Available online at www.sciencedirect.com

ScienceDirect

Information and Software Technology 50 (2008) 67–75

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

Requirements engineering: In search of the dependent variables

Tony Gorschek [a,*], Alan M. Davis [b]

[a] Department of Systems and Software Engineering, School of Engineering, Blekinge Institute of Technology, P.O. Box 520, SE-37225 Ronneby, Sweden
[b] College of Business, The University of Colorado at Colorado Springs, 1420 Austin Bluffs Parkway, P.O. Box 7150, Colorado Springs, CO 80933-7150, USA

Available online 13 October 2007

Abstract

When software development teams modify their requirements engineering process as an independent variable, they often examine the implications of these process changes by assessing the quality of the products of the requirements engineering process, e.g., a software requirements specification (SRS). Using the quality of the SRS as the dependent variable is flawed. As an alternative, this paper presents a framework of dependent variables that serves as a full range for requirements engineering quality assessment. In this framework, the quality of the SRS itself is just the first level. Other higher, and more significant levels, include whether the project was successful and whether the resulting product was successful. And still higher levels include whether or not the company was successful and whether there was a positive or negative impact on society as a whole.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Requirements engineering; Project focus; Product focus; Dependent variables; Requirements process; Product management; Project management; Process improvement; Organizational perspective

1. Introduction

Organizations throughout the world are attempting to improve their ability to perform requirements engineering. All of them need to determine if they have been successful in their efforts. The basis of this determination is the subject of this paper.

Assessment in most companies targets one of two bases:

- The requirements *process* itself, e.g., some measurement of the time and/or resources consumed during the performance of requirements engineering, and/or benchmarking the process itself against a set of "best practices".
- The primary *product* of the requirements process, e.g., some measurement of the quality of the software requirements and/or the entire requirements specification.

However, as pointed out by Davis and Zowghi [1], performing well on either or both of these "tests", does not necessarily translate into true success. Most requirements engineering process improvement efforts are performed with an aim of somehow improving the *projects* with little or no concern for the resulting *products*. This is supported by the following:

- General software process improvement (SPI) frameworks address the area of requirements engineering in the context of development projects. Initial process assessments are performed mostly on projects, and improvements are assessed on projects. This is true for model-based prescriptive SPI frameworks such as CMMI [2–4] and SPICE [5,6], as well as for inductive bottom-up frameworks like QIP [7]. Thus, improvement is determined to be successful if the project ends up costing less or completing earlier, not if the resulting product is more successful in the marketplace.
- Process improvement experiences reported from industry (for example, [8–11]) have largely focused on assessment and improvement efforts performed from project perspectives. This should not be a surprise of course, considering the previous bullet.

* Corresponding author.
E-mail addresses: tony.gorschek@bth.se (T. Gorschek), adavis@uccs.edu (A.M. Davis).

0950-5849/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2007.10.003

- Requirements engineering process models and best-practice guides (for example, [12–17]) center on activities performed on development projects.

Our argument is that the effect of requirements engineering transcends project instances. The success of activities performed as part of the requirements engineering process cannot be determined simply by examining a development instance, but are instead a part of something larger involving product management activities [18,19]. A requirements engineering process change that, say, halves the effort to construct a system, but lowers the resulting product's acceptance by its intended customers is a failure, not a success. Although this applies equally to every software process improvement effort, it is most crucial to apply this to requirements engineering because requirements engineering's *primary* purpose is to increase the likelihood that the as-built product meets the needs of the intended customers. Requirements engineering improvement efforts as well as process assessment measurements have to take this into account.

Researchers and engineers alike attempt to make changes to their processes with the hope that some positive outcome will result. The processes that they change are typically called independent variables, and the outcomes that are to be observed are called dependent variables. As requirements engineers and as requirements engineering researchers, we attempt to change the way requirements engineering is conducted and expect that positive outcomes will result. The question, however, is what are these dependent variables? If the dependent variables are selected deviously, then we can likely prove that almost any change to the requirements engineering process was positive. Dependent variables for requirements engineering can vary tremendously from the shortest term (e.g., minimizing the time we spend doing requirements engineering) to the longest term (e.g., having the best possible result for the health of the planet Earth). Deciding what is the right dependent variable for any specific case is nontrivial, yet of major consequence. This paper presents a framework of dependent variables, each of which can serve as a requirements engineering quality assessment basis.

This paper is structured as follows. In Section 2, we present the framework for dependent variables suitable for requirements process quality assessment. In Section 3, measures are proposed for each of the dependent variables. Suggestions for future research are proposed in Section 4, and finally our conclusions are summarized in Section 5.

2. Levels of quality

As requirements processes are changed, we may assess their impact on many dependent variables. These dependent variables reside within at least five distinct levels, as shown in Fig. 1. Starting at the center and working out:

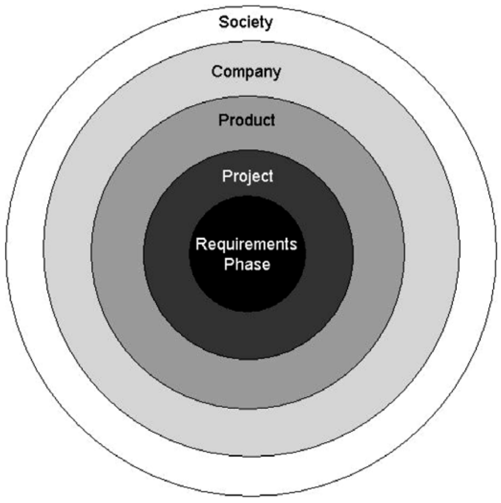

Fig. 1. Levels of requirements process change dependent variables.

