Journal of Technology Management & Innovation

Special Issue on Selected Papers from ALTEC 2011.
Selected February 11, 2013

J. Technol. Manag. Innov. 2013, Volume 8, Special Issue ALTEC.

Test-Driven Development as an Innovation Value Chain

Ana Paula Ress[1], Renato de Oliveira Moraes[2], Mário Sérgio Salerno[3]

Abstract

For all companies that consider their Information Technology Department to be of strategic value, it is important to incorporate an innovation value chain into their software development lifecycles to enhance their teams' performance. One model is TDD (Test-Driven Development), which is a process that detects failures and improves the productivity and quality of the team's work. Data were collected from a Financial Company with 3,500 employees to demonstrate that software projects that require more than 4,000 hours of development benefit from TDD if a clear knowledge conversion step occurs between the client and the developers.

Keywords: tdd; test-driven development; innovation value chain; software quality.

Universidade de São Paulo (USP). Departamento de Engenharia de Produção, Avenida Prof. Luciano Gualberto, Travessa 3 N° 380.
E-mail: [1]ana.ress@usp.br, [2]remo@usp.br, [3]salerno@usp.br

Introduction

Global competition propels companies to identify a competitive advantage that will focus the market on their products. According to Porter (1993), competitiveness depends on a company's capacity to improve and innovate. Companies obtain a competitive edge through their innovation initiatives, which identify new technologies and methods to execute processes and projects.

Companies must make a strategic decision to develop the capabilities to innovate and improve. As Porter mentions, "the choices concerning point of view determines not only the activities the company will develop and how these activities will be configured but also how these activities will relate with each other" (PORTER, 1996).

According to Parasuraman and Colby (2002), companies that desire product and service competitiveness need to adequately manage the company-client (external marketing), company-employee (inside marketing) and employee-client (interactive marketing) communication channels. This approach is interesting because it segregates the moments and the goals of client-company communication. Kotler (2002) states that while it is important for a company to define and understand its target market, the company also requires a mechanism for adequately capturing the needs of its clients. Understanding all of the software requirements is important for having a successful end result that can actually be launched in the market. Regardless of how well the software has been planned or coded, it will not meet the end users' expectations if it is poorly analyzed and the requirements implemented incorrectly. The developer will thus face difficult inquiries for not having met the client's goal. Leite (2001) states that well collected requirements are key inputs that must be inserted into building a system such that the actual needs of the users are considered.

It is a challenge to change the software development process. but any change in a company's culture is rather difficult. According to Hurley and Hult (1998), the capacity to innovate within a company can be understood as the ability of the organization to adhere to or execute new ideas, processes and products with success.

To Van de Ven et al. (1999), innovation is a process of developing and implementing new processes and projects or generating new ideas, such as a new technology, product, organizational process or new arrangements that reinforce Rogers's (1995) concept that innovation is a "perception" of the new.

Jonash and Sommerlatte (2001, p. 2) concluded that innovation is an organizationally defined strategy that is not limited to R&D, as occurs in most traditional organizations, but they affirm that innovation is the propelling force that drives the company.

According to Powell and Dentmicallef (1997), technology alone is not enough to maintain a company's competitive edge. Information Technology productivity must be continuous and integrated into the companies' processes. In addition, this software development model proposes improving productivity and quality, which motivates the study and practice of TDD.

The goal of this article is to discuss a software development model called TDD. This model offers a new way to organize the activities of software development to promote a better understanding of the clients' system requirements clients. The TDD model will be explored using the innovation concepts approach.

The structure of this article is based on a strong theoretical foundation, aiming to demonstrate the main concepts in the following sections: Agile Methodology Management and Test-Driven Development (TDD), the Product Development Model, and the Innovation Value Chain. The methodology used, the case study premises and the results are then presented. The article ends with Conclusion and References. Academic Board

Agile Methodology Management

Innovation and complex projects are usually present in dynamic organizations. In this context, there is difficulty in forecasting the future, which is uncertain and challenging, due to the limitations of traditional project management techniques that perform tests at the end of a project (SUIKKI; TROMSTEEDT; HAAPASALO, 2006). According to Senhar (2001), these limitations are present in project planning and monitoring.

