The ORDIT Approach to Requirements Identification

J.E. Dobson, A.J.C. Blyth, J. Chudge and M.R. Strens
Department of Computing Science
University of Newcastle upon Tyne
NEWCASTLE UPON TYNE NE1 7RU
United Kingdom

Abstract
The most crucial aspect of software engineering is the gathering of requirements. Increasing the amount of time and improving the quality of effort expended at this stage will lead to fewer and more manageable problems later on in the development life-cycle. In this paper we describe the ORDIT approach to requirements identification and expression within the context of organisational change. ORDIT focuses on the representation of organisational requirements in the design of socio-technical systems which are intended to emphasise the relationships between organisational structure and information technology (IT) systems.

1: Introduction
1.1: The ORDIT project

Conventional systems analysis has largely focused on defining information processing requirements rather than looking at IT from a wider perspective, and so it is common for systems to be created which do not satisfy the needs of their human operators although they are technically sound. Therefore it is often the case that technical solutions to problems (the so-called 'hard' systems) are created which do not adequately support the way in which the human components of the work system are organised. This problem is partly addressed by the soft systems approach [2], but is taken a step further by the ORDIT project which takes a socio-technical approach in which the system is viewed as a whole by placing it within the broad operational environment with the user as an integral part of the system.

ORDIT, which is an Esprit II project, includes collaborators from industry and academia with expertise both in human factors and organisational issues, and software design and engineering (Algotech srl, HUSAT Research Institute, the MARI Group Ltd, the Department of Computing Science of the University of Newcastle upon Tyne, and Work Research Centre). In this paper we describe the ORDIT (Organisational Requirements Definition for Information Technology) methodology for the design of socio-technical systems [5] and discuss our experience in three years of its development and application using the case-study approach.

Designers have generally ignored the importance of organisational issues in the design of IT products, and hence many of the difficulties encountered have been due not to limitations in technology but to the disregard of organisational requirements. It is only fairly recently that organisational requirements, while acknowledged to be essential for successful implementation of socio-technical systems, have become an explicit focus of attention in systems design. The central tenet of the ORDIT philosophy is that design methods appropriate for technical systems cannot simply be applied to socio-technical ones, and that consideration must be given equally to both human and technical issues, with success being seen as the construction of a relevant socio-technical system that meets the 'real' requirements of the organisation.

Hence the aim of the ORDIT project is to develop a methodology that will enable systems designers to reason about organisational goals, policies and structures, and the work roles of intended end users in a way which will facilitate the identification and expression of organisational requirements for IT systems, and furthermore, one which will support these structures and roles. Our concern therefore is not solely with the creation of a framework in which such issues can be identified, but also with the development of a language with which to discuss human requirements of socio-technical systems, and to demonstrate how these are linked to the technical features of the system design.

We believe that our approach addresses the problems that arise when designing large and complex systems. The areas of complexity are both 'technical' and 'organisational'. Technical complexity is fairly well understood, the principal tool being that of abstraction whereby the overall picture is constructed, with more detail being added as the design process proceeds. Organisational complexity, however, is another matter. The traditional notion of the software development life-cycle with requirements capture being completed before the design stage is no longer satisfactory. Requirements capture and design are now seen to be symbiotic. The initial set of requirements needed to start off the design process is gradually refined into a systematic and coherent statement of requirements hand in hand with the refinement of the design.

1.2: Components of the ORDIT methodology

There are five main components of the ORDIT methodology:
A Process Model. This is a model of the process of eliciting and modelling requirements. One of the main characteristics of the ORDIT process model is the way it has separated these two functions and shown the relation between them. Requirements are not considered as butterflies: they are not there to be captured and pinned down in a specification cabinet. Rather, the process of finding them is one involving at least three roles (requirements owner, requirements elictor, requirements modeller) and a number of separate tasks involving some complex feedback loops.

