# The Management of Socio-technical Systems using Configuration Modelling

Simon Lock

*Computing Department, Lancaster University. Lancaster, LA1 4YR*

*Phone: +44 (0) 1524 592792*

*Fax: +44(0) 1524 593608*

*lock@comp.lancs.ac.uk*

**Abstract**

In this paper we describe a new approach and support tool for the modelling and analysis of socio-technical system configurations. This novel approach has been developed for use on systems composed of a wide variety of different components including social and organisational elements, in addition to the more traditional software and hardware aspects. Configuration models of such systems are lightweight and quick to construct and can help to promote understanding by the various stakeholders involved in system development, operation and evolution. These models also provide the data required for performing various useful forms of automated analysis. The results of such analysis can allow managers, administrators, developers and end users to investigate various efficiency, productivity and dependability attributes of the current configuration of a system. This can help support decisions about the evolution of a system by allowing the assessment of proposed changes such as the addition or removal of components, processes and structures. In this paper we utilise a real world case study in order to demonstrate and evaluate the utility of the described approach.

**Keywords**

Socio-technical systems, Component Structure, Configuration modelling, Configuration analysis, System management, Dependability investigation

## 1. Introduction

This paper provides a description of Strider, an approach and support tool that permits the modelling and analysis of socio-technical system configurations. Dealing with such socio-technical systems is particularly challenging due to their often complex structure and diverse makeup that can consists of a wide variety of human, social, political, organisational, software and hardware components. Both the approach and support tool developed for this purpose are intended to be simple to use and lightweight enough so as to not significantly impinge on the work of the various parties involved in system engineering and operation. Such parties include managers, ethnographers, system designers, implementers, administrators and so on.

The problem of communication and knowledge transfer between the various stakeholders of a system is well known. We see this primarily as a problem of the translation and structuring of system information in order to make it more digestible to all parties. This transfer of information can be greatly assisted by the use of configuration models. Such models are particularly suitable for management level investigation and understanding, as it provides a high level appreciation of a system's nature. This ensures that managers are not forced into considering a system in unnecessary depth nor requiring them to comprehend the low level operation of the components and process which are involved.

A fundamental consideration in the development of Strider was a desire to avoid the modelling overhead involved in the capture of deep structure, complex concepts and dynamic behaviour of systems. Configurations only model the abstract “skin deep” nature of a system and thus they are relatively simplistic and lightweight to construct. It is thus our aim to attempt to assess exactly what can be achieved by modelling just the top level configuration on its own.

The simplistic nature of configurations ensures that effort required in the comprehension and understanding of these models is as low are possible. This makes such models an ideal concrete topic of conversation to support discussion and understanding between ethnographers, developers, end users and managers. To help support such discussion and understanding, support is also provided for the investigation of the configuration models through browsing, query and automated analysis features.

We base our configuration models on well established fundamental modelling concepts currently in use for the representation of purely technical systems. These include structures from entity-relational, control flow and data flow modelling. In order to be able to apply concepts from such approaches to the modelling of socio-technical configurations, a variety of abstractions, refinements and simplifications have been made.

The utilisation of the Strider approach has a number of efficiency, productivity and dependability implications for systems upon which it is applied. The act of modelling a configuration itself can be beneficial due to the increased understanding possible by all interested parties. Automated analysis also offers a more structured, complete and rigorous appreciation of the often complex features, properties and implications of a system and it's configuration.

Using this approach it is possible to identify bottlenecks and single points of failure within a system so that potential dependability weakness may be identified and appropriate action taken to rectify them. Alternatively, Strider can be used to help identify redundant or under-utilised elements of a system. This information can then be used in the process of streamlining and optimisation of the system in order to improve efficiency or reduce operational costs. Configuration models can not only assist managers in achieving an optimal and efficient system, but can also help ensure that enough redundancy, failsafes and backups exist to ensure dependable and resilient operation.

Finally, increased quality of operation is specifically supported by the identification of problems, weaknesses and anomalies within a system's configuration. We see the modelling and management of both social and technical aspects of a system's configuration as essential for ensuring the smooth operation of complex systems.

In this paper we first highlight the aims and objectives of the Strider approach. We then provide a background review of existing and relevant approaches that provide facilities for the modelling and analysis of system configurations. Next we define some of the key concepts and mechanisms that are at the core of the Strider approach and support tool. Following this we introduce a case study system that we shall apply the Strider approach upon. We then use this case study system to evaluate and demonstrate the abilities and facilities of the Strider approach. In so doing we discuss the results and the insight which have been gained into the case study system, with particular emphasis being placed upon issues of dependability.

### 1.1. Aim of the Strider approach

Strider is a pragmatic approach that is sensible to the difficulties of scale involved in the modelling and analysis of complex systems. The aim of Strider is to support the construction of simple system models based on natural language descriptions such as documentation, manuals, requirements definitions and ethnographic reports.

Due to the often observational and discursive nature of information available on such socio-technical systems, it is not always possible for solid and concrete models of all aspects of a system to be created. In addition to this, we may not be able to abstract, generalise and formalise implicit information or embedded knowledge enough to be able to model it adequately. For these reasons, it is essential that Strider is able to deal with abstract structures and concepts as well as the possibility of missing fragments of configuration information

Even if complete modelling of the systems were possible, in most complex systems the overhead of modelling such a large quantity of information would be prohibitive. The focus on modelling purely system configurations frees us from the laborious task of modelling the deep operational behaviour of a system. Instead we are able to focus on the lightweight structures of configuration models and the possibilities such models offers us for in-depth analysis and appreciation.

The models produced during the Strider process give managers, ethnographers, developers and end user access to in-depth information regarding the configuration of a system. Investigation and analysis of such models can provide illuminating insights into the structure and operation of the system. Particular emphasis is place on supporting the investigation of flexibility, dependability and diversity attributes of individual configurations. Strider permits the specification and comparison of different configurations of a system. Thus it is possible to model and assess the current configuration or any number of proposed future configurations to aid in the redesign and evolution of a system.

Strider aids managers in understanding the nature and operation of an existing configuration and allows them investigate the interplay between workers, technology, organisational structures and procedures. The models produced also allow managers to perform experimental re-configurations and observe their consequences, without the need for costly alteration and upheaval of the actual system or organisation.

Strider is very much a "thought support" tool that can be used to help an operator manipulate and investigate a socio-technical system. The results of such analysis require much interpretation and places significant emphasis on the operator's knowledge of the domain. In so doing, Strider achieves a division of labour, where the two partners (i.e. Strider tool and it's operator) specialise in the tasks to which they are most suited. The tool performs all complex and repetitive "bulk" data processing and the operator undertakes comprehension, collation and implication assessment.

This symbiotic partnership works very effectively since in many situations it is enough for Strider to know of the existence of a process, entity or structure from the domain, without maintaining any additional information about it. The operator of the tool can then provide the contextual and domain specific knowledge to fill in the gaps in the analysis. Thus the output of analysis has the affect of "keying in" to the domain concepts held in the mind of the operator and stakeholders.

## 2. Background

In this section, we provide some background information and a contextual setting to the modelling and analysis of socio-technical system configurations. In order to do so, we define some of the important concepts that underlie such activities and also discuss a number of existing approach which may provide suitable input to our task. In this section we also describe some existing approaches that appear potentially useful in the attainment of our modelling and analysis objectives. Many of the approaches that we consider have been developed for purely technical domains, such as software and hardware engineering. Considerable effort has to be expended in adapting concepts and processes from these techniques in order to apply them fruitfully to systems in a socio-technical context.

## 2.1. System configurations

Before we progress any further, it is first worth defining exactly what we mean by a "configuration". The concept of a configuration means many different things to many different people. Designers, developers, administrators, end users, managers and other system stakeholders all have a distinct idea of what a configuration means to them. In order to facilitate the development of the method and support tool described in this paper, we have attempt to reach a generic consensus of these various different perspectives.

Abstractly, we define a configuration to be all of the entities that are present in a system and the interrelationships between them. Such entities include people, buildings, organisations, documents, management structures, hardware, software, domain artifacts and so on. In these terms, a configuration can be viewed as a skeletal structure that is not concerned with how work processes are actually performed nor the internal attributes of atomic entities.

A configuration model has many similarities with schematic diagrams used in the construction of electronic hardware devices. Both record and present the components that form part of a system and the linkages between those components. In addition to this, only a structural representation is provided, with no consideration being given to dynamic operational qualities and behaviour of the system.

Configurations tend to be very flexible in nature due to the internal re-configurability and adaptability of both technical and social components of systems. For example, a particular worker in an organisation may be very flexible in their skill set and work activities and thus have considerable ability to adapt themselves to many different tasks. In addition to this internal flexibility, configurations may contain multiple "paths" to achieve desired objectives. This provides configurations with significant flexibility and resilience to the failure of single or groups of configuration items and processes.

A configuration may evolve significantly over time in order to adapt to changing environment, altered governmental legislation or shifting needs of an organisation. In some situations the configuration of a system may also be altered to improve it's operation in some way. This may be to optimise on time, money or resources or may be to increase system dependability and thus reduce the risk of configuration failure. This may take place after a complete failure has occurred in an attempt to prevent that failure from re-occurring.

