Software and the Social Production of Disorder

Jonathan Marshall
Faculty of Arts and Social Sciences
University of Technology, Sydney
jon.marshall@uts.edu.au

Didar Zowghi
Faculty of Engineering and IT
University of Technology, Sydney
didar.zowghi@uts.edu.au

Abstract

Software development is inherently an ordering process. When implemented in a workplace it orders the ways that people go about their work, the work they do, and the ways they interact and communicate with each other. This new mode of ordering may conflict with existing orders, existing distributions of power and knowledge, and arrangements of, and between, groups. Ordering is almost always the subject of dispute, so software development can easily become enmeshed in the politicking between competing groups with deleterious effects. Removing all these conflicts may not be possible, as they can be an essential part of the ways relevant groups interact. Better communication, for example, may actually increase conflict, and not produce harmony. Rather than thinking of order and disorder as mutually exclusive polarities, it is more effective and realistic to think of them as constituting an 'order/disorder complex' and to expect disorder to appear alongside the ordering. This paper explores the problems of ordering and disordering through a study of changes in the Australian Customs' "Integrated Cargo System". We suggest that acceptance of some untidied mess, or openness to dispute and unclarity may be useful in implementing functional software.

1. Introduction

Software development is a social and political act, whether those engaged in it want it to be or not, because it:

a) organizes and distributes work,

b) organizes the communications between groups of people,

c) can change the balance of power between groups, upsetting status relations or opening them to new struggles, and

d) tends to impose one model of how things are done, when different groups have different models, and may want different outcomes to preserve different types of social and reward arrangements.

Software engineers may not find it possible to fully address these conflicting demands, no matter how good the software development process is, as conflict can be an inevitable part of the way work is conducted. Closer communication between people or groups may not result in harmony but can magnify the dispute, generating polarized positions which separate people further the more the communication continues and the better it gets.

Allocating blame for failure is also part of the ongoing politics between groups especially if frictions were exacerbated during the software implementation. Allocation of blame may reflect distributions of power and popularity, or involve ritual scapegoating; the point of which is to purge the mess and locate it elsewhere. This can normalize power relations, assuage people's consciences, and restore harmony to a group troubled by dissention or failure [1]. As a result, reports investigating software failures can continue the process of producing disorder.

We illustrate these issues with a summary of our documentary investigation of the Australian Customs' "Integrated Cargo System" (ICS) installation. The ICS was declared to be possibly "the world's first fully integrated imports and exports system", combining "into one coherent and technologically modern system, the numerous cargo systems that have been developed over the past 30 years". It was to be paperless and use a single window [2]. We show how the politics surrounding the implementation and installation of the ICS contributed to the production of disorder and speculate if anything could have been done to make that disorder less disruptive. The main contribution of this paper is to increase our understanding of the interplay between social order and disorder in software development.

2. Order and disorder in software

We assume, following Marshall's study of Internet communication [3], that disorder is important in its own

right, and that the means of making social order can also be disruptive of that order.

This is relevant as software systems fail regularly. In 2003 Saran wrote that: "In a survey of 450 IT directors across the UK, Germany and France, 73% said they had suffered major faults in their IT systems…. lack of quality in software had a direct impact on their business. Thirty six per cent reported that IT failures had led to 'considerable reduction in turnover', and 43% said poor software quality led to a substantial drop in staff productivity. Forty five per cent said poor software quality had damaged the company's image among clients and prospective clients" [4]. In 2008 El-Emam and Koru reported from two years of web surveys that "26 percent to 34 percent of IT projects fail" [5].

That such failures should be consistently reported after 60 years of software development suggests there is a major gap in our understanding of software and organizations, or that software failure and disorder is not an aberration, but an essential part of development and should be treated as such.

Western societies have a metaphysical bias which labels order as 'good' and disorder as 'bad' and to be corrected or eliminated. Despite disorder always being present, it tends to be dismissed as error, as a transitory phase between ordered states, as a negligible random effect, or as pathology. In ordinary life the remedy proposed for many problems is greater organization, while we ignore the potential costs of neatness [6]. For example, is the endless managerial restructuring or tidying of organisations actually useful?

Ways of creating what appears to be order to some people can create what appears to be disorder to other groups of people. The practical consequence of this is that we cannot always give a definition of disorder. What appears ordered or disordered depends upon who does the reporting. However, as disorder has so many more ways of occurring than order does, different people may agree on what constitutes disorder, but find it much harder to agree whether something is ordered correctly or effectively [7].