An Enterprise Modelling Language. The ORDIT project has devised an enterprise modelling language to represent the structure of the organisation in order to serve two related but distinct purposes: to determine the requirements owners and their positions and roles within the organisation; and to determine the user community (and others affected by the proposed IT system) and their roles and responsibilities within the organisation. The first purpose is in order to demonstrate completeness of the requirements elicitation process, and the second is required in order to demonstrate completeness of the requirements modelling process.

An Information Modelling Language. The purpose of the information modelling language is not only to derive the data models that are required, but also to determine the 'non-functional' requirements (such as security and privacy) on the data, so that the data needs can be shown to be complete and fit for purpose in terms of the organisational responsibilities of the data owners and users.

A Role Reference Model. The ORDIT concept of 'role' is perhaps the most sophisticated of all current role models, covering such things as functional and structural relationships, responsibilities, information access modes and rights, conversation structures and role evaluation criteria. The concept of the Role Reference Model has been developed by ORDIT in order to capture all these aspects of 'role' that are so important for a full definition of the organisational requirements placed on a system by the roles with which the system will interact.

Supporting Tools. The ORDIT methodology is supported by a number of modelling tools which combine the power of hypertext-like structures with easy to use graphical interfaces and logical and analytic power. The development, prototyping and evaluation of these tools is a major feature of years 4 and 5 of the project.

In this paper we shall describe only the first two of these components.

2: Requirements engineering

2.1: Setting the context for organisational requirements

Within organisations, large tasks tend to be devolved to groups of people who work together in complex ways to achieve an overall objective. This has always been the case, and yet technical systems design tends to assume a single user with a discrete task. The failure to recognise that users work in a collaborative or co-operative way, and to design systems to support this way of working, can account for the relatively low success rates of many complex technical systems. One of the aims of the ORDIT methodology is therefore to enable design teams to address these organisational requirements, and thereby to produce IT systems that match not only the organisational and functional needs of the individual end user, but also those of groups of users and their associated usability and acceptability requirements.

The requirements identified in order to achieve the aim of supporting different members of the design team are as follows:

* to identify the full range of relevant requirements in a specific organisational context;
* to derive appropriate functional specifications for any IT systems which take account of organisational as well as individual task requirements;
* to compare the organisational requirements match of different design alternatives;
* to represent the range of organisational requirements to the problem owner and the systems designers in an iterative way;
* to identify organisational mechanisms or processes for fulfilling critical non-functional requirements.

The method used to determine requirements must allow the system designer to explore possible solutions (involving both the IT system and possible organisational change) and their consequences at the same time as specifying the problem, thereby refining the understanding of the problem and developing the solution by an iterative process. We have found it convenient to represent the general requirements process within ORDIT as four broad interactive component subprocesses, namely scoping, modelling, requirements capture and solution options, as shown in Figure 1.

![Figure 1](image-url)

The important point to note is that there is no set route through the diagram, the whole process being iterative.
with feedback to the client at every stage. For example, in contrast to other methods, modelling may be started at a very early stage to help in exploring the system boundaries and in identifying stakeholders. This process is in contrast to the traditional ‘waterfall’ approach to modelling in which the output from one stage forms the input to the next stage and so on, with all the stages following a pre-determined order. The objective of the ORDIT model is to support the identification and transformation of organisational requirements into precise statements which can be operated upon by systems designers without being prescriptive as to the order in which the various operations involved in this process are carried out.

2.2 The requirements process

The four major sections in Figure 1 outline a process for arriving at a set of organisational requirements on an information technology system.

Scoping

This component subprocess deals with determining the scope of the requirements determination team and the contractual agreement with the client, establishing boundaries for the contract, gaining an understanding of the purpose and structure of the organisational unit(s) which are to be involved, and identifying the principal stakeholders involved.

