

REQUIREMENTS ENGINEERING FOR WEB-BASED INFORMATION SYSTEMS DEVELOPMENT: AN EMPIRICAL STUDY

Kieran Conboy, Michael Lang
Department of Accountancy & Finance
National University of Ireland, Galway

University Road, Galway, IRELAND

Telephone: +353 91 750301 Fax: +353 91 750565 Email: Michael.Lang@nuigalway.ie

Keywords: Web-based Information Systems; Systems Development Methodologies;
Requirements Engineering; Requirements Prioritisation

Abstract

No system is perfect in so far as requirements are always the subject of ongoing negotiation, and there will generally be a subset of the requirements that get pushed aside. This is especially true of the volatile environments that characterise Web-based information systems (WIS) development. A pragmatic resolution is to apply a requirements prioritisation methodology that aims to produce a best possible wish list under constrained circumstances. Herein are reported the findings of a case study of a WIS development project which uses such an approach.

In practice, requirements engineering methodologies are not executed in the structured, methodical way advocated by researchers. This is not surprising, as the underlying philosophy of most of these methodologies is that systems development is a rational process, whereas in actuality it is more accurately portrayed as creative, somewhat improvised behaviour. It is therefore important to determine if the key issues suggested by the normative view of requirements engineering corresponds with the approaches being used in the real world.

This paper outlines the theories of “rationalism” and “improvisation”, and describes the major pitfalls of each. It then discusses how these pitfalls were encountered in the case study. Interestingly, although the methodology in this case study was very specific and firmly based in the “rational” paradigm, the users often improvised by privately making decisions on how certain aspects should or should not be implemented. As such, elements of both rationalism and improvisation were experienced. The paper concludes with the assertion that even where methodologies are developed and revised based on actual experiences in practice, as opposed to academic theory, it is not possible to devise an approach that is wholly methodical because amethodical aspects will always creep in during implementation.

1. INTRODUCTION

In less than a decade, the Web has grown from humble text-only interfaces and static documents to now embrace dynamic, large-scale information systems that extend beyond the traditional boundaries of organisations. Academic researchers have accordingly responded so that the literature on Web-based information systems (WIS) development is strewn with a multitude of methodologies that all aspire to the same elusive goal: to deliver a WIS that satisfies all cost, time, functionality and quality constraints.

This paper concentrates on the requirements engineering aspect of WIS development, as opposed to the entire life cycle. No information system is perfect in so far as the requirements are always the subject of ongoing negotiation, and there will generally be a subset of the requirements that get pushed aside. This is especially true of the volatile environments that characterise WIS development. There are a number of methodologies that fall under the general classification of “requirements prioritisation” which aim to produce a best possible wish list under constrained circumstances. Herein are reported the findings of a case study of a WIS development project which uses an in-house requirements prioritisation methodology.

In practice, requirements engineering methods are not executed in the structured, methodical way advocated by researchers [8]. This is not surprising, as the underlying philosophy of most of these approaches is that systems development is a rational process, whereas in actuality it is more accurately portrayed as “situated action” [16], “improvisation” [2], or “amethodical” behaviour [1]. It is therefore important to determine if the key issues suggested by the normative view of requirements engineering corresponds with the approaches being used in the real world.

Section 2 outlines two theories of information systems development (ISD): those of “rationalism” and “improvisation”, and describes the major pitfalls of each. Section 3 then discusses how these pitfalls were encountered in the case study. Interestingly, although the methodology in this case study was very specific and firmly based in the “rational” paradigm, the users often improvised by privately making decisions on how certain aspects should or should not be implemented. As such, elements of both rationalism and improvisation were experienced. This is then discussed in the Conclusions.

2. THEORIES OF SYSTEMS DEVELOPMENT

The intricacies of ISD are poorly understood and, broadly speaking, attempts to describe ISD processes generally fall into two opposite viewpoints. On the one side, there are processes that fit within the logical framework traditionally associated with the scientific or “rational” approach to problem solving, and on the other are processes that are apparently of a creative, ad hoc, “improvised” nature.

