1-1-2005

OZCHI 2005 doctoral consortium application

Robert B. K. Brown
University of Wollongong, bobbrown@uow.edu.au

OZCHI 2005 doctoral consortium application

Abstract

Formal methods of generating and specifying requirements have a chequered past when it comes to dealing with interface design. The Human Computer Interface (HCI) community have not adopted formal methods with open arms [Paterno 96]. If the accurate determination of stakeholder requirements is a significant factor in determining software project success, this thesis turns to a theory based in psychology and sociology to understand these capricious actors, the misunderstanding of whose whims can bring down 70% of software projects. This thesis will deploy Activity Theory as a basis for a proposed system design method and present initial conceptions and some preliminary case-study findings in support. It is anticipated that such a method could bring significant benefits, especially to a particular and growing class of projects; those whose functionality relies heavily upon user interaction.

Keywords

application, consortium, 2005, doctoral, ozchi

Disciplines

Engineering | Science and Technology Studies

Publication Details

Brown, R. B. K.. (2005). OZCHI 2005 doctoral consortium application. In T. Bentley & S. Balbo (Eds.), OZCHI '05 - Proceedings of the 17th Australia Conference on Computer-Human Interaction: Citizens Online: Considerations for Today and the Future (pp. 1-5). Australia: Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

Doctoral Consortium Submission.
Proceedings of OZCHI 2005, Canberra, Australia. November 23 - 25, 2005.

OZCHI 2005 Doctoral Consortium Application

Robert B. K. BROWN - University of Wollongong [rbkb@uow.edu.au]

Title of the Project

An Activity Theoretic Methodology for Eliciting and Specifying Requirements in High-IQ Systems

Introduction

Formal methods of generating and specifying requirements have a chequered past when it comes to dealing with interface design. The Human Computer Interface (HCI) community have not adopted formal methods with open arms [Paterno 96].

If the accurate determination of stakeholder requirements is a significant factor in determining software project success, this thesis turns to a theory based in psychology and sociology to understand these capricious actors, the misunderstanding of whose whims can bring down 70% of software projects.

This thesis will deploy Activity Theory as a basis for a proposed system design method and present initial conceptions and some preliminary case-study findings in support. It is anticipated that such a method could bring significant benefits, especially to a particular and growing class of projects; those whose functionality relies heavily upon user interaction.

Literature Review

Exploring the construction of human consciousness, Vygotsky proposed that humans conceive actions upon an internalised plane of reality. The actions of the human actor (Subject) upon objective reality occur via various mediating tools (both physical and psychological).

Leont’ev focussed upon specific activities and proposed a hierarchic model of components. Any given Activity has a Motive. Within that Motive are Goals oriented Actions. At the base level, atomic Operations are taken depending upon prevailing Conditions. Leont’ev’s conception was a powerful and dynamic vision which encompassed the notion of components ‘sliding’ to another level (typically upwards) as the Subject devotes more cognitive attention upon them in the face of some unforeseen complication. Downwards ‘slides’ may arise as Subjects become more familiar with Actions and they devolve into near autonomic operations.

Interest turned more to the role of the people engaged in the Activity, and Yrjo Engström developed a conceptual matrix, which expands on Vygotsky’s earlier work. The Subject is the person or sub-group whose point of view is analysed. The relationships between these socio-cultural nodes are defined in the Division of Labour node and also in the Rules node which contains social norms, regulations, paradigmatic procedures and other constraints.

The socio-contextual notion extends to encompass the relationships and frictions between neighbouring and inter-related Activities such as those described by Kuutti [Kuutti 91]. Consider, that the outcome of one Activity may become a Tool or Rule of another. One Activity may determine the Subject of another and so on. [Vrazalic 04]

This third generation of AT marks the current state of the art of Activity Theory. The reader should not feel satisfied that this brief sketch has done justice to the domain. Engström’s matrix is a powerful conceptual system whose applications are widespread and much debated in a variety of fields.

Bødker’s landmark PhD thesis and text “Through the Interface” [Bødker 91] and later, Nardi’s text “Context and Consciousness: Activity Theory and Human-Computer Interaction” [Nardi 96] laid out Activity theory as a useful tool and theoretical framework for Human-Computer Interaction (HCI) study. Several proposals have come to light, notably the checklist idea [Kaptelinin 99], however it has been stated that HCI has yet to benefit directly from AT [Vrazalic 04].

