Strider: Configuration Modelling and Analysis of Complex Systems

Simon Lock
Computing Department, InfoLab 21, Lancaster University, Lancaster, UK, LA1 4WA
lock@comp.lancs.ac.uk

Abstract

This paper describes an approach and support tool for the modelling and analysis of proposed reconfigurations to complex systems. The configuration models used are quick to construct and can help to promote understanding by the various stakeholders involved in system development and evolution. These models also form the basis of automated analysis, the results of which can help us to understand the full social and technical implications of the proposed changes to a system.

1. Introduction

Managing the evolution of large systems is a complex and difficult task. The full social and technical implications of any proposed changes must be fully appreciated before a decision is made whether or not to proceed with their implementation. This task is particularly difficult when the system in question is a complex, real-world system where technical components are intertwined with the social, political and organisational factors.

In many cases it is not feasible to deploy a new version of a system into its operational context for verification and validation purposes. This could for example be due to the fact that the system performs a critical role and failure of untested and unevaluated technical components may not be tolerable.

There are a number of possible solutions to this problem, including the use of simulators, test harnesses, small-scale trials, pilot studies and so on. Although it maybe be possible to verify and validate individual technical components before deployment, it is much more problematic to investigate the interactions these will have with social and organisational components of the final real world system. This is a very challenging problem since the interplay between technical and non-technical components is often very complex and the various human factors that are involved inject much variability and unpredictability into a system.

For these reasons we have developed Strider, an approach and support tool that permits the investigation and exploration of different configurations of social and technical components to aid in the maintenance, modification and evolution of complex systems. Without suitable support such as this, it is impossible to fully predict how changes made to the individual components or the overall configuration of a system will affect the operational behaviour of that system during real world operation.

This paper provides a description of the Strider approach and support tool and demonstrates how it may be used to assist in the evolution and maintenance of complex systems. The Strider approach is based on well-established fundamental modelling concepts used in the representation of purely technical systems. These include structures from entity-relational, control flow and data flow modelling. In order to apply concepts from such approaches to the modelling of system configurations a variety of abstractions, refinements and simplifications have been made.

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 this work is to explore how modelling and analysing just the top-level configuration structure of a system can benefit the maintenance process.

2. 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. 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 are present in 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.

3. 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.

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 or lesser extent in order to apply them to full socio-technical systems.

In this section we will describe a number of the most interesting and potentially useful approaches for the purposes of socio-technical configuration modelling. 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 implementators and those for system administrators. The following sections will consider each of the three different classes of approach in turn.

3.1. Architects

Past research in the area of systems architecture has resulted 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 [1] 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 [2], ZCL [3], PCL [4] and so on. These subsequent approaches are based upon the standard MIL concepts, but with the addition 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 [5]. 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. ACME is an interchange language used by numerous architecture description methods [5]. ACME utilises Components, connectors, systems, ports, roles, representations and representation maps to model the structure and composition of a system. ACME augments the structural representation of a system by allowing the specification of properties for each element of the system model.

3.2. 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) [6]. 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. Class and object models can be used to capture various static relationships between the entities that constitute a system. Activity diagrams can be used to represent dynamic configuration aspects of a system. Additional configura-tional information is illustrated in the description of how sets of such processes are chained together in order to perform larger granularity "Activities".

Conic [7] is an 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.

3.3. 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 [8] is one such approach that provides organisation wide configuration of Linux machines. LCFG provides a configuration language for the automated installation and configuration of multiple distributed computers. 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 single, or group of machines.

KUANG [9] 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.

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 regarding a system’s configuration is a solid and definite thing. 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 socio-technical systems.

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 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 systems. In addition to this, many of these deep concepts and structures may be hard or even impossible to determine from the available system descriptions.

Additionally, the notations utilized 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 be simplistic, high level, flexible and provide suitable structures and concepts to deal with the wide variety of domain entities that must be modelled. In developing the Strider approach we have extended, generalised and augmented structures and processes from existing approaches to support socio-technical 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.

5. 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 interested in modelling the "configuration, not operation" of a system. For this reason, we are not concerned with how processes behave internally, but rather limit ourselves to the pattern of inter-relationships that they imply.

5.1. 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.

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 that 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").