2.1. Traditional ISD Theory: “Rationalism”

The conventional theory of systems development is that of “rationalism”, which is deeply rooted in the work of Descartes four centuries ago. In its essence, the Cartesian method may be summarised by four principles: (1) accept nothing as true which is not clear and distinct; (2) analyse a problem into its parts and discuss it part by part; (3) arrange thoughts from simple to complex as the order of study; and (4), enumerations must be full and complete.

Most of the well-known systems development methodologies (SDMs), – such as SSADM, Information Engineering, and the “Waterfall” SDLC, – are firmly based on the principles of rationalism. Empirical studies reveal that, in reality, SDM usage is rather limited, and those who use SDMs mostly tend to judiciously mix-and-match combinations and parts of methodologies rather than following all the steps required by a particular methodology [7,4]. A number of major philosophical and pragmatic problems with the use of formalised SDMs in practice have become evident, outlined in the following paragraphs.

Firstly, the Cartesian method demands that the problem being solved is well defined and fixed before design starts. Based on this assumption, there arises a tendency to separate the “ends” and the “means”; that is, the finished product and the method used to get there [14]. However, the development of software systems is one of the most complex endeavours that man has ever undertaken, as software by definition does not have a clearly visible structure. Because traditional SDMs are based upon the “fixed point theorem” of finite engineering project timeframes, – that is, the assumption that “there exists some point in time when everyone involved in the system knows what they want and agrees with everyone else”, – they are doomed to inevitably fail when

confronted by the reality of “living” information systems where requirements are in constant flux [12]. Therefore the Cartesian method does not hold for ISD. The “ends” are often undefined and likely to change, and any “means” that assume otherwise, such as a methodology which freezes the requirements long in advance of delivery, is likely to fail. After Schön, we mean by the **blindness** problem that whereby the “means” are fixed even though the “ends” are unclear.

Methodologies are tools to get the job done, and ought to be invisible to the task. The general trend within ISD has been an increase in the visibility of tools and methodologies. Many users of methodologies became weighed down with the intricate documentation required, instead of focusing on the actual problem at hand. We refer to this as the **methodology visibility** problem.

According to the philosophy of rationality, a perfectly rational person is one who always has a good reason for what he does. In practice of course, this is often not the case. What therefore may happen is what Parnas & Clements [11] refer to as “faking the process”, tied to the concept of accountability. In precarious environments such as characterise WIS development, it is clear why rational justification is important because a misguided action could lead to time and cost over-runs. Developers therefore follow a specified method, or at least are overtly seen to be following a specified method, as a politically astute safety mechanism that affords an excuse against failure [4]. We refer to this as the **accountability** problem.

Robinson [13] highlights another shortcoming of SDMs in practice, that of **theory above practice**. He asserts that the relationship between theory and practice has traditionally been “one of theory standing logically above practice, informing and dictating practice”. Few academics have a sufficiently thorough understanding of the context of real-world systems development. Furthermore, few of the methods they devise are adequately tested in live situations [5]. Nevertheless, academic researchers persist in developing new methods that, not surprisingly given the two previous points, are often arcane, impractical, and unworkable.

The **definition of rationality** itself is also an issue. Stolterman & Russo [15] distinguish between public and private rationality. If a method is viewed as a way to put specific concepts of rationality into practice, it is obvious that the developer will need to understand the rationale, known as “public rationale”. However, “private rationality” is the term given to the unique judgments and interpretations that a developer will make. These types of rationality may, and probably will, differ.

2.2. Against Rationalism: Theories of “Improvisation”

Striving for predictability and control is a paradox in design since the most valued and desirable characteristic of a design process is creativity. In his book *Software Creativity*, Glass [6] strongly sets forth his views that software development is a very complex problem-solving activity that requires the ultimate in creativity. In reviewing the debate between the need for discipline and formality on the one hand, and the need for creativity on the other, his considered conclusion is that software design, being part art and part science, requires both aspects and one should not suppress the other.

There is a growing body of opinion that ISD resists method and is essentially amethodical [9,1,16]. Ciborra [2] refers to an implicit process of improvisation that typifies ISD in practice:

“Improvisation is simultaneously rational and unpredictable; planned but emergent; purposeful but opaque; effective but irreflexive; discernible after the fact, but spontaneous in its manifestation”.