- *Requirements phase.* By far the most commonly used measure among companies trying to improve their requirements process, this level includes dependent variables that relate to [20–22]:
  - *Requirements cost and time.* Such as total cost of the requirements effort, average cost per requirement, percentage of total development duration used for requirements.
  - *Requirements quality*. That is, the quality of the requirements specification itself, and so on.
- *Project*. This level is classically equated with "project success" [23,24], i.e., whether the project was completed on time, within budget, and did it meet the requirements. It includes dependent variables that relate to:
  - *Project cost and time*. Such as total cost of the project and project duration.
  - *Project estimates.* Such as the degree of meeting the budget, degree of meeting the schedule, degree to which the as-built product meets the as-specified requirements.
  - *Degree of requirements change.*

  Notice that a requirements process change that is assessed as successful using dependent variables from only the requirements level could still fail miserably at the project level. For example, a project could reduce the cost of the requirements phase by skipping it altogether, and the result could be so much project entropy that the project ends up costing twice as much as planned.
- *Product.* Dependent variables within this level determine the degree of *product* success, e.g., did the product succeed in fulfilling the needs of its intended customers/users. It includes measures relating to:
  - *Requirements selection.* That is, the degree to which the requirements selected for the product reduces errors, increases efficiency, and so on, for its users and or customers.

  - *Degree of Impact*. That is, the degree to which the product increases revenues, decreases costs, and so on, for the company developing the product. If the product is to be sold to a market, then it includes measures of the units sold, revenues produced, margins attained, average cost of sales, average order size, and so on.
    Notice that a requirements process change that is assessed as successful using dependent variables from only project level could still fail miserably at the product level. For example, a project could come in on budget, on schedule, and meet its written requirements, but because the documented requirements did not reflect the true needs of the users (or because the user needs changed during development without the project team noticing), the resulting product could fail in the marketplace. Another example could be that unnecessary requirements were included in the project effectively increasing development time and cost without adding product value.
- *Company*. Suppose a project is delivered late and/or over budget, but the resulting product succeeds tremendously in the marketplace. Using dependent variables from the previous level, we could declare the product successful. However, what happens if the lateness and/or excessive cost is the cause of another project within the same development company failing. Furthermore, what if the adversely affected project was more critical to the company's success than the original project? In such a case, we cannot declare the project to be successful. The company level includes such measures as
  - *Portfolio management*. The degree to which products compliment each other, and the degree to which they each demonstrate optimal use of resources.
  - *Strategic alignment*. The degree to which the product aligns with corporate strategies.
  - *Degree of impact*. Total corporate revenues, revenue growth, meeting of revenue targets, profits, profit growth, meeting of profit targets, market share, return on investment, cash situation, and so on.
- *Society*. Although a company may provide great financial returns to its shareholders, a corporation also has a greater responsibility to society at large. Thus, a project that contributes to a company's bottom line but pollutes the environment or kills people must be considered a failure. Products always give rise to
  - *positive and negative externalities* [25] that have to be taken into consideration.

3. Related research

Measures for all five levels have been proposed by others. For example:

- *Requirements phase*. Impact on the requirements phase may be measured in terms of its outputs and in terms of its activity. As the easiest of the five levels to measure, no shortage exists of research aimed at determining dependent variables related to the quality of software requirements specifications. Boehm [26] was the first to define measures of quality of an SRS. This was followed by Alford and Burns [27], Zave and Yeh [28], the IEEE [29], and many others, all of which were summarized by Davis et al. [21]. More focused work has been reported by Hunter and Nuseibeh [30] for consistency of requirements, Yue [22] for completeness of requirements, Zowghi and Gervasi [31] for consistency, completeness, and correctness, Agusa [20] for verifiability, and Kovitz [32] for ambiguity. Meanwhile, in terms of the activity, COCOMO [33] reports that on average the requirements phase consumes 8% of the total project resources during a period of time averaging 20% of the total project duration. In a study of NASA projects, Forsberg [23] presents a graph (see Fig. 2) that shows optimal project success occurs when 7% to 15% of total project resources are used during the requirements phase.
- *Project*. Project performance is often measured in terms of cost, schedule and how many of the documented requirements were actually implemented. At least four distinct requirements-related causes exist for a project succeeding or failing at the project level:
  (1) Poor estimation techniques. Because requirements define what is to be done in a project, they serve as a natural basis for estimation of project costs and schedules. The primary attempts to estimate costs and schedules using requirements employ function points or feature points [34,35]. Obviously, if a project completes its mission and significantly exceeds the resource expectations, one cause could be that the requirements were stated in a manner that rendered them ineffective as an estimation basis.
  (2) Mismatch between requirements specified and requirements satisfied. The level of fulfillment of the specified requirements are usually measured through verification and validation (V&V) activities, of which *system testing* is the most common [36]. Failure of V&V can be an indication of a poor requirements engineering process, e.g., the requirements could be too poor to serve as a basis for cre-