Configuration item classification hierarchies such as this include a variety of predefined classes for use in modelling the configuration of a system. This 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.

5.2. 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 [6]).

The information present in the structural model is normally relatively easy to determine by observation and discussion with workers and managers within an organisation as well as 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 1 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.

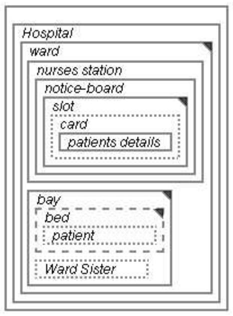

Figure 1. Structure

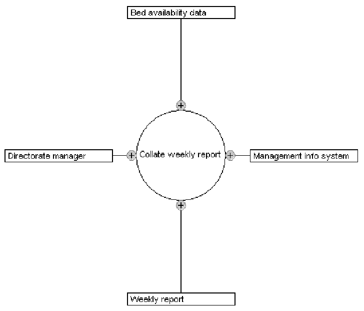

Figure 2. Process

The small triangular flags in the top right hand corner of some of the configuration items depicted in figure 1 indicate elements of the configuration that may be plural in nature. 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. 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 it presents an easily understandable facet of system configuration that can help ethnographers, developers, managers and other interest 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.

5.3. 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 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. 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 [6]. 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 2. 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.

5.4. Objectives

An objective is defined as a goal that can be achieved using the particular 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". It is possible to identify *Action objectives* (objectives whose purpose is to cause some action or activity to take place within a configuration) as well as *Information objectives* (objectives whose purpose is to derive, collate or retrieve some information from within a configuration).

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 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.

6. 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 system 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.

Automated analysis allows us to determine if a specified objective is achievable within a given configuration. Such analysis will identify all of the paths that are available to achieve that objective. For an existing system, this may not just be limited to the paths that are used in the day-to-day operation of that system, but may also incorporate possible, yet currently unused mechanisms. 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 same objective

To perform analysis, the operator specifies a particular objective by selecting a start and endpoint configuration item or process. To help optimise analysis, the operator also indicates whether the objective type is either action or 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.

Output from analysis is a list of 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 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.

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.

7. Applying the approach

In order to demonstrate the utility of the proposed configuration modelling and analysis approach, we will now apply it to a real world case study. Due to the size and complexity of real world systems, it is not possible to manually consider the effects, implications and ramifications or all configuration changes. This is where the Strider approach and support tool can be most useful, by assisting an 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 nature 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 consider the utility and practicality of the Strider approach.

7.1. Example case study

In order to help illustrate the Strider approach we have selected a real world, complex, socio-technical system to act as a case study. We shall focus upon the domain of ballot based voting, and in particular the Chaum scheme for electronic voting [10]. This scheme is a proposed mechanism to provide a dependable, high security and trustworthy method of electronic voting in national and regional elections or referenda. We have primarily chosen this scheme as our case study because it is a well-documented example of a complex socio-technical system.

In addition to this, the application domain of this system will require only minor explanation, since most people should have a general understanding of how an election operates and the context in which it takes place.

The introduction of the Chaum electronic voting scheme can be viewed as a reconfiguration or modification of the traditional paper based voting mechanism. Many of the components present in the whole system (such as voters, political parties, monitoring organisations, electoral registers etc.) would remain the same. However, new social and technical components associated with electronic voting would replace their paper-based counterparts.

We find this case study particularly interesting because it not only involves both social and technical components, but also some very interesting interplay between the different elements.

The integrity of any voting mechanism is essential to maintaining the electorates trust in the system and the party that it places into power. For this reason, it is not acceptable to sustain failures in the operation of the voting mechanisms. This fact is evident from the outcry arising from failures within the voting mechanisms of the 2000 US presidential elections.

Trust in the voting mechanisms is particularly important with the recent trend of falling voter turn out, where a poor reputation of the mechanisms could discourage individuals from voting.

For these reasons, it is highly undesirable to incorporate the new electronic voting components into the overall socio-technical system that is “the election”, without first gaining considerable understanding of how the final system will operate and what problems might be encountered. Configuration modelling and analysis provides us with an ideal opportunity to gain some insight into this proposed re-configuration of the system.

In order to understand the motivations behind the Chaum scheme, let us first look at some of the key 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 the 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 [10]. The following sections do however present Strider configuration models that will shed more light on the operation of the Chaum scheme.