System Model

The purpose of this component subprocess is to represent the current understanding of the socio-technical system by producing a set of models. This provides not only information about the environment in which the IT system is to function, but also a context for understanding later policy and design decisions. One particularly important use of the description is to act as a focus for discussion on what sort of system the new system should be, and on how responsibilities and authorities in the organisation are going to change as a result of introducing the new system. Another important function is to assist in the scoping subprocess, by providing models for the determination of relevant stakeholders and system boundaries.

The existing system is described in terms of the system itself and its organisational environment and then agreed with the problem owners. The next step is to model the existing system in terms of abstract agencies (see Section 3) and a general information and computational system. In addition, models are annotated with relevant requirements policies concerning issues such as combinations of agents into roles, access, authorisation, scope and objectives.

User Requirements

The purpose of this component subprocess is to identify those organisational requirements which must be observed in any implementation. Natural language statements of requirements are elicited from problem owners in order to agree upon how the system should be rather than how it is, and to identify defects in the current system and how these might be overcome. Statements of requirements are then fed back to problem owners and agreed. Requirements are then classified in order to identify conflicts, particularly conflicts in definitions of boundary objects and organisational aspects. It is an important feature of the ORDIT methodology that this stage of the requirements process is more concerned with the determination of what organisational responsibilities need to be supported by the IT system than what functions the IT system is to perform. Experience in a number of case studies has convinced us that taking a functional or activity-based view of a system at this stage leads to significantly lower quality (i.e. less ‘fit-for-purpose’) requirements than taking a view of a system as supporting contractual and work-related obligations.

Solution Options

Requirements and some priority ordering are used to generate possible design options for the socio-technical system, with conflicts and trade-offs being resolved with the client. One of the purposes of the earlier stakeholder analysis is to answer not only the question of how and by what criteria to prioritise stakeholders, but also whether different prioritisations are needed for different purposes (e.g. data collection, validation/feedback, policy decisions). The implications of the design options are analysed and discussed with relevant stakeholders in an iterative fashion. The acceptability of the preferred option is agreed with the problem owners and other stakeholders, ensuring that the option meets the formal model of requirements.

The ‘space’ in which solution options are discussed is a socio-technical space, that is it encompasses possible changes in the organisation and new organisational constructs as well as changes in the technical system and new IT constructs. These must be considered together. For example, an organisation which wishes to improve its repair service carried out at customer premises might wish to consider making its repair service an integral part of its operations, or a separate organisation (wholly-owned or independent), or make its repairers a set of independent agents with a common coordinating service to which they all subscribe (the taxi-driver model). Each of these options will place its own distinctive set of requirements on an IT system to support the requesting, scheduling, use and monitoring of the repair service.

This is the final step in problem space of requirements analysis. Subsequent stages consist of a response by the system architects and designers to the requirements, and are therefore expressed in solution space terms.

3: Enterprise modelling

The modelling concepts are perhaps the key aspect that makes ORDIT different from more conventional approaches to design. Enterprise modelling provides a framework for representing and reasoning about the IT system as a component of a wider environment which is the organisation whose needs it is designed to serve. We have found that this form of modelling ensures an adequate representation of the structural and organisational aspects of the problem making explicit policy issues and assumptions which cannot so easily be stated. One of the main characteristics of the ORDIT approach is that we model responsibilities and relationships rather than activities and base our understanding of the architectural framework on these models and not on the current implementation. In an analogous way to this we model
information from the contractual view rather than the current structure of the information flow.

The ORDIT project has devised a diagrammatic enterprise modelling language to represent the structure of the organisation [3] in order to serve two related but distinct purposes: 1) to identify the requirements owners and their positions and roles within the organisation in order to demonstrate completeness of the requirements elicitation process, and 2) to identify the user community (and others affected by the proposed IT system) and their roles and responsibilities within the organisation in order to demonstrate completeness of the requirements modelling process.