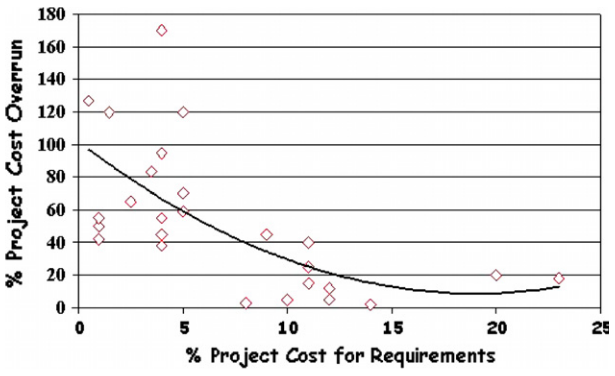

Fig. 2. Forsberg study of NASA projects.

ating test cases. Inspections of requirements can be good way to ascertain and in fact measure the inherent testability of requirements [37–40].

(3) Evolving requirements. Some see changing requirements during a project as a problem, and use negative expressions such as *requirements creep*, as if requirements change is somehow evil. The reality is that if the real-world requirements are actually changing (or even if our perceptions of them are actually changing), we have just two choices: ignore the changes and build the wrong product, or change the requirements being addressed by the project (see Table 1). In any case, the degree of requirements change will significantly affect the success of the project. The gross number of requirement changes made to the project's requirements and severity of these changes are examples of dependent variables used at the project level [41–44].

(4) Mismanagement. This cause is unrelated to the subject of this paper.

- *Product*. The success of an individual product (or a product line/family) is directly linked to *what* requirements were selected for realization, i.e., the degree of alignment between what the product does and what the needs really are. This is not the same as assuring that the requirements were realized in the right way, which is measured by *project* success, as described in the previous section. Two issues are directly responsible for the success or failure of requirements engineering at the product level:

(1) Requirements selection. Requirements in a market driven environment come from several sources both internal (e.g., developers, marketing, sales, support personnel, bug reports, etc.) and external (e.g., users, customers and competitors, often gathered via surveys, interviews, focus groups, competitor analysis, etc.) [45–47]. The volume of requirements that need to be handled makes initial screening important in order to decrease the risk of overloading in the evaluation and realization process [48]. Regnell et al. present an analytical model based on product strategies and business strategies aimed at enabling initial handling of large volumes of requirements [49]. The use of requirements abstraction and indirectly product strategies has also been tried with promising results through the use of a Requirements Abstraction Model [77].

After initial screening, requirements have to be prioritized, interdependencies need to be defined, and cost estimations have to be established in order for the selection process to take place. Several methods for attaining requirement priority exist, including AHP [50], the 100-point method [51], triage [52,53], attainment [54], negotiation [55], and the planning-game [56]. Prioritization can be performed by customers and/or customer representatives, as well as by internal experts from marketing and product management. Prioritization of requirements can be used as input to the requirements selection before the final choices for realization are made. Interdependencies between requirements are also a determining factor in selection; studies show that only about 20% of requirements are relatively independent [57]. This may influence the selection radically as high priority requirements can be dependent on low priority requirements, and vice versa. In addition to priority and interdependencies, cost estimations have to be made before final selection can be made, determining what (customer) needs are to be satisfied by the product/release, and which requirements are to be postponed or dismissed.

Being able to measure the success of the requirement selection process is crucial. Regnell et al. present a model that allows for principal reasoning about both the quality and the current capacity of the requirement selection process [49]. Karlsson et al. presents the PARSEQ method which focuses on post-release analysis of requirements selection quality, examining the requirements actually selected for realization [58]. In addition, the perceived customer value of a product can be measured through the use of GAP analysis [59–61]. GAP analysis measures positive and negative "gaps" between what the product offers and what the customer perceives. Features and characteristics of the product are identified and their fulfillment of customer needs is mapped. A positive gap represents when a product delivers more than is expected, a negative gap the opposite. One of the earliest descriptions of the need to measure this "gap" was described in [62]. Customer Value Analysis (CVA) is similar to

Table 1
The impact of requirements change

	Real-world (e.g., market) requirements changing	
	Yes	No
Project's requirements changing		
Yes	Good for the *Product*, but *Project* may complete late or over budget	*This* is the situation that should be called requirements creep. it is bad for the *Product* and bad for the *Project*.
No	Guaranteed that the *Product* Will Fail, but *Project* may be completed on-time and on budget	Good for *Product* and *Project*

GAP analysis but also includes the perspective of using competitor products in the analysis and the evaluated product is graded with regards to the value of a need in comparison to alternatives [63]. Both GAP and CVA can be used to measure selection quality post-release, but the results can also be used actively as input to the next round of requirements selection for the product in question. The relationship between requirements change, development time, and customer satisfaction was explored by [64].

