

Making sense of Design & Requirements Perspectives - & their Inter-relations

Liam J. Bannon
Interaction Design Centre
University of Limerick
Ireland

18th March 2007
DRAFT 1.0

Preamble

This NSF Design Requirements Workshop has a very ambitious and wide-ranging set of objectives. It invites contributions from a wide range of disciplines and encourages a variety of perspectives on requirements and design, as applied to socio-technical systems. In attempting to position myself in relation to the Workshop themes and objectives as outlined in the invitation letter, I have extracted certain key sentences from the letter to frame my remarks here. The organizers ask for ideas on “how to address the diverse and multidisciplinary realities of software intensive designs in the 21st Century”, and wish to include social and organizational, as well as technical perspectives. They also request us to explore “the envisioned future of requirements in a variety of design contexts” and the issues involved in “managing design requirements in heterogeneous and rapidly-changing environments”. They are also open to “the ways in which modern systems design can be informed by ideas from a range design fields (e.g., engineering, architecture, economics, and the arts).” I have highlighted these excerpts as they relate most closely to some of my own concerns and expertise. Similarly, I have pulled out 2 of the 8 listed objectives as ones on which I might be able to make a contribution at the Workshop, namely: *Elaborating the need for new avenues in requirements research* – the new landscape and emergent challenges; and *Promoting intellectual cross-fertilization across disciplines in design and requirements*

Thinking about the nature of “requirements”

There are a range of perspectives on the nature of “requirements” as evidenced by the different languages used in discussing the concept. So, for some people, systems design begins with the need for “requirements capture” - which to me inspires an image of requirements as well-defined entities just waiting to be plucked from the environment. It goes without saying that this particular viewpoint is less widely held today than heretofore. A less extreme view, yet one which is still quite popular in the engineering community is the notion of requirements “gathering”, which again has an implicit if not explicit conception of requirements as things that are waiting to be harvested. Continuing on this line, one can hear discussion of requirements “elicitation” which begins to acknowledge that requirements as such may not exist in some independent objective sense, and that even in consultation with users, requirements may not be immediately apparent, or accessible, and it may require some effort to “bring forth” these requirements from the user community. Going one step further, we can argue that requirements are not “out there” awaiting collection, but are themselves constructions, jointly and severally produced by a range of actors, including users and analysts and developers in specific contexts. This view

thus requires that we pay close attention to the ways in which we investigate the use situation and work context, and takes into account the social, political and economic factors involved in the requirements process.

Despite these diverse views on the nature of requirements, it is still the case that in most projects, there is still the need for a requirements document that contains the statement of the requirements for the system, which must be signed off by the end user organization, and is often the basis for the legal contract between the client and the design organization. Indeed, in many contract situations, especially in situations where competitive bidding is required, as it is by EU law in many instances nowadays, one has the situation where often the people who “generate” the requirements which are the basis for the tender are not the people who will be involved in the contract. While there are many reasons for having such a procedure, in terms of transparency and accountability, it must be noted that such contracts militate against a more fluid conception of requirements which is what we are arguing for here.

A number of commentators have noted how requirements, viewed as texts can impede a good design process. The designer Chris Jones (1988) notes: “Both parties (designers and clients) have to give up the use of the requirements as a semi-legal basis for control and measurement and agree to work together in the continuous meta-process of evolving the design brief and sharing in the eventual decision as to how the problem is to be seen and solved.” Continuing, he notes: “Functions, statements of requirements, are essential but temporary. Without them we cannot begin, but unless we can change them we cannot finish, cannot discover..” Jones concludes with the claim that “[we must] recognize that the ‘right’ requirements are in principle unknowable by users, customers, or designers at the start.” This position calls into question the nature of most formal software development contracts today. Similarly, the consultant Tom Gilb (1990) stresses the need to focus on process, not method or static product. He notes that current development methodologies “...are based on a static product model. They do not adequately consider our work to be a continuous process—derived from the past and being maintained into the future.” Yet another voice in support of this shift, coming from academic software engineering, is that of Floyd (1987). She argues for more emphasis on the process of software development than on the efficiency of the resulting code: “The product-oriented perspective regards software as a product standing on its own, consisting of a set of programs and related defining texts... considers the usage context of the product to be fixed and well understood, thus allowing software requirements to be determined in advance,” while the process-oriented perspective “views software in connection with human learning, work and communication, taking place in an evolving world with changing needs... the actual product is perceived as emerging from the totality of interleaved processes of analysis, design, implementation, evaluation and feedback, carried out by different groups of people involved in system development in various roles.” This shift in perspective calls into question the separability of development and use and the concept of a “complete” requirements specification. A static view of the user organization is behind the assumption that a task analysis is a sufficient basis for design, i.e. once uncovered, the tasks are assumed to remain the same throughout the design process. In opposition to this stands again the human activity approach where it is a fundamental belief that we cannot design the future totally, and that design is a learning and change process for *all* the involved parties, both designers and users. Again, Gilb (1990) admonishes software developers to accept that requirements are always changing and to focus on handling this situation by catching problems early through user involvement in the design and specification phase. He