Configurations can in fact evolve on their own, without change being driven by any external influences at all. Internal aspects such as natural wastage, replacement of parts of the configuration, learning by workers, wear and tear of assets, obsolescence of technology, staff turnover, retirement of workers, upgrade of technical systems and so on may all cause a significant amount of configuration drift. As long as we are aware of this fact and are able to update the system models, understand the changes and control the fluid motion of the system then serious problems should not arise.

**2.2. Configuration modelling mechanisms**

There are many different approaches that are concerned with the modelling of system configurations for a variety of engineering, management and analysis purposes. Such models support the activities of architects, designers, implementers, administrators and system end users. In this section we will describe a number of the most interesting and potentially useful approaches for the purposes of socio-technical configuration modelling.

Most technical system configuration techniques currently available tend to focus on hardware, software, operating system artifacts and so on, ignoring to a great extent social and environmental aspect of systems. The concepts and processes advocated by such techniques must therefore be adapted to a greater of lesser extent in order to apply them to full socio-technical systems.

Given the size of this paper, it is not possible to cover all of the currently available modelling approaches, however we describe a broad spectrum of some of the most interesting to give a flavour of the type of mechanisms which are possible.

The approaches upon which we will focus fall broadly into three categories: those developed for use by system architects, those for system designers and implementers and those for system administrators. The following sections will consider each of the three different classes of approach in turn.

Despite this range of application domain, many similarities exist between the approaches under consideration, with numerous shared and common concepts, structures and processes. Having said this, much of the work in this area has focused particularly on software and hardware artifacts, with an occasional extension to support elements such as documentation or system test artifacts. Thus when attempting to apply facets of any of the described approaches to socio-technical systems considerable effort has to be expended to adapt them appropriately. This has to done to take into account the abstract, variable and enigmatic nature of many of the social components of any socio-technical system. It is however possible to build upon, generalise and extend various element of existing approaches in order to make them applicable to socio-technical systems.

### 2.2.1. Architects

Past research in the area of systems architecture has resulting in the development of a number of different approaches that provide support for the capture of configuration information. Early work in this area focused on the development of various Module Interconnection languages (MILs). The original MIL approach was first proposed by DeRemer and Kron [5] and is based around the capture of fundamental processing modules and associated relationships such as "provides", "has-access-to" and "consists-of". Since the proposal of this original approach, numerous MILs have been developed including CL [8], ZCL [12], PCL [15] and so on. These subsequent approaches are based upon the standard MIL concepts, but with the additional of various enhancements and extensions, including additional relationships, formal specification with proofs and structural variability.

Generally such approaches break down a system into a number of generic templates (or components) each of which have one or more interfaces (or ports). Each interface is associated with a set of attributes that classify the mode of operation, nature and type of that interface. Abstract templates are instantiated into concrete nodes that represent the individual elements present in a particular system. The node instances can then be related together by linking the interface of one to that of another using a connector [7]. Using a combination of nodes and connectors, a tree or graph structure is constructed in order to represent the configuration of a system.

Architecture description languages (ADLs) can be viewed as an extension to the standard MIL concept. Additional semantic information is included to provide more information about the components that constitute a system [7]. ACME is an interchange language used by many architecture description methods including Aesop, C2, Darwin, MetaH, Rapide, SADL, UniCon and Wright [7]. ACME utilises Components, connectors, systems, ports, roles, representations and representation maps to model the structure and composition of a system. As with MILs, components are the primary computational elements of a system and connectors represent interactions among those components. Similarly, each component provides one or more interfaces which are defined using different types of ports. ACME also supports the hierarchical description of any component or connector, in that each may be decomposed into one or more lower level sub-elements. ACME augments the structural representation of a system by allowing the specification of properties for each element of the system model. These properties allow ACME models to be extended to include a representation of aspects such as the run time behaviour of the system.

### 2.2.2. Designers and implementers

Recent standardisation and convergence in the area of technical systems design has been led by the widespread uptake of the unified modelling language (UML) [13]. UML includes numerous features that can contribute to the capture and presentation of system configuration information. Many of the numerous graphical models which UML supports have various aspects of configuration modelling embedded in them. The three most configuration oriented models within UML are however "activity models", "class models" and "object models". These provide both structural and behavioural oriented perspective on system configuration issues. Class and object models can be used to capture various static relationships such as "Association", "Composition", "Dependency", "Aggregation" and "Generalization" between the entities that constitute a system. Activity diagrams can be used to represent dynamic configuration aspects of a system by illustrating the relationships between individual processes that are performed. Additional configurational information is illustrated in the description of how sets of such processes are chained together in order to perform larger granularity "Activities". It should however be noted that activity diagrams also have additional behavioural aspects not relevant to configuration modelling (such as parallel activity synchronisation).

Despite the fact that UML is rapidly becoming an industry standard for designing technical systems, there are various other approaches that also deserve a mention due to their particular emphasis of capturing configuration information. Conic [9] is one such approach which focuses on supporting both the design and implementation of software based systems. This approach allows developers to capture the main processing components, data flow paths, processes and process interconnections within a system. Conic goes further and allows the specification of the data types of information flows and as well as the interfaces of components in the system. Hierarchical decomposition is also employed to represent the structure of complex system components. A combination of both data flow and object oriented entity identification and classification is used to provide two different perspectives on the configuration of a system.

### 2.2.3. Administrators

Due to the demanding nature of many of today's application domains and the resultant built-in flexibility of many modern systems, significant configuration activity is often required on the part of a technical administrator before or during operation of a system. Many approaches exist to support this process and most include mechanisms for modelling current and future configurations of the system. LCFG [1] is one such approach that provides organisation wide configuration of Linux machines (although it is latterly being extended to support general Unix and windows communities). LCFG provides a configuration language for the automated installation and configuration of multiple distributed computers. The main aim of LCFG is to automate the administration of all the machines in a department or organisation. Multiple configuration "source files" are stored on a centralised configuration server. Each of these source files focuses on a different aspect of system configuration specified in a high level language. The configuration server can compile a number of these source files into a lower level XML configuration description "profile" for a particular machine, or group of machines. Each individual machine on a site is then responsible for acquiring a suitable configuration description profile from the centralised server and applying it to it's own configuration.

The European DataGRID modelling language [3] is a high level configuration description language to describe the configuration of grid computing nodes. Each description consists of a hierarchy of configuration key-value pairs each of which is referred to as a system parameter or property. These configuration descriptions are stored in a centralised location known as the configuration database (CDB). When performing configuration on a particular machine an appropriate description is selected from the database, converted into a lower level configuration representation and then applied to the machine in question. The European DataGRID modelling language uses the notion of "templates" which are sets of statements that are executed to derive the configuration property values for a particular system. Templates can include assignments, conditional statements, iterators, arithmetic operators, logical operators, string operators, includes (for importing other templates), type definitions, validation statements and various operations for including and excluding sets of properties and relationships. European DataGRID modelling language, are also extensible and allow administrators to specify additional custom entity types.

KUANG is an element of the Computer Oracle Password and Security system (COPS) [6] that uses a rule base of predicate logic to express the current configuration of a particular system. The aim of this approach is to provide a configuration model of a system upon which can be performed automated checks and validations to assess various aspects of security. A set of basic facts (e.g. "file" is owned by "user", "user" is a member of "group" etc.) is used to represent various aspects of the configuration of a system. This knowledge base is dynamically updated by the addition and removal of facts as the system's configuration evolves over time. KUANG uses the concept of a "goal" to specify the objectives of an attacker or unauthorised user of the system. As such these goals represent the individual failure modes of the system since, if an attacker achieves a goal, the system has been compromised and security breaks down.

A particular point of interest regarding this approach is that it relies heavily on various detection mechanisms to identify changes in the system configuration and update the rule base model appropriately.

The Arusha [11] project focuses on the description of the "value added" to a system by it's administrator and the sharing of that information cross the Unix administration community. "Value added" refers to any enhancements and additions that have been made over and above the default factory configuration of a system. Such features could include installed applications, added servers, performance enhancement, tightened security settings etc. This approach utilises an object oriented model and classification hierarchy support to represent configuration information in a clear and consistent manner. The novel aspect of the Arusha project is that it attempts to provide a centralised, "open source" style repository of configuration knowledge. This information may be used simply as examples that administrators can consult when performing manual configurations. However a tool is provided for automatically applying Arusha configuration knowledge to a Unix machine. The utility of Arusha hinges on the fact that many administrators at many sites need to perform the same (or similar) configuration tasks on the same (or similar) machines in the same (or similar) situations. Arusha supports the description of applications, machines, users, vendors, printers, equipment orders, maintenance contracts, licence agreements, serial numbers, spare parts, help line numbers, addresses etc. in a consistent manner.

## 2.3. Configuration analysis

In addition to approaches that attempt to model configurations for a variety of different reason, there also exist a number of approaches that provide analysis mechanisms for the configurations held. Reliability graph analysis [14] is once common mechanism that can be used in the analysis of system configurations. A reliability graph can be generated from a configuration model of a system and then a number of different algorithms may be used to analyse them. A reliability graph contains a set of nodes and a number of directed edges between those nodes. Each edge represents a component that can fail, or a structural or dependency relationship between two components in a system configuration. The system represented by a reliability graph fails when there is no path from the source to the sink node. Each edge in the graph can be associated with a probability of failure so that it is possible to gain an insight into the likelihood of configuration failures. Popular analysis mechanisms for reliability graphs include factoring algorithms, inclusion-exclusion analysis, sum of disjoint products analysis and so on.