Furthermore, what some people consider to be disorder may be useful to the organisation. What some call creativity can be experienced by others as vandalism [8]. So if an organization is going to change, adapt or generate new ideas (including implement a new software system), then there will likely be disagreement or conflict.

Disorder does not only arise from failure, it may equally result from success. Success in improving computer technology has led to a massive increase in the noxious garbage resulting from superseded machines [9]. Success with oil and coal technologies helps foster climate change. Technologically generated disorder does not just appear with computing.

Rather than assuming that order and disorder are mutually exclusive poles, or that we can achieve perfect order, it is more useful to think in terms of a politically based order/disorder complex. This approach accepts that a *degree* of mess, conflict, redundancy, misalignment and misfit can keep the system resilient and functioning, and that we cannot require everyone involved agree on the nature of order, and that we ourselves may be mistaken in our view of order.

3. Software and social life

The major determinates of social life include the structures of communication, the structures of interaction, the distribution of labour and the environment within which action occurs (e.g. [3], [10], [11], [12]), therefore, as software intrinsically structures communication, work and action, and provides the functional environment for these activities, it affects social organization, and new software produces conflict.

Even software that only supports a single person's tasks can affect the organisation. Word processing software, for example, eliminated the typing pool, and made it compulsory for most 'white collar' workers to type and organize documents. In so doing, channels of gossip and socializing were removed, thus altering patterns of information exchange throughout organisations everywhere. While we may approve of the results, this new order disrupted old orders and displaced people from patterns of work and life they were used to. It is doubtful that those who introduced word processors intended this result. It is always possible that software installations may threaten security of work and the status that comes with having recognized skills, and thus provoke resistance.

Changing software can also change the arrangement of hierarchy in an organization, although we need not assume that this will necessarily make organizations more democratic or open as argued by Friedman and others [13]. A change in hierarchy could prevent lower levels of the hierarchy from hiding their considered response to local conditions, against the directives of central managers, with possibly disastrous results for the organization. If the power in an organisation is pathological, software can extend that pathology to places it was previously unable to reach.

Software can also distribute power through a system. This also may not make the organization more democratic or responsive, but simply make it unclear to everyone where power and responsibility lies, so that less gets done and problems are not dealt with.

4. Software and worldview

Software almost inevitably involves the application, or enforcement, of a worldview (or model of the world), of work, of ideal organisation, of the way that items and events are constituted, interconnected or separated. As different groups often have different models, models are rarely neutral, and are often drivers of social conflict. Software models may be based on how only one sector of the organization (usually management, or perhaps the requirements analyst) thinks the organization works. Paraphrasing the famous remark of George Box and Norman Draper, we can say that not only all models may be wrong, but they may be more useful to some people than to others [14].

Some of these modelling problems arise because software uses categories, conceptual or data models, to represent the world, its entities and their relationships. In general, software categories are considered static and limited in number during run time, while human cognitive categories are both changeable and flexible in what they include, depending on context for their resolution. Old categories are constantly modified and new categories invented for new situations. It is difficult to know in advance how categories will need to change in response to changes in the environment (perhaps produced by the software), or to new discoveries. Software can therefore enshrine procedures that no longer work and render the users less able to respond to unpredicted or unclassifiable events [3].

In general, technology is intended to simplify reality and therefore is likely to clash with the complexities of what it organizes. Technology can:

a) Break, or circumvent, previously existing links between factors. What constitutes a significant 'factor' may not be apparent until the technology is installed and the break is made.

b) Add unexpected links between factors.

c) Create new factors.

d) Add complexity, as when extra data is collected because it can be.

e) Make it harder to carry out previous actions.

Through the way it orders, technology can produce what appears to be disorder. There is no certainty we can predict these difficulties before observing the interaction between the technology and the world it organizes. Neither can we completely specify what strategies software users will discover and exploit for their own advantage.

5. Methodology

These assumptions about the order/disorder complex, the importance of software in structuring work and communication, the relevance (and frequent inadequacy) of models of the world, the additions and subtractions of complexities and links, and the connection of all of these factors to social politics, were deployed in a study of the Australian Customs Integrated Cargo System (ICS). Our data was gathered from newspapers, magazines, press releases, parliamentary debates, official reports and online sources found through Google searches. We accumulated over 800 A4 single-spaced pages of documentation.

