Chapter 7

Interdisciplinary Design Teams: Translating Ethnographic Field Data Into Design Models: Communicating Ambiguous Concepts Using Quality Goals

Jeni Paay
Swinburne University of Technology, Australia

Sonja Pedell
Swinburne University of Technology, Australia

Leon Sterling
Swinburne University of Technology, Australia

Frank Vetere
The University of Melbourne, Australia

Steve Howard
The University of Melbourne, Australia

ABSTRACT

Translating ethnographic field data to engineering requirements and design models suitable for implementing socio-technical systems is problematic. Ethnographic field data is often “messy” and unstructured, while requirements models are organized and systematic. Cooperation and communication within an interdisciplinary design team makes the process even more complicated. A shared understanding between ethnographers, interaction designers, and software engineers is vital to ensure that complex and subtle social interactions present in the data are considered in the final system design. One solution for supporting team conversations uses the quality goal construct as a container for complex and ambiguous interaction attributes. Quality

DOI: 10.4018/978-1-5225-5969-6.ch007
goals in system modelling promote shared understandings and collaborative design solutions by retaining a high level of abstraction for as long as possible during the design process. This chapter illustrates the effectiveness of abstract goals for conveying complex and ambiguous information in the design of a socio-technical system supporting social interaction between couples.

INTRODUCTION

In designing socio-technical systems, it is necessary to discover what discriminates social requirements from other types of requirements. Designers need to focus on the identification of user goals, and in particular those non-work-related, non-goal-oriented interpersonal interactions that happen between individuals. An interdisciplinary team is important for the translation from ethnographic field data to requirements modelling to achieve a solution that takes account of both function and non-functional aspects of social interactions. Embracing ambiguity in requirements modelling for socio-technical systems helps withhold design commitment on social requirements for as long as possible. Recognizing and working with how models of ambiguous concepts, such as joking and play, facilitates rich conversations between ethnographers, interaction designers and software engineers with respect to the social needs and interactions of users of proposed systems. The need for the design team to create a shared understanding becomes particularly relevant when designing digital technologies for activities outside of the work domain, for example, mediating personal and intimate relationships between couples (Paay et al., 2009).

Cultural probes produce ethnographic field data that is particularly suitable for learning about intimate and personal aspects of people’s lives (Gaver et al., 1999). Data generated from cultural probes can include user-generated photographs, postcards, drawings, and scrapbook entries, and can be supplemented with contextual interviews to enrich understanding. However, it is a substantial challenge to take the “messy” user data and translate it into the formal models required for system implementation. A design team needs to identify non-work-focussed and non-goal-oriented interpersonal interactions that happen between individuals, to discover social requirements of a system. These interactions, for example, flirting or joking, constitute much of what occurs between intimate couples, families, friends and social groups engaged in leisure. Ambiguity plays a vital role in maintaining openness and flexibility in the interpretation of cultural probe data throughout the design process. Ambiguity, as described by Gaver et al. (2003) becomes “a resource for design” when designing technologies to support activities outside of work. Ambiguity should be regarded as an opportunity, rather than a problem, because it allows for multiple possible meanings and individual interpretations of concepts. According to Gaver
et al., ambiguity in design allows people to interpret situations for themselves, and establish deeper and more personal relations with meanings offered by systems.

Software engineering is proficient at modelling individual purposeful needs but it does not yet convincingly address requirements elicitation of non-goal-oriented social interactions. Additionally, ethnographers and interaction designers come to the design process with their own interpretations of the data and hence different implications for system design. It therefore remains an ongoing challenge to merge the differing viewpoints, skill sets and understandings of members of an interdisciplinary team, to create an effective and relevant technology solution for end users (Paay, 2008). Using ambiguity in requirements modelling can provide a solution to the problem, through the use of quality goals from the Agent Oriented Software Engineering (AOSE) method (Sterling & Taveter, 2009). Quality goals are essentially non-functional goals encapsulating social aspects of the context into software requirements models. They represent quality attributes that both enrich and constrain the goals and roles of a multi-agent system. This is particularly relevant when data about people’s experiences are gathered using ethnographically inspired methods. The construct of quality goals, attached to functional goals in the requirements models, represent quality attributes of social interactions as abstract, ambiguous and unresolved concepts. The ability to articulate quality goals and carry them through to the design phase in an undischarged form, adds the necessary level of ambiguity for interpretation and translation of social requirements. Quality goals can carry subtle nuances of social interactions from field data through to final system design. They are able to stimulate conversations on design alternatives during collaborative conceptual design, because they remain flexible and open to interpretation. They do not need to be resolved early in the process. This is important when the team is interdisciplinary, allowing for translation of different concepts and ontologies, toward a shared understanding.

This chapter presents the development of the Secret Touch system, illustrating the design of a novel technology supporting flexible and meaningful social interactions between people in intimate relationships. An interdisciplinary team of ethnographers, interaction designers and software engineers modelled system requirements for Secret Touch collaboratively. User data was collected in a cultural probes study exploring social interactions between couples in the “Mediating Intimacy in Strong-Tie Relationships” project (Kjeldskov et al., 2005; Vetere et al., 2005). Although technological innovations are typically created to address commercial interests, technologies are often appropriated to mediate personal and intimate relationships (e.g. typewriter for love letters, telephone for maintaining personal ties, mobile phone for texting personal interactions). Rather than wait for the consequences of commercially oriented innovation to permeate personal lives, the mediating intimacy project sought to examine the design of technologies for
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

strong-tie relationships explicitly. The project proposed a framing of intimacy that supported a design oriented discussion. Intimacy was characterised by Antecedents (pre-conditions for intimacy), Constituents (elements of the intimate acts), Yields (the consequences of an intimate exchange). The themes were used to stimulate 22 design ideas, one of which was Secret Touch which enables partners to “virtually hold hands” while physically separated. Couples exchange tactile impulses over the Internet by padding or squeezing a device in their hand or pocket. These impulses would be sensed as vibrations, heat or pressure by the partner.

The secret touch concept was explored through contextual interviews to supplement and interpret the probe data with participants. Probe data was analysed using affinity diagramming (Beyer & Holtzblatt, 1998) and scenarios were created (Carrol, 1997), before cooperative modelling by the team. Quality goals and modelling notation inspired by the ROADMAP methodology (Sterling & Taveter, 2009), were used during requirements elicitation. However, quality goals can be equally well elicited in most requirements modelling methodologies. The Prometheus methodology (Padgham & Winikoff, 2004), an AOSE methodology, inspired modelling at the design level.

The Secret Touch design process demonstrates the benefits of using quality goals during requirements modelling when using cultural probe data about the social interactions of a couple. The final design of Secret Touch runs on a small pocket device, communicating with a similar partner device. Couples in intimate relationships interact discretely and remotely by physically moving a device in their pocket which makes their partner’s device move in an identical fashion. Detailing the design process shows how quality goals maintained ambiguity throughout the elicitation, analysis, and conceptual design stages of requirements modelling. They supported richer on-going conversations between ethnographers, interaction designers and software engineers. This shows that by combining ethnographic techniques for understanding complex and subtle human relationships with best practice from interaction design and software engineering disciplines the outcome is a socio-technological system incorporating rich, valued aspects of people’s social interactions. Recognizing ambiguity in an adequate and open way is particularly relevant when developing social technologies.