Failure goal analysis [16] can be used to assess security of networked computer systems and to specifically find particular system vulnerabilities. This is achieved by performing a systematic analysis of computer configurations in order to uncover possible security breaches that can make a system vulnerable to attack. An operator specifies a failure mode (e.g. an unauthorised attacker gains root access) and a set of initial conditions (the attackers initial security privileges). Using a simple backwards chaining algorithm, the system recursively expands the attackers goal into one or more sub-goals. If the original goal can be expanded into a set of facts that hold true for their initial privileges and the current configuration of the system then a security vulnerability has been identified.

This form of analysis is particularly useful since the steps that must be taken by the attacker in order to exploit the weakness are described by a record of the goal expansions. By processing this rule base, it is possible to perform automated checks and validations on a configuration. Mounji [10] uses such a rulebase configuration model combined with a real-time intrusion detection expert system, in order to identify operating system or application vulnerabilities and thus assess the security of a given system at a given time. This approach acts as an "always on" agent which can not only infer the weaknesses of changes in a configuration, but can also adapt the behaviour of the intrusion detection aspect of the system to monitor exploitations of the identified vulnerability.

ZCL [12] is based around a module interconnection language that is specified formally using Z. With such a formal specification of a system's configuration, it is possible to perform various transformations and proofs to analyse various aspects of the system. Such analysis can include assessing the effects of a failure of one component on the others within a system, verifying compliance of the entire configuration to a set of system wide invariants, as well as verifying the effects of performing a reconfiguring the system.

## 2.4. Problems with existing approaches

There are numerous problems and deficiencies associated with many of the previously mentioned approaches. The majority of these problems arise from the fact that these approaches have been developed for technically oriented modelling activities, where information about a system’s configuration tends to be both concrete and complete. Such approaches tend to be very exact and do not provide enough flexibility to support the high level, abstract modelling and analysis required when working with informal system descriptions.

This is exacerbated by the fact that these approaches focus on technical artifacts, such as software and hardware components. These types of component have their own set of properties and behaviours that are not shared by organisational, social or socio-technical components. By way of example, let us consider the technical artifact specific concepts of "interfaces" and "data types". Obviously these cannot be directly applied to social aspects of a system's configuration.

Although the approaches described above provide support for the modelling of system configurations they often include additional features and deeper modelling structures that can result in the construction of over complex models. Much of this complexity is not required in the modelling of configurations and can add significantly to the effort required in constructing suitable system models. Such complexity makes these models especially impractical for large scale and complex socio-technical systems. In addition to this, many of these deep concepts and structures may be hard or even impossible to determine from the available information.

Additionally, the notations utilised by many of these approaches are often oriented to technically trained readers. Such notations may take the form of complex graphical representations or even formal methods. The major drawback of these representations is that their meaning is not immediately intuitive to all readers and requires some training before interpretation is possible. Such notations are clearly inappropriate in a situation where communication involving non-technical stakeholders is required. In addition to this, these notations are often overly complex, capturing as they do many types of information in addition to those that relate purely to the configuration aspects of a system.

Any successful configuration modelling approach must include simplistic, high level, flexible and provide suitable structures and concepts to deal with the wide variety of domain entities that must be modelled when considering full socio-technical systems. In developing the Strider approach we have extended, generalised and augmented structures and processes from existing approaches to support socio-technical and purely social components and concepts. We have also employed simplified, especially pictorial graphical model notations and have attempted, wherever possible to minimise graphical clutter and complexity.

At the forefront of the considerations that drove the creation of our new approach was a desire to support maximal configuration analysis whilst at the same time requiring only the minimum amount of modelling effort. Any suitable approach must be lightweight and agile enough to deal with rapidly evolving systems and improvements in modeller’s appreciation and understanding of a particular system.

## 3. Strider configuration models

Strider configuration models capture only a very thin "skin deep" layer of system configuration whilst at the same time maximising management, support and analysis opportunities. This not only helps to keep any additional modelling overhead low, but also aids the scaleability of the approach. A configuration model captures the static components and relationships within a system at a particular point in time. It models the content, structure and form of all social and technical components. So for example, a configuration model would represent the software, hardware, worker and resource components of a system, as well as the organisational, structural and operational relationships between them. It is important to bear in mind that we are interesting in modelling the "configuration, not operation" of a system. For this reason, we do not concern ourselves with how processes behave internally, but rather limit ourselves to the pattern of inter-relationships that they imply.

To aid in the construction of configuration models, Strider provides support for the direct recovery or "reclamation" of data from ethnographic reports, manuals, documents and other original source materials. We do not utilise fully automated grammatical analysis techniques due to issues relating to accuracy, scaleability and generalisablity of such approaches. Rather we prefer to keep the human "in the loop" by providing tools to enhance and augment the activities of a human operator. To support this process the tool provides mechanisms for the selection of fragments of media from original source documents and the population of the configuration models using a specialised "cut and paste" mechanism.

The overall process of modelling should be viewed as a human mediated "harvesting" of pertinent data from verbose system description to be used in the automatic construction of configuration model entities. Support is provided for the importation of electronic texts, hand written notes, photographs, diagrams, scanned artifacts and audio files. With these facilities in place, the generation of the configuration models can be achieved as quickly and as cleanly as possible from the available original source material. The model entities that are created by this process encapsulate and contain references to the fragments of raw data from which their existence was initially determined.

### 3.1. Configuration items

Configuration items are the atomic elements that make up a configuration model. These configuration items may represent software, hardware, buildings, artifacts, people, documents and so on. Some less tangible configuration items such as knowledge for example, can be particularly problematic to model. This is partially due to their complex nature, the difficulty in understanding and decomposing them, and the fact that the configuration models supported by Strider are not detailed enough to adequately permit their modelling. Identifying suitable configuration items for modelling can be assisted by interviews with managers and workers, examination of procedural manuals or as a result of ethnographic study.

Within Strider, a hierarchy is used to classify the various classes of configuration items that may exist. The user may extend and enhance the set of configuration items present in the classification hierarchy to suit the types of system that they are attempting to model. Within the Strider support tool, this hierarchy acts like a palette from which the user may instantiate concrete configuration items into the configuration models. A single configuration item may be an instance of multiple classes from this hierarchy. Perhaps the best example of this is the fact that a single worker within an organisation may fall into multiple worker classifications, due to the fact that they may perform multiple roles within that organisation. In addition to this, configuration item classes may be instantiated into a single entity (e.g. "the hospital") or alternatively a multiple entity (e.g. "the beds"). An example configuration item classification hierarchy provided by Strider is illustrated in figure 1.

Configuration item classification hierarchies such as the one depicted above in figure 1, include a variety of predefined classes for use in modelling the configuration of a system. The initial set of item classes in this hierarchy are provided as part of the Strider tool, however these can be extended and augmented by an operator before or during the modelling process. These items can include various types of software, hardware and other technical resources as well as the variety of roles a person may undertake and the artefacts, structures, social constructs and buildings that may be involved in a configuration.

**3.2. Structural models**

An important aspect of a configuration that the Strider approach explicitly models is the relative structural organisation of the configuration items within a system. Such models allow us to record which configuration items are located within, installed on, placed in or operate within other items. In this way, the structural model captures relationships that are a generalised, socio-technical approximation to the "part-of" relationship identified in technical modelling approaches (such as UML for example [13]).

The information present in the structural model is normally relatively easy to determine by observation and discussion with workers and managers within an organisation and investigation of procedural manuals and so on. The structural model indicates some of the fundamental relationships within a configuration, namely those relating to the organisation and physical location of components. The relationships contained within such models indicate the pervasive top-level configurational structure, the physical organisation and spatial positioning of the various elements of a system.

Take for example a hospital ward: Strider allows modellers to describe the "contains" relationships which link the ward with the bays and the bays to the beds that they contain. Figure 2 illustrates part of the structural model for a typical healthcare domain system configuration. As we can see, the structural model provides us with an intuitive, easily understandable representation of a particular facet of a configuration, providing as it does a great degree of flexibility in the types of relationship that can be modelled.

The small triangular flags in the top right hand corner of some of the configuration items depicted in figure 2 indicate elements of the configuration that may be plural in nature. So for example, a ward may have many bays, each of which may have many beds, but each bed may have only one occupant patient. This notion of plurality provides a quick and simple indication of the cardinalities associated with the structural relationships within the model. Such information provides important additional data to aid in the comprehension and analysis of the system configuration.

The degree of mobility also varies from one configuration item to another and this can be used to help describe some of the dynamic characteristics of the structural model. Static configuration items are ones which cannot be relocated within a structural organisation; Moveable configuration items are ones which are present in a particular location, but which may be relocated for a particular reason; Roaming configuration items are ones which freely move around a configuration and are rarely static for any length of time. Each type of mobility is illustrated using a different graphical notation in the structural model. The more static a particular configuration item is, the more solid the box which represents it on the diagram.