McGrath and Uden [McGrath 00] observed, as have the authors of this paper, that there is a near total lack of any prescriptive procedures for applying AT in software development. They found it difficult to apply Engström and Kuutti’s theoretical frameworks using prior AT case studies as a guide, as these were, whilst interesting, almost uniformly narrative in style lacking a well defined and replicable process.

An exception to this lack of system and procedure is the 2004 PhD Thesis by Vrazalic, who proposed a method for evaluating the usability of a system after its completion. This technique is concerned with the broader social context in which the system is used. The user, her social environment, the system and all mediating technologies must be considered. Vrazalic adopts a broader distributed definition of usability in the manner of Spinuzzi [Spinuzzi 99] that incorporates assorted genres, practices, uses and goals. Under this notion of distributed usability, Vrazalic considers the typical usability laboratory to be an artificial environment that has a number of shortcomings that can skew the results [Standish 94]. Vrazalic's Distributed Usability Evaluation Method (DUEM) deploys a comprehensive series of tests based upon Activity Scenarios generated from intensive observation of the user in their native work context, interviews with users and moderated focus group discussions.

DUEM consists of three interacting phases: understanding user activities; evaluating the role of the system in user activities and analysing and interpreting the results. The first phase produces a shared understanding of user tasks and goals. The second phase produces rich qualitative descriptions of the users' interaction with the system. The third phase concentrates on identifying points of breakdown, where the system and the activity map contradict. The problems are described via deeply contextual definitions which aid in reaching any negotiated solutions. DUEM uses the notion of distributed usability and AT principles to define contexts of a system's use by humans. Evaluation is adjudged against criteria derived from these initial findings, based upon user activity rather than system specific requirements. Users are deeply involved in an iterative process through interviews, workshops and observations. [Vrazalic 04].

It has been observed [Brown 04] that one drawback of DUEM is that evaluators must have an understanding of AT principles to inform their analysis and to help them guide users through the process. This precludes deployment by most software analysts and requirements engineers.

Vrazalic [3] proposes in her PhD thesis a method for evaluating the usability of a system after its completion. This rich technique is informed by Cultural Historical Activity Theory (CHAT) and thus is concerned with the broader social context in which the system is used. The user, her social environment, the system and all mediating technologies must be considered. Vrazalic adopts a wider definition of usability in the manner of Spinuzzi [4] that incorporates assorted genres, practices, uses and goals. Under this notion of distributed usability, Vrazalic considers the typical usability laboratory to be an artificial environment that has a number of shortcomings that can skew the results [5].DUEM thus involves an involved and comprehensive series of tests based upon Activity Scenarios that are generated from intensive observation of the user in their native work context, interviews with users and moderated focus group discussions.

DUEM consists of three interacting phases: Firstly, understanding user activities, then evaluating the role of the system in user activities, and finally analysing and interpreting the results. The first phase produces a shared understanding of user tasks and goals. The second phase produces rich qualitative descriptions of the users' interaction with the system. The third phase concentrates on identifying points of breakdown, where the system and the activity map contradict. The problems identified have a deeply contextual definition that aids in any negotiated solutions. DUEM uses the notion of distributed usability and CHAT principles to define contexts of a system's use by humans. Evaluation is adjudged against criteria derived from these initial findings, based upon user activity rather than system specific requirements. Users are deeply involved in an iterative process through interviews, workshops and observations. Evaluators must have an understanding of CHAT principles to inform their analysis and to help them guide users through the process.

The evaluations that result from DUEM show considerable promise however the wide scale deployment of the method may be inhibited by several factors; namely that the method requires trained evaluators, occupies a great deal of time (and is thus quite expensive) and requires that intended users be available at the time [6]. We must also consider that any usability evaluation method that is conducted after the fact can at best only indicate the quality or otherwise of the finished product or facilitate a late stage repair. It is questionable that any meaningful savings in the software production lifecycle will result, as each test is case-specific. If such an evaluation method is widely used, we might expect to see a slow generational or evolutionary improvement in the usability of systems, as each new product is evaluated and its success measured.

This paper proposes that remediation of the software production process may be best achieved by modifying the design-side of the process. DUEM offers some crucial elements to be addressed in this design process, any resulting method would not only benefit from the principles of DUEM, but make DUEM itself easier and cheaper to apply. DUEM is based on user-system interactions rather than system requirements, because these requirements do not address such issues. If they did however, DUEM need not be such an end-phase-heavy process.