By considering that traditional methodologies, which forecast the total size of the initial phases of the project covering all of the specification requirements were somewhat adaptive, is that in February 2001, a group of developers thus proposed an alternative methodology for software development entitled Agile Project Management (APM), document published on the internet (http://agilemanifesto.org/) titled "Agile Manifesto for Software Development".

Beck (1999) defines TDD as one of the practices that belongs to the XP core and should be implemented during the design and software coding phases. TDD is a practice that is focused on simplifying the Software Design Activities to improve the hand-in speed.

TDD is a design and development technique that focuses on building a system incrementally by keeping in mind that we are never too far away from a functional and deliverable baseline. The test-code-refactor cycle dictates the rhythm of development through a series of short and controlled steps in which refactor is the activity performed by the developer to increase the internal structure of the code without altering its functionality. The classic sequence is to first produce the design, implement it and then test the software for bugs is the opposite of the test-code-refactor methodology. According to Lewis (2004), quality cannot be reached after a product is completed. Therefore, the goal is to prevent quality failures or deficiencies in the first place using a quality assurance program.

TDD – Test-Driven Development

According to Jansen and Saiedian (2005), TDD appeared in a set with agile processes models and has its roots in the interactive, incremental and evolutionary processes models used before 1950. The authors add that NASA's project Mercury from 1950 applied TDD, which was not known until 2003, when Kent Beck unveiled this method. The authors still stresses the ability of TDD to improve software quality and that this method should be a part of Software Engineering coursework.

According to Pressman (1995), this classic approach forecasts system development prior to conducting analyses, the project and its implementation. These models follow the traditional execution of final tests for coding.

Janzen and Saiedian (2008) compared the traditional (i.e., tests conducted at the end of the project) and TDD (i.e., tests conducted at the beginning) approaches. The result obtained through their survey is that TDD has smaller modules with fewer methods and greater coverage on code (i.e., the number of coding lines that are executed in test cases). It is thus possible to assume that by using smaller modules, the authors reached or exceeded their expectations in terms of requirements, compared to the traditional approach. A direct result is less complexity in development, which results in easier maintenance.

Figure 1 compares the traditional software development (process A), which applies tests at the end of software development, and TDD (process B), which applies tests at the beginning.

According to Crispin (2006), by applying TDD, clients are satisfied with the quality obtained during development and the improved communication among the clients, business analysts and developers. This communication makes it easier to understand the requirements and scope of the product.

Innovation Concept

Product Development Model

According to Thier (2005), the development process for a product with many particularities must follow a process, i.e., a sequence of events, as is common in other ventures. The process of developing new products (PDP) requires management to focus on quality control during product development to minimize production errors and failures. It is appropriate to apply these concepts to the model of software development. Ulrich and Eppinger (2000) state that the organization must decide to build the product, and figure 2 shows their model, which contains 6 phases.