This chapter presents background information on social concepts in requirements modelling, eliciting requirements from cultural probe data, quality goals in requirements modelling and ambiguity in the modelling process. The design of the Secret Touch system is described, with detailed requirements elicitation, analysis and design modelling, tracing the progression and resolution of ambiguous concepts through the use of quality goals, throughout the process. This is followed by a discussion on the benefits of the approach, as well as insights on how the research has influenced current methodologies in software engineering of socio-technical systems. The chapter concludes with implications of the study.
BACKGROUND

Software engineering has a long tradition of identifying constraints on software behaviour, representing functional services, and modelling them as functional goals that the system is expected to deliver (Jureta et al., 2006). However, with increased development of socio-technical systems supporting people’s activities outside of work, such as leisure activities, functional goals are no longer sufficient to represent all system requirements. This is because people’s social interactions are not easily decomposed into measurable functional requirements. Domestic and social pursuits comprise a range of purely communicative activities, not necessarily serving any external productive purpose. In designing socio-technical systems, capturing social concepts accurately but flexibly at a high level is essential, while not losing their vitality through over specification.

Social Concepts in Requirements Modelling

In software engineering requirements modelling, social issues have been considered since the early 1990’s. Goguen and Linde (1993) introduced techniques for requirements elicitation acknowledging that work takes place within a social context. These techniques used concepts from ethnomethodology and sociolinguistics, including conversation, interaction and discourse analysis, to gain access to social interactions. Quality issues such as security, portability, reliability and modifiability, although not easily quantified, were identified and their imprecise nature embraced as desirable for making trade-offs during the design phase (Goguen & Linde). Other software engineering approaches including social needs modelling are: i* framework (Yu, 1995); TROPOS (Castro et al., 2001); ROADMAP (Sterling & Taveter, 2009); and the Softgoal Concept (Jureta et al., 2006). These approaches include social context but focus on coordination and collaboration tasks, which are social but with deliberative purposes, for example, arranging a time to meet.

Ethnographically inspired methods have been successfully used in Human Computer Interaction (HCI) to understanding how people interact and to gather requirements for interaction design since the mid 1990’s (e.g., Hughes et al.,1995; Rouncefield et al., 2003; Sommerville, 2015). These methods include: field observations, field studies, contextual interviews and cultural probes. HCI designers want to understand the current practice of people in a particular situation in order to design technologies to support them or improve their experience of that situation. The ethnographic approach to understanding user needs puts the user firmly at the centre of data gathering. Viller and Sommerville (2000) used ethnographic studies of work to inform systems design. They demonstrated how ethnographic results could be integrated into the object-oriented software engineering (OOSE) process.
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

They identified use cases and generated initial use case models, using the Unified Modelling Language (UML) to represent social aspects of work. However, because their case study was in air traffic control, the focus was on defining functional requirements around distributed coordination, plans and procedures, and work awareness. They were also faced by the problem that OOSE does not explicitly encourage the inclusion of information pertinent to social contexts. According to Sommerville (2015), ethnographically inspired data collection can provide a rich source of ideas for new technological products. Especially when data is collected about a social activity, as and when it happens.

For the representation of non-functional social requirements, it is essential to capture social concepts while also communicating the rich quality of people’s social interactions. Several approaches have been made in software engineering to achieve this, the most successful being the introduction of actors and agents using the AOSE paradigm. In an agent-based system, agents have characteristics of autonomy, reactivity, pro-activeness and social ability (Wooldridge & Ciancarini, 2000). The social ability characteristic allows agents to interact with other agents, and sometimes even with humans. Agents have the ability to engage in social activities. They are primarily used for cooperative problem solving and negotiation toward achieving functional goals of a system. However, the construct of the agent, and agent roles, opens up the possibilities for representing human qualities of social interactions. The AOSE paradigm is promising for supporting social concepts in software development because the characteristics of agents can be equated to human actors. Agents can exhibit human-like behaviours such as autonomy, flexibility, intelligence, learning, and dynamic adaptability to the surrounding environment, increasing their suitability for socio-technical software development. Agent-oriented concepts help design teams collaborate toward a deeper understanding of the implications of quality requirements in design, and work together to find solutions.

Constructs for capturing social concepts in requirements modelling can be found in the i* framework (Yu, 1995), Gaia (Wooldridge et al., 2000; Zambonelli et al., 2003), TROPOS (Castro et al., 2001), ROADMAP (Sterling & Taveter, 2009) methodologies. The i* framework approaches social concepts in requirements engineering from the broader perspective of describing a network of social dependencies among “actors” in the system. The i* framework inspired approaches for explicitly handling qualitative system aspects using quality requirements, often called Non-Functional Requirements (NFRs) (Chung et al., 2000). Gaia is an early and well-cited AOSE methodology. It takes an organisational view, describing systems in terms of the roles that agents take on. However, Gaia starts modelling from a requirements statement, assuming requirements have been gathered by some other process. Conversely, TROPOS is requirements-driven and strongly involved in the early requirements phase. TROPOS defines systems in terms of autonomous, intentional and social
software actors, incorporating concepts from the i* framework, and focussing on social dependencies amongst actors. TROPOS documents requirements in a way that is compatible with a business environment and facilitates software engineering from design through to implementation. However, it is a rather formal methodology, so modelling quickly becomes quite complex. Complexity precludes it from being suitable for capturing ambiguous social concepts. ROADMAP (Role-Oriented Analysis and Design for Multi-Agent Programming) is an AOSE methodology that started as an extension of Gaia. Analysis and design in ROADMAP differs from the other AOSE methodologies because it enables extensive, detailed and fine-grained capture of the different roles and goals that agents have. It separates analysis and design, and is independent of design architecture and agent platform. ROADMAP exhibits strong flexibility, enabling uncertainty and ambiguity to be maintained during requirements elicitation. It does so through the representation of quality goals in goal model diagrams. Quality goals, directly associated with system goals, constrain the ways in which agents enact their roles. The flexibility of ROADMAP makes it particularly suitable for conveying ambiguous social concepts encapsulated in quality goals throughout system modelling.