7.2. 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 3 below. A description of some of the key, Chaum specific configuration items are then shown in table 1.

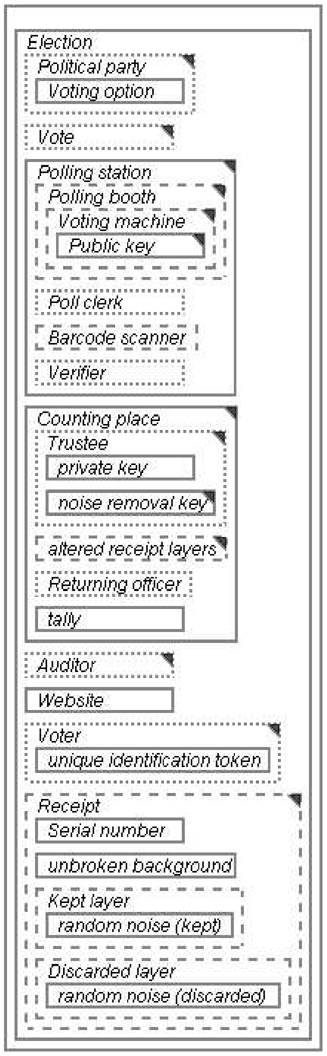

Figure 3 Structural model for Chaum

Table 1. Configuration Items

Voting option - One of the candidates which can be voted for in an election (can include 'abstain', 'spoil' and 'other')
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
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. The serial number can either be just the next available consecutive number or a voter specific number (this decision is independent of scheme)
Unbroken background - A visual security feature present on the receipt which is used by the voter to ensure it's validity
Public key - A PGP public key used to encrypt a noise removal key for a particular trustee
Trustee - Person (or machine) charged with the decrypting, mixing, publishing and counting votes
Private key - A key used to partially remove noise from a layer of the receipt
Altered receipt layer - A layer of the receipt from which some noise has been removed

7.3. Process model fragments

To supplement the description provided by the structural model, a set of 27 processes model fragments have also be derived for the Chaum scheme. A selection of the processes that were identified are shown in table 2 below.

Table 2. Selected processes

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
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
Published mappings - the act of partially publishing mix mappings on a website
Votes tallied - the cumulative summation of all votes for the different parties
External audit - the verification of the counting process by an external independent auditor
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
Check website – checking by various interested parties of the voting information published on the website
Reassures – an attempt by auditors to increase voters confidence in the accuracy and security of the voting scheme

7.4. Objective analysis

The objective that we will focus on is the ability for a political party to find 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 an individual.

This is an undesirable objective from the perspective of a free and fair election since its attainment can support intimidation as well as the "sale" of votes by a voter. The purpose of assessing this objective to investigate how the configuration of the Chaum scheme prevents such objectives from being achieved.

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 which 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.

8. 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 into the proposed configuration of a particular system.

The analysis of objective paths allows us to not only investigate the technical operation of the proposed re-configuration, but also to explore how any changes will effect the social constructs that surround and inter-operate with those technical components. In particular, Strider allows us to assess aspects of systems that are not possible using traditional testing mechanisms. Such analysis is thus essential in order to fully validate the appropriateness and correctness of a proposed reconfiguration.

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 changes to a system's structure and its interrelationships. This is particularly true for complex 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 an analyst to become more confident about their ability to 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. This can help to ensure higher dependability of the configuration in case of partial failure of one or more paths. Conversely, analysts 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; the identifying and highlighting of grey areas that might not have been given adequate consideration in the construction of a system configuration; identifying and highlighting of potential pitfalls specific to particular configurations; justification of the use of the various elements present in a configuration (e.g. the use of receipts).

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 configuration items present in a system. Strider provides a unique style of high-level analysis that offers support to human operatives attempting to understand the consequences and implication of complex system reconfiguration.

9. 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: the assessment of varied aspects of system configurations to assistance in managing the evolution of complex systems. By allowing developers to gain insight into the full implication of changes to a system it only then becomes possible to make informed decisions about the evolution of that system.

Strider is part of the ongoing DIRC project [11].

10. References

[1] 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

[2] 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

[3] 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

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

[5] D. 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

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

[7] 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

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

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

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

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