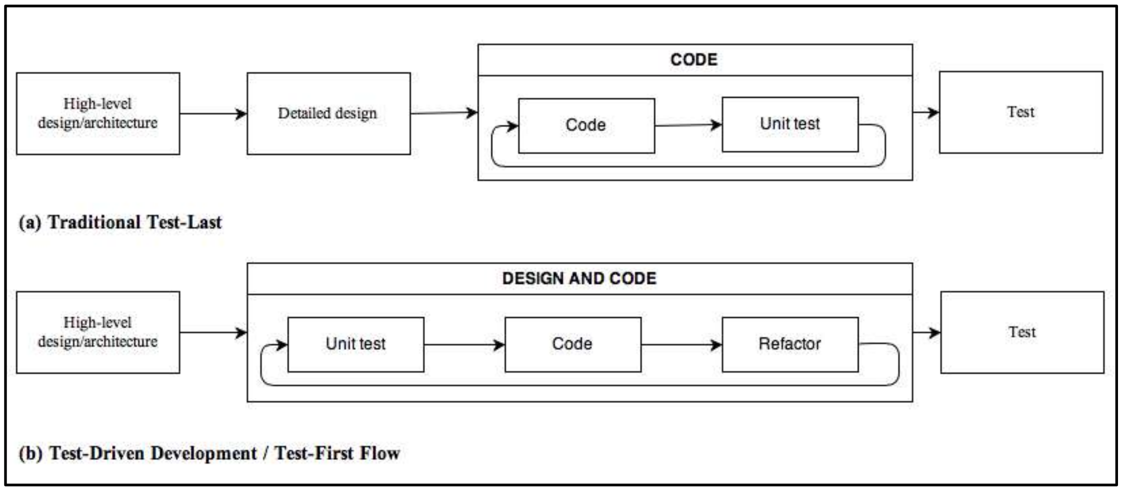

High-level design/architecture
Detailed design
CODE
Code
Unit test
Test
(a) Traditional Test-Last
High-level design/architecture
DESIGN AND CODE
Unit test
Code
Refactor
Test
(b) Test-Driven Development / Test-First Flow

Figure 1: Model of Janzen and Saiedian (2008) - Traditional Development vs. TDD

In this above proposed organization of work, the test phase comes before the production phase, the goal is to produce sufficient evidence that the product will satisfy the needs of the client. In software development, the application of TDD focuses on looking for the same target, e.g., when writing the test specifications before coding, the functional requirements are prioritized. Performing the steps in this order is a way to guarantee that the clients' requirements will be achieved and then apply the coding needed for its operation, promoting even a lean code.

Innovation Value Chain

The term "Value Chain" was originally adopted by Michael Porter in the 1980s. According to the author, "every company is a set of activities which are executed to design, produce, trade, deliver and support its product" and "all these activities can be represented using a Value Chain" (PORTER, 1993).

Each organization desires to improve the way they create and generate value. In the case of innovation, it is an important factor to evolve the competitiveness, productivity and quality of the product.

Afuah (1998) states that companies should seek to differentiate themselves through their attributes, skills and innovation processes, and states that one of the sources are the internal factors of the company's internal value chain. Similarly, Dosi et al (1982) indicates that innovation is the quest for new solutions, imitations and experimentation, which leads to new products or processes and new organizational structures. Ducker (2002) adds that for internal sources of innovation have improved production processes and changes resulting from the strategic positioning.

Hansen and Birkinshaw (2007) allege that upon observing innovation from point A to point B, we have an Innovation Value Chain composed of three phases: idea generation, idea development and the propagation of the developed concepts. Within these phases, there are several activities that compound the link of this chain. As observed in table 1, the TDD concept is used.

Phase 0 –Idea Generation: The client expresses their requirements, and a business analysis of the IT work required to conceive the product is performed. The result of this phase is the specification in textual form.

Phase 1 – Idea Conversion: The ideas created in phase 0 must be converted into the target product. In the traditional approach, the prototype is a resource that can be shared be-