For each non-static configuration item present in the structural model, all of the important activity locations of that item should be recorded. Although this leads to multiple instances of a particular item, it does allow us to capture all of the possible structural relationships between an item and the other items in a configuration. This is particularly important for performing analysis that relies on the specification of all structural relationships.

One important feature of this type of model is that is it equally applicable to both social and technical components of a system. As a result of this, the structural model unifies both aspects of a system allowing all considerations to be represented in a single, consistent model. The structural model is particularly useful since is presents an easily understandable facet of system configuration which can help ethnographers, developers, managers and other interested parties in the comprehension of a system. The model not only aids in the understanding of the internal composition of complex components, but can also provide much information on the context of configuration items within the system. As we shall see in a later section, the relationships present within structural models also provide an excellent input to automated analysis.

## 3.3. Process models

In addition to the relationships recorded in the structural model, Strider also uses the concept of 'processes' to relate configuration items together. A process is an identifiable low level atomic activity from the socio-technical system under consideration. Examples of such processes could include *print document*, *make phone call* and *fill out form*. By modelling these processes, it is possible to record fragments of interrelationships and collaborations between the various configuration items of a system. Identifying suitable processes for modelling can be assisted by interviews with managers and workers, examination of procedural manuals or as a result of ethnographic study. This mechanism is particularly flexible and can be used to model a wide variety of operational relationships from both social and technical aspects of a system.

The process models utilised by Strider can be viewed as augmented fragments of UML activity diagrams [13]. Each process modelled is associated with a number of *Initiators* (items which begin or activate the process), *Targets* (items upon which the process acts or which perform the process), *Inputs* (items which are required to perform the process) and *Outputs* (items which are produced by the process).

We can illustrate the various elements of a process using the example shown in figure 3. In this example initiators are to the left (i.e. directorate manager), targets are to the right (i.e. management info system), inputs come from the top (i.e. bed availability data) and outputs are produced at the bottom (i.e. weekly report).

It is important to note that, due to our focus on configuration, we are only interested in the patterns of configuration item relationship. Thus we do not model what the targets do, how inputs are used or how outputs are produced. As a result of this, Strider does not incorporate state based or temporal models of process activity. These are both time consuming to produce and do not provide any additional information for the modelling and analysis of configurations as we perceive them.

The concept of a process is a useful one for the modelling of socio-technical configurations due to its general and generic nature. Such processes allow us to model a verity of disparate relationships between the configuration items of a system. This ranges from *used_by* relationships, through *constrained_by* relationships to a general *involves* relationship. One beneficial aspect of the use of processes to model relationships is that there is no need to attempt to classify the type of relationship. For the purposes of modelling and analysis it is sufficient to simple know of the existence of the relationship. If further information regarding the nature of the relationship is required, then Strider can refer the user to the supporting segments of original source material from which the relationship was derived.

## 3.4. Objectives

An objective is defined as a goal that can be achieved using the particular socio-technical system configuration under consideration. An objective path is a chain of relationships through a configuration by which a particular objective may be achieved. A single objective may be associated with a number of different paths though a configuration, each of which represents a different way of "getting the job done". We can define two different types of objective:

- Action objectives - objectives whose purpose is to cause some action or activity to take place within a configuration (e.g. discharging of a patient)
- Information objectives - objectives whose purpose is to derive, collate or retrieve some information from within a configuration (e.g. ascertaining the number of currently available beds)

Within strider an objective path can be traced by following the flow of control or information along the various relationships within the configuration models. These relationships include those taken from both the structural as well as the process oriented models of a configuration.

Redundant objective paths often exist so that when a configuration item on one path fails, alternative paths may be followed. These are the "work arounds" commonly identified in many work practices. Without redundant paths in a system, a failure of one component can quickly lead to the failure of the system to achieve its supported objectives. This obviously has serious implications for the overall dependability of the system.

Due to the existence of redundant objective paths in most configurations, the complete failure of the entire socio-technical system is relatively rare. What does often occur however is the degradation in operation of the system due to failure of individual or clusters of configuration items and processes. This degradation may manifest itself as an increase in time to complete objectives, increases in system running costs and so on. This is a direct consequence of having to following sub-optimal or inefficient paths due to the unavailability of the usual and most efficient paths.

We should be careful to bear in mind however that for any number of reasons, the most optimal path may not always be utilised in the attainment of an objective. The reasons for this may include both internal and external political constrains, imperfect knowledge of workers, cost implications, legacy processes or procedures and so on.

## 4. Configuration model analysis

Much can be gained from the modelling of configurations in terms of aiding understanding and communication between the various parties involved in socio-technical systems development and management. In addition to this, significant benefits can be gained from the use of tools to support both the modelling and investigation of configurations. However the most significant benefits of the Strider approach are obtained through automated analysis of the modelled configurations.

Using Strider, it is possible to model the existing configuration of a system and then perform analysis on that model in order to derive an insight into various dependability attributes of that configuration. The automated analysis performed by Strider is based on the processing of objective paths and configuration item dependencies. By processing these entities it is possible to perform a variety of different analysis tasks. Using these automated analysis features it is possible to:

- Check availability of paths - Analysis allows us to determine if a specified objective is achievable within a given configuration.
- Ensure multiple paths - The identification of multiple paths within this set for a given objective indicates some resilience to the failure of individual configuration items. If one path is rendered inoperable due to the failure of a configuration item on that path, then alternative paths are available to achieve that objective.
- Check pattern of paths – Strider can reveal the patterns that objective paths follow through a configuration. In doing so it is then possible to identify single points of failure and areas of high utilisation within the system. In addition to this, it is possible to highlight redundant or under utilised components and processes.
- Find alternative paths – Strider allows us to identify all the possible paths to achieve a particular objective. This is not just limited to the paths that are used in the day-to-day operation of a system, but can also include currently unused paths. Using this information, it may be possible to identify and utilise alternative paths that are faster, more efficient or more cost effective paths than those that are currently in use.
- Identify need for additional paths - If only a single path between start and end point is identified, then it may be desirable to perform a reconfiguration of the system in order to produce a more resilient configuration in which it is possible to achieve a particular objective by more than one means. Thus when a single item within a configuration fails, it is less likely to prevent the attainment of the desired objectives.

- Identify unnecessary redundant paths – If there exists many duplicate or overly complex redundant paths, there may be scope for the optimisation and streamlining of a system or organisation. By removing the configuration items and processes associated with such paths, and provided that enough redundancy still remains to provide an appropriate level of dependability, significant cost and efficiency savings can be made.

To perform analysis, the operator first specifies a particular objective by selecting a start and endpoint configuration item or process. It is also possible to specify configuration items and processes that are to be included or excluded from the final set of identified paths. To help optimise analysis, the operator also indicates whether the objective type is "action", "information" or both. The Strider support tool then uses various relationship extraction rules to determine all of the possible objective paths. These rules use knowledge about the nature of configuration structure and process relationships to derive data and control flow paths. Inputs to a process imply a flow of data into that process; outputs to a process imply a flow from data out of that process; initiators imply a flow of control to a process as well as a potentially bi-directional flow of information; targets imply a flow of control from a process as well as a potentially bi-directional flow of information. Structural model relationships can also be used since a relationship whose target is a composite configuration item implies relationships to all atomic parts of that item. Information relating to the type and direction of flow can later be used to perform certain types of analysis on a system model.

Output from analysis is a list of possible objective paths each of which represents a route through the configuration by which the specified objective can be attained. A path in this list is composed of a sequence of configuration items and processes, extracted from the configuration models, via which the objective is reached.

If a path between a start point and end point is not found, yet the objective is actually achieved in the operation of the real world system, then this can indicate a potentially incomplete configuration model.

## 5. Investigation

In order to demonstrate and assess the utility of the proposed configuration modelling and analysis approach, we will apply it to a real world case study. Due to the size and complexity of real world systems such as this, it is not possible to manually consider all of the effects, implications and interactions of the various socio-technical elements of a configuration. This is where the Strider approach and support tool can be most useful, helping to assist the operator in investigating enigmatic aspects of the system's configuration.

We will first give an overview of the case study system before continuing on to provide a more detailed description based upon the Strider configuration models themselves. This will not only illustrate the utility of the configuration models, but will present the data upon which automated analysis will be performed. Finally we will present and discuss the results of analysis and examine the utility and practicality of the Strider approach.

**5.1. Example case study**

To help evaluate the Strider approach, we have chosen to apply it to the Chaum e-voting scheme [2]. This scheme is a proposed system to provide a dependable, high security and trustworthy mechanism for electronic voting in national and regional elections or referenda. We have chosen this scheme because it is a well documented example of a complex socio-technical system. The need to generate trust in the scheme has lead to the creation of much information regarding various aspect of the system's operation. In addition to this, the subject of e-voting is an important and topical area currently generating much research interest. Finally the application domain of this system will require little explanation since most people should have a general understanding of how an election operates and the context in which it takes place.

In order to understand the motivations behind the Chaum scheme, let us first look at some of the key socio-technical requirements that underpin it. These high level requirements are as follows:

1) Anonymity - It is essential that the scheme maintains the anonymity of every voter. In order to prevent vote selling, a voter should not be able prove who they voted for. Additionally, to help prevent intimidation of voters, votes should not be traceable back to voter who cast them and the names of those who voted should never be published.