(2) Product strategies. The development of product strategies is a way of planning for a product, and documenting in what way the product will support business strategies [65]. Product strategies can be seen as roadmaps for the long-term development of a product and should reflect not only current market knowledge and customer priorities but also the long-term goals set for a certain product. The requirements selection described above should be done within the boundaries set by product strategies. Ignoring product strategies (and only looking at current priorities) may mean that a product is successful short-term, but at the expense of the long-term goals [40,47]. For example, security requirements may be deemed less important at present but the long-term plans for the product is to eventually break into new market segments where security is deemed crucial. One of the fundamental aims of a product strategy is to explicitly plot the goals and the limits of a product – focusing all efforts and aligning them in one deliberate direction [47,66]. On the other hand, sometimes constraining yourself too much to a business strategy could cause a company to miss new, out-of-the-box business opportunities that can leverage business technology strengths.

Requirements selection within the boundaries of product strategy does not guarantee success, as the strategies followed can be flawed. However having up-to-date product strategies, which reflect all vital knowledge – from both technology and marketing perspectives – will most certainly increase the chance of developing a successful product [40,47,67].

Internal Value Analysis (IVA) is a technique to measure whether or not a product is in line with the product strategies (and the company strategies), taking limited resources and other products into account [40,47]. Using IVA, factors like resources available (time, money, risk, and knowledge) can be taken into consideration, complementing data from both GAP/CVA analysis, prioritizations of requirements, dependency mapping, and cost estimates. Combined, this can form decision support material for requirements selection taking product strategies into consideration.

(3) Did the product make money (revenue and/or profit)? Selecting the "right" requirements from a customer perspective and thus satisfying customer expectations should in theory have a strong correlation with market success. However, many factors, including the presence of unexpressed, tacit knowledge, can adversely effect the correlation (see [68] for discussions of elicitation techniques suitable for discovering tacit requirements). Whether or not the product is really successful requires a wider perspective, assessing the internal value of realizing requirements, risks taken, resources used, and so on, into consideration, as well as working within the long-term perspective of product strategies. On the product level, this may suffice, especially since many of the larger issues are filtered down through product strategies, but regarding taking the "best alternative investment" into consideration the company scope has to be taken into consideration.

- *Company*. The company level of dependent variables in requirements engineering is closely related to the product scope described above. However, at the company level, an overview of the entire product portfolio consisting of products already deployed, those under development, and those just being planned, is essential. The product strategies mentioned in the previous section are created and compared to the overall strategy of the company (a.k.a. business strategies) as such [47,66].

  Product line literature uses the term "Product Portfolio Scoping" with regards to the activity of deciding the overall focus of strategies on a company level as it pertains to product lines – helping to leverage existing investments where it pays [69]. This activity is based on traditional portfolio management [46], but has been adapted to the software product line domain where reuse is paramount and core assets are established [70]. The selection of requirements for realization has to be viewed from the company level when it regards product line organizations, as the selection of one requirement can potentially impact several products that share core assets. This is one of the reasons why development in product line organizations often take the company level into consideration with regards to requirements engineering.

  The main idea at the company level is to maximize the total economic benefits of several products, as opposed to the product perspective of maximizing the benefits of a particular product. Many of the same tools, e.g., GAP, CVA, and IVA, can be used. The use of bubble diagrams like the Boston Consulting Group Matrix is also fairly common [71]. They are used to visualize the placement of individual products in relation to market maturity, as well as in relation to each other. The idea is to get a distribution of products over time relative to market maturity, risk, and reward in an attempt to minimize risk and maximize reward in the long run [72]. These tools help the company choose between products (and thus indirectly requirements) in a manner that maintains a balance between high risk – high reward and low risk – low reward development.

Another approach to insure company success is to align all product requirements with explicitly stated corporate business goals. This has been the subject of a stream of research by Bleistein and his colleagues at NICTA in Australia [74,75]. As we have discussed earlier, it is extremely difficult to assess the degree to which requirements contribute to the satisfaction of quality at the levels of company and society. The work of Bleistein et al. shows considerable promise in this regard because it insures a priori that the requirements are aligned with corporate goals. Of course, it presupposes that (a) corporate goals can be stated in an unambiguous fashion, and (b) that corporate goals do not mutate (e.g., changes to corporate goals add a third dimension and four more boxes to Table 1 thus if neither the real-world nor the project's requirements change, as indicated in the lower right box of Table 1, but the corporate goals change, the project could still fail!).

- *Society*. From the perspective of society, the development and consumption of products can be associated with *positive and negative externalities* [25]. This means that neither the developer nor the customer bears all of the costs or reap all of the gains generated by the product. Developing a defective piece of avionics software can potentially have negative externalities – the plane crashes. On the other hand, offering a new solution to improve airport landing scheduling can have positive externalities, both economic and safety related.
  By actively taking positive and negative externalities into account when formulating company strategies (which filter down to product development and requirements engineering through product strategies) a company can try to maximize the positive effects of their product development. This can demand that some resources be added for analysis of possible externalities, but it may also reap benefits, e.g., a positive response from society. An example of this can be seen in the automotive industry where research on alternative fuels is not immediately profitable in terms of revenue, but it generates good-will and possibly even improvement in long-term competitiveness. By including the societal perspective in the selection of requirements for realization a company can attempt to take a proactive stance regarding product impact on society at large, which can be seen as a pre-requisite for long-term survival.