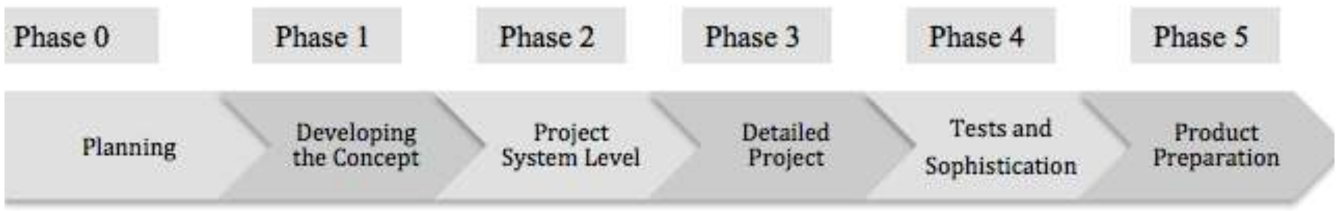
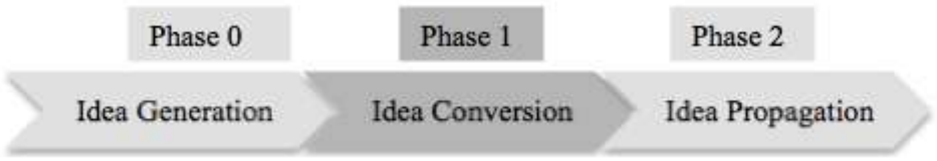


Figure 2: Modelo f Ulrich and Eppinger (2000) - Product and Development Process



Communica-tion	Client + Business Analyst	Client + Developer	Developer Team
TDD Activities	Requirements Document Technical Requirements	Tests	Code Refactor
Traditional Activities	Requirements Document Technical Requirements		Code Tests

Table 1: Adapted Model of Hansen and Birkinshaw (2007) - TDD vs. Traditional

tween client and developer, but it cannot be used as a framework for coding after it is a user interface. The developers must interpret the specifications to code the functional requirements. In the TDD case, the client will be able to collaborate with the developers in creating the test classes, which will be the groundwork for coding. In other words, the developer will code, and the client will be the information resource.

Phase 2 – Idea Propagation: In the traditional approach (test in the end), there is not a knowledge conversion document. Each developer must interpret the requirement document, thus making it difficult to understand the project scope. In the TDD case, as there is a knowledge conversion step before beginning product development by applying a test case, the developers can develop a common communication channel while having a common understanding of the project.

As mentioned by Hansen and Birkinshaw (2007), it is important that the links of the chain are equally strong and weak because the productivity of the chain is measured by its weakest link. Because the traditional approach does not contain a conversion phase, we presume that a weak link in the chain is addressed in the TDD case because it has a conversion phase.

Method

The development of this study was divided into two distinct parts. The first constitutes a biographical review to highlight the key components of TDD and to search for synonyms. The second part is composed of a case study that is used to investigate the assumptions of TDD in an actual case.

Biographical Review

To verify the publications referring to TDD, the following search was used in the Web of Science:

Search for the term Test-Driven Development

- Criteria: Key Word, Test-Driven Development; Type of Document, Article
- Search Result: 36 articles

In the analysis, three articles were excluded because they referred to related subjects in sociology and education. This left 33 articles to be analyzed.

Seventeen articles demonstrated a productivity gain in TDD compared to the traditional methodology of software development. The articles stress that productivity gain comes from the early detection of failures in the beginning phases of process development or product quantity [ERDOGMUS (2005), JANZEN; SAIEDIAN (2008), PITT-FRANCIS (2008), PITT-FRANCIS (2009), ENQUOBAHRIE (2007), SFETSOS (2006), NAGAPPAN (2008), JANZEN; SAIEDIAN (2005), ZHANG (2004), GERAS (2004), ANDREA (2007), JOHNSON (2007), TURNU (2006), RICCA (2009), CRISPIN (2006), KOU (2010), HUMMEL; ATKINSON (2007)]. Among them, we can examine the profit gained in the requirements analyses, as the developers can criticize or even promote discussions during the conversion of knowledge (before the coding) made through the test cases [PITT-FRANCIS (2008),PITT-FRANCIS (2009), RICCA (2009),KOU (2010)]. Some articles question the productivity gain mentioned by Madeyski (2006) and Madeyski and Szala (2007). These articles indicate that their studies do not result in a productivity gain and do not even improve the software quality. These results could be explained by the research of Muller and Hofer (2007) and Martin (2007), who express that the success of TDD depends on the team's seniority.