2) Verifiability - Mechanisms should be present that allow external checking of the voting mechanism to help ensure that votes are counted correctly. To help support this, a voter is provided with a receipt that can be used to (partially) verify vote. A voter can then check that their vote has been counted by finding its entry a published list of counted vote. Partial traceability through the counting process is also provided to allow external auditing of the election.

3) Accuracy - To help ensure a high level of trust in the scheme, it should exhibit a low error rate and features to allow the checking for such errors. This includes checks at various stages to help ensure that once the vote has been cast no one is able to change it and avoid detection. Additionally, partial traceability ensures that no votes are "lost" by the system and that additional "phantom" votes cannot be inserted.

The Chaum scheme provides an interesting technical solution to the apparently conflicting requirements of a traceable audit trail and maintaining the anonymity of the voters. Due to the limited space available, we are not able to provide a full description of the Chaum scheme. For a more complete description, readers are referred elsewhere [2].

The following sections do however present Strider configuration models that will shed more light on the operation of the Chaum scheme.

## 5.2. Structural model

In this section, we attempt to further describe the Chaum scheme using the configuration models utilised by the Strider approach. The structural model produced for the Chaum scheme is shown in figure 4. The key configuration items from this figure are as follows:

- Election - An event organised to select a local council, national government or other elected body
- Political party - an organisation (or representative for that organisation) for which a voter may vote
- Voting option - One of the possible choices which can be voted for in an election (can include 'abstain', 'spoil' and 'other')
- Voter - Someone who is eligible to vote in the election
- Vote - an individual voter’s selection of one of the voting options
- Tally - the total count for each voting option, representing the final result of the election
- Polling station - The building or location containing voting booths where voting takes place
- Unique identification token - An identification number or card given to the voter upon entry to the polling place which is used to activate a voting machine
- Polling booth - private area in a polling station within which a voter may make their voting selection
- Voting machine - Electrical or mechanical equipment which supports the selection and recording of votes and the printing of receipts
- Receipt - A receipt consists of two transparent layers, both of which have half of the receipt printed onto them (in such away that the whole receipt may be viewed when the two layers are combined and held up to the light). The data on each layer is obscured by random noise to hide the option that was voted for. The random noise is generated such that when the two layers are combined, the noise from each layer cancels each other out, leaving only the voting data.
- Random noise - A random pixel pattern used to obscure the text on each layer
- Kept layer - the layer of the receipt retained by a voter
- Discarded layer - the layer of the receipt discarded and destroyed by a voter

- Website - Web page for recording and publishing details of various stages of the trustees work (especially the first and last phases of counting)
- Serial number - Unique number printed on the receipt to identify it. This is used to allow verification via the website.
- Unbroken background - A visual security feature present on the receipt which is used by the voter to ensure it's validity
- Public key - A public key used to encrypt a noise removal key for a particular trustee
- Noise removal key – A key which can be used to partially remove noise from a layer of the receipt
- Counting place - The building or location where the trustees perform decryption, counting and tallying
- Trustee - Person (or machine) charged with the decrypting, mixing and publishing votes
- Returning officer - Person (or machine) charged with tallying results and returning a result for part or all of an election/vote
- Poll clerk - Person responsible for confirming the registration of the voter, and shredding the discarded portion of the receipt.
- Auditor - Person responsible for performing checks on the trustees and their activities
- Private key - A key used to decrypt a noise removal key for a particular trustee
- Altered receipt layer - A layer of the receipt from which some noise has been removed
- Verifier - member of an organisation or and individual interested in the free and fair running of the election who is able to verify the validity of kept receipt layers
- Barcode scanner - Device used by verifier to perform consistency checks on the retained portion of the receipt.

## 5.3. Process model fragments

To supplement the description provided by the structural model, a set of processes model fragments have also be derived for the Chaum scheme. The processes that were identified are as follows:

- Identifies themselves - this involves a voter identifying themselves to a poll clerk (either using some documentation or by answering some questions) and receiving a voting token
- Activate voting machine - this is the enabling of a voting machine to accept a vote using a token (e.g. id number, smart card etc)

- Generate serial number - the creation of a serial number which will identify a vote and appear on the voting receipt
- Indicate selection - the selection of one of the voting options by a voter
- Generate receipt - the creation and printing out of the voting receipt
- Verify correct choice - the checking by the voter that the correct choice has been recorded by the voting machine (by checking the receipt or the voting machine display)
- Visually inspect - the manual, optical checking of both layers of the receipt together to check it's authenticity
- Retaining a single layer - this involves the voter keeping one layer of the receipt for later verification purposes
- Surrender layer - this involves the voter surrendering one layer of the receipt to prevent their choice being determine by an unauthorised individual
- Destroys the surrender half - the destruction of the surrender layer of the receipt
- Retained layer check - an automated check to verify the authenticity of the layer of the receipt which was retained by the voter
- Encrypted multiple times - multiple levels of encryption are used to obscure the vote cast and make it un readable to any unauthorised individual
- Grouped into batches - the collection of groups of votes into batches to ease the process of decryption and counting
- Decryption - the use of trustee private keys to create noise removal keys
- Remove the noise - the use of noise removal keys to partially remove noise from the votes during the counting process
- Mix batches - this involves the mixing of the batches as occurs before batches are moved from one trustee to another
- Published mappings - the act of partially publishing mix mappings on a website
- Pass onto next trustee - the passing on of vote batches from one trustee to another
- Final decryption – the final phase of decryption, after which the votes which have been cast are evident and my be counted
- Votes tallied - the cumulative summation of all votes for the different parties
- Publish tally - the act of publishing the final results on a website
- External audit - the verification of the counting process by an external independent auditor
- Reassures – an attempt by auditors to increase voters confidence in the accuracy and security of the voting scheme

- Check website – checking by various interested parties of the voting information published on the website
- Controlled – the manipulation of one of the officials associated with an election by a political party
- Acts under coercion – the persuasion of voters by a political party, using fair means or foul

**5.4. Objective analysis**

With the above configuration models in place for the case study system, it then becomes possible to perform automated analysis based on identified objectives. We have proposed three objectives that may be achieved with the configuration implicit in the Chaum scheme. These form a varied set of objectives that can be used to investigate and demonstrate the abilities and benefits of the Strider approach.

Some of the objectives identified are positive ones and are explicitly supported by the scheme. In the case of such objectives, we utilise Strider to help assess the extent to which these aims are achievable. Some objectives on the other hand are negative and focus on phenomenon that the scheme attempts to explicitly prevent. In this case, we use strider to help identify the paths by which such objectives may be achieved and then assess the extent to which these paths are blocked by the current configuration of the scheme.

The objectives that we have identified for this investigation are as follows:

1) Political party finds out who a particular person voted for - As part of the anonymity requirements of the scheme, it should not be possible for anyone (especially member of a political party) to identify the voting option selected by a particular individual. The purpose of assessing this objective to investigate how the configuration of the Chaum scheme prevents such objectives from being achieved.
2) Voter checks that their vote has been counted - To help improve voters confidence in the scheme, it is important that they are able to check that their vote has been counted after tallying has taken place. This objective allows us to identify all of the different paths by which this activity may be achieved.
3) Political party effects tally - A key aspect of any voting scheme is to control the ways in which a political party may affect the outcome of the election. Some of the paths by which this objective may be achieved will be permissible but others may be deemed unfair or even illegal. By identify all possible paths it becomes possible to ensure that all of those that are not permissible have been blocked in some way.

Each of these three objectives have been specified and analysed using the Strider tool. In the following three sections we present the results of this analysis and highlight key areas of interest and any important issues that are raised during this process.

**5.4.1. Objective 1: Political party finds out who a particular person voted for**

The aim of this objective is to attempt to identify the paths by which a political party may access information on the votes cast by people when they make their selection in the voting booth. This is an undesirable objective from the perspective of a free and fair election since it's attainment can support intimidation as well as the "sale" of votes by a voter. The objective path that is specified in order to investigate this aim is an information based path that originates with the "Political party" configuration item and terminates with the "indicate selection" process that is performed by the voter.

During analysis 112 different objective paths were identified, a number of which were of particular interest. These included the identification of the known and acknowledged paths such as the placing of pressure on the voter by the political party to reveal there voting selection (*Political party :: acts under coercion :: Voter :: indicate selection*) and the controlling of a trustee or returning officer to the extent that they can provide information about the selections made by the voters (*Political party :: controlled :: Trustee :: votes tallied :: Vote :: indicate selection*) and (*Political party :: controlled :: Returning officer :: votes tallied :: Vote :: indicate selection*). Various mechanisms are currently available to interrupt these known objective paths and the Chaum scheme competently combats such abuses.

Another particular issue addressed by the Chaum scheme is the provision of a voting receipt for the voter. This is done in such a way as to prevent this receipt being used by any unauthorised personnel to determine the voting option that was selected. The objective path around which this concern is based and which the scheme explicitly blocks was clearly identified during analysis (*Political party :: acts under coercion :: Voter :: generate receipt :: Vote :: indicate selection*).