Eliciting Requirements From Cultural Probe Data

Cultural probes are a novel data collection technique developed by Gaver et al. (1999) gaining prominence in interactive systems design (e.g., Burrows et al., 2015; Cheverst et al., 2003; Dey & De Guzman, 2006; Hassling et al., 2005; Mayasari et al., 2016; Paulos & Jenkins, 2005; Petrelli et al., 2009; Rodden & Benford, 2003; Rouncefield et al., 2003; Schorch et al., 2016; Soro et al., 2016). Cultural probes, an ethnographically-inspired method, are particularly suited to investigating people’s everyday life in situations otherwise difficult to reach with traditional social science methods, such as questionnaires, interviews, focus groups or observation. Like deep sea or planetary probes, cultural probes are ‘sent-out’ by researchers and return fragmentary data over time. The data returned gives subjective, inspirational glimpses into the situational context and lives of participants and captures motivations that shape domestic life. Cultural probes are carefully designed packages of materials, including: diaries, scrapbooks, postcards, cameras, pens, pencils, glue, sticky tape and other materials. Participants use these to document everyday activities as they occur or shortly afterward, and then return them to researchers. It is left to the participants to record their own experiences. The probes both encourage and empower subjects to collect data themselves, rather than relying on the presence and intervention of a researcher (Arnold, 2004). Participants provide researchers with insight into their daily lives, but at their own discretion.
Cultural probes facilitate collection of research materials over longer periods of time and in multiple locations concurrently, compared with resource intensive observational ethnographic approaches. In particular, cultural probes garner an understanding of the playful character of human life and the multifaceted ways in which people “explore, wonder, love, worship, and waste time” (Gaver, 2001, p.1). The initial intention behind cultural probes was simply to inspire design, not to gather system requirements (Gaver et al., 2004). Cultural probes are looking for clues about how people get on in the world, and how technology might support their interactions with each other and their environments. They enable elicitation of information otherwise difficult to obtain about participants’ habits, and emotional, aesthetic and social values (Hemmings et al., 2002). Interaction designers have embraced cultural probe data as providing rich descriptions of real social phenomena (Rouncefield et al., 2003) useful for informing technology design. However, probes, by their very nature, provide fragmentary data, making it difficult to elicit detailed requirements for systems design, since they were not designed to provide data easily transformable into technical specifications. Hence, difficulties arise when using cultural probes for requirements gathering in software engineering requirements modelling. It is because the cultural probe method is so good for learning about the subtleties of social interactions in the domestic domain, that researchers persist with it as the data gathering method of choice for the design of innovative social technologies. They provide rich information to designers about people’s daily lives (Vetere et al., 2005). Cultural probe data is also useful for team design. It inspires “interdisciplinary iterative interpretative and design work where stories are generated, renewed and reworked over time” (Graham et al., 2007, p.35). Provocative and traceable discussion points arise within the design team as they develop a shared understanding of system requirements through insights garnered from the rich probe data.

With ethnographic field data, the problem is how to derive usable and useful system requirements while retaining the insights and inspirations collected. Eliciting requirements from probe data involves working in a way that captures concepts accurately but flexibly, at a high level, without losing their liveliness and vitality through over specification. The rich information gathered in the field needs to be transformed to fit the formalized and rigorous models of software requirements elicitation and design. The team identifies goals for the system, and then cooperatively designs how they should be operationalised. Quality goals are a necessary part of the abstraction, maintaining the richness of data while concrete system requirements are generated to support technology implementation. Quality goals are closely related to system use and yet are not necessarily always translated directly into system requirements. In socio-technical systems, the quality of the final design tends to be greater than the sum of the individual single functions it is composed of, that is, “the whole is greater than the sum of the parts” (Sterling &
Taveter, 2009, p.11). High-level quality goals associated with social activities act as a point of reference for discussing design alternatives. Quality goals function as a bridge between “messy” cultural probe data and requirements elicitation models, providing a focus for conversations within an interdisciplinary team.

Quality Goals in Requirements Modelling

Quality goals provide a constant reminder of the overall goals and intentions of people interacting in social situations. They capture the essence of human interactions better than other elicitation mechanisms found in software engineering practices, because they are more dynamic and fluid. A design team uses quality goals to understand the reasons why people do things, or the essence of a relationship, rather than the actions taken. Quality goals can be used to withhold design commitment beyond the elicitation phase, keeping quality attributes of an interaction unresolved and interpretably flexible throughout all stages of the process. From a software engineering point of view, requirements and design models enable a design team to take the outputs from a cultural probe study and use them to inform socio-technical software design. Thus taking account of the richness of human social interaction provided by the cultural probe data and encapsulating quality attributes of that interaction into quality goals in the models. These models then facilitate high-level discussions in designing a system that supports and enhances domestic social interaction. Agent-oriented modelling, in particular, facilitates the expression of non-functional requirements by attaching quality goals to goal models, and putting constraints in role models (Sterling & Taveter, 2009). Goal model diagrams specify roles required to enact a goal. Role model diagrams detail responsibilities and constraints relevant to enacting goals. These roles are consolidated into agents that form part of a multi-agent software system. The success of a design in achieving its goals can really only be confirmed after implementation. However, to build technologies that genuinely facilitate social connections and relationships, it is important to maintain an openness in the process and not resolving all aspects of requirements too early. The fact that quality goals are not directly implemented, but influence design by providing a rich picture of the social interactions of people and their domestic roles being modelled, means that openness can be achieved.

Quality goals in requirements modelling, as introduced by ROADMAP (Sterling & Taveter, 2009), represent quality attributes of the system as high-level abstractions that remain unresolved throughout the design process. Multiple interpretations and design alternatives for social aspects represented in the final product are encouraged. The quality goal container facilitates conversations between ethnographers, interaction designers and software engineers on quality attributes of social interactions, until a collaborative solution is reached. Innovation and creativity in design are stimulated,
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

as people with different expertise and skills sets communicate their different ideas, provoke “out of the box” solutions, evaluate them as a team, and negotiate holistic solutions. The process brings together the deep social understanding of the ethnographer, the human factors and HCI expertise of the interaction designer and the abstraction and modelling expertise of the software engineer to create a technology design embedded into real social contexts. By externalizing quality attributes in quality goals, they become shared artefacts that sustain multiple interpretations across disciplines (Paay et al., 2009).

Focus on quality is a well-established aim within software and systems engineering. Software engineers realise the need to express quality attributes of software, as well as functional capabilities, as digital technologies move into the home and leisure domains. Some form of “quality attributes” are included in several modelling methodologies, using a variety of terms, including: non-functional requirements (NFRs), constraints, quality attributes, quality goals, or quality of service requirements (Gross, 2005; Kirikova et al., 2002; Reekie & McAdam, 2006; Sommerville, 2015; van Lamsweerde & Letier, 2004). Technology that supports interpersonal contact in social situations is only valuable if it fulfils the felt needs of the people acting in these environments, who will use technology to augment and support their social interactions. People’s social needs are typically high-level, cognitive, emotional, and hard to measure. For example, playfulness, the act of engaging in an activity or expressing feelings for no functional purpose, is an important part of close relationships. Playfulness helps define the nature and closeness of a relationship (Davis et al., 2007). It is therefore important that any device facilitating interaction between close couples should be able to represent playfulness in the user experience. Playfulness encapsulated in a quality goal means that it becomes a visible part of the model, to be remembered and incorporated into the design at some point.

Quality goals are an effective container for quality attributes in requirements modelling, conveying a more positive connotation than the term “non-functional requirements”. They are seen as different from functional goals, and not reduced to measureable goals. Quality goals, by their very name, infer concern with the quality of a system, and aspects of user experience that go beyond functionality. Quality goals are most effective in requirements modelling when directly paired with system goals. The direct relationship is necessary for quality attributes to be carried through the stages of the elicitation, analysis and design in an unresolved state. System goals are user goals that are satisfied by using the system and are not necessarily implemented as specific functions in the final system. Relating an abstract, unresolved quality attribute to a system goal maintains ambiguity for as long as possible within the design process. By openly defining quality goals during socio-technical system design, quality concerns are kept visible throughout the design process, ensuring they do not become an afterthought in system development.
Quality goals in a process, emphasize a more active searching for qualities of the interaction from the ethnographic data, distinct from identifying functional goals. They purposefully use words directly linked to people’s interactions in social environments, such as, flirting, joking, teasing.