Externally, design activity may appear as chaotic and maybe even “slightly out of control” [3], but it is guided by the hidden rationality of skilled individuals. Of course, absolute improvisation is a potential license for anarchy, so improvised approaches should always be founded upon competent decision-making, or what Ciborra calls “smart improvisation”. This is situated action that contributes to individual and organisational effectiveness, rather than improvising merely for the sake of dispensing with formalised methods.

Although this theory of “improvisation” is more intuitively satisfactory than that of rationalism, it is not without its difficulties. While formalised methodologies can be overly visible, the opposite is the case here. It is difficult, particularly in “Web time” to record and pass on knowledge gained through iterative processes. The improvisation approach depends on continuity. Large volumes of hard-learned design lessons must be passed from one generation to the next if development processes are to become efficient. Whereas inexperienced developers can learn easily by following rational, prescriptive methods, it is more difficult to pass on knowledge if purely improvised approaches are being relied upon. We refer to this as the **transport of knowledge** problem.

Improvisation can also have a negative impact on project management. By shifting the control towards the developers, it becomes more difficult to tell exactly where a project currently stands. Predicting where an implementation is going and how long it will take to get there becomes harder to determine [10]. This is a critical issue in relation to WIS development, where time is a major constraint. This we refer to as the **shift of control** problem.

A third issue is that whereas rational approaches attempt to eliminate biased judgment or opinion, improvisation can actually encourage developers to “embrace their biases to the point that alternative views are occluded” or to “inflate the importance of their own point of view at the expense of others” [10]. We refer to this as the **bias** problem.

3. FINDINGS OF CASE STUDY

This case study is based on a WIS development project conducted by a multinational IT consulting organisation on behalf of a government department. In the interests of confidentiality, the pseudonyms ABC Consulting and GovDept shall be used to refer to the consulting organisation and the government department respectively.

The purpose of the WIS under development was to facilitate the online registration of births, marriages and deaths. The primary users of the system were registrars located at various bases nationwide, such as hospitals and government offices. However, the general public can also use the system to obtain various certificates and information.

This project was selected for a case study because it exhibited many of the classical characteristics of WIS development environments, as regards the pressures of accelerated “Web Time”, an outward extra-organisational focus, a diversity of professional disciplines in the development team, and application complexity. The project lasted six months, and the development team was quite small, with a maximum of 12 members at one stage during the final build and test. Team members “rolled on” and “rolled off” the project at various times. No specific person performed a requirements engineering role; rather, different team members elicited requirements particular to their individual area of expertise.

The methodology used was a requirements prioritisation methodology devised in-house by ABC consulting which used a 500-point marking scheme and was administered across various groups and sub-groups. Although this methodology was very specific and firmly based in the “rational” paradigm, the users often resorted to improvisation and privately made decisions on how certain aspects should or should not be implemented. As such, elements of both the methodical (rational) and amethodical (improvised) approaches were experienced. In this section of the paper, we discuss how the pitfalls of each approach, as described in Section 2, were experienced.

3.1. Pitfalls of a Methodical (Rational) Approach

3.1.1. Methodology Visibility

In this case, the GovDept was undergoing a decentralisation process, resulting in a distance of 100 miles between the client and the requirements acquisition team. Both the development team and the users found distributed collaboration very difficult. The project manager said that this is always a problem, but the users felt that the rigidity of the methodology did not ease matters and placed an unnecessary burden on them. A particular concern for users was being forced into groups, who had to then facilitate times and locations when every member of a group could attend. For example, there is one registrar in each hospital throughout the country. Therefore, a registrar group meeting forced one person from every hospital in the country to meet at the same location. The users felt that this was rather unnecessary and cumbersome, as one registrar could have adequately prioritised requirements with the same degree of accuracy.

Other users felt that the organisation of users into sub-groups was pointless, given that individual roles within these sub-groups meant that requirements would be prioritised by the most expert user, whose opinion was rarely questionable and met little resistance.

Understandably, the attitude of the users did not change for the better when various inconsistencies with the usage of the methodology came to their attention. Also, they found it hard to understand why such strict guidelines regarding attendance and marking were in place, when finalised requirements were selected in an

abstract manner and did not have a well-defined cut-off point as would be expected of any good prioritisation process.