Accuracy of our description is limited by the accuracy and bias of the sources. Sources reflect the aims of particular groups or their attempts to articulate their worldview. However, these inaccuracies and the politics of the reporting express the political divergences between the groups involved, and thus count as data themselves. There is no particular reason for assuming an interview subject would be more accurate. The divergence between what people say, or believe, they do and what they actually do is well known in social theory.

Analysis was conducted qualitatively, as the data collected did not lend itself to the rigor of statistical analysis. A more detailed history of the ICS installation and the related issues, using this data, may be found in [15].

6. Troubles in the ICS

In October 2005 the Australian Customs Service's Integrated Cargo System came online, several years later than expected with a cost increase from the initial estimate of Aus$30 million to Aus$212.7 million [16]. Many users said the system was bug-ridden, slower and more complex than the systems it replaced. Some people resorted to pen and paper to process their cargo, and cargo piled up on the docks during the 'Christmas rush' period [15]. In June 2007 newspaper reports claimed customs was still dealing with some of the 568 claims for compensation [17]. In 2008 adjustments to the software were still being reported.

In this section we present some inferences drawn from the analysis of available data for the ICS installation.

6.1 Group backgrounds

We identified different groups involved in the software installation, and their *possible* conflicts. This included:

a) *Software companies* with conflicting agendas, differing workloads, differing organizational structures, mutual secrets, and an interest in locking in contracts and making it hard for them to be replaced.

b) *Computer companies*, wishing to make the installation safe for their machines, and to lock in contracts.

Software engineers are not devoid of political positions and ambitions, and these may impact on the project. They may, for example, think that they know better than the people 'on the ground' how the software should structure work, or be keen to use existing software. Programmers may have restrictions placed on their ability to communicate with programmers from other companies, or on describing how their software works.

c) *Sales people*, aiming to sell systems rather than represent their capabilities accurately.

d) *Customs Brokers* who can be divided between those affiliated with large retail or other companies, and those contracted with smaller importers who acted as middlemen between them and the Customs department.

e) *Competing businesses* who aimed to take business advantage of the situation.

f) *Customs management* who are responsible to the Government and hence to a political party, and who according to the ideology of the business groups will be ignorant of business procedure and thus automatically at fault.

g) *Politicians*, who will oppose or defend anything the government or its instruments do, and look after the interests of important constituents. Difficulties that arise are likely to become public political issues.

Various federations represented the differing groups, such as the 'Customs Brokers and Forwarders Council of Australia', the 'Australian Chamber of Commerce and Industry', the 'Australian Air Transport Association', the 'Australian Federation of International Forwarders' and the 'Australian Exporters and Importers' Association', all with varying interests, elected officials, budgets, internal power blocks, and some overlap.

The installation intensified the possibility of conflict between, and within, all these groups. They would, almost inevitably, have differences about what constitutes good order, whatever the success of the project. Bringing these groups into better contact did not help diminish the conflicts. Any failures were likely to involve public dispute, which increased separation, although making large amounts of information available.

6.2.Changing structures of communication

Changing the structures of communication, leads to the possibility of changing the relationships between groups. A possible goal of the software design was to eliminate 'middleman' small brokers [18]. Large companies did their own brokerage, while the new system allowed smaller companies to communicate directly with Customs via the Internet. Small brokers disputed this potential removal of income, and could not feel that Customs had communicated with them.

ICS changed the ways that brokers could communicate with each other, or communicated accidentally. At one stage it was reported that brokers could occasionally see each other's invoices [19]. Although rectified quickly this created much mutual suspicion and provided a handy tool to berate Customs with, given that 'security' had become one of the focuses of the installation.

Customs also tried to co-ordinate communication between the parties through itself. This in theory should have given Customs greater awareness of problems and a quicker response time. However, it added delays, made it harder for parties to communicate easily and meant that if there was anything groups wanted to hide from Customs (if temporarily while they found out how serious the problem was), then it would be less likely to be communicated to anyone. This could make small problems cascade into large problems and increase areas of mutual ignorance. Attempting to introduce control and order into communication almost certainly had the contrary result.

New patterns of communication may have distributed power in new ways, making it hard to allocate responsibility for particular parts of the operation. The Audit Office report [16] suggests that people in Customs were unsure of their responsibilities, or unable to enforce them, particularly as the relationship between parts of that organisation changed. High-level Customs management frequently appointed people to newly created positions after the problems became visible.