Quality goals are often subjective, context-specific, and imprecise (Sterling & Taveter, 2009), and hence actively promote ambiguity in the process. Quality goals that are intentionally ambiguous, represent an opportunity rather than a problem to be solved. They encompass ambiguous elements in social systems, such as flirtatiousness, fun and play, and carry them through to design in a system independent form. Delaying specification enables the design team to ideate on solutions and negotiate trade-offs while evolving the functional design. Functional goals are generally hierarchical and are decomposed into sub-goals. Quality goals are not. They require a more “holistic” design approach. Rather than specifying solutions for resolving quality goals as a set of sub-tasks, they are incorporated into the overall design through inspirational and collaborative leaps, requiring a “designerly way of thinking” (Cross, 2011; Kjeldskov, 2014).

Scenarios, used in conjunction with quality goals, are an addition to the requirements modelling process that further deepen the design teams understanding of the user’s situation and help translate those goals into system design (Sterling & Taveter, 2009). Scenarios are a well-established tool for requirements engineering (Carroll, 1997; Sutcliffe et al., 1998). They are typically textual descriptions of situations of use, described from the perspective of an end user. Carroll (2000) states scenarios keep the future use of the technology in view, making the technology easier to design: “The defining property of a scenario is that it projects a concrete description of activity that the user engages in when performing a specific task, a description sufficiently detailed so that design implications can be inferred and reasoned about” (p.3). Stories enable developers to empathise with the people in the situation, which leads to questions about motivations, intentions, reactions and satisfaction. Therefore, scenarios are powerful when designing and evaluating systems with requirements that involve user engagement. While goal model diagrams are purposefully simple they are reinforced through more extensive information captured in motivational scenarios and domain models.

Motivational scenarios, in requirements elicitation, can be informed by cultural probe data. They hold more detailed information about the qualities of a system from a person-centred view and describe the roles people take on. The addition of scenarios to a modelling methodology, further socializes quality goals and deepens the richness of understanding available to the design team. Scenarios attached to quality goals help identify key steps in user interaction with the system where a particular quality attribute is important, helping the design team interpret quality goals appropriately. Domain models capture important information about the use
context, for example, the home. Quality goals rely on context from domain models for the design team to understand and resolve ambiguity in quality attributes.

Quality requirements are an important part of any software project, they are a major factor in determining system success, and yet quality attributes are often forgotten about during software development (Sterling & Taveter, 2009). The ability of socially oriented quality goals to carry quality attributes as abstract and ambiguous concepts makes it possible for design teams to evaluate and resolve these concepts at any point during elicitation, analysis or design. While some quality goals, such as performance targets or system availability, are readily quantifiable, others such as a system being secure or a game being fun, are less so. Quality goals are used to define quality attributes in the data, without the need to resolve them. Resolution of quality requirements can be done while solving functional requirements, impacting the way both are resolved. Through the use of the quality goal container, quality aspects found in the data are able to influence design decisions throughout the process, because they remain visible and yet still ambiguous.

**Ambiguity in the Design Process**

The value of embracing ambiguity in the design process is in giving space for creativity in the design process. Gaver et al. (2003) define ambiguity “as uncertainty in the interpretative relationship linking person and artifact”. They identify three aspects in design where uncertainty can exist: the ambiguity of information or attributes of the artefact itself; the ambiguity of context in the sociocultural discourses that are used to interpret it; and ambiguity of relationship in the interpretative and evaluative stance of the individual or his attitude towards the artefact. Several HCI studies have investigated benefits of preserving ambiguity in design products, to give users space for their own interpretation of the artefacts (Aoki & Woodruff, 2005; Dalmau, 2003; Gaver et al., 2003). The benefits of ambiguity can equally be applied to the process of requirements modelling. However, using “ambiguity as a resource for design” in modelling, is most beneficial during design specification. The set of models are the design artefact where ambiguity sits, and the design team are users of those evolving models.

Although ambiguity seems antithetical to the routines of software engineering, it has always been part of it. However, software engineers typically view ambiguity as having a negative impact on development because it is a source of impreciseness that can lead to misinterpretation and incorrect formal specifications (Kamsties et al., 2001). Software engineering research has been primarily concerned with detecting ambiguity in requirements specification and dealing with it in (Blaha et al., 2005), involving either reducing it to a minimum (Tjong et al., 2007) or removing it completely (Fabrini et al., 2001). Research has explored the use of tools...
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

(Fabrini et al.), inspection techniques (Kamsties et al.), and rules (Tjong et al.) to detect ambiguity in the design process or in requirements documents (Buckeridge & Sutcliffe, 2002). It is a generally held belief that a requirements specification is of high quality if there is no ambiguity (Fabrini et al.).

The use of natural language in requirements specification is regarded as one reason that ambiguity finds its way into the process (Berry, 2008; Kamsties et al., 2001; Tjong et al., 2007). The use of a formal requirements specification language is an often-suggested solution for removing ambiguity (Berry et al., 2003). By dealing with ambiguities in data at the informal level, before formal specifications are produced, misinterpretations and incorrect formal specifications can be avoided (Kamsties et al.; Fabrini et al., 2001). However, very few requirement documents are written in formalized language because the translation from informal natural language requirements to formal requirements documentation is difficult (Berry; Fabrini et al.; Sawyer et al., 2005). The difficulty is exacerbated by a “messy” data set informing those the requirements, for example, gathered using cultural probes or other ethnographically inspired methods, and especially within a non-work context.

Software engineering research, since the early 1990’s, has used a social science perspective to recognize that models, and other documentation, are not simply feeding user data into a formal process of modelling for system design (Button & Sharrock, 1994; MacLean et al., 1990; Randall et al., 1994). They provide ways to think through problems, to reach agreements, and elaborate needs. For example, MacLean, et al., found the most useful form of design rationale was the very rough documents made early in the design process. These documents provide a resource for revisiting, used as orienting documents and artefacts later in the process. Similarly, the value of ambiguous attributes in quality goals is in giving the design team a more structured way of elaborating and confronting the complexity of the design space for socially-oriented technologies. Thus, considering not only ambiguity per se, but the role of models in the design process. If ambiguity is removed from the modelling process too early, designers lose clarity on where it occurred and the context of the resolution (Kamsties et al., 2001). Early disambiguation leads to unrecognized or unconsciously assumed meaning, that could be entirely wrong, finding its way into design specification (Berry et al., 2003; Gause & Weinberg, 1989). It also leads to ambiguous concepts not being represented properly in the final design, because they are resolved before their impact on the overall design is completely understood. It is difficult to detect missing non-functional attributes once a formal specification document is made (Kamsties et al.). Therefore, the essence of non-functional quality social interactions could be lost to the final design, if they are interpreted too early.