3.1: Basic concepts of the enterprise model

The three essential elements of the Enterprise Model are agents, actions and resources. The relations between these entities are defined in Figure 2 [4]. Agencies (embodied in agents) are regarded as primary manipulators of the state or structure of the system, and agency is the only object that can create, modify or destroy other objects. Actions are the operations that change the state of the system, and they are performed by agencies. All actions must induce state changes in the system that are visible to one or more agencies. The resources can be of two types: physical or logical, where physical resources are tangible objects such as nuts and bolts, and logical resources include information, time etc. When modelling organisations at the enterprise projection level [1], resources act either as tokens of responsibility signifying that an agency has a binding responsibility upon it, or as objects for which some agency is responsible. An important type of logical resource is data. When data are passed from one action to another interactions occur, the data being the bearer of those interactions.

The enterprise model also provides us with a framework from within which it is possible to examine the kinds of relations between entities of the same type and of different types. By examining these relationships a problem owner and solver can begin to understand how the basic objects influence and are related to each other. In our enterprise modelling we have chosen to concentrate on the concepts of 'role' and 'agency' as being central to our procedure.

3.2: The concept of role

We choose to describe an organisation as a set of related work roles for the following reasons:

1. A role is a descriptive concept that can be used to represent many different organisational realities from the formal and structured to the fluid and unstructured.
2. Treating role as a basic building block makes it possible to move between organisational requirements and the requirements of individual users (e.g. from the organisation's role in a project to the way these responsibilities devolve to the roles of members of the project team).
3. A role defines task responsibilities and thereby functionality requirements.
4. A role defines the relationships between role holders and the behaviour they expect of one another which in turn defines many non-functional requirements.

Hence our concept of role allows us to distinguish: a) an agent with associated obligations such as accountabilities and responsibilities to other agents; and b) activities that interact through information flows and are structured into tasks and operations. This enables us to represent and analyse the relations between these concepts and to represent the way in which they operate in real organisations.

Figure 3 is an example of the basic building block of our enterprise modelling diagrams, which shows the roles involved in repairing a customer's appliance. We have used these diagrams in an actual case study to discuss options for reorganising an electricity supply company's policy on the future organisation of its customer premises repair service.

The main point to note about this diagram is that each box contains the names of what we term a functional role and a structural role. The functional roles (problem owner, problem repair) are shorthand for the framework of obligations that permit and give meaning to these behaviours.

The lines that join the functional and the structural roles represent the fact that a relation between two agents may be one of interaction or one of commitment. Commitments and interactions are related to each other through activities such that commitments only arise and are fulfilled through interactions. So, properly speaking, the links between them are in fact functional and structural relationships.
Figure 3 thus embodies the policy that the repair is effected by the supplier. An alternative scenario, in which the problem repair is effected by an approved independent agency might be represented by the following diagram, Figure 4.

Finally, the following diagram represents the policy that the problem repair is effected by a separate agency within the supplying organisation. Note that we have used nesting to represent the organisational relationships involved.

3.3: The concept of agency

ORDIT aims to describe and reason about organisations that embody both a social and a technical system. These however comprise one single system, a socio-technical system, and, as such, cannot be described or modelled in terms of state and behaviour only as a purely technical system might be, since there is a fundamental difference between social and technical systems. It is to be able to differentiate between social and technical objects (i.e. between people and computers) that we introduce the idea of agency. A machine may perform the same tasks as a person, but the person will hold responsibilities for those tasks in contrast to the machine which cannot hold responsibility. The person is said to be an agent and hold the agency.

It is important to realise that an agent is distinct from both an individual human and a role. An agent holds the particular set of responsibilities that comprise an agency. Thus depending on how responsibilities in a social system are allocated and combined, so agencies are composed and decomposed. An agent also differs from an individual in that an individual may hold more than one agency simultaneously. An agent differs from a role in that a role is not merely an agency or a collection of agencies but also includes a set of relationships with other agents. These are structural or social in nature, arising from responsibilities that relate to the other agents.