6.3 Corporate software conflicts

Differing software or computer companies wanted to keep the software, or the machines used, in-house. When the original IT service management company EDS was replaced on part of the system, the replacement company ported everything onto its computers and its favoured operating system, with some difficulty and disorder. At the same time, EDS withdrew staff previously embedded in the Customs Department, leaving Customs with fewer staff who understood the technical and planning side of the operation. Local knowledge was lost and Customs

was penalised for attempting integrated ties with a company. Again, ordering produced vulnerability [16].

Splitting work between different software vendors and programmers seemed efficient, as they were chosen by price or specialty, but the businesses had little interest in coordinating activities efficiently to make the others look good, so a loss in co-ordination was inevitable. Further loss in coordination arose as brokers and importers had to contract their own software vendors or programmers to write the interface between their existing machines and software, and the ICS [15].

Another political difficulty arose as the older independently developed 'Tradegate' system was to be phased out, yet was indispensable for cargo clearance until the ICS was fully functional. As the managers of Tradegate implied, there was little incentive for them to invest in keeping Tradegate working smoothly. Conflict also arose over whether importers should go back to using Tradegate when the ICS failed. Officially Customs refused to risk exceeding the contract termination period. There were, however, persistent rumours that this decision was influenced by the large importers wanting to add stress to the lives of smaller importers [15]. If true, this is an example of disorder being maintained for political effect.

6.4 Deadlines

As the software industry seems to expect that deadlines will be broken and software will not work perfectly at first, then people will delay their own implementation until the software they are integrating with is ready. Some brokers did not get their interfaces ready on time, because of this expectation, and because they felt they could not write or test their software until the software they had to integrate with was running [20]. The mutual interdependence of software implementation, together with expectations about delay, means that when different vendors are involved there will nearly always be disputes over the timelines.

Deadlines are gripped in paradox. Trying to meet them causes pressures which may lead to programming failure, poor software quality and insufficient testing, yet without them it is impossible to co-ordinate, manage or complete the software construction.

6.5.Miscellaneous conflicts

Conflicts arose over the cost of the services with big importers resisting changes which allowed smaller importers to pay less for customs brokerage. Large importers may have seen commercial advantage in pressing for the changes to go ahead to cause difficulties for less well resourced small companies, thus potentially removing competitors and reducing competition, at a major 'make or break' time of year [15].

Brokers and software companies clashed over whether the problems were arising because of bug ridden software or incompetent users.

Conflict was displaced into Parliament, as problems were taken as 'showing' the inefficiency of the Federal government [15]. State governments joined this attack. In response the Federal Minister, although having a great many meetings with interested parties, tended to assert that things were going relatively smoothly and, during the crisis period, put the responsibility for failure onto the users; an approach condemned by small users in particular.

6.6 Intensification of precision

The ICS installation also created disorder through its capacity to provide more data to Customs and to the government in general. The new Cargo System was seen as a way to improve Australian border security and, post September 11 2001, of protecting Australia against terrorism. This demand added to the complexity of the original specifications, and led to conflicts with brokers and importers over the degree of precision required. Categories for imports were multiplied and numbers representing each category where enlarged. During entry, categories had to be checked; new ambiguities crept in, old ambiguities in categories were not acceptable, and the system rejected mistaken numbers demanding complete re-entry. This put greater pressure on workers, took more time, and old solutions to category problems no longer worked [19]. New staff had to be hired and trained, adding confusion. The extra detail did not match many brokers' accounting systems [21]. Improving precision for security overwhelmed other requirements.

6.7 Compounding

Trivial incidents can compound and become disruptive. Adding Internet connection meant that when the ICS began to fail, the server was overwhelmed with people trying to access, and retrying to access, the online help desk. Attempts to enter customs data was stymied by the added delay; people didn't know if data had been received, and some data did drop out due to the overload, having to be re-entered. Time delays and drop outs, compounded on top of each other.

Similarly, delays on the docks meant that new cargo arrived and blocked access to old cargo, further delaying

attempts to remove the cargo even when it had been cleared. Queues of trucks at the cargo gates blocked access for trucks whose cargo was cleared, adding to disorder.

Other parts of the computer system proved unable to cope with larger than expected demand on computer time, even when the software was purchased off-the-shelf and had a good reputation for the tasks that Customs had set.

Failure interacts with failure and increases the effects.