also stresses the need for prototyping and frequent iteration. In recent years, more fluid, and flexible approaches to all aspects of systems development have come to the fore. The wide takeup of “agile” approaches to systems design and implementation have radically re-structured aspects of traditional requirements and specifications activities. There is much more realization on the part of both clients and providers, that requirements are not fixed, and that there needs to be on-going flexibility in the development phases. This has led to more open and innovative practices in this area.

For a number of important applications and software installations, the whole idea of going through a requirements phase as traditionally envisaged, is no longer sustainable. Rather, in many more open design contexts, there is a much more fluid movement between discussions with stakeholders and possible design approaches and even solutions, without there being any formal requirements document being produced. Rather, the emphasis is on close collaboration and open communication between the interested parties, with users/stakeholders in some cases trusting to the competence of the developers to understand their concerns and produce relevant design solutions, which can then be examined and approved or rejected. Thus it would appear that discussing where the requirements field is heading must take into account these radically different environments where software-intensive systems are being developed. For example, in safety-critical applications, adherence to tried and trusted procedures, working through checklists, documenting fully agreed requirements and decisions in the early stages would appear to be mandatory and eminently desirable. On the other hand, an organization requiring an innovative web design for their intranet or WWW presence might be much more open to leaving the whole “requirements” activity at a very general level, trusting in the expertise and competence of the software provider to come up with design and client solutions that can be tried out and accepted based on prototypes, without the need for extensive formal documentation in the early phases.

Likewise, the increasing use of off-the shelf software products or components has had a significant impact on the ways software developers work. Much software work involves creating software products, their work now resembles a *bricoleur*, adapting and fitting various existing software applications or modules to work together and attempting to fit the resulting product to the use situation, which again makes little use of the more traditional phase-oriented software development process, moving from a requirements phase, through specification to implementation. A paper by Potts (1995) offers some perceptive observations: “The requirements engineering community concentrates on requirements disambiguation and correctness. But attributes like time to market and usability matter more for off-the-shelf products.Off the shelf product requirements are not elicited or played back to the customer; they are proposed, invented or designed”. Potts (1995) notes several key features of this change: “developers of off-the shelf software produce requirements through a process of collective and incremental design...requirements for a new package depend more on the marketability of named features and feature comparisons with competing products than on “elicitation” of requirements from “ the customer”.....designers of such artifacts do think about user needs, and may even undertake some task analysis, but they do not do what we call “requirements engineering”.

Another change in recent years has been in how we attempt to understand the work context. The attempt to more adequately “represent” user requirements has led to an