This concept of agency is one of the strengths of the ORDIT approach to reorganisation of a socio-technical system, since it facilitates the reallocation of agency in a way that takes into account as fully as possible the structural and organisational implications of the change. Since agency is considered as a coherent set of responsibilities, it permits the discussion of issues related to the change in and reallocation of responsibilities when some functions or agents in the system are proposed to be automated.

3.4: Decomposition

A marked advantage of our notation is the way in which we can compose and decompose our role-relation diagrams for the purposes of ascertaining requirements at the levels of agent (or individual), groups (or collections of distributed agencies), and organisations (policy holders). It is through using the features of role-relation diagrams in this way that we are able to specify the differences between relationships. In order to achieve this, it is essential that our diagrams (de)compose so that we can view the organisation at whatever level best fits the purpose. Furthermore, an advantage of this abstraction is the freedom it gives for the binding between the abstract and the concrete. Our use of the abstract term 'agency', for example, is deliberate so that we can discuss who or what corresponds to the agency. For instance, it allows us to investigate differing bindings and boundaries associated within agencies encapsulated within a role.

It may be the case that binding turns out to be a fundamental policy issue — for example, it might be agreed that there should be a beneficiary from the existence of a market (e.g. the market landlord), but in the case of a public utility supply market, is this beneficiary to be the public state or the private shareholder? Similarly for boundary drawing: it is useful to distinguish between for example the diagnosing agent and the prescribing agent in medical health care, but are these to be combined into one role (as in the U.K. primary health care service) or separated (as in certain secondary health care services)?

An important point that this serves to highlight is that differing requirements are expressed at differing levels within an organisation. At the organisation level the most important requirements may be to do with policy issues, at a group level they may be concerned with being able to support collaborative work in an efficient manner, and at
an individual level priority may be given to job satisfaction. We believe that enterprise modelling is capable not only of capturing what may be conflicting requirements within an organisation's structure but that it can represent these qualitatively different requirements at their appropriate levels and then discuss issues of their conflict resolution, compatibility, desirability and so on.

4: Summary

We have found that our experience in ORDIT of constructing enterprise models and determining their bindings and boundaries has aided us in suggesting issues of job design and exploring organisational issues in a manner that pays heed to an organisation's requirements, goals, and policies. For example, the case presented, where the choice between direct contact between the customer and the repair agency, or indirect contact, with the repair agency being either internal or external to the supplier, is an organisational choice that can easily be explained, the implications on job design, responsibilities and information systems structure having been explored using the diagramming techniques we have outlined. The problem of determining system boundaries of complex IT systems has meant that mistakes have occurred where the boundaries turned out to have been drawn in the wrong place. We claim that our modelling techniques provide a sufficiently rich environment in which organisational structure and the roles of agents, information flow, resource management, and the relationships between all of these are capable of being represented, and we are therefore able to capture the complexities of organisational structure.

The most common problem of requirements engineering in the design and implementation of complex IT systems is combining differing representations of the system and its environments; the operational, organisational, and social environments of a system all possess different characteristics. Hence the driving thrust of the ORDIT philosophy is its advocacy of involving policy makers/problem owners throughout the design of the system. It is a process of shifting the balance of responsibility between system owner and system designer away from the 'owner states, designer solves' model towards a relationship in which the problem solver helps the problem owner understand the problem, and the problem owner helps the problem solver understand the implications of possible solutions.

To sum up, the ORDIT approach recognises that the elicitation, representation, and re-presentation of requirements is an iterative process that is best accomplished by a methodology that combines the social aspects of a system with the technical, and adheres to the principle of giving the customer what the customer needs and not what the system designers think the customer wants.

Acknowledgements

We gratefully acknowledge the contributions made by our colleagues in the ORDIT project to the ideas presented here. We also gratefully acknowledge the support of the University of Newcastle upon Tyne and financial support afforded by the ESPRIT programme of the CEC (Project 2301) and by SERC (Bainbridge Project).

References