conduct the test and PTO related requirements are used to launch the test process. | Creating a detailed implementation plan and test schedules, and planning of performance metrics | Implementation of the test process and if Unsuccessful recall and iterate | Result report and recommendations, and Documentation |

A generic model derived from IEEE software engineering standards is adopted for the design and implementation of the framework for PTO analysis [21]. Even though this particular model is chosen due to its generic nature, the framework can be adopted for different kinds of process models and development stages. For instance, this model can be adopted for Agile development methodology easily by incorporating the components of the model into every stage of the agile development stages.

The PTO knowledge base represents a repository in which every encounter in the PTO analysis during the development process is recorded for reference. It is also crucial in the iteration process of the development stages to validate the overall development process.

### *3.2 Selection/Decision Support Model for Collaborative Trade-off Analysis and Evaluation*

In general, an architectural evaluation process involves the collaboration of major process stages of Selection & Decision, Analysis & Evaluation, Output and Post Analysis Execution. The major area of focus for the collaborative analysis and evaluation of trade-offs is the Selection & Decision stage. Overall, the trade-off analysis and evaluation is conducted as part of the PTO Design agent. The Selection/Decision Model contains three major phases of Planning, Analysis and Implementation as can be seen in Figure 3.

## 4. Discussion

In order to produce a precise solution to the problem of the research area, a case study on the current trends of selected software development companies in Addis Ababa, Ethiopia, is conducted as part of the quantitative aspect of the research and an experimental testis conducted to test the PTO Framework and the Selection/Decision Model. The findings of the empirical study have identified gaps towards the issues in the current operational settings of companies and professionals. Some of the critical gaps that were identified in the findings were, although 78% of the respondents have an experience on the field for more than 3 years, 80% consider quality attributes and trade-offs during a development process and 90% have experienced products with problems of performance and trade-offs with various side effects, 75% of them do not have experience on employing standard procedures to incorporate such issues in the development processes and 82% of them have no experience on using performance and trade-off analysis and evaluation tools.

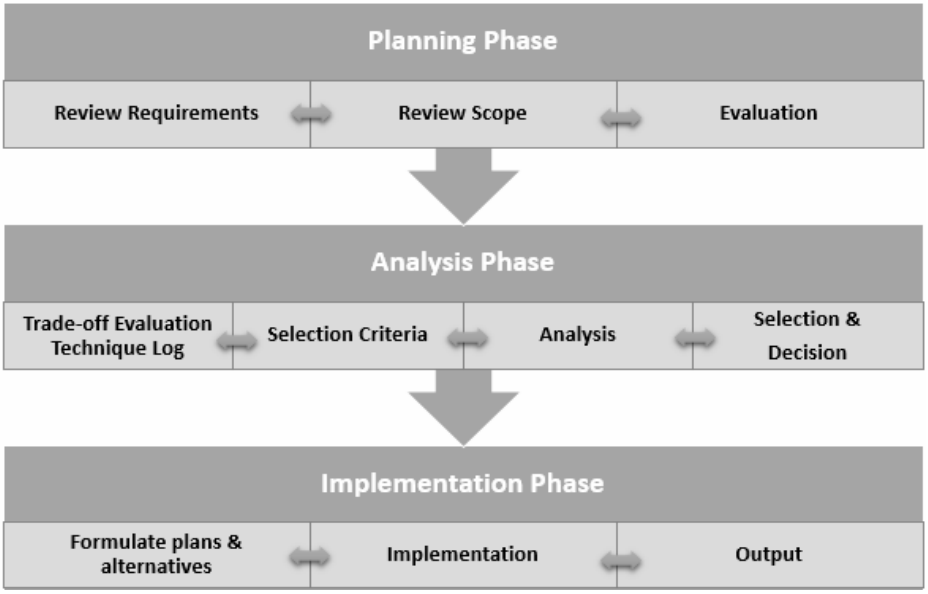


*Figure 3:* Selection Model for the Collaborative Analysis and Evaluation of Trade-Offs

The implementation of the framework and Selection/Decision Model focused on the core parts of the PTO Analysis Framework, i.e., the PTO Design Agent and the PTO Verification Agent. Hence, a standalone desktop application (Automated Child Sponsorship Management System -ACSMS) developed with C# and SQL Server, and a web based system (Educational Management Information System – EMIS)powered by PHP 5.0 and MySQL was used for the experimentation of the Selection/Decision Model and the PTO Framework respectively using the PTO Design Agent and the PTO Verification Agent. In addition, a scope has been set for the overall implementation process and both systems are implemented in accordance with the scopes. Various kinds of testing platforms and tools have been explored to determine the best testing solution for a precise scrutiny based on the criteria of process and output reliability, universal recognition among professionals and simplicity of the overall process.