4. Discussion and future research

No shortage exists of best-practice guides [13] for RE where "best" means "best at the requirements-level or the project-level". New best-practice guides need to be developed that take into account broader bases. Similarly, plenty of dependent variables have been used to assess whether or not requirements process changes have had a positive or deleterious affect of the requirements phase activities or project success. However, to enable us to improve requirements practices and actually have an effect on something important, we will need to invent new dependent variables for all levels. There are examples of empirical work where changes on one level are partially mapped to impact on another, e.g., Damian et al. present a case study where process improvements in the requirements phase are evaluated in terms of impact to project activities [78].

Table 2 gives an overview of the quality levels and groupings of dependent variables presented in this paper. As requirements engineers, we often handle and measure dependent variables on the requirement and project levels, but as we continue to product and especially company levels, the dependent variables and the tools/techniques used (CVA, IVA, GAP, etc.) are often considered a part of economics and management research.

There is a pressing need for us to see beyond traditional boundaries and utilize the multidisciplinary nature of requirements engineering to develop not only new dependent variables but also tools and techniques for their measurement, seizing the opportunity to cooperate with management researchers, and thus addressing the issues

Table 2
Overview of the quality levels from Sections 2 and 3

Dependent variables				
Requirements phase	*Cost* and *time* (RE effort, per requirement, % of total dev. effort)	*Requirement quality* (consistency, completeness, ambiguity, verifiability)		
Project	*Cost* and *time* (project cost, project duration)	*Estimates* (degree of meeting budget and Schedule, planned vs. realized req.)	*Requirement change* (creep, addition, new req.)	
Product	*Requirement selection* (screening, prioritization, packaging to projects)	*Impact* (revenue, profit, market share, short- and long-term)		
Company	*Product portfolio scoping* (selecting what products to commit to, best alternative investment, reuse)	*Portfolio management* (balance between products in different lifecycles, balance between high risk/low risk, etc.)	*Strategic alignment* (support business strategies, long-term plans)	*Impact* (corporate revenue, profit, market share, short- and long-term)
Society	*Positive* and *negative externalities* (reap benefits/bear costs, good-will)			

together top-down as well as bottom-up. This will allow us to take consequential dependencies into account, e.g., how should product strategies be formulated to be useful for requirements engineers performing initial screening/triage of requirements? Also, through cooperation it should be possible to develop dependent variables for measurement across the levels, or at least mapping dependencies between the variables, making it possible to develop useful holistic models predicting actual impact of a process change.

For example, inventing new dependent variables and mapping dependencies between levels can provide a basis for finding an appropriate balance between technology-push and market-pull [76] in a product development organization. As mentioned before, optimal product strategies can be used to assist in initial requirements screening activities (requirements selection), and the requirements themselves can be used as input to management when developing the product strategies themselves. This can be seen as premiering market-pull, i.e., what is to be developed depends on the market needs, not on the creation of innovative ideas and new technologies. Obviously premiering technology-push is also possible, e.g., requirements creation can be driven by technologists rather than by the market. Premiering either or these over the other can have negative consequences. The complete understanding of the relationships between the dependent variables can help a company find the right balance between market-pull and technology-push. However, this requires cooperation between managers (formulating strategies), engineers (analyzing requirements and inventing new technologies), and marketing personnel (understanding market needs). Requirements engineering can be used as a link between these three areas. The dependent variables are the common tool that all three own and use together to assure that balance can be obtained.

An effective requirement selection process can help assure that the “right” requirements are put forward, putting minimal effort on inappropriate requirements that will be dismissed in later stages or just are not worth implementing, freeing up resources to either realizing more “good” requirements or putting resources into new technology high-risk/high-potential development. In addition, as an organization’s requirements screening capabilities increase, overloading becomes a less critical issue, making it possible to elicit more requirements internally. This could be a powerful way to enable technology-push in a structured way as internal innovative ideas are compared by product managers in relation to incoming market-pull requirements.

As the distance between the independent and dependent variables increases, more independent variables have their influence, making it increasingly more difficult to produce empirical results that show a correlation between the RE process change and dependent variables. Most researchers who have studied the relationship between RE process change and downstream success have acknowledged the difficulty of understanding the correlation, but that is not enough of a reason to not pursue this type of research.

5. Conclusions and contributions

At first glance, it appears to be quite easy to determine whether a RE process change has been beneficial: simply measure its impact on the requirements phase itself or on the project. However, the impacts of successful RE process change are not limited to just projects. A process change that is highly successful at the requirements phase and project levels (e.g., one that reduces the efforts required for both) can be a total disaster if it results the creation of a product that nobody will buy or use.