Janzen and Saiedian (2008) and Zhang (2004) show the importance of TDD as an extension of the recognition that TDD is a new practice in Software Engineering.

After its diffusion in 2003 by Kent Beck, works were published in 2007 that proposed improvements to the TDD technique, especially concerning the conversion phase, which suggested new techniques or improvements in the test cases [JURECZKO; MLYNARSKI (2010), NAGAPPAN (2010), CZIBULA (2008), VODDE; KOSKELA (2007)]. By this time, some works suggested that the TDD concepts should be applied to specific applications such as embedded development, databases or even spreadsheets [RUIZ; PRICE (2007), MCDAID; RUST (2009), HAMILL (2008), CHEN (2008), GRENNING (2007), AMBLER (2007), DOHMKE; GOLLEE (2007)].

Search for synonyms for Test-Driven Development

As mentioned by Janzen and Saiedian (2005), TDD has synonyms such as Test-First Programming, Test-First Design and Test-Driven Design

Test-First Programming

- Criteria: Key words, Test-First Programming; Type of document, Article
- Search result: 4 articles

Of these 4 articles, 2 use the term Test-Driven Development, 1 discusses the productivity of Erdogmus (2005) and 1 questions that the quality results are deemed inconclusive to affirm that there are benefits, as mentioned by Madeyski (2006). The other 2 articles either discuss applying TDD in larger projects, i.e., the study by Talby D (2006), or investigating improvements in productivity and quality after using TDD, i.e., the study by Huang L (2009).

Test-First Design

- Criteria: Key word, Test-First Design; type of document, Article; area, Computer Science
- Search Result: 4 articles

In this case, it was necessary to shorten the search in the field of computing; even with this shortening, it was necessary to exclude 1 article from the analysis, as it referred to a Physician school. Of the 3 articles analyzed, 2 are presented with the key word Test-Driven Development, 1 discusses the productivity of Johnson (2007), and another questions whether the improvements in quality are conclusive for affirming that there are profits, as Madeyski (2006) suggests. Only one article, i.e., that by Kay (2009), raises the question of tests in the usage approach.

Test-Driven Design

- Criteria: Key words, Test-Driven Design; Type of document, Article; Area, Computer Science or Operations Research & Management Science or Engineering or Automation & Control Systems
- Search Result: 14 articles

In this case, there was also a need to limit the subject areas. As a result, 2 articles were excluded from the analysis because they refer to the fields of explosives and education. Twelve articles were left for analyzes, all of which included the term Test-Driven Development. In closing, the term Test-Driven Development is used most frequently, and the term Test-Driven Design is identified as a synonym.

Case Study

Data were collected from projects that followed either the traditional approach or TDD. The focus of the investigation was the presence of and/or problems with the assumptions, compared with the results of the bibliographical review. Considering the value chain of innovation, it was expected that the conversion phase in the TDD approach would actually improve the productivity of development because there is cohesion among the links in the chain.

Company A

The Financial Company with 3,500 employees where the TI department occupies a strategic position in the company. The data collected are from 2010, and the organization has the following features:

- Projects are classified according to the development effort: large is above 4,000 hours; medium is between 2,000 and 3,999 hours; small is between 1,000 and 1,999 hours; and tasks that take under 1,000 hours are not considered projects.
- The closed projects are considered a success when the deviation does not exceed 5% compared to the total term (actual x estimated) and the total cost (actual x estimated)
- All projects, when they complete the development phase, are submitted to the quality area to validate their functional requirements. This area must create black box test cases, which will consider the system interface, not the source code. The goal is the functional validation, which runs through a minimum of 80% of the system, and the success rate must be equal to or greater than 90%. The project can go through this cycle several times until it reaches the required indicators.

The table below presents the data obtained following the traditional development (test in the end).

In 2010, the company did not successfully close one large project. The area of IT management investigated this phenomenon and concluded that these software failures were responsible for the rising costs and increased software development time. To minimize the software failure rate, it was decided to apply TDD practices to a large project.