Software engineering (SE) centres on two key concepts: 'engineering discipline' and 'all aspects of software production' [Sommerville 01]. Such 'traditional' software developments are product-centric. Some traditional software 'myths' disregard stakeholder objectives and the work then required after initial presentation to adjust the product to suit clients' needs [Pressman 97].

The SE community is recognising that inadequate requirements lead to increased likelihood of failure, especially on 'softer' socio-political grounds [Goguen 93]. This highlights the need to elicit stakeholder requirements and target their softer, qualitative objectives.

Maciaszek draws the production-centric distinction between 'business rules' – a functional requirement describing an 'always on' (invariant) aspect of the system, and 'constraint statements' which define restrictions on system behaviour or the production process [Maciaszek 01]. Whilst this distinction may not strictly address stakeholder preference, it allows acceptance or rejection on these grounds.

Stakeholders generate their own notations and terminologies, complicating the business of capturing such details [Sommerville 98]. This difficulty informs approaches to requirement elicitation that are sensitive to consistency and viewpoint. It is necessary therefore to use a systematic approach when capturing requirements. A significant risk of failure exists in marginalizing the stakeholder's softer objectives, despite their inherent messiness.

One approach to identifying stakeholder preferences adopts a goal-orientation and asks 'what does the stakeholder want to achieve'. Goal formulations express intended system properties [Lamsweerde 01]. Goal-centrism offers stability as top-level goals are often invariant under decomposition, and facilitate back-tracking when re-design issues arise.

Shifting to process orientation necessitates consideration of the software in its environment. It becomes necessary to identify active elements that have choice [Lamsweerde 01]. Such active elements (actors) are the loci for the formulation of goals and preferences. Process-centrism thus infers examination of actor goals, motivations (intentions), dependencies and cultural-historical activities.

A rough taxonomy of these might be visualised as follows.

- Product centric
- Process centric
- Goal centric
- User centric
    - Intention-Dependency centric
    - Motive-Activity centric

Examination of users/actors may take several slices, each of which should reveal some useful description. It is unlikely that these analytical slices reveal incompatible data to each other, indeed the goals, motivations, dependencies and activities should become visible under each. The issue therefore is choosing an approach which serves best for the task at hand.

Whilst numerous proposals exist for design methodologies, most appeal at some point to the experience and expertise of the analyst who is, by some unspecified process, apparently able to identify and extract the relevant data and synthesise a coherent holistic understanding of the project at hand. It has been noted that there is a distinct lack of any prescriptive or systematic process by which these 'hazy' areas may be conducted. There are grounds for suspecting that some methodological sleight-of-hand is at work, some appeal to 'magic'.

There is much written regarding the use of Scenario based design (in which specific instances of deployment are considered) and to the defining of Requirements by "Use Cases" which under certain instances, somehow encapsulate the pragmatic functionality of the component under examination. Each of these seemingly common terms seems to have a distinct and separate heritage, yet they have been merged in various ways and appear in a wide range of methods.

Typically a 'Scenario' is presented as an instantiation of a Use Case, and serves as some kind of colour-text justification beside the Requirement. In other cases the Scenario is synonymous with the UseCase. A significant concern here is that the published 'reason' for almost any given design decision seems to have been constructed after the fact to justify the choice. Despite the use of rhetoric which appeals to User centrism, the user seems to appear, methodologically, as an afterthought. Experts bemoaning the difficulties of dealing with ignorant users are commonplace, and with very little digging most systems designers can be cajoled into revealing a belief that the expert knows best.

In yet other methods, the Scenario seems to refer to a human process essentially indistinguishable from that designated as "Activity" under AT theory.

Yet another commonly deployed design method is that of Pattern use. Patterns represent engineers' collations of experiential and empirical practice, gathered, classified and applied without the burden of any underlying theoretical base; and with little if any regard for the originating or final deployment context. Research in the field of Pattern based design centres largely around the generation of various Pattern Languages, under which the practitioner may semi-automate the sorting, selection and use of an appropriate 'standardised' solution to any given design issue.

Some core issues are arising here:

(1) Design as a process, is a black art. Little more than lip service is paid to the notion of user centrism (as though it were some kind of political correctness one must be seen to comply with) and the lack of any clear methodology makes dealing with users difficult if not downright occult.

    This gives rise to gaps in most methods which preclude them from becoming end-to-end processes, or from being coherently joined to other methods for a complete process.

(2) Both giving rise to, and stemming from problem (1) above, there is a considerable confusion and duplication of terminology and method components.