The implementation of the PTO Verification Agent involves the major stages of Analysis – selection of a testing tool, Planning & Preparation – creating a performance metrics and test scenarios, Execution – implementation of the testing process with iteration and Evaluation – assessment and generating results of the overall process. The overall testing process also involves the documentation of all the processes on the Knowledgebase for future reference. Based on this, two testing platforms are chosen, i.e., Microsoft Visual Studio Testing Tool and HP Load Runner. Some of the performance metrics set include: ’50 concurrent users’, ‘Response time <30 sec’, ‘Batch completion time from 20 to 25 minutes on worst case scenario’, and ‘CPU utilization <75%, memory <1GB, Disk <500GB and Bandwidth’.

Performance testing using the two testing tools involves different set of processes to complete the PTO Verification Agent stages such as creating a test project, creating and recording a Vuser script, recording the test based on the scenarios chosen and output of the test project. The testing output basically contains all the performance metric results for the specified system. Another important stage of the PTO Verification Agent is the evaluation stage that determines the state of the system whether it has met the targeted performance metrics or not. During this stage the performance analysis outputs have to be compared with each of the performance metric objectives for the systems. This will determine whether there needs to be a system tuning or not. For instance, during both testing processes, the evaluation

stage has determined that the expected 'Memory Consumption' for the systems has exceeded the targeted amount. In this case, a system tuning has to be conducted in order to meet the targeted objective by going through the development stages and refining requirements, refining design related targets and finally adjusting the construction of the system to incorporate the issue. Then a re-testing procedure will determine whether the issue is resolved or not.

The implementation of the PTO Design Agent also involves the incorporation of the Selection/Decision Model that includes the major stages of Planning– involves refining architectural requirements and scopes of the ACSMS, Analysis – involves the creation of a selection criteria and a log of trade-off analysis tools then analysis of these tools will be conducted based on the selection criteria which results in filtered trade-off analysis tools, and Selection and Decision–involves the selection and decision on single or multiple trade-off analysis tools together with relevant stakeholders.

## 5. Conclusion and Future work

It is believed that the framework and model together with its underlying implementation and detailed processes is a realistic approach to address the issues of PTOs in software products. Hence, the most appropriate and tangible approach for addressing issues of PTOs is the incorporation of the issue into every stage of the development life cycle. The effectiveness of the framework and model had been practically experimented on standalone desktop application and a web based system which had proven the ability of the framework and model to achieve a product that is PTO oriented. Even though additional resources might be required for the overall implementation of the framework and model, the ultimate outcome is by far better than fixing problems due to its incomparable advantages such as cost and time. On the other hand, the selection of performance testing tools and trade-off analysis and evaluation tools also matters on the ultimate results of the process and on the decisions that will be made subsequently. Implementation of the framework and model will play a significant role in the long-term reliability of a typical software product. Additionally, during a typical software development process an intensive focus has to be given for PTO analysis since late mitigations will not guarantee the success of the software meeting PTO requirements. The standardization and adaptation of such frameworks and models by governmental bodies and institutions is also highly advised.

The framework and model proposed in this research will be used as a basis for a number of research and development works. Some of the areas can be implementing the framework and model on a prototype system from scratch to attain its full capability, i.e., including the requirement, construction, and deployment phases. Afterwards implementing the 'implementation' part of the selection/decision support model, studying the framework and model in different development methodologies and process models other than the generic ones such as Agile, incremental, spiral, etc. is required. Additionally, the current framework can be expanded to include development stages such as software evolution, human resource, budget and project schedule implications of the framework and model, formulation of a testing platform selection/decision support model for efficient test outputs that will enable informed decisions to be made, conducting an in-depth analysis of trade-off analysis and evaluation techniques such as AHP & QNM on how to adopt a prediction model of performance, trade-off and sensitivity points of software architectures and integrating it with this framework and studying how to adopt such frameworks and models in policies and governmental regulations for further strengthening the software development industry in Ethiopia.