3.1.2. Accountability

Accountability was an important factor in the methodology process. Because the client was a government body, the usual attention to detail and justification was required. This explains the visibility issues earlier outlined. Even though one registrar fulfilled the same role as many others, it was insufficient to rely on the opinion of a single user or even a few similar users. An unwillingness on behalf of the client to depend on the prioritisation process resulted in a refusal to use a strict cut-off point. They were willing to accept the results as a strong indicator of user priorities but not as a basis on which to include or dismiss requirements.

3.1.3. Theory above Practice

The methodology aimed to produce an ordered ranking of requirements. However, there were major gaps between the prescribed methodology and what was really needed in practice. According to the project manager, these gaps were only apparent in hindsight.

An analysis of the outcomes of the prioritisation process showed that many stakeholders gave the same priority to a number of different requirements. This caused a lot of problems as it became very difficult to discriminate between requirements because of clustering. When each group's prioritisation document was examined, over 80% of requirements on each document were not uniquely prioritised. For example, one group awarded 1 mark of 100, 8 marks of 50, – thus fully absorbing the quota of 500 points, – so all the remaining 118 requirements were awarded marks of 0.

The project manager felt that the detailed steps relating to the 500-point scale were insufficient. He felt that the users had trouble using “absolute values”, and that every group seemed to award marks in a different way. For example, one group prioritised 97 requirements with an average mark of 5.17. Another group only prioritised 9 requirements, giving an average mark of 55.5. This hindered the accuracy and validity of statistical analysis, as a group who rank such a low number of requirements will ensure those requirements are highly ranked in the consolidated list regardless of what other groups try to do. Although the project manager realised the fallacies of the methodology, he could not think of any practical way to improve it.

3.1.4. Definition of Rationality

There was substantial evidence of a conflict between public and private rationality, where the users' interpretation of the methodology differed from its intended use.

Each group of individuals were asked to “rate the following requirements in order of importance” using the 500-point marking system. The conflict occurred in what the individuals deemed as “important”. Some groups interpreted “important” as a requirement that was important to them or their role within the organisation. Others defined “important” as a requirement that was important to the organisation as a whole, irrespective of its direct impact on them. Legal issues was an example of the latter, where everybody knew how critical legal compliance was, even though the legal team were the only people with direct legal knowledge.

3.2. Pitfalls of an Amethodical (Improved) Approach

3.2.1. Transport of Knowledge

Users' rationale behind the ratings was not recorded for future reference. When it was found that the users were applying different rationale to their marking systems, and some were using underhand tactics, it was difficult to identify the rationale of each user and to determine exactly what went wrong, because each user's script comprised only of a list of numbers without any explanations.

3.2.2. Bias

A status meeting after the process was completed suggested that the users exhibited an element of bias when prioritising requirements, indicating a flaw in the 500-point rating system. The project manager found that a high number of users gave zero points to certain requirements, because they knew that these requirements would receive adequate marks from other users and groups. This left them with more points to distribute to the requirements that they preferred.

4. CONCLUSIONS

4.1. Pitfalls of Methodical Approaches

Because blindness only becomes a problem where the objectives change during a project, which did not happen in this case study because the objective was simply to “get an overview of stakeholders’ priorities”, it can be said not to apply in this case study. That aside, every other foreseeable pitfall of the methodical approach was experienced.

It is easy to understand why methodologies developed from theory would succumb to these pitfalls. Many researchers develop approaches without the hindsight of practical implementation, and so would be unable to evaluate such factors as the impact of human fallacies. The revelation in this study is that an in-house methodology, created and updated by practical experience alone, not directly influenced by any academic involvement, still fails to overcome these problems. The reason, according to the project manager, is that these problems occur in every project and that they are impossible to overcome.

Two conclusions can be drawn from this study of methodical pitfalls:

- Even an in-house methodology, developed from practice and subjected to repeated validation and revision from numerous projects, did not eliminate methodical pitfalls.
- Managers consider these pitfalls to be a quite natural feature in the implementation of a structured, methodical approach, and efforts to eliminate them are futile. Therefore, no effort was made.

4.2. Pitfalls of Amethodical Approaches

It is remarkable that even though the approach taken in the case study was of a highly methodical nature, two of the three pitfalls of amethodical approaches as earlier outlined were experienced.