The Chaum scheme is less clear on the mechanisms used to prevent poll clerks from determining how a voter has voted, either by observation or questioning, and then reporting this fact back to a political party (*Political party :: controlled :: Poll clerk :: destroys the surrendered half :: Voter :: indicate selection*). Similarly, the contact that external auditors may have with voters or the tallying process could allow them to determine or imply individual's voting selections (*Political party :: controlled :: Auditor :: reassures :: Voter :: indicate selection*) and (*Political party :: controlled :: Auditor :: external audit :: tally :: votes tallied :: Vote :: indicate selection*). Although certain aspects of the Chaum scheme may make such objective paths problematic, there are currently no mechanisms in place that directly address these issues.

An area of particular interest identified during analysis relates to the use of the public web site (used for verifying the decryption and tallying of votes) and the serial numbers generated during the voting process. (*Political party :: check website :: Website :: publish mappings :: Serial number :: generate serial number :: Voting machine :: indicate selection*). The detection of this path indicates the need to exercise caution when generating such numbers to ensure no traceability is maintained which could allow an interested party to link voters to individual selections.

Finally, the analysis of the specified objective resulted in various suggestions involving controlled trustees manipulating receipt layers in the counting phase of the election. However, due to the encryption and mixing of votes by the Chaum scheme, the individual trustees will either have access to the identities of the original voters, or the votes that they cast, but not both. The implication here is that for such paths, all trustees must be under the control of a political party in order to determine the voting choice of particular voters.

### 5.4.2. Objective 2: Voter checks that their vote has been counted

The aim of this objective is to attempt to identify the paths by which a voter may reassure themselves that their vote has been counted and not "lost" somewhere in the complex socio-technical system which is the Chaum scheme. The objective path that is specified in order to investigate this aim is an information based path that originates with the "Voter" configuration item and terminates with the "votes tallied" process that is performed by the trustees and returning officers.

For this objective, 73 suggested paths were produced by analysis. This objective forms an important part of the Chaum scheme since the confidence of voters is an essential part of any voting system. For this reason there are a number of mechanisms provided by the Chaum scheme to support this objective. These include the provision of external independent auditors to check the tallying process on behalf of the voters (*Voter :: reassures :: Auditor :: external audit :: tally :: votes tallied*) as well as various paths which allow the voter to be reassured by the information published on the website (*Voter :: check website :: Website :: publish mappings :: Trustee :: votes tallied*) , (*Voter :: check website :: Website :: publish tally :: Returning officer :: votes tallied*) and (*Voter :: check website :: Website :: publish tally :: tally :: votes tallied*).

Another objective path identified during analysis indicates the utility of the receipts provided by the Chaum scheme. This path (*Voter :: indicate selection :: Voting machine :: generate receipt :: Vote :: final decryption :: Trustee :: votes* *tallied*) demonstrates how the receipt provides feedback to the user that their vote has been cast and is thus an indicator of inclusion in the count. Although such a receipt does not provide conclusive proof of a vote being counted, it can increase voter confidence by providing them with a tangible piece of evidence.

Less obvious and unintentional paths to achieve this objective exist if a voter is being coerced by a political party (*Voter :: acts under coercion :: Political party :: controlled :: Trustee :: votes tallied*) since they can be confident that if a trustee is also being controlled, their vote will be counted correctly. Although this is an unexpected path, it does illustrate Strider's ability to identify the existence of potentially significant paths that are not usually considered.

**5.4.3. Objective 3: Political party effects tally**

The aim of this objective is to attempt to identify the paths by which a political party may affect the final tally of votes in an election. The objective path which is specified in order to investigate this aim is more complex than the previous objectives since it encapsulates both information and action aspects. Information is required due to the flow of information implicit in many of the mechanisms used to affect the tally. Action is required because the political party must invoke various processes in order to affect the tally. The objective path used originates with the "Political party" configuration item and terminates with the "tally" configuration item.

Analysis of this objective resulted in the detection of 67 suggested paths. Some of the most direct and immediately obvious of these included the controlling of trustees and returning officers (*Political party :: controlled :: Trustee :: votes tallied :: tally*) and (*Political party :: controlled :: Returning officer :: publish tally :: tally*) as well as the coercion of the voters themselves (*Political party :: acts under coercion :: Voter :: indicate selection :: Vote :: final decryption :: Trustee :: votes tallied :: tally*). Such coercion may be as extreme as threats or intimidation, or may simply take the form of voters being compelled to support a particular party due to belief in the party’s policies. All of these forms of coercion have varied acceptability in different counties around the world. Strider can be used to identify the existence of these paths, it is up to the particular authorities in question to determine which paths should be blocked.

Due to the series of backups and fail safes in the Chaum scheme, some objective paths require the manipulation of multiple operatives involved in an election. For example one path (*Political party :: controlled :: Trustee :: remove the noise :: Kept layer :: grouped into batches :: Returning officer :: votes tallied :: tally*) indicates the need to control the trustee as well as the returning officer in order to have an effect on the final tally. Yet other paths indicate the need for action to be taken to cover up the side effects of alterations in the tally and prevent detection by external auditors, voters and so on (*Political party :: controlled :: Auditor :: external audit :: tally*) and (*Political party :: acts under coercion :: Voter :: check website :: Website :: publish tally :: Returning officer :: votes tallied :: tally*)

As a side issue some interesting paths were also identified which, rather than affecting the final tally, can be used to question it's validity. This can occur if voters disagree with the findings of Auditors or the information published on the website (*Political party :: acts under coercion :: Voter :: reassures :: Auditor :: external audit :: tally* and *Political party :: acts under coercion :: Voter :: check website :: Website :: publish tally :: tally*). Although this cannot be used to directly affect the results of an election, it can be used to gain political advantage by promoting mistrust and thus undermining confidence in the election's outcome.

## 5.5. Discussion

As we can see from the results laid out in the previous section, the objective path analysis performed by Strider can provide us with many useful insights in the configuration of a socio-technical system. Strider's ability to detect well known and commonly cited objective paths supported clearly demonstrates the validity of the analysis mechanisms in use. However, Strider goes further than this and allows us to detect previously unidentified, unintentional and unexpected paths that also exist. In addition to the explicit identification of objective paths, we have also seen how Strider can be used to highlight a variety of other interesting and insightful phenomenon from the information held in the configuration models.

Due to the "naïve" analysis provided by the Strider approach, the number of paths detected was usually quite large for the three example objectives (an average of 84 paths). This resulted from the inability of the analysis mechanism to distinguish between realistic and unrealistic paths. It is the job of the operator to determine which paths are irrelevant or illogical.

Although this may seem relatively large at first sight, results sets of this scale are perfectly acceptable. This number of paths is manageable and can easily be investigated by an operator in a reasonable amount of time. This set of suggested paths provides an invaluable set of "prompts", guiding the consideration of the operator to the most fruitful areas of the system.

Without the assistance of configuration modelling and analysis as provided by Strider, it is difficult for an individual to envisage all of the implications and ramifications of a system's structure and it's interrelationships. This is particularly true for complex socio-technical systems where important paths are easy to miss and it is essential to have some systematic way of undertaking investigation.

Aided by the support of the Strider approach, it is possible for a stakeholder to become more confident about their ability appreciate all aspects of a system. This includes their ability to ensure that all intended paths are available and that all desired objectives are achievable within a given configuration. Implicit in this is their ability to check for the availability of multiple paths to achieve an objective as well as the patterns that those paths follow. This can help to ensure higher dependability of the configuration in case of partial failure of one or more paths. Conversely, stakeholders can also be aided in ensuring that all paths to reaching undesirable objectives are suitably blocked and thus become unobtainable.

In addition to these core benefits, the analysis of the Chaum scheme has highlighted a number of peripheral advantages that are offered by the use of the Strider approach. These include:

- Support for checking the negative implications and side effects of desirable paths
- Identifying and highlighting of grey areas which might not have been given adequate consideration in the construction of a system configuration
- Identifying and highlighting of potential pitfalls specific to particular configurations
- Justifying the use of various elements present in a configuration (e.g. the use of receipts)
- Directing the operator or analyst towards interesting side issues which may impinge upon the attainment of the main objectives

Obviously we must be realistic and face the fact that it is not possible to identify all weaknesses and potential failures of a configuration using the Strider approach. This is partly due to the limited focus of the Strider models on purely configurational aspects of the system. What Strider does provide however is a very fast and lightweight modelling approach that can facilitate analysis and the gaining of considerable insight into the implicit and explicit paths present in a system. Strider provides a unique style of high level analysis results that are very supportive to a stakeholder attempting to understand the consequences and implication of a complex system configuration.

## 6. Conclusion

This document has described the Strider configuration modelling and analysis approach and support tool. Strider combines a lightweight configuration modelling approach, combined with tool supported investigation facilities and in-depth, automated analysis mechanisms. These features aid Strider in achieving its primary goal, to support the appreciation and investigation of various aspects of socio-technical system configurations. Particular emphasis in this area has been placed on the investigation of the efficiency, productivity and dependability attributes of the system under consideration.