“Soft” goals, such as fun, are difficult to precisely define, independent of the domain, because these quality attributes tend to be imprecise, subjective, idealistic and context-specific (Jureta et al., 2008). Goal models are used early in design
to arrive at a common understandings of system goals. There is a directive in the software engineering process to detect and resolve system issues as soon as possible, often without seeking clarification (Jureta & Faulkner, 2007; Kamsties et al., 2001). With quality goals, it is more difficult to arrive at a shared understanding early in the process. During early requirements elicitation, designers might not be able to clarify social concepts sufficiently to resolve them, therefore social goals should be kept at a high level of abstraction, even after formal specifications are written. Identifying ambiguities in social data creates awareness in the design team of the need to maintain the high level of abstraction. Team discussions around highly ambiguous complex concepts that are not easily clarified, helps complexity reduction without losing the richness of the social concepts themselves. Resolving design through discussion, embraces interpretation and openness amongst design team members and supports correlation with overall goals of the system and the context of use. Rather than eliminating ambiguity early in the process, withholding design commitment gives the design team time to discuss and clarify what it might mean to disambiguate with respect to the overall design.

Ambiguity in the requirements modelling process helps the design team to recognize, design for, and evaluate systems with the view that multiple, competing and divergent interpretations of a system can co-exist, and solutions to design problems do not need to reside in a single correct interpretation (Sengers & Gaver, 2006). The value of ambiguity is in the flexibility it provides in the design process. Ambiguity, as embodied in goal models, allows for multiple interpretations of a problem to co-exist throughout the requirements modelling and design phases. By integrating quality goals into system goal models, the ambiguous nature of social interactions is acknowledged, and can impact the final design of the system. Some ambiguity may be resolved during requirements modelling; some may be carried through to the design phase. By keeping quality goals at an informal and ambiguous level throughout the analysis and design processes, they remain available as a resource for discussion. They facilitate an interpretive relationship between team members and models and can be used to prompt and influence design decisions without the need to clarify and specify quality attributes within fixed system boundaries. These discussions around models and motivational scenarios, bring to the table as many different interpretations of the models as there are members of the design team. Conversations around complex and abstract social concepts such as “having fun” and “playing” help the design team to identify concepts at goal level and include qualities related to the achievement of those goals in the overall system.
THE SECRET TOUCH SYSTEM

The design of the Secret Touch system demonstrates the translation of data from a cultural probes study to system requirements and a final design of a socio-technical system. The models include a scenario to motivate the design, goals, roles and ambiguous concepts in the form of quality goals. Presentation of a sample set of models shows how quality goals contribute to the process of interpreting data for system design. By identifying important aspects of social interactions, such as “Risky” and “Playful”, and representing them as quality goals associated with system goals in the models, they were made visible. The design team of ethnographers, interaction designers and software engineers had a construct for analysing, interpreting and designing around complex social concepts. The interdisciplinary design team worked together in interpreting quality attributes, resulting in a design that was more holistic in its resolution of social interactions. However, the models presented do not provide a step-by-step process for designing socio-technical systems, rather, they show the role of ambiguity within a requirements and design modelling methodology. By tracing specific quality goals through the different models, it can be seen how uncertainty, inherent in the data, is carried through the process of design. The Secret Touch quality goals represent information about intimate interactions that occur between couples, which in themselves are sometimes intentionally ambiguous. These quality goals actively promote and tolerate ambiguous attributes of intimate interactions, maintaining them throughout the elicitation process at a high level of abstraction. They are not discharged before they are passed to the design process, demonstrating a sensitivity to the non-functional aspects of the data, and a radical extension and application to the synthesis of probe outcomes into the technology design process. Through the processes of requirements elicitation, analysis and design, the software engineering models were used in a flexible way, acting as artefacts for discussion between team members.

Requirements Gathering

Requirements gathering for The Secret Touch system used the findings from the Mediating Intimacy project (Vetere et al., 2005). The study focused on supporting intimates (i.e. life partners, parents and younger children, adult offspring with dependent parents, etc.) when both co-present or separated by distance. Data gathering in the Mediating Intimacy study was a multi-method approach to understand intimate interactions between participants, extending the work of Gaver et al. (1999) by combining cultural probes, interviews, focus groups and scenario-based acting out sessions. The cultural probe data collected included workbooks and diaries produced by six couples, documenting their interactions throughout the day.
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

Scenarios drawn from the Mediating Intimacy data were used to drive design ideation. A series of motivational scenarios were created, each suggesting technological solutions to assist in mediating intimacy. One of these was the Secret Touch scenario (see Table 1), which described a device designed to facilitate intimacy between couples by maintaining a digital connection with each other throughout the day. The Secret Touch scenario was chosen for design and implementation of a socio-technical device for mediating intimacy in couples.

Requirements elicitation from the data was an iterative one. Software engineers driving the requirements elicitation phase would regularly present their ideas and models through presentations, discussions and survey questionnaires to the team of six ethnographers and interaction designers, who conducted the fieldwork for the Mediating Intimacy project. The team would then discuss the abstractions being made from the data and cooperatively form revised models and design features. Through team interactions, Secret Touch became a robust and flexible system reflecting needs of target users. The communicative and integrated design process was able to cater for and embrace the diverse opinions within the design team about the desirability of different intelligent Secret Touch options. The process resulted in a range of different features for the system representing complex and subtle needs of end users. Table 2 contains the complete feature range, listed from the simplest to most complex activity enabled by the system.

“Flirt” is the basic feature of the Secret Touch system, where all device movements are transformed into touches. These touches are instantaneously sent to the partner’s device, and received as a movement of the device. If there are simultaneous incoming and outgoing movements, the device resolves them and moves in a direction that is the vector sum of both touches, simulating a playful tug-of-war situation between users, where they can actively counteract touches from their partner.

“Discrete Flirt” enables partners to enter into a turn taking dialogue of interactions. If the user is busy, for example, attending a meeting, they can either switch their

**Table 1. The Secret Touch motivational scenario generated from the Mediating Intimacy Project**

<table>
<thead>
<tr>
<th>Scenario: Secret Touch</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Inspiration:</strong></td>
</tr>
<tr>
<td>• Couples wanting to communicate in private</td>
</tr>
<tr>
<td>• Feeling each other</td>
</tr>
<tr>
<td>• Being playful</td>
</tr>
<tr>
<td>• Individuals like fiddling with toys</td>
</tr>
<tr>
<td><strong>Description:</strong></td>
</tr>
<tr>
<td>They both reach in their pockets during work. She feels that he is fiddling with the device. She turns the device in the other direction, engaging into playful activity</td>
</tr>
</tbody>
</table>
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

Table 2. The Secret Touch features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flirt</td>
<td>Risk and openness to flirtation</td>
<td>Open channel, full-duplex communication, i.e. always on</td>
</tr>
<tr>
<td>Discrete Flirt</td>
<td>Partner chooses level of accessibility</td>
<td>Choice of open channel or modes: ON, OFF, PASSIVE</td>
</tr>
<tr>
<td>Fiddler’s Choice</td>
<td>Response possibly from agent</td>
<td>Add learning or remembering to either of the above</td>
</tr>
<tr>
<td>Guessing Game</td>
<td>Who or what is that?</td>
<td>An open, dynamic system – partners and devices change</td>
</tr>
</tbody>
</table>

device off, or they can set it to discretely receive touches from their partner. The discreet flirt feature collects incoming flirts, which can be replayed later when the meeting is over. The user is able to control when they are not available for flirting, or they would like to remain connected, but review collected flirts when it is more convenient.