7. Official reports and recommendations

The main analyses of the ICS software problems were firstly the Booz Allen Hamilton Report (BAHR) [21] and later the Audit Office Report which supported the earlier report's conclusions [16]. They both blamed bad management, as is usual in post software disaster reports; leading to the problem of why it never gets any better. Although hindsight shows obvious problems, Customs did carry out what would have normally been considered good management procedures. People did set deadlines, and have timelines. They were made 'accountable', they did have 'performance indicators', there were endless meetings, and they attempted 'proper budgets'. None of these appeared to help, and it is doubtful more management would have completely solved these problems.

Management theory is an interpretive art, and it is always possible to find an opposite action which might have worked. Customs was blamed by BAHR for not altering business protocols, but had they altered protocols and the software failed then they would probably have been blamed for interfering with the protocols. BAHR castigated Customs for not using a clear pyramidical administration structure, but if they had, then Customs could have been blamed for not using a more responsive distributed system. Customs was blamed for contracting out programming, but that could have been recommended had they kept programming in-house. 'Best practice' in software development or project management under these circumstances is not well defined nor understood in the literature or industry. This leaves room for speculation and unsubstantiated claims.

8. What can be done?

If we accept that:

a) the social world is not just ordered or orderable but comes as an order/disorder complex,

b) different groups and people often see order and disorder differently,

c) knowledge is incomplete and biased, and

d) software structures social interaction,

then we can improve our understanding of the interplay between order and disorder during software development and installation and see Information Technology (IT) and software engineering in a new light.

Firstly, software installation will been seen as part of organisational politics, as people try to advance their position, or resist the advances of others, and engineers will find themselves being exploited in the disputes. Software installation affects different people differently but, in any case, changing the organisation of communication and the tasks people are able, or compelled to do, will produce disruption and resistance, and may interfere with the organisation's work.

Secondly, by accepting that the order that software may attempt to impose will often generate the appearance of disorder, software engineers can leave space for this disorder to be dealt with, or incorporated into the system as it manifests. Disorder can 'tell' us that what we are attempting is not viable, so suppressing disorder can be dangerous. It may not be possible to completely plan a software project at the outset, and implementers may get the best results if they can engage in coordinated redesign as the software related problems emerge and unfold during development.

Thirdly, the models of the world incorporated into software, can be recognized as political models with political consequences. Models by their very nature are always incomplete, no matter how well the requirements elicitation activities are conducted and how many stakeholders may be interviewed. However, the fewer people in different positions, whose views are taken into account, the more likely that the model will be partial. Even so, much of what people do will probably not be consciously expressible, and thus we recommend that implementation be preceded by ethnographic observations of what people actually do in their work place, as recommended elsewhere in the literature [22,23]. In particular, we recommend observation of how people interact with each other, and the collaborations, competitions and factions that arise in that interaction. Sometimes groups need to establish boundaries between each other. If the software violates these boundaries then it can produce resentments and undermine the work.

By recognising incomplete knowledge, engineers can expect categories, conceptual or data models to change, as all relevant categories for all future actions cannot be anticipated in advance [24]. Ease of use of the category schema, and whether it serves all the purposes expected of it, should be investigated.

Fourthly, while technology aims to simplify, it can add complexity, making connections between previously separated factors (human and material) or rendering simpler ways of doing things difficult. The main activities carried out by people should become easier to perform and not buried under hoards of options. Trivial problems can compound, so no problem is inherently trivial; the question is 'will failure cascade?'

At the beginning of the Customs installation, we would have recommended investigating and listing all the different groups of stakeholders involved, noting their potential rivalries, and how extraneous groups could get involved. What the installers did about potential rivalries would involve a moral and political stance, whether they wanted or not. However, without expecting that dispute can be bypassed, it could be useful to have a recognized and comparatively neutral mechanism whereby groups who feel they are being badly done by can be heard and pass on neglected information. These views may need to be gathered, rather than waited for, as workplace politics can mean that workers deliberately hide their actual work from managers; ignoring this problem is ignoring essential data.

More work needed to be done in investigating the categories deployed by the programs to check these were relevant to users. Intense demands for order failed to recognise that categorizing objects might be difficult, and gave people little room to correct mistakes. The project needed to expect category disorder. Changing categories over a short time frame may put unreasonable demands on human memory, which generally learns categories gradually through use.

Deadlines and budget proved a problem throughout the project, and compounded through expectations. Legislating deadlines, or legislating budgets (had the latter been done) only advertises failure. When the work is new, deadlines need modification as the job progresses. Deadlines could become political and competing groups will attempt to gain advantage from them.