The configuration models produced by strider also provided a clear, clean and easily understandable representation of a system suitable as a target of conversation amongst system stakeholders. As such they form a concrete "topic" of conversation which provides a common point of reference for managers, ethnographers, developers, end users and so on. In this capacity, the configuration models become a vehicle for knowledge transfer between all interested parties.

Applicability and utility of the Strider approach has been illustrated with the aid of a significantly sized real world socio-technical system as a case study. The analysis of the configuration models for this example system revealed many interesting aspects not evident from the initial, purely textual descriptions. With the aid of the analysis features provided by Strider, it was possible to gain considerable insight into configuration of this system and the achievement of objectives based upon it. The benefits of Strider increase rapidly as the configuration becomes larger and more complex and gaining a knowledge and understanding of the whole system becomes more and more problematic.

Strider can offer a great deal of assistance in revealing such insights to the operator. However, due to the abstract nature of analysis, this process still relies on a significant amount of comprehension on the part of the operator. With such interpretation from the operator, some particularly interesting and not immediately obvious insights into the system being modelled were revealed. The various analysis mechanisms available were able to reveal insights into numerous different facets of the system's operation.

Strider is part of the ongoing DIRC project [4]. As part of this work we will be utilising the Strider approach for modelling and analysis of a number of large scale, real world socio-technical systems. Due to the current emphasis of the project, these are likely to be in the healthcare domain.

**Acknowledgements**

This work is funded by the UK Engineering and Physical Sciences Research Council as part of the Dependability Interdisciplinary Research Collaboration (DIRC) http://www.dirc.org.uk/

## References

[1] Anderson, P., Scobie, A., *LCFG: The Next Generation*, UKUUG Winter Conference, London, UK UNIX User Group, February 2002

[2] Chaum, D., *Secret-Ballot Receipts and Transparent Integrity: Better and less-costly electronic voting at polling places*, http://www.vreceipt.com/article.pdf

[3] Cons, L., Poznanski, P., *High level configuration description language specification*, http://hep-proj-grid-fabric-config.web.cern.ch/hep-proj-grid-fabric-config/documents/cfglan.pdf, CERN, March 2002

[4] Dependability Interdisciplinary Research Collaboration (DIRC), http://www.dirc.org.uk/

[5] DeRemer, F., Kron, H. H., *Programming-in-the-Large versus Programming-in-the-small*, IEEE Transactions on Software Engineering, IEEE Computer Society Press, June 1976, pp80-86

[6] Farmer, D., Spafford, E. H., *The COPS Security Checker System*, Proceedings of the Summer Usenix Conf., Anaheim, CA, 1990, pp165-170

[7] Garlan, D., Monroe, R., Wile, D., *ACME: An Architecture Description Interchange Language*, Proceedings of the Centers for Advanced Studies Conference (CASCON) '97, Toronto, November 1997, pp169-183

[8] Justo, G. R. R. , Cunha, P. R. F., *Programming Distributed Systems With Configuration Languages*, Proc. of the Int. Workshop on Configurable Distributed Systems, London, 1992, pp118-127

[9] Kramer, J., Magee, J., Sloman, M., *The CONIC Toolkit for Building Distributed Systems*, IEE Software Engineering Journal, 134-D(2), IEE press, 1987, pp73-82

[10] Mounji, A., Le Charlier, B., *Continuous Assessment of a Unix Configuration Integrating Intrusion Detection and Configuration Analysis*, In Proceedings of the ISOC'97 Symposium on Network and Distributed System Security San Diego, California, 1997

[11] Partain, W., The Arusha Project home page, http://ark.sourceforge.net/

[12] de Paula, V. C., Justo, G. R. R., Cunha, P. R. F., *Specifying Dynamic Distributed Software Architectures*, XII Brazilian Symposium on Software Engineering, BCS Press, October 1998

[13] Quatrani, T., *Visual modeling with Rational Rose 2000 and UML*, Addison Wesley, 2000

[14] Sahner, R., Trivedi, K. S., Puliafito, A., *Performance and Reliability Analysis of Computer Systems*, Kluwer Academic Publishers, 1996

[15] Warren, I., Sommerville, I., *PCL: A configuration language for modelling evolving system architectures*, IEE Software Engineering Journal, 11(2), IEE, February 1996

[16] Zerkle, D., Levitt, K., *NetKuang - A Multi-Host Configuration Vulnerability Checker*, In 6 th USENIX Security Symposium, San Jose, California, July 1996

**Figure captions**

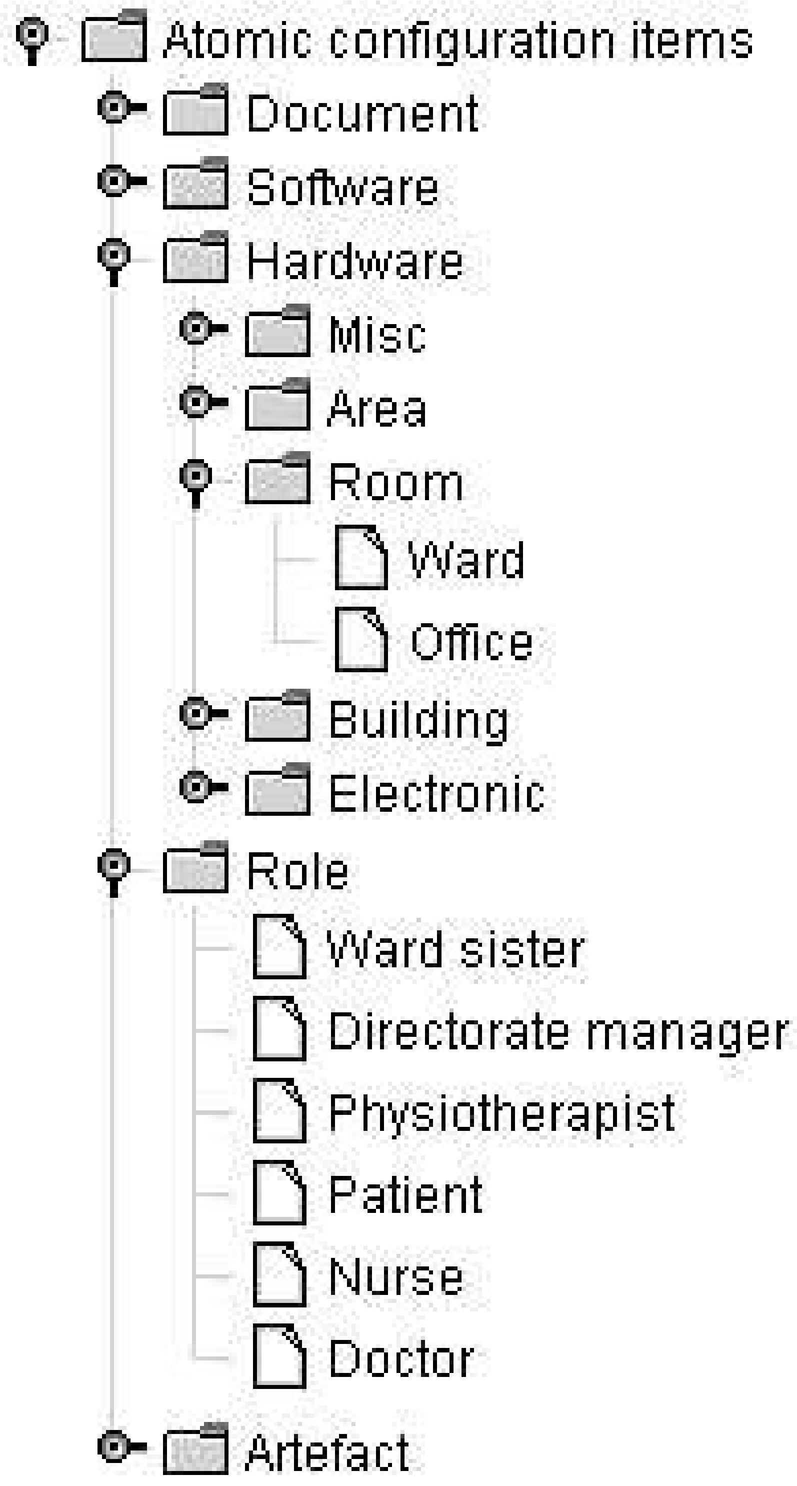
Atomic configuration items
Document
Software
Hardware
Misc
Area
Room
Ward
Office
Building
Electronic
Role
Ward sister
Directorate manager
Physiotherapist
Patient
Nurse
Doctor
Artefact