“Fiddler’s Choice” is an intelligent, learning feature of the system. When a user receives an incoming touch, but is unable to personally engage in the flirting episode, the device takes control and responds to their partner on their behalf. The feature can also be used by individuals who would like the system to actually flirt with them. An important application for personal use is in learning and behavioural education. It can be used to assist in reducing the kinds of nervous fiddling associated with anxiety or autism. Both awareness and satisfaction comes to the user by giving them a device movement in response to their fiddling.

“Guessing Game” is designed for playing “hard and fast” in personal relationships, with connectivity between groups of devices. There is no longer only a set partnership or connection between two devices. Connected devices may still be aligned with a single intimate couple, but can also involve multiple partners. It is an open, dynamic system, where devices participating in the exchange of touches can appear and disappear from the game, seemingly at random.

Requirements and Analysis Models

In requirements elicitation, requirements modelling diagrams, based on the ROADMAP methodology (as detailed in Sterling & Taveter, 2009), were used to elicit a set of roles, goals and quality goals from the data collected during the Mediating Intimacy project. These models offer capacities at the requirements analysis level, enabling various levels of abstraction, hiding complexity, or focussing on details of a section of the model. The models facilitate high-level abstraction, flexibility in design, and ease of use, which are important in the process of interdisciplinary
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

design when transforming field data into system requirements. System requirements and quality constraints can be captured independent of later decisions concerning design and architecture. For the final design, modelling diagrams, based on the *Prometheus* methodology (Padgham & Winikoff, 2004), were used to create the AOSE system design for *Secret Touch*. *Prometheus* arose out of a BDI (Belief-Desire-Intention)-based agent platform *JACK* (2015), is more fully developed for design than requirements analysis, and offers a well-documented and usable set of methods for designing agent software.

Goal and Role Modelling Notation

Goal models represent system goals, in a loosely hierarchical fashion. Goal models can have quality goals associated with them, which influence how that goal should be fulfilled. Quality goals are the intangible goals of a system, such as, privacy, risk taking, and timeliness. Goal models show overall goals of a system at a high level. Quality goals make complex and ambiguous constructs visible in the models, making them available for discussion by the design team. Roles are attached to goals, and indicate who is responsible for achieving that goal. While goals retain ambiguity through quality goals, roles allow different members of the design team to identify different alternatives on how goals might be achieved, based on their own experiences. Role models comprise a name, description, list of responsibilities, and list of constraints. The concept of roles is particularly useful in modelling socio-technical systems, as an analogy can be drawn between agents and human actors. Agents take on one or more roles within the social situation being modelled and are usually responsible for a set of goals. Role models capture a lot of detail, which is an advantage when eliciting requirements and initiating discussions between ethnographers, interaction designers and software engineers about the different responsibilities and constraints of the roles that people play that in the situations being modelled.

*Figure 1. Requirements and Analysis Modelling Notation: a) Goal, b) Quality goal, and c) Role*

---

(a) Goal: to be fulfilled by the agent system. Roles often have responsibility for specific goals.

(b) Quality goal: or “soft” goal, are attached to goals and may be addressed by a role’s constraints.

(c) Role: has responsibility and constraints. An agent may have many roles.
The notation used includes: goals represented in diagrams as parallelograms, with goal names written in the centre (Figure 1a.); quality goals, attached to goal models, drawn as clouds, with names in the centre (Figure 1b.); and roles represented as stick figures with the name of the role written underneath it, indicating their analogy to agent/human roles (Figure 1c.). These three different constructs are connected in modelling diagrams by lines indicating their relationships to each other.

**Goal Model Diagrams**

The *Secret Touch* highest-level goal model diagram, *Mediate Intimacy*, is shown in figure 2. There are two sub-goals, *Flirt* and *Communicate*. The *Flirt* sub-goal (see Figure 2), has the *Partner* role associated with it, as well as the quality goals of *Risky* and *Playful*. The *Flirt* goal is achieved using the sub-goal *Initiate Flirt*. A *Touch Initiator* role describes how a flirt should be initiated, leading to the achievement of the sub-goal, *Capture Touch* which has the quality goal of being *Timely*, and constraints associated with capturing the touch defined by the *Touch Perceiver* role. Guided by the *Device Manager* role, the goal of *Translate Movement Into Touch* has the quality goal of being *Accurate*. The other outcome of a *Flirt*, (as seen in Figure 2), is that it can be responded to using a *Respond To Flirt* goal. Responding is important for the intelligently learning “Fiddler’s Choice” feature of the system (as described in Table 2). The *Touch Responder* role is responsible for how it is done. There are two sub-goals associated with the goal, *Recognise Incoming Touch* and *Propose Touch Sequence*. A *Choreographer* role is responsible for both sub-goals. The *Choreographer* recognizes an incoming touch and proposes the appropriate counter-movement. The response must have the qualities of being *Matching* and

---

**Figure 2. Secret Touch Requirements Model, with sub-goals Flirt and Communicate**
Appropriate, as it builds on habits that often develop within relationships, for example, finishing a telephone call with “Love You” instead of “Bye”.

The Communicate sub-goal, (see Figure 2), described in general terms, involves noticing incoming touches in a timely manner, and accepting and responding to those incoming touches. It is controlled by a communication manager. Remembering incoming touches is available only in the “Fiddler’s Choice” design. Remembering past touches is done using a library, and based on past responses, either rejecting the touch, or responding to it by translating that touch into movement using a device manager, while ensuring that the movement response is accurate. Alternatively, the system can mediate a touch exchange, in a discrete, fair and appropriate manner, available in the “Discrete Flirt” feature, as controlled by an intimacy mediator. Finally, a touch can be given by a touch giver.

The goals represented in the Secret Touch requirements model are at a high level, but easy to understand as the goal names used are familiar social concepts. Ambiguous concepts in the social interactions, such as risky, playful, and appropriate, are captured using quality goals and are available for discussions in the design team, with respect to associated goals.

Role Model Diagrams

Role Models are specified in table form, inspired by the REBEL toolkit (Sterling & Taveter, 2009). These role models represented important details of envisioned system behaviour. Details of the role model for Touch Acceptor are shown in Figure 3. The Touch Acceptor role, (seen in Figure 2), is associated with sub-goals of Notice Incoming Touch and Respond To Incoming Touch, in the Communicate section of the requirements model. The Touch Acceptor role model shows the list of responsibilities required to enact accepting touches when they are noticed, and responding to them, using a “Responsibilities List” (as shown in Figure 3). These responsibilities include: “Notice incoming touch”; “Check partner availability”; “Accept/Reject/Remember a touch”; and “If appropriate, inform of touch’s fate”.

The “Constraints List”, shown in Figure 4, is used to describe how a goal might be achieved, including: “If partner is currently openly flirting, pass touch on to be felt”; “If partner is currently interactively available, inform of accepted touch to be felt”; “If partner is currently passively available, inform of accepted touch to be remembered”; and “If partner is unavailable, reject touch”. The Touch Acceptor role model gives the design team information needed to ideate design ideas for dealing with incoming touches, and discussing how the associated quality goal, Timely, might influence that design. Thus, the role model construct facilitates design team discussions around an ambiguous concept while maintaining the ability to withholding design commitment. Actions resulting from partner availability are defined, but the
degree of timeliness of these responses, that is, the quality of the response, remains open for alternative design solutions.