opening within requirements engineering to ethnographic approaches to understanding the setting and user needs. However, this approach, while opening up some new vistas, is not without its own problems. There is an issue as to whether the developing relations between such unlikely bedfellows as technical systems developers and social scientists, particularly ethnographers, and more narrowly ethnomethodological ethnographers, should be seen as a virtuous coupling or a “deadly embrace”. While I am in favor of any and all approaches to requirements that open-up this phase to a richer appreciation of the work context and work practices of people, I also feel that this recent courtship between developers and sociologists may turn sour due to a misalignment of motives and interests. If we are to have a useful interplay between these two professions then perhaps we also need to be aware of their different agendas, so as to reduce confusions and misunderstandings. Indeed at some gatherings where these two groups have been in evidence, signs of these misunderstandings and frustrations were palpable. The developer may be hoping that the sociologist can (a) help produce better “requirements” and possibly (b) obviate the need for the developers to enter the field and be subjected to the messiness therein. The sociologist, or more particularly, in the context of our current discussion, the ethnographer, is pleased to be provided with resources and a chance to investigate an interesting work setting, and can provide copious field notes and interpretations of what they observe. Sommerville et al (1992) see sociologists as the “user’s champion”, and also “the designer’s conscience”, a position which I feel is not warranted and unhelpful. As argued eloquently by Button(1993) and Shapiro (1994) the descriptive language and sociologically-generated analytical categories constructed in ethnographic studies are likely to be of little relevance to the practical problem of designing computer systems. Those who attempt to show explicitly the relevance of their research, may find that in the process of translating their detailed accounts into more formal requirements, the richness and significance of their work gets lost, distorted or misconstrued. But if researchers find it problematic reconceptualising their findings, what is it like for designers and consultants (whose job it is to implement new technology and redesign work) to translate descriptions of "the sociality of work" into the language of design and workflow procedures? The paper by Bob Anderson (1994) puts forward one trenchant view on the current confusions. While this is not the place to fully engage in detailed discussions on his arguments, a few points need to be made here. Certainly, it is a good thing that developers learn more about how to do field work, interviewing, observing, documenting, etc. And this should help them in drawing up their requirements. But, as Anderson points out, this does not exemplify what ethnography is really about. The particular analytic form of reportage characteristic of ethnography is not necessarily of direct relevance to the requirements gathering process *per se*. Also, the notion that this turn to the social can produce yet another “method” in the armory of systems analysis and development methodologies is mistaken. Just as there has been a misconception about the nature of the work done by people in participative design, so the same misunderstanding is apparent here. What ethnographers are about is NOT simply some method for data collection, and viewing it solely in this light is the source of some of the current confusions. If both sides in the courtship are clearer about their interests and concerns, however, then, as Anderson outlines, there is the possibility of opening up the design discourse, at a “higher” level, to different “sensibilities of design”: “The contribution that ethnography may make is to enable designers to question the taken-for-granted assumptions embedded in the conventional problem-solution design framework. (Anderson, 1994)” But this is quite a different sort of endeavor to data collection. Note that if anyone, I would lay more blame on some of the sociologists for not making this distinction explicit, and for apparently acquiescing at times to the view of their contributions to systems development as simply data collection and expanding

requirements methods, and even playing the role of user surrogates. The result of such a re-assessment may lead to a more modest “proposal” concerning relationships between the two parties, and yet one which may be achievable and lead to more useful interactions.

Thinking about the nature of “design “

I would like to review and critique some of the different perspectives on the design process, just as I have done above re. the “requirements” process, as again, there are many different views on design, and many different kinds of designing. There is a need for delineating the contexts we are discussing if we are to make any headway in understanding how to move forward in the field of “design requirements”. My own more recent work is in interaction design, and design research more generally, so this is the topic on which I feel I might have most to contribute.. Unfortunately, I am now over the word limit for the position paper so I will stop here, for the moment, but this is the area I hope to expand on at the meeting.

Conceptualizing the notion of “Design Requirements”

Yet another topic for analysis and creative re-synthesis!.....

References

- Anderson, R. (1994). Representations and Requirements: The value of ethnography in system design. *Human-Computer Interaction*, 9, 151-182.
- Button, G. (ed.) (1993) *Technology in working order*. London: Routledge.
- Jackson, M. (1995). *Software Requirements and Specifications: A lexicon of practice, principles and prejudices*. Addison-Wesley (ACM Press): Wokingham, UK.
- Floyd, C. (1987). Outline of a paradigm change in software engineering. In G. Bjerknes, P. Ehn, & M. Kyng (Eds.), *Computers and democracy – A Scandinavian challenge* (pp. 191-212). Aldershot, UK: Avebury.
- Jirotko, M. & J. Goguen (Eds.) (1994) *Requirements Engineering: Social & Technical Issues*. London: Academic Press.
- Jones, J.C. (1988). Softecnica. In J. Thackara (Ed.), *Design after modernism: Beyond the object* (pp. 216-226). London: Thames & Hudson.
- C. Potts. (1995) Invented requirements and imagined customers: Requirements Engineering for off-the-shelf software. *Proceedings 2nd IEEE International Symposium on Requirements Engineering*, March, York, UK, p 128-130.
- Sommerville, I., Rodden, T., Sawyer, T., & Bentley, R. (1992) Sociologists can be surprisingly useful in systems design. In A. Monk, D. Diaper, & M. Harrison (eds.) *People & Computers VII*, pp. 341-353. Cambridge: Cambridge University Press.
- Shapiro, D. (1994). The Limits of Ethnography: Combining Social Sciences for CSCW. *Proceedings CSCW'94*, Raleigh, North Carolina, USA. pp.417-428.