There is evidence to suggest that the amethodical pitfalls are encountered for the same reasons methodical ones are. They are simply too difficult to eradicate entirely. When discussing these pitfalls and the reason for their occurrence, the manager again dismissed them as problems that occur as part of every project.

There is evidence to suggest that even though every attempt was made by ABC Consulting to achieve a purely methodical approach, they were trying to achieve an impossible goal. Academic research indicates that most approaches in industry lie somewhere between being purely methodical and purely amethodical in nature. There is a consensus that the advantages and disadvantages of each have driven project managers and method developers to adopt an approach that finds some middle ground between the two. In this case however, there was no conscious decision to find that middle ground. Instead, every attempt was made to adopt a purely methodical approach to such an extent that the project manager considered the approach taken in this project to be purely methodical and structured in nature. However, there are examples to show that each amethodical pitfall was encountered simply because a purely methodical approach was not achieved.

In hindsight, it is clear that amethodical tendencies arose where insufficient controls were in place. However, it is very difficult, perhaps impossible, to predict such problems and to then be able to resolve them. The conclusion to be drawn from this is that in a turbulent development environment with so many variables and the unpredictable nature of human actions, it is impossible to develop a completely methodical approach that can control every situation.

5. REFERENCES

- [1] Baskerville, R., Travis, J. & Truex, D. (1992) Systems without Method: The Impact of New Technologies on Information Systems Development Projects. In Kendall, K. E. et al. (eds), *IFIP Transactions A8, The Impact of Computer Supported Technologies on Information Systems Development*, pp. 241-269. Elsevier Science Publishers (North-Holland).
- [2] Ciborra, C. U. (1999) A Theory of Information Systems Based on Improvisation. In Currie, W. L. & Galliers, B. (eds), *Rethinking Management Information Systems*, pp. 136-155. Oxford University Press.
- [3] Cusumano, M. A. & Yoffie, D. B. (1998) *Competing on Internet Time / Lessons from Netscape and Its Battle with Microsoft*. New York. The Free Press.

- [4] Fitzgerald, B. (1997) The Use of Systems Development Methodologies in Practice: A Field Study. *Information Systems Journal*. **7**(3), 201-212.
- [5] Fitzgerald, G. (1991) Validating New Information Systems Techniques: A Retrospective Analysis. In Nissen, H.-E. et al. (eds), *Information Systems Research: Contemporary Approaches and Emergent Traditions*, pp. 657-672. Elsevier Science Publishers B.V. (North-Holland).
- [6] Glass, R. L. (1995) *Software Creativity*. Englewood Cliffs, NJ. Prentice-Hall.
- [7] Hardy, C. J., Thompson, J. B. & Edwards, H. M. (1995) The use, limitations and customization of structured systems development methods in the United Kingdom. *Information and Software Technology*. **37**(9), 467-477.
- [8] Hooks, I. & Fellows, L. (1998) A Case for Priority - Classifying Requirements. In *Proceedings of International Council on Systems Engineering 8th Annual Symposium*, Vancouver, British Columbia, Canada, July 26-30 1998.
- [9] Introna, L. D. & Whitley, E. A. (1997) Against method-ism: exploring the limits of method. *Information Technology & People*. **10**(1), 31-45.
- [10] McPhee, K. (1997) *Design theory and software design*. Technical Report TR-96-26. Department of Computing Science, University of Alberta, Edmonton, Canada. May.
- [11] Parnas, D. L. & Clements, P. C. (1986) A Rational Design Process: How and Why to Fake it. *IEEE Transactions on Software Engineering*. **12**(2), 251-257.
- [12] Paul, R. J. (1994) Why Users Cannot 'Get What They Want'. *International Journal of Manufacturing Systems Design*. **1**(4), 389-394.
- [13] Robinson, H. (2001) Reflecting on Research and Practice. *IEEE Software*. **18**(1), 110-112.
- [14] Schön, D. A. (1984) *The reflective practitioner : how professionals think in action*. New York. Basic Books.
- [15] Stolterman, E. & Russo, N. (1997) The Paradox of Information Systems Methods: Public and Private Rationality. In *Proceedings of the 5th British Computer Society Conference on Information System Methodologies*, Lancaster, England.
- [16] Suchman, L. A. (1987) *Plans and Situated Actions*. Cambridge University Press.