(3) Desipte claims to the contrary, Design does not seem to occur in the design phase! Rather design, as it is conducted, can too easily become after-the-fact attempts at damage control and justification. The common notion that "usability is all about usability testing" is a testament to this endemic problem.

Analysis of the literature indicates the presence of numerous underlying weaknesses or gaps in extant System Analysis and Design (SAD)

Problem 1.
Current production processes for generating Requirements Specifications do not seem to be addressing the crisis level rejection rate of IT products.

Problem 2.
Many SAD conceptions differentiate the activities of requirements gathering and system analysis, rather than viewing these as complimentary and simultaneous facets of the same process.

Problem 3.
Eliciting specification from stakeholders, users and clients remains a black-art, with few if any structured or systematic approaches in common use.

Problem 4.
Interpretation of user utterances remains a black art. Most current SAD methods leave this up to the insight of the analyst, without addressing the inherent issues, to whit:

a. An *in-house* analyst may be too close to the issues to perceive them objectively.
b. A current user of an extant tool has difficulty in envisioning new tools without their analyses carrying echoes of the current tools and practices.
c. Few *in-house* analysts have the depth and breadth of current technical options and practices to assemble a system for their organisation, and even if they do, their management will more frequently consult external analysts to insulate the process from any internal agendas and preconceptions.
d. An *external* analyst however, lacks tacit knowledge of organisational practice.

Problem 5.
Once gathered, utterances from numerous users may not be consistent. A reliable process for minimising or resolving inconsistency in user utterances has not been adopted into a prescribed, systematic method for extracting feasible Requirement Specifications from user utterances.

Problem 6.
No clear system exists for *processing* the consistent set of utterances ($\underline{U}$) into usable, consistent, feasible, traceable & reusable requirements.

Problem 7.
There is a fundamental problem in SAD, that of the coincidence of users in roles. Most SAD conceptions adopt a function-role-based paradigm, independent of the occurrence, deployment interaction and interdependence of the users within the client organisation.

Problem 8.
Whilst reusability if an oft-cited necessary quality for good Requirements Specifications, there remains as a result of Problem 7, no consistent method for reusability in the face of organisational restructuring, nor for deploying a comparable technical solution to another organisations whose personnel deployment and structure differs even in all other technical factors remain invariant.

Problem 9.

Few if any SAD methods maintain a consistent terminology or taxonomy throughout the process.

Problem 10.

Few if any SAD methods are informed by a consistently applied theoretical base.

Problem 11.

Whilst Activity Theory *has* been identified as a strong candidate for a potential theoretical base, in response to Problem 10, as yet no generic, systematic or prescribed method has been proposed or demonstrated.

Research Question(s)

The rsearch Questions are largely embedded in the Normative Research Methodology (detailed below): namely that a design method be designed which attempts to address the Issues identified under Literature review of extant design Methods, and Normatively tested against those issues.

Methodology

The research project is to build and validate a method. Following the work of Goldkuhl, method components are investigated for their procedures, concepts and notation.

Investigation is conducted under a Normative Framework. From the Literature Review of numerous design methods and approaches a number of normative issues are identified. The new Method is tested against these. Thesis conclusions are based upon an analysis of how well the new Method applies to these normative issues, in the hope that it is at least as effective across most issues, and perhaps exhibits some areas of improvement in some.

Initial normative issues under investigation include that the method should:

1. Be prescriptive, yet flexible.
2. Minimise appeals to analyst expertise or experience.
3. Have a coherent and consistent informing theoretical base.
4. Use consistent and unambiguous terminology.
5. Contain a framework for eliciting and recording user requirements which requires minimal training, yet drives or compels near completeness.
6. Conducts the analysis and production of usable Requirements in the initial phases of the project, and provides criteria under which the project may be continually assessed and validated throughout production.
7. Produce Requirements which have the potential for re-use under changed circumstances with minimal added work.

Schedule

2004	Initial research and focussing on the problem
	Formulation of the research direction
	Initial Literature Review
	Decision on research methodology
	Conference publication of a preliminary positional paper
2005	Clear expression of the project and its goals
	Initial formulation of the Research Question
	Identification of the design norms and issues
	First pass at creating the new Design Method
	Conference publication of the early model method
	Initial testing of the method under controlled conditions
	Write up of the results – conference publication of same
	Refinement of the Method
2006	Finalisation of the Method
	Normative testing of the Method
	Conference publication of results
	Journal submission
	Final thesis write up & submission