**Figure 1**

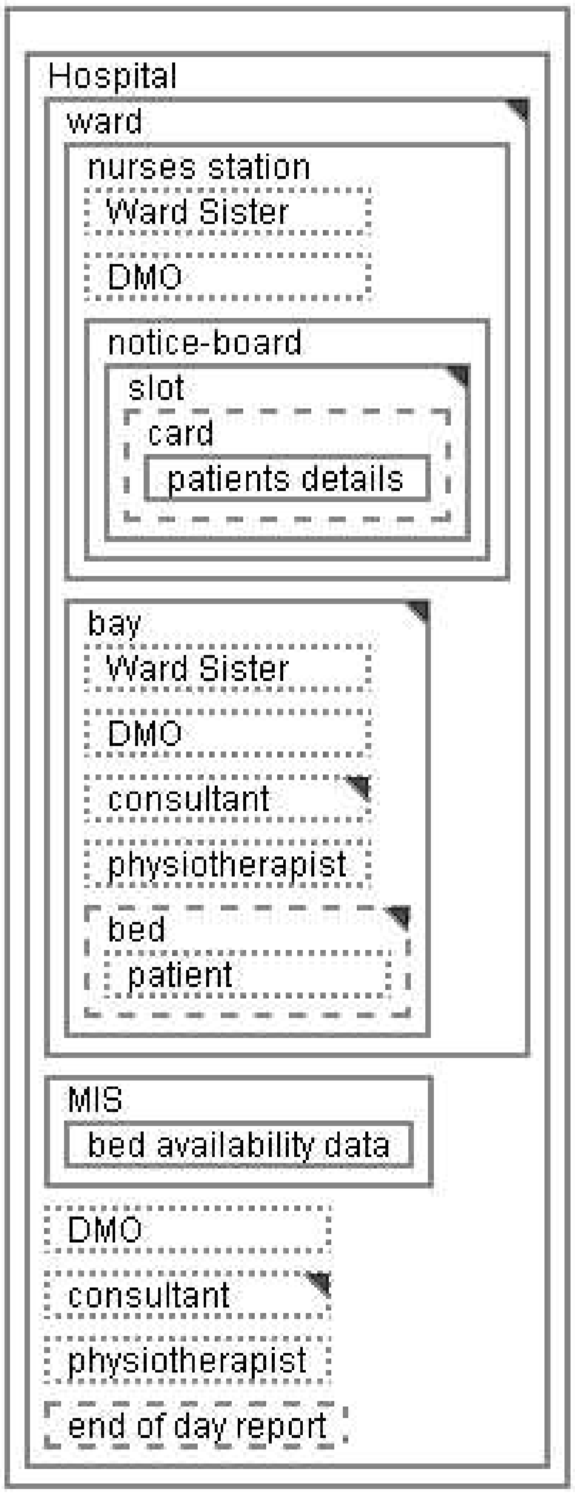
Hospital
ward
nurses station
Ward Sister
DMO
notice-board
slot
card
patients details
bay
Ward Sister
DMO
consultant
physiotherapist
bed
patient
MIS
bed availability data
DMO
consultant
physiotherapist
end of day report

**Figure 2**

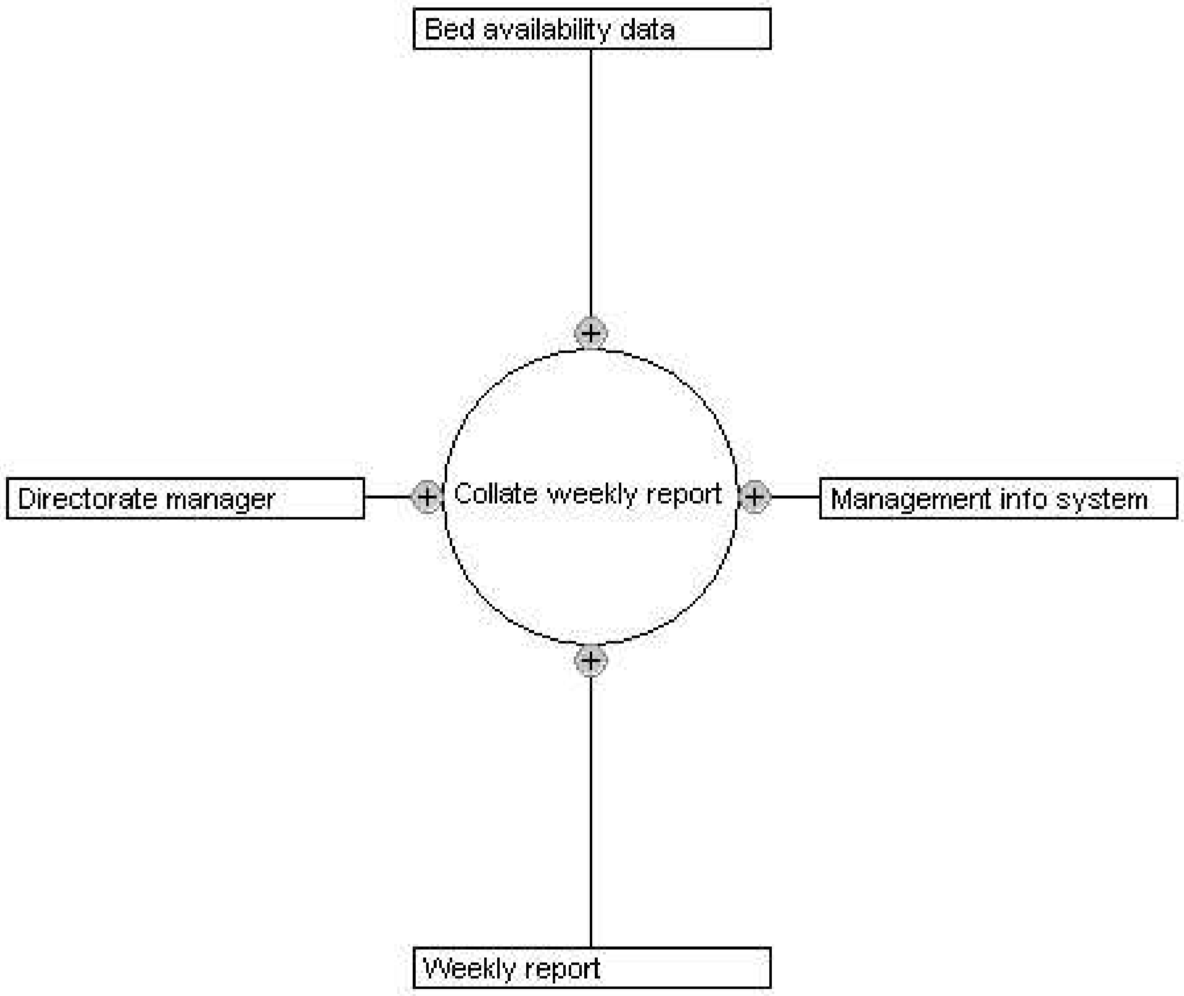
Bed availability data
Directorate manager
Collate weekly report
Management info system
Weekly report

**Figure 3**

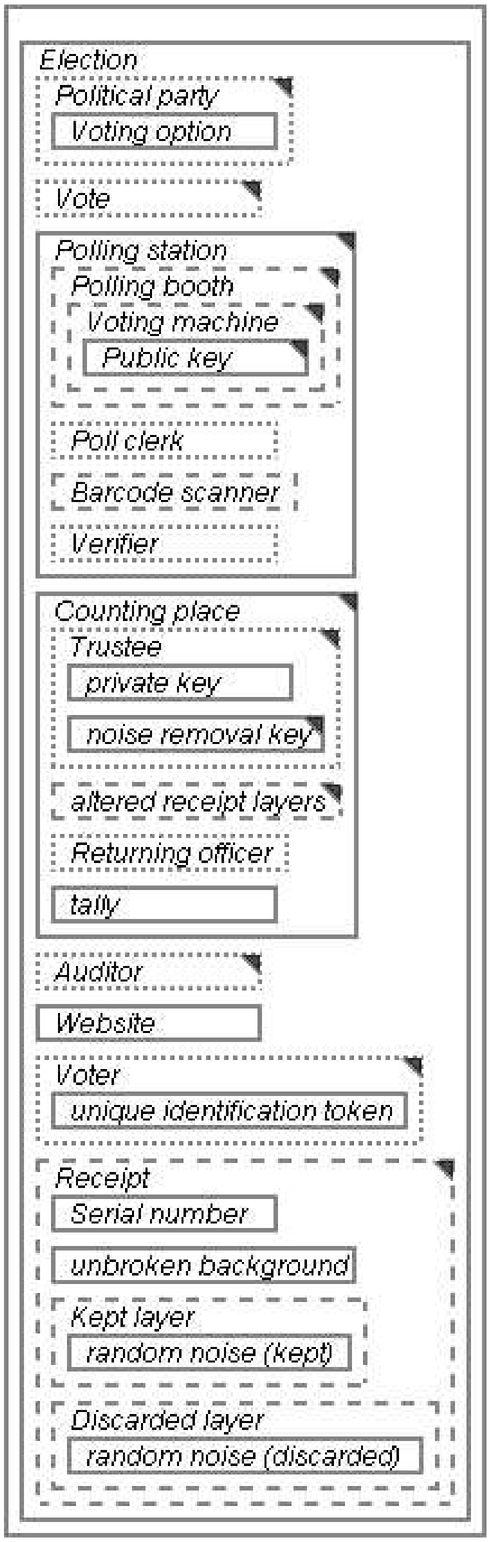
Election
Political party
Voting option
Vote
Polling station
Polling booth
Voting machine
Public key
Poll clerk
Barcode scanner
Verifier
Counting place
Trustee
private key
noise removal key
altered receipt layers
Returning officer
tally
Auditor
Website
Voter
unique identification token
Receipt
Serial number
unbroken background
Kept layer
random noise (kept)
Discarded layer
random noise (discarded)

**Figure 4**