The lists of responsibilities and constraints attached to role models proved to be a useful communication tool for achieving a shared understanding between ethnographers, interaction designers and software engineers. They encouraged each team member to offer different interpretations of quality attributes and their impact on design. For ethnographers, it involved retaining strong links to social interactions observed in the field data. For interaction designers, it involved bringing human-centred design guidelines and usability considerations into the discussion. For software engineers, it involved sharing expertise on software implementation and technology capabilities. Role models, associated with goals and quality goals, acted as shared artefacts used to generate lively discussion within the design team during requirements elicitation sessions. The coming together of different disciplines, experiences and understandings, resulted in the emergence of additional quality goals.
**Detailed Design**

Agent-oriented design models, including scenario and agent diagrams, were used in detailed design of the *Secret Touch* system. Plans and capabilities, which are sets of functions and abilities within each agent, and any messaging internal to that agent, were created. Each plan represents a subset of functionality that becomes a chunk of code in implementation. Capabilities include the sub-capabilities, plans and messages needed for agents to fulfil their functionality. The relationship between plans and capabilities are shown in an agent overview diagram. A plan may include several alternate versions, based on different triggers or pre-conditions. The detailed design serves as instructions for implementation. As an exemplar of the detailed design modelling for *Secret Touch*, one set of high-level design diagrams are presented, including detailed design agent overview diagrams for one agent.

**Design Notation**

The constructs of design modelling used include: Agents, Protocols, Percepts, and Actions. Agents are the central component of an agent oriented system, represented by rectangles containing stick figures (Figure 5a.). They act as humans might, sometimes even representing the roles and behaviours of people. Protocols are represented by a double-ended arrows and define the ways in which agents communicate, containing the communication patterns between agents (Figure 5b.). Percepts and actions represent interactions with the external environment. Percepts represent interactions from the outside world that cause the system to react in some way, diagrammed as splats (Figure 5c.). Actions represent how the system acts outwardly upon the world, diagrammed as arrows (Figure 5d.). In the notation used, connecting lines indicate relationships, and arrows on the ends of connecting lines show the direction of the interactions. The models show inter-agent interactions within a system, via protocols. They also show system boundaries through actions and percepts related to each agent.

**Scenario Diagrams**

Scenarios have tabular constructs to represent their descriptions in design modelling. These scenarios are built from goals, actions, percepts and sub-scenarios. They are different from motivational scenarios created during requirements modelling. Actions associated with scenarios indicate how the system acts on the external world. Percepts associated with scenarios indicate how the system reacts to events in the outside world. Scenarios are useful during consolidation of roles into corresponding agents. Interactions between agents are based on scenarios. The role responsible
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

Figure 5. Design Notation: Agent, Protocol, Percept, Action

for each step in a scenario is documented in the table. Scenarios are useful during the process of consolidating sets of roles into corresponding agents. Interactions between agents are based on scenarios, and interaction diagrams are created after coalescing roles into agents.

The *Tug-of-War* design scenario from *Secret Touch* is shown in Figure 6. The artefact was defined through discussions in the design team involving the translation of quality goals into the underlying details of these scenarios. The design team considered ambiguous concepts associated with quality goals, attached to system goals, during creation of these design scenarios. Additionally, design scenarios remain flexible, and open to change through their descriptions. At the highest level, the *Tug-of-War* design scenario consists of a back and forth between giving touches in the *Flirt* sub-scenario and receiving them in the *Feel Touch* sub-scenario, in any order, at any time. Each of these sub-scenarios is detailed in a similar form, defining related goals, actions and roles.

For example, the *Feel Touch* scenario, as illustrated in Figure 8, involves the activation of the sub-scenario of *Notice Touch*, which involves the role of *Touch Acceptor* (detailed in Figures 3 and 4). It involves the goal of *Translate Touch Into Movement*, which the *Device Manager* role is responsible for, (as seen in Figure 2). The goal of *Translate Touch Into Movement* carries with it the quality goal of *Accurate*, ensuring that touch responses are authentic and genuine. The role *Device
Manager is also associated with the subsequent action of Move Device. In the final system, when a touch is noticed by the device, it is translated into movement in an accurate manner, and the device then moves (vibrates), to indicate that a partner has sent a touch.

Using scenarios in the detailed design process ensured continuity from the requirements role and goal models, through creation of these scenarios, to defining software agents of the final system. Goals defined during requirements elicitation, and their associated quality goals, were included in the definitions of design scenarios. Thus, quality goals remained in an undischarged form and continued to represent ambiguous concepts into the final agent and interaction diagramming phases.

Agent Models

The detailed design process involved assigning roles to agents, where one agent had one or several roles. Different features of the system, as defined in Table 2, required different sets of roles and agents. During assignment, Agent-Role coupling diagrams acted as shared artefacts for design team discussions. Software engineers presented final system designs using these diagrams. Agent models were a great communication tool, as both ethnographers and interaction designers could clearly see analogies between the roles of the agents, and the roles and responsibilities of people in social interactions. Quality attributes attached to goals informed team discussions during allocation of roles to agents, enabling checking of role assignments with respect to qualities of the social interactions being modelled.

The Secret Touch system has four agents, each corresponding to a system interface to the external environment: Device Handler, Intimacy Handler, Partner Handler, and Resource Handler. These are required in the different system features. The simplest feature, “Flirt”, consists of the Device Handler agent interfacing to the physical device and the Intimacy Handler agent interfacing to the partner system. The Intimacy Handler agent (detailed in Figure 8) has the Touch Acceptor role assigned
to it, because the agent is responsible for noticing touches, and ensuring that they are responded to in an Accurate manner. Accurate is the quality goal carried into agent modelling through the Feel Touch scenario (Figure 7), by virtue of its direct association with the Translate Touch Into Movement goal. The Intimacy Handler agent also handles the roles of Evaluator, Touch Giver, Communication Manager and Intimacy Mediator.

More complex features of the device require the other agents. The Partner Handler agent is used in the “Discrete Flirt” feature, so that the user can indicate availability to their partner. The Resource Handler agent is used in the intelligent feature, “Fiddler’s Choice”, because it requires access to a library (knowledge base).

**Design Model**

The detailed design model of Secret Touch is presented in Figure 10. The coupling between agents can be seen by the links between agents via protocols, where each protocol contains the inter-agent messages for connected agents. The Intimacy Handler agent is at the centre of the system. The Exchanges protocol contains the basic inter-agent communications needed to flirt and enables interaction between the Intimacy Handler agent and the Device Handler agent. The Discretion protocol adds the ability to change availability, thus communicating between the Intimacy Handler agent and the Partner Handler agent. Discrete, an important quality goal in the requirements model associated with the Mediate Touch Exchange goal (see Figure 2) finds its way into the design model through the Discretion protocol, indicating that agents must communicate with discretion. The design model also shows the Intelligent protocol influencing communication between the Intimacy Handler agent.
and both the Resource Handler and Partner Handler. These agents are involved in the “Fiddlers Choice” feature, and rely on information about partner availability or past touches when deciding how to respond. The percepts and actions in the design model represent interactions with the external environment. For example, the Arrival of Touch percept associated with the Intimacy Handler agent represents an incoming touch, while the Give A Touch action represents an outgoing touch. For the remainder of the process, example sequences of interactions between agents are documented using interaction diagrams, which are similar to UML interaction diagrams in OOSE. From the interaction diagrams, protocols of inter-agent messaging are built, and include all possible variations in communication between agents. The system can then be implemented from this detailed design. This part of the process has not been described here but can be found in Paay et al. (2010).