The motivation of IT managers for choosing the TDD practices was the observation that the quality department created some ad hoc test cases to determine what was important for the system, as it was already a post-development activity. The selected project was estimated to have 5,300 hours of development. Initially, it was necessary to adjust the team's expectations, as TDD seemed less productive and the level of employee frustration increased due to the time invested in creating test cases. As the guidance provided was to cre-

			Medium Quality	
Classification	# of projects	Successful Projects	# of Cycles	Quality of the 1st Cycle
Large Project	12	0%	3	62%
Medium Project	38	40%	2	76%
Small Project	63	58%	2	84%

Table 2: Data of Company A - Traditional Development

ate one test class for each code class, it was necessary to have two quality cycles to reach the minimum 90% of the expected quality. The project thus had a 7% of deviation in time and cost. The lessons learned included the following:

- For the construction of the test classes is required rigor to certain input parameters that will be inserted. The simple fact of having a test class not guarantee the expected quality (above 90%) in the first round of tests.

- To create the test classes, it is necessary for the team to have certain experience in development, and the ad hoc developers had difficulties in applying the TDD concepts.

It was thus decided to choose a second project that also had approximately 5,800 hours of development time. Developers were instructed to focus on understanding the features, with the goal of writing good tests. For the main classes of tests, the user was asked to provide the expected entry and exit parameters. In the first quality cycle obtained 93% of the total expected quality, and the project total had a 4% deviation in time and cost and was thus considered a success. The lessons learned included the following:

- It is necessary to write specific test classes for performance improvement (i.e., including nonfunctional requirements, such as processing or response time).

- The development led to tests tend to have the technical documentation also guided the tests, and developers unfamiliar with this technique may present some difficulty in reading. It is recommended to pay special attention to this scenario.

Discussion

In the traditional development (tests in the end), we observe that a larger project incurs a larger gap between the project and the time x cost, according to the rules applied by the organization studied. It is safe to say that one of the reasons that this gap occurs is due to the lack of a knowledge conversion phase between the client and the developer. At the end of the development, it will be revealed whether the coding reflected the client's expectations.

When applying TDD techniques, we can observe that the second project's results were better, as the developers were more concerned with understanding the requirements so they could prepare better tests. The client also provided the expected entry and exit parameters. It is thus possible to conclude that there was knowledge conversion between the client and developer. Considering the principles of Hansen and Birkinshaw (2007), it is possible to expect a higher quality chain, as the weak link (knowledge conversion) was reinforced.

Considering the model of Ulrich and Eppinger (2000), in which the test phase occurs before production, the projects applying TDD were concerned with detecting failures before coding began. This model was valuable in software development, as more attention was paid to the elements to be implemented and less was paid attention to the logical structures.

It is important, however, to pay attention to the implementation of TDD by teams with limited experience in development, as having knowledge of test class constructions is necessary to profit from the technique.

In general, we can suppose that applying TDD enhances the productivity of the software development chain process and that, as the case study showed, the company should be attentive to the seniority of the development team and should maintain close communication with the client to build good test cases.

A limiting factor of this study is the lack of data from other companies that applied TDD, which prevents us from forming a broader conclusion. We would like to suggest that a future research project should focus on gathering such data.

References

AFUAH, A. Innovation management: strategies, implementation and profits. New York: Oxford University Press, 1998.

AGILE Alliance. Manifesto for agile software development. http://www.agilemanifesto.org/ [Accessed May 20, 2011]

AMBLER, S. W. Test-driven development of relational databases. Ieee Software, v. 24, n. 3, p. 37-+, 2007.

ANDREA, J. Envisioning the next generation of functional testing tools. Ieee Software, v. 24, n. 3, p. 58-+, 2007.

CHEN, W. K. et al. Integration of specification-based and CR-based approaches for GUI testing. Journal of Information Science and Engineering, v. 24, n. 5, p. 1293-1307, 2008.