This paper’s primary contributions are:

- It creates a taxonomy of levels on which the impact of RE process changes can be assessed.
- It exposes the fact that at higher levels in the taxonomy, we need to recognize that impacts are multidisciplinary and multi-perspective.
- It raises awareness that strategies (both company level and product level) must be explicitly stated [73,74], and that the individuals performing RE are aware of this work as they are the ones that have to make decisions within the limits posed by the strategies in question. Requirements engineers are the link between management and development [75] as they directly realize strategies through the choices they make when prioritizing between products, and between requirements within a product.
- It suggests that product strategies should reflect long-term goals, but these goals also have to be aligned to both current market and technology trends. To this end, the requirements themselves (from multiple sources – both internal and external) can be used as input to the creation of viable and up-to-date strategies, and also as a way to “test” the current company and product strategies. This “operational feedback” could be used continuously to modify product strategies as needed.
- From a wider perspective, the assessment of requirements engineering process change is a complex undertaking. The implications reach much further than project level; thus, the capability of current assessment and improvement frameworks do not suffice. For example, as requirements engineering transcends several perspectives, so do the activities performed as part of requirements engineering. The implication being that the cost of requirements engineering work cannot be calculated based on what is done in association with a development instance (project).

If engineers, managers, and marketing personnel develop and share responsibility for monitoring dependent variables in collaboration, each group realizing that a process change affects more than one quality level, success and failure will be judged on a different scale than today.

References

[1] A.M. Davis, D. Zowghi, Good requirements practices are neither necessary nor sufficient, Requirements Engineering 11 (1) (2006) 1–3.
[2] D.M. Ahern, A. Clouse, R. Turner, CMMI Distilled: A Practical Introduction to Integrated Process Improvement, Addison-Wesley, Boston, 2003.
[3] CMMI-PDT, Capability maturity model integration (CMMI), Version 1.1, in: CMMI for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing Version 1.1 (CMMI-SE/SW/IPPD/SS, V1.1), Pittsburgh, 2002.
[4] SCAMPI-ADT, Standard CMMI Appraisal Method for Process Improvement (Scampi) Version1.1 (CMU/SEI-2001-Hb-001), Carnegie Mellon – SEI, Pittsburgh, 2001, p. 245.
[5] K. El Emam, J.-N. Drouin, W. Melo, SPICE: The Theory and Practice of Software Process Improvement and Capability Determination, IEEE, Los Alamitos CA, 1998.
[6] SPICE-DT, SPICE, 2003, Available from: <http://www.sqi.gu.edu.au/spice/>.
[7] V.R. Basili, Quantitative Evaluation of Software Methodology, University of Maryland, College Park, Maryland, Technical report TR-1519, 1985.
[8] V.R. Basili, F.E. McGarry, R. Pajerski, M.V. Zelkowitz, Lessons learned from 25 years of process improvement: the rise and fall of the Nasa Software Engineering Laboratory, in: The Proceedings of 24th International Conference on Software Engineering (ICSE02), Orlando, 2002, pp. 69–79.
[9] J.A. Calvo-Manzano Villalón, G. Cuevas Agustín, T. San Feliu Gilabert, A. De Amescua Seco, L. García Sánchez, M. Pérez Cota, Experiences in the application of software process improvement in SMEs, Software Quality Journal 10 (2002) 261–273.
[10] J.D. Herbsleb, D.R. Goldenson, A systematic survey of CMM experience and results, in: The Proceedings of the 18th International Conference on Software Engineering, Los Alamitos, CA, 1996, pp. 323–330.
[11] K. Kautz, H.W. Hansen, K. Thaysen, Applying and Adjusting a software process improvement model in practice: the use of the ideal model in a small software enterprise, in: The Proceedings of the 2000 International Conference on Software Engineering, Los Alamitos, CA, 2000, pp. 626–633.
[12] REAIMS, 2003, Available from: <http://www.comp.lancs.ac.uk/computing/research/cseg/projects/reaims/index.html/>.
[13] I. Sommerville, P. Sawyer, Requirements Engineering: A Good Practice Guide, John Wiley & Sons, Chichester UK, 1999.
[14] L. Scott, R. Jeffery, L. Carvalho, J. D'Ambra, P. Rutherford, Practical software process improvement – the impact project, in: The Proceedings of the Australian Software Engineering Conference, Los Alamitos, CA, 2001, pp. 182–189.