In summary, in designing the Secret Touch system, cultural probes were used to gather data on how people appropriate existing technologies to support their intimate relationships. A mixed team of ethnographers, interaction designers and software engineers then analysed the data. During the process of translating cultural probe data to system requirements, we found that coordinative and collaborative goals were not sufficient to model social interactions between people in the domestic space, because there are a range of purely communicative activities that do not serve any
productive purpose, for example, communication for the sake of pleasure. Instead, we discovered that the use of quality goals attached to system goals enabled the design team to model social needs while keeping them at a high level of abstraction, hence ambiguous and open for conversations throughout the modelling process. Quality goals added a richness to the goal models that supported conversations about ambiguous concepts related to social interactions, throughout the design phase. In this way, ambiguity became a resource for the design team in achieving a shared understanding of user needs in the final system requirements.

PUTTING EMOTION INTO SOFTWARE ENGINEERING

Scenarios as they are currently used in software engineering lack the kinds of emotional aspects often associated with social interactions. Carrol (1997) refers to the inclusion of emotions in scenarios as “a condition of comprehension”, rather than just something that would be nice to have. Hence, it is important when understanding the user and their intentions through scenarios, that some component representing people’s emotions is incorporated into scenarios. Understanding people’s emotions in a situation increases understanding of it, and hence the ability to build technologies that suit their context.

Considering emotions is particularly important when modelling social interactions in a non-work context, as in the case of Secret Touch. It is not straightforward to model emotions, however, because emotions felt by a specific user can vary based on their cultural background, age, gender, belief systems and living situations. Our experience suggests that it is better to link emotions to individual roles and goals acting within a scenario, rather than represent emotions in the scenario itself (Lopez-Lorca et al., 2014).

We have extended our experience through modelling a range of situations. The mediating intimacy project was followed by a project which developed models which guided the building of software systems to encourage grandparents and grandchildren to have fun over the internet (Davis et al., 2007). A subsequent application domain was the design of emergency alarm monitor systems to help older people wishing to live at home. In this domain, emotions of the older people are critical (Pedell et al., 2014).

Continuing from the aged care case study discussed in Pedell et al. (2014), we have used quality goals in modelling in several other domains, especially in the area of mental health. Wachtler et al. (2018) describes the development of a screening app to match people with current depressive symptoms with appropriate treatment recommendations. Thomas et al. (2016) describes the development and pilot study of a novel digital intervention to promote personal recovery in people with persisting
Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

psychotic disorders. In this case, the goal models were used during the requirements elicitation phase which involved different groups of stakeholders including mental health workers, people suffering from psychosis, psychologists and the families of the patients. The high level of abstraction of the goal models made them accessible for stakeholders without a technical background and acted as a boundary object to support interactions between the development team and clients. Sterling et al. (2017) describes the development of SleepWell, an app to help people with sleep difficulties. The most recent example of use was in tele-audiology as described in Sterling et al. (2018).

Quality goals and goal models are now routinely used in interaction design and software engineering teaching at tertiary education level (Marshall, 2018). The models have been incorporated with requirements elicitation methods and incorporated in agile development, within a semester-long requirements elicitation subject, and within a year-long software engineering project subject. While the models were inspired by agent concepts, students are able to understand and build the models without any special training in agents. The high-level models are intuitive and easy to understand from our experience.

CONCLUSION

Software engineering of socio-technical systems is difficult, because people’s social interactions cannot be easily decomposed into measurable requirements. The domestic space gives rise to a range of purely communicative activities, which do not necessarily serve any external productive purpose. During system design, it is essential to capture social concepts accurately but flexibly at a high level, without losing the liveliness and vitality of those concepts through over specification. Modelling socially nuanced requirements while maintaining the richness of social interactions and generating concrete requirements ready for software implementation is problematic. Communicating ambiguous concepts using quality goals is an important part of solving the problem. Quality goals can remain unresolved for as long as possible, without committing to a specific design solution. Requirements and design models, incorporating quality goals and scenarios, embrace ambiguity a resource for design teams, prompting ongoing cooperation and communication around design alternatives and achieve a shared understanding of both user needs and final system requirements. In the Secret Touch case study presented, the ambiguous concepts were made explicit in the process and accessible for team discussion. The use of AOSE notations also supported effective communication and translation of complex and subtle social interactions into the modelling process, helping to retain the essence of a couple’s intimate social interactions.
Detailed requirements, analysis and design modelling of the *Secret Touch* system illustrated how data from a cultural probes study exploring social interactions between couples, can be translated to create design requirements for a technology that supports them keeping in touch with each other. Quality goals were used to contain ambiguous concepts found in the social data. These quality goals remained visible and traceable throughout the modelling process, giving the design team focal points for discussions on representative modelling and design alternatives. By combining expertise and techniques from ethnography, interaction design and software engineering, the interdisciplinary team was well placed to translate data collected about people’s appropriation of existing technologies in maintaining their close relationships, into system requirements for new and novel technology supporting social interactions.

**ACKNOWLEDGMENT**

The authors acknowledge the contributions of Professor Steve Howard to this chapter. He participated in the field work, analysis, and the interdisciplinary design team for this research, as well as the writing of earlier versions. Sadly, Steve passed away unexpectedly in April 2013, but lives on in his words and in the legacy of all the HCI researchers that he guided, nurtured and inspired.

This project was supported by Smart Internet CRC [project UE-06].

**REFERENCES**


Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models


Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models


Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models


Interdisciplinary Design Teams Translating Ethnographic Field Data Into Design Models

ADDITIONAL READING


KEY TERMS AND DEFINITIONS

**Agent:** A software agent is a computer program that acts for a user or other program in a system.

**Ambiguity:** The quality of being open to more than one interpretation.

**Cultural Probe:** A technique used to gather fragmented data about people’s lives, values, and thoughts for the purpose of inspiring ideas in a design process.

**Ethnography:** The study of people and cultures, traditionally through observational methods.

**Interaction Design:** The practice of designing interactive digital products, environments, systems, and services.

**Interdisciplinary:** Crossing traditional boundaries between academic disciplines.

**Intimacy:** Close familiarity or friendship.

**Requirements Engineering:** The process of defining, documenting, and maintaining requirements for systems and software engineering.

**Socio-Technical System:** Systems that include technical components but also operational processes and people who use and interact with the technical system.

**Strong-Tie Relationships:** Strong-tie relationships are intimate, well developed, with frequent interactions, such as experienced in families or married couples.

**User-Centered Design:** A design process that focuses on the end user, through understanding usability goals, user characteristics, environment, tasks, and workflow in the design of an interface.