As complete clarity is not possible, it is important to expect mess, and to be prepared for *expected* disorder such as: deadlines being broken, categories proving insufficient, models of the workplace and work being incomplete or incompatible, and rivalries between various groups with people exploiting the software for their relative benefit. Recognition of these disorders will give the installation greater resilience. If everything is perfectly within specifications then the system will have no slack or adaptability. Disorder cannot be eliminated but can be tolerated or even become usefully recognised.

9. Acknowledgements

The authors acknowledge the Australian Research Council for supporting the project from which this paper emerges. Jon Marshall would like to acknowledge the comments of Luke Kendall on an earlier version of the paper.

10. References

[1] Girard, R. *The Scapegoat*, Athalone Press, 1986.
[2] Australian Customs Service, *Annual Report, 2004-5*.
[3] Marshall, J.P. *Living on Cybermind: Categories, communication and control*, Peter Lang, NY, 2007.
[4] Saran, C. "Software failures damage business", *Computer Weekly*, 5/27/2003: 5.
[5] El Emam, K. & Koru, A.G. "A Replicated Survey of IT Software Project Failures", *IEEE Software*, 25(5), pp. 84-90.
[6] Abrahamson, E. "Disorganization Theory and Disorganizational Behavior: Towards an Etiology of Messes", *Research in Organizational Behavior* 24, 2002, pp. 139-80.
[7] Bateson, G. *Steps to an Ecology of Mind*, Chandler, San Francisco, 1972.
[8] Peckham, M. *Explanation and Power*, The Seabury Press 1979.
[9] Grossman, E. *High Tech Trash: Digital Devices, Hidden Toxics and Human Health*, Shearwater Books, 2006.
[10] Durkheim, E. *The Division of Labor*, Macmillan, NY, 1933.
[11] Radcliffe Brown, A.R. *Structure and Function in Primitive Society*, Cohen & West, London, 1952.
[12] Harris, M. *Our Kind: who we are, where we came from & where we are going*, Harper & Row, NY, 1989.
[13] Friedman, TL. *The World Is Flat: A Brief History of the Twenty-first Century*, Farrar, Straus and Giroux, NY, 2005.
[14] Originally "Essentially, all models are wrong, but some are useful" in Box, G.E.P. & Draper, N.R. *Empirical Model-Building and Response Surfaces*. Wiley 1987.
[15] Marshall, J.P. "Information Technology, Disruption and Disorder: Australian Customs and IT". http://uts.academia.edu/jonmarshall/attachment/515173/full/Information-Technology--Disruption-and-Disorder--Australian-Customs-and-IT
[16] Australian National Audit Office, "Customs' Cargo Management Re-engineering Project," *Audit Report No.24* 2006–07. http://www.anao.gov.au/uploads/documents/2006-07_Audit_Report_24.pdf
[17] "Customs fiasco claims hit $12.3m" *The Australian*, Tuesday, June 19, 2007.
[18] Bajkowski, J. "Customs issues blunt warning over cargo system", *ComputerWorld*, 27 September 2005.
[19] Bajkowski, J. "Enterprise import data exposed by Customs system", *ComputerWorld*, 20 October, 2005. http://www.computerworld.com.au/article/142673/enterprise_import_data_exposed_by_customs_system
[20] Paul Zalai of the Customs Brokers and Freight Forwarders Council of Australia in the *Australian Financial Review*, 25 October 2005.

[21] Booz, Allen, Hamilton. “Final Report: Review of the Integrated Cargo System”, 2006. http://www.customs.gov.au/webdata/resources/files/boozallenhamilton_icsreport.pdf

[22] J. Hughes, J. O'Brien, T. Rodden, M. Rouncefield, I. Sommerville, “Presenting ethnography in the requirements process”, *Second IEEE International Symposium on Requirements Engineering* (RE'95), pp 27, 1995.

[23] S. Viller, I. Sommerville, “Social Analysis in the Requirements Engineering Process: From Ethnography to Method”, *Fourth IEEE International Symposium on Requirements Engineering* (RE'99), pp 6, 1999.

[24] Zowghi D., Gervasi V., “On the Interplay Between Consistency, Completeness, and Correctness in Requirements Evolution”, *Journal of Information and Software Technology, Volume* 46 (11), pp. 763-779, 2004.