CRISPIN, I. Driving software quality: How test-driven development impacts software quality. Ieee Software, v. 23, n. 6, p. 70-71, 2006.

CZIBULA, I. G. et al. Comdevalco Development tools for procedural paradigm. International Journal of Computers Communications & Control, v. 3, p. 243-247, 2008.

DOHMKE, T.; Gollee, H. Test-driven development of a PID controller. Ieee Software, v. 24, n. 3, p. 44-+, 2007.

DOSI, G. Technological paradigms and technological trajectories. Research Policy, v.11, p.147-162,1982.

DRUCKER, P.E. The discipline of innovation. Innovation. v.42, p.75-87, 1991.

ENQUOBAHRIE, A. et al. The image-guided surgery toolkit IGSTK: An open source C++ software toolkit. Journal of Digital Imaging, v. 20, p. 21-33, 2007.

ERDOGMUS, H. et al. On the effectiveness of the test-first approach to programming. Ieee Transactions on Software Engineering, v. 31, n. 3, p. 226-237, 2005.

GERAS, A. M. et al. A survey of software testing practices in Alberta. Canadian Journal of Electrical and Computer Engineering-Revue Canadienne De Genie Electrique Et Informatique, v. 29, n. 3, p. 183-191, 2004.

GRENNING, J. Applying test driven development to embedded software. Ieee Instrumentation & Measurement Magazine, v. 10, n. 6, p. 20-25, 2007.

HAMILL, P. Unit Testing Web Services. Dr Dobbs Journal, v. 33, n. 11, p. 53-+, 2008.

HANSEN M. ; Birkinshaw J., The Innovation Value Chain. Harvard Business Review, 2007.

HUMMEL, O.; Atkinson, C. Improving the retrieval efficiency of software component markets. Wirtschaftsinformatik , v. 49, n. 6, p. 430-438, 2007.

HURLEY, Robert F.; Hult, Tomas M. Innovation, market orientation and organizational learning: an integration and empirical examination. Journal of Marketing, vol. 62, July, 1998. pp. 42-54

JANZEN, D. S.; Saiedian, H. Does test-driven development really improve software design quality? Ieee Software, v. 25, n. 2, p. 77-84, 2008.

JANZEN, D.; Saiedian, H. Test-driven development: Concepts, taxonomy, and future direction. Computer, v. 38, n. 9, p. 43-+, 2005.

JOHNSON, M. J. et al. Incorporating performance testing in test-driven development. Ieee Software, v. 24, n. 3, p. 67-+, 2007.

JONASH, R. S.; Sommerlatte, T. O valor da inovação (the innovation premium) como as empresas mais avançadas atingem alto desempenho e lucratividade. Rio de Janeiro: Campus, 2001

JURECZKO, M.; Mlynarski, M. Automated Acceptance Testing Tools For Web Applications Using Test-Driven Development. Przeglad Elektrotechniczny , v. 86, n. 9, p. 198-202, 2010.

KOTLER, Philip. Administração de marketing: a edição do novo milênio. 5° ed. São Paulo: Prentice Hall, 2002

KOU, H. B. et al. Operational definition and automated inference of test-driven development with Zorro. Automated Software Engineering, v. 17, n. 1, p. 57-85, 2010.

LEITE, J. C. S. P.: Gerenciando a qualidade de software com base em requisitos, Qualidade de software: Teoria e Prática, cap. 17. A.R.C. Rocha, J.C. Maldonado, K. Weber (orgs), Prentice-Hall (2001)

LUNA, E. R. et al. Incorporating Usability Requirements In A Test/Model-Driven Web Engineering Approach. Journal of Web Engineering, v. 9, n. 2, p. 132-156, 2010.

MADEYSKI, L. The impact of pair programming and test-driven development on package dependencies in object-oriented design - An experiment. Product-Focused Software Process Improvement, Proceedings, v. 4034, p. 278-289, 2006.