[15] K.E. El Emam, N.H.E. Madhavji, Elements of Software Process Assessment & Improvement, Wiley-IEEE, Los Alamitos, CA, 1999.
[16] G. Kotonya, I. Sommerville, Requirements Engineering: Processes and Techniques, John Wiley, New York, 1998.
[17] T. Gorschek, M. Svahnberg, K. Tejle, Introduction and application of a lightweight requirements engineering process evaluation method, in: The Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Essen, Germany, Available from: <http://www.bth.se/fou/Forskinfo.nsf/>. 2003, pp. 101–112.
[18] R. Wieringa, C. Ebert, Guest Editors' introduction: RE'03: practical requirements engineering solutions, IEEE Software 21 (2004) 16–18.
[19] L. Karlsson, Å. Dahlstedt, J. Natt och Dag, B. Regnell, A. Persson, Challenges in market-driven requirements engineering – an industrial interview study, in: The Proceedings of the Eighth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'02), Essen, Germany, 2003, pp. 101–112.
[20] K. Agusa, et al., Verification of requirements description, in: The Proceedings of Twelfth Hawaii International Conference on System Science, Los Alamitos, CA, 1979.
[21] A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, M. Theofanos, Identifying and measuring quality in a software requirements specification, in: The Proceedings of First International Software Metrics Symposium, Los Alamitos, CA, 1993, pp. 141–152.
[22] K. Yue, What does it mean to say that a specification is complete? in: The Proceedings of Fourth International Workshop on Software Specification and Design, Los Alamitos, CA, 1987.
[23] K. Forsberg, H. Mooz, System engineering overview, in: R.H. Thayer, M. Dorfman, A.M. Davis (Eds.), Software Requirements Engineering, second ed., IEEE, Los Alamitos, CA, 1997, pp. 44–72.
[24] Standish Group, The Chaos Report, 1995, Available from: www.standishgroup.com.
[25] J.M. Perloff, Microeconomics, Third ed., Pearson Addison Wesley, Boston, 2004.
[26] B. Boehm, Software engineering, IEEE Transactions on Computers 25 (1976) 1226–1241.
[27] M. Alford, I. Burns, R-Nets: a graph model for real-time software requirements, in: The Proceedings of Symposium on Computer Software Engineering, New York, NY, 1976, pp. 97–108.
[28] P. Zave, R. Yeh, Executable requirements for embedded systems, in: The Proceedings of Fifth IEEE International Conference on Software Engineering, Los Alamitos, CA, 1981, pp. 295–304.
[29] IEEE, Recommended Practice for Software Requirements Specifications (Standard 830-1984), IEEE Press, New York, NY, 1984.
[30] A. Hunter, B. Nuseibeh, Analyzing inconsistent specifications, in The Proceedings of IEEE International Symposium on Requirements Engineering, Los Alamitos, CA, 1997, pp. 78–86.
[31] D. Zowghi, V. Gervasi, The three C's of requirements: consistency, completeness, and correctness, in: The Proceedings of International Workshop on Requirements Engineering: Foundations for Software Quality, Essen, 2002, pp. 155–164.
[32] B. Kovitz, Ambiguity and what to do about it, in: The Proceedings of Tenth International IEEE Conference on Requirements Engineering, Los Alamitos, CA, 2002, p. 213.
[33] B.W. Boehm, Software Cost Estimation with Cocomo II, NJ, Prentice Hall, Upper Saddle River, 2000.
[34] C. Jones, Applied Software Measurement: Assuring Productivity and Quality, second ed., McGraw-Hill, New York, 1997.
[35] A.J. Albrecht, J.E. Gaffney, Software function, source lines of code, and development effort prediction, IEEE Transactions Software Engineering SE-9 (1983) 639–647.
[36] S.R. Rakitin, Software Verification and Validation for Practitioners and Managers, second ed., Artech House, Boston, MA, 2001.
[37] O. Laitenberger, T. Beil, T. Schwinn, An industrial case study to examine a non-traditional inspection implementation for requirements specifications, in: The Proceedings of the Eighth IEEE Symposium on Software Metrics, Los Alamitos, CA, 2002, pp. 97–106.
[38] A.A. Porter, L.G.J. Votta, V.R. Basili, Comparing detection methods for software requirements inspections: a replicated experiment, IEEE Transactions on Software Engineering 21 (1995) 563–576.
[39] F. Shull, I. Rus, V. Basili, How perspective-based reading can improve requirements inspections, Computer 33 (2000) 73–79.
[40] S.A. Ross, R. Westerfield, B.D. Jordan, Essentials of Corporate Finance, Third ed., McGraw-Hill, Boston, 2001.
[41] V. Basili, D. Weiss, Evaluation of a software requirements document by analysis of change data, in: The Proceedings of International Conference on Software Engineering, Los Alamitos, CA, 1981, pp. 314–323.
[42] K. El Emam, D. Holtje, N.H. Madhavji, Causal analysis of the requirements change process for a large system, in: The Proceedings