## References

[1] Mario Barbacci, Mark H. Klein, Thomas A. Longstaff, and Charles B. Weinstock, "Quality Attributes", 1995.

[2] Karen Henricksen and Jadwiga Indulska, “A Software Engineering Framework for Context-Aware Pervasive Computing,” 2004.

[3] J. S. Glider, C. F. Fuente, and W. J. Scales, "The Software Architecture of a SAN Storage Control System,” IBM Systems Journal, 2003.

[4] Capers Jones and Olivier Bonsignour, “The Economics of Software Quality,” 2011.

[5] Why Ethiopia’s Software Industry Falters, 2013, retrieved from http://addisfortune.net/columns/why-ethiopias-software-industry-falters/, Last accessed on June 14, 2015.

[6] Lloyd G. Williams and Connie U. Smith, “Five Steps to Solving Software Performance Problems,” 2002.

[7] Rick Kazman, Mark Klein, Mario Barbacci, Tom Longstaff, Howard Lipson, and Jeromy Carriere, “The Architecture Tradeoff Analysis Method,” 1998.

[8] Bridget Spitznagel and David Garlan, “Architecture-Based Performance Analysis,” 2005.

[9] Paul Clements, Rick Kazman and Mark Klein, “Evaluating a Software Architecture,” 2001, retrieved from http://www.csee.wvu.edu, Last accessed on May 28, 2015.

[10] Liming Zhu, Aybüke Aurum, Ian Gorton and Ross Jeffery, “Tradeoff and Sensitivity Analysis in Software Architecture Evaluation Using Analytic Hierarchy Process,” 2005.

[11] Mario Barbacci, Paul Clements, Anthony Lattanze, Linda Northrop and William Wood, “Using the Architecture Tradeoff Analysis Method (ATAM) to Evaluate the Software Architecture for a Product Line of Avionics Systems: A Case Study,” 2003.

[12] Michel R. Dagenais, Karim Yaghmour, Charles Levert and Makan Pourzandi, “Software Performance Analysis,” 2005.

[13] Jorgen Boegh, Stefano Depanfilis, Barbara Kitchenham, and Alberto Pasquini, “A Method for Software Quality Planning, Control and Evaluation,” 1999.

[14] Creswell, J. W., “Research Design: Qualitative, Quantitative, and Mixed Methods Approaches,” 2003.

[15] Maura Borrego, Elliot P. Douglas, and Catherine T. Amelink, “Quantitative, Qualitative and Mixed Research Methods in Engineering Education,” 2009.

[16] Daniel A. Menasce, “A Framework for Software Performance Engineering of Client/Server Systems,” 1997.

[17] Paul Ringmacher and Poorna Bhimavarapu, “Integrating Performance Engineering with Software Development Life Cycle,” 2013.

[18] Rasha Tawhid and Dorina Petriu, “Integrating Performance Analysis in the Model Driven Development of Software Product Lines,” 2009.

[19] Peng Liang, Anton Jansen, and Paris Avgeriou, “Collaborative Software Architecting through Knowledge Sharing,” 2010.

[20] Piyush Maheshwari and Albert Teoh, “Supporting ATAM with a Collaborative Web-based Software Architecture Evaluation Tool,” 2004.

[21] Michel Chaudron, Jan Friso Groote, Kees van Hee, Kees Hemerik, Lou Somers, and Tom Verhoeff, “Software Engineering Reference Framework,” 2002.

[22] Ian Sommerville, “Software Engineering,” 9th Edition, 2011.

[23] Mohan V. Tatikonda and Maia Lorence, “Towards Effective Software Development: A Conceptual Framework of Software Project Types, Development Processes, and Functional Outcomes,” 2002.

[24] Verifysoft Technology, Software Quality, 2011, retrieved from http://www.verifysoft.com/fr_software_quality.html, Last accessed on April 6, 2015.

[25] Wikipedia, Web Application Frameworks, retrieved from http://en.m.wikipedia.org/wiki/Web_application_framework, Last accessed on May 04, 2015.