MADEYSKI, L. The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment. Information and Software Technology, v. 52, n. 2, p. 169-184, 2010.

MADEYSKI, L.; Szala, L. Impact of aspect-oriented programming on software development efficiency and design quality: an empirical study. Ieee Software, v. 1, n. 5, p. 180-187, 2007.

MARTIN, R. C. Professionalism and test-driven development. Ieee Software, v. 24, n. 3, p. 32-+, 2007.

MCDAID, K.; Rust, A. Test-Driven Development for Spreadsheet Risk Management. Ieee Software, v. 26, n. 5, p. 31-36, 2009.

MULLER, M. M.; Hofer, A. The effect of experience on the test-driven development process. Empirical Software Engineering, v. 12, p. 593-615, 2007.

NAGAPPAN, N. et al. Realizing quality improvement through test driven development: results and experiences of four industrial teams. Empirical Software Engineering, v. 13, n. 3, p. 289-302, 2008.

PARASURAMAN, A.; Colby, Charles L. Marketing para produtos inovadores: como e por que seus clientes adotam tecnologia. Porto Alegre:Bookmann, 2002

PITT-FRANCIS, J. et al. Chaste: A test-driven approach to software development for biological modelling. Computer Physics Communications, v. 180, n. 12, p. 2452-2471, 2009.

PITT-FRANCIS, J. et al. Chaste: using agile programming techniques to develop computational biology software. Philosophical Transactions of the Royal Society a-Mathematical Physical and Engineering Sciences, v. 366, n. 1878, p. 3111-3136, 2008.

PORTER, Michael E. Estratégia competitiva: técnicas para análise de indústrias e da concorrência. Rio de Janeiro, Campus, 1996 .

PORTER, Michael E. Vantagem competitiva das nações. Rio de Janeiro, ed. Campus, 1993.

POWELL, T. C.; Dentmicallef, A. Information technology as competitive advantage: The role of human, business, and technology resources. Strategic Management Journal, v. 18, n. 5, p. 375-405, 1997.

PRESSMAN, Roger S. Engenharia de Software, São Paulo: Makron Books, 1995.

RICCA, F. et al. Using acceptance tests as a support for clarifying requirements: A series of experiments. Information and Software Technology, v. 51, n. 2, p. 270-283, 2009.

ROGERS, E. M., Diffusion of Innovations. New York: The Free Press, 1995.

RUIZ, A.; Price, Y. W. Test-driven GUI development with testNG and abbot. Ieee Software, v. 24, n. 3, p. 51-+, 2007.

SENHAR, A. Strategic project leadership: toward a strategic approach to project management. R&D Management, v. 34, n.5, p..394-414, 2001

SFETSOS, P. et al. Investigating the extreme programming system - An empirical study. Empirical Software Engineering, v. 11, n. 2, p. 269-301, 2006.

SUIKKI, R; Tromstedt, R.;Haapasalo,H. Project management competence development framework in turbulent business environment. Technovation, v. 26, n.5, p.723-738, 2006

THIER, Flávio. Modelo para o processo de desenvolvimento de máquinas para a indústria de cerâmica vermelha. 2005. 198p. Tese (Doutorado em Engenharia de Produção) – Universidade Federal de Santa Catarina. Florianópolis, 2005.

TURNU, I. et al. Modeling and simulation of open source development using an agile practice. Journal of Systems Architecture, v. 52, n. 11, p. 610-618, 2006.

ULRICH, Karl T.; Eppinger, Steven D. Product design and development. 2º ed. New York: McGraw-Hill, 2000

VAN DE VEN, A. H. et al.; The Innovation Journey. New York: Oxford University Press, 1999

VODDE, B.; Koskela, L. Learning test-driven development by counting lines. Ieee Software, v. 24, n. 3, p. 74-+, 2007.

ZHANG, Y. F. Test-driven modelling for model-driven development. Ieee Software, v. 21, n. 5, p. 80-+, 2004.