of International Conference on Software Maintenance, Los Alamitos, CA, 1997, pp. 214–221.
[43] S.D.P. Harker, K.D. Eason, J.E. Dobson, The change and evolution of requirements as a challenge to the practice of software engineering, in: The Proceedings of IEEE International Symposium on Requirements Engineering, Los Alamitos, CA, 1993, pp. 266–272.
[44] W. Lam, V. Shankararaman, G. Saward, Managing requirements change: a dissection of management issues, in: The Proceedings of Fifth International Workshop on Requirements Engineering: Foundations for Software Quality, Heidelberg, 1999, pp. 19–32.
[45] P. Kotler, G. Armstrong, Principles of Marketing, Ninth ed., Prentice Hall, Upper Saddle River NJ, 2001.
[46] D.R. Lehmann, R.S. Winer, Product Management, Third ed., McGraw-Hill, Boston, 2002.
[47] H. Mintzberg, B.W. Ahlstrand, J. Lampel, Strategy Safari: A Guided Tour through the Wilds of Strategic Management, Free Press, New York, NY, 1998.
[48] M. Weber, J. Weisbrod, Requirements engineering in automotive development: experiences and challenges, IEEE Software 20 (2003) 16–24.
[49] B. Regnell, L. Karlsson, M. Host, An analytical model for requirements selection quality evaluation in product software development, in: The Proceedings of the 11th International Conference on Requirements Engineering, Los Alamitos, CA, 2003, pp. 254–263.
[50] T.L. Saaty, L.G. Vargas, Models, Methods, Concepts & Applications of the Analytic Hierarchy Process, Kluwer Academic Publishers, Boston, MA, 2001.
[51] D. Leffingwell, D. Widrig, Managing Software Requirements: A Unified Approach, Addison-Wesley, Reading, MA, 2000.
[52] A.M. Davis, The art of requirements triage, IEEE Computer 36 (2003) 42–49.
[53] E. Simmons, Requirements triage: what can we learn from a medical approach? IEEE Software 21 (2004) 86–88.
[54] M.S. Feather, T. Menzies, Converging on the optimal attainment of requirements, in: The Proceedings of IEEE Joint International Conference on Requirements Engineering, Los Alamitos, CA, 2002, pp. 263–270.
[55] I. Hoh, D. Olson, T. Rodgers, A requirements negotiation model based on multi-criteria analysis, in: The Proceedings of Fifth IEEE International Symposium on Requirements Engineering, Los Alamitos, CA, 2001, pp. 312–313.
[56] L. Karlsson, P. Berander, B. Regnell, C. Wohlin, Requirements prioritisation: an experiment on exhaustive pair-wise comparisons versus planning game partitioning, in: The Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004) – (in: proceedings of ICSE 2004), Los Alamitos, 2004, pp. 145–154.
[57] P. Carlshamre, Release planning in market-driven software product development: provoking an understanding, Requirements Engineering 7 (2002) 139–151.
[58] L. Karlsson, B. Regnell, J. Karlsson, S. Olsson, Post-release analysis of requirements selection quality – an industrial case study, in: The Proceedings of the Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03), Essen, Germany. Available from: <http://www.bth.se/fou/Forskinfo.nsf/>, 2003, pp. 47–56.
[59] T.J. Redling, A methodology for developing new product line requirements through gap analysis, in: The Proceedings of 22nd Digital Avionics Systems Conference, Indianapolis, 2003, pp. 10.A.1–101-11.
[60] M. Zairi, Best Practice: Process Innovation Management, Butterworth-Heinemann, Oxford, Boston, 1999.
[61] N. Hill, J. Brierley, R. MacDougall, How to Measure Customer Satisfaction, Brookfield, Gower, Aldershot, 1999.
[62] A. Davis, E. Bersoff, E. Comer, A strategy for comparing alternative software development life cycle models, IEEE Transactions on Software Engineering 14 (1988) 1453–1461.
[63] P. Kotler, G. Armstrong, J. Saunders, V. Wong, Principles of Marketing, Third European ed., Prentice Hall, Harlow, England, New York, 2002.
[64] A. Davis, E. Bersoff, E. Comer, A strategy for comparing alternative software development life cycle models, IEEE Transactions on Software Engineering 14 (1998) 1453–1461.
[65] L. Gorchels, The Product Manager's Handbook: The Complete Product Management Resource, second ed., NTC Business Books, Lincolnwood, Ill, 2000.
[66] D.J. Teece, Managing Intellectual Capital: Organizational, Strategic, and Policy Dimensions, Oxford University Press, New York, Oxford, 2000.
[67] T. Sasaki, A. Nagata, R. Toyama, T. Hirata, K. Hasegawa, Coevolution of patent strategy and product strategy, in: The Proceedings of Portland International Conference on Management of Engineering and Technology PICMET'01, Los Alamitos, CA, 2001, pp. 481–484.
[68] M. Jirotka, J. Goguen (Eds.), Requirements Engineering: Social and Technical Issues, Academic Press, London, UK, 1994.
[69] W. Lam, A case-study of requirements reuse through product families, Annals of Software Engineering 5 (1998) 253–277.
[70] K. Schmid, A comprehensive product line scoping approach and its validation, in: The Proceedings of the 24th International Conference on Software Engineering, New York, 2002, pp. 593–603.
[71] C.W. Stern, G. Stalk, Boston Consulting Group Perspectives on Strategy: From the Boston Consulting Group, J. Wiley, New York, 1998.
[72] R.G. Cooper, S.J. Edgett, E.J. Kleinschmidt, Portfolio Management for New Products, second ed., Perseus Pub., Cambridge, MA, 2001.
[73] S. Bleistein, K. Cox, J. Verner, Modeling business strategy in e-business systems requirements engineering, in: S. Wang, K. Tanaka, S. Zhou, et al. (Eds.), Lecture Notes in Computer Science, Springer-Verlag, 2004, pp. 617–628.
[74] S. Bleistein, K. Cox, J. Verner, Validating strategic alignment of organizational IT requirements using goal modeling and problem diagrams, Journal of Systems and Software 79 (3) (2006) 362–378 (March.
[75] N. Rosenberg, Inside the Black Box: Technology and Economics, Cambridge University Press, Cambridge, MA, 1982.
[76] A.M. Davis, Just Enough Requirements Management: Where Software Development Meets Marketing, Dorset House Publ., New York, NY, 2005.
[77] T. Gorschek, C. Wohlin, Requirements abstraction model, Requirements Engineering 11 (1) (2006) 79–101, March.
[78] D. Damian, J. Chisan, L. Vaidyanathsamy, Y. Pal, An industrial case study of the impact of requirements engineering on downstream development, in: Proceedings. 2003 International Symposium on Empirical Software Engineering, 2003, pp. 40–49.