COMPUTING
SCIENCE

Dependability Explicit Metadata: Experimental Results and Research Issues

Carl Gamble and Steve Riddle

TECHNICAL REPORT SERIES

No. CS-TR-1249  May 2011
Dependability Explicit Metadata: Experimental Results and Research Issues

C. Gamble and S. Riddle

Abstract

This report presents the results of from the development of prototype tool support to demonstrate metadata-based dynamic resilience, and a set of research issues that outline further work to improve the fidelity of the tool support and mature the resilience framework.

The problem addressed by the work is the provision of support for dynamic reconfiguration in the context of Network Enabled Capability, by maintaining resilience dynamically. Resilience is defined as the ability of a system to maintain a dependable service while assimilating change, without loss of functionality. To achieve the required level of resilience requires the availability at run-time of dependability explicit metadata, information about system components which can govern decision-making about reconfiguration.

This report is in two parts. The first part describes the experiment, which is used to illustrate how metadata-based reconfiguration can be more resilient to change than a system with a static configuration. This experiment is a deterministic simulation which is instrumented to allow the properties and configuration of the system to be read and altered. The second part discusses the simplifications used in the experiment, describes the issues which need to be considered in order to improve the fidelity of the experiment, and presents a set of research issues for further development of the resilience approach in terms of the metadata, provenance and policy definitions which make up the framework.

The key results from the experiment efforts are the understanding of the properties that would be required of a simulation platform to gain a realistic confidence in the approach. The main requirements are a high fidelity simulation of a system-of-systems (SoS), with a valid model of the component failures. This would allow the effectiveness of the transfer of metadata and the decisions made upon it to be assessed.

A set of six properties is provided that may be used to guide the definition of metadata for a component or SoS. The main research issues are the generation of ontologies of metadata by examination of existing systems, and the exploration of protocols used to ensure that components within the SoS get the metadata they need in a timely manner.
The means for generating policies is seen as the biggest research issue in that particular area, with the policies presented in an earlier deliverable been programmatic and too fine grained for use in a battlespace situation. An approach using natural language terms is suggested and related work from the security community is discussed.

The final research issue relates to provenance metadata. Here the provenance structure used during in the experiment is presented and shown to be of use for detecting data incest and data repudiation.

The work reported in this document remains at an early stage and is still relatively low Technology Readiness Level (TRL). The research issues section addresses the developments in the approach which would be required in order to increase its maturity.
Bibliographical details

GAMBLE, C., RIDDLE, S

Dependability Explicit Metadata: Experimental Results and Research Issues
[By] C. Gamble, S Riddle
(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1249)

Added entries

NEWCASTLE UNIVERSITY

Abstract

This report presents the results of the development of prototype tool support to demonstrate metadata-based
dynamic resilience, and a set of research issues that outline further work to improve the fidelity of the tool support
and mature the resilience framework.

The problem addressed by the work is the provision of support for dynamic reconfiguration in the context of
Network Enabled Capability, by maintaining *emph{resilience}* dynamically. Resilience is defined as the ability
of a system to maintain a dependable service while assimilating change, without loss of functionality. To achieve
the required level of resilience requires the availability at run-time of *emph{dependability explicit metadata},
information about system components which can govern decision-making about reconfiguration.

This report is in two parts. The first part describes the experiment, which is used to illustrate how metadata-based
reconfiguration can be more resilient to change than a system with a static configuration. This experiment is a
deterministic simulation which is instrumented to allow the properties and configuration of the system to be read
and altered. The second part discusses the simplifications used in the experiment, describes the issues which need
to be considered in order to improve the fidelity of the experiment, and presents a set of research issues for further
development of the resilience approach in terms of the metadata, provenance and policy definitions which make
up the framework.

The key results from the experiment efforts are the understanding of the properties that would be required of a
simulation platform to gain a realistic confidence in the approach. The main requirements are a high fidelity
simulation of a system-of-systems (SoS), with a valid model of the component failures. This would allow the
effectiveness of the transfer of metadata and the decisions made upon it to be assessed.

A set of six properties is provided that may be used to guide the definition of metadata for a component or SoS.
The main research issues are the generation of ontologies of metadata by examination of existing systems, and the
exploration of protocols used to ensure that components within the SoS get the metadata they need in a timely
manner.

The means for generating policies is seen as the biggest research issue in that particular area, with the policies
presented in an earlier deliverable been programmatic and too fine grained for use in a battlespace situation. An
approach using natural language terms is suggested and related work from the security community is discussed.

The final research issue relates to provenance metadata. Here the provenance structure used during in the
experiment is presented and shown to be of use for detecting data incest and data repudiation.

The work reported in this document remains at an early stage and is still relatively low Technology Readiness
Level (TRL). The research issues section addresses the developments in the approach which would be required in
order to increase its maturity.

About the authors

Carl received his BEng (Hons) in Manufacturing Systems and Mechanical Engineering from the University of
Leeds in 1996. Between 1997 and 2003 he worked as an Industrial Engineer for Faurecia in Washington where he
had responsibility for developing a synchronous delivery system and the programming of ABB water jet robots. In
2003 Carl left Faurecia and started an MSc in Computing Science at the University of Newcastle, graduating with
a Distinction in 2004. Carl is currently involved in two main activities. The first is finishing the corrections in his
PhD thesis on using architectural styles to detect architectural mismatch when composing web services, this is
supervised by Dr Cristina Gacek. His main role however is as an RA on the Ministry of Defence funded SSEI
project. He is involved in the dependability sub task, specifically looking at using dependability metadata to drive
software reconfgurations in the field of Network Enabled Capability (NEC) systems.

Dr Steve Riddle is a lecturer in Computing Science, with modules in requirements engineering, high integrity
software development, formal specification and software engineering to undergraduate and masters level students.
He is also involved in coordinating final year projects, and has responsibilities for industrial liaison and student placements. Steve's research contributions have included novel safety analysis techniques (INCO-COPERNICUS ISAT), component contracts and protective wrapper architectures (EPSRC project DOTS) and resilience (FP6 NoE ReSiST). He is a Theme Lead in the SSEI project and leads work in dependable dynamic reconfiguration in that project. Beside these projects his research interests include requirements volatility, evidence-based argumentation and risk management. Steve worked in the BAE SYSTEMS DCSC (Dependable Computing Systems Centre) from 1997, first as a Research Associate then as technical lead on the requirements and specification strand from 2002-2007. Steve obtained his BSc in Computer Software Technology at the University of Bath in 1991, and completed his PhD at Bath concerning the use of partial specifications and formal refinement theory to aid the process of explaining complex systems.

**Suggested keywords**

DEPENDABILITY EXPLICIT METADATA
METADATA PROPERTIES
RECONFIGURATION POLICIES
PROVENANCE
SIMULATION
EXPERIMENTAL RESULTS
Abstract

This report presents the results of from the development of prototype tool support to demonstrate metadata-based dynamic resilience, and a set of research issues that outline further work to improve the fidelity of the tool support and mature the resilience framework.

The problem addressed by the work is the provision of support for dynamic reconfiguration in the context of Network Enabled Capability, by maintaining resilience dynamically. Resilience is defined as the ability of a system to maintain a dependable service while assimilating change, without loss of functionality. To achieve the required level of resilience requires the availability at run-time of dependability explicit metadata, information about system components which can govern decision-making about reconfiguration.

This report is in two parts. The first part describes the experiment, which is used to illustrate how metadata-based reconfiguration can be more resilient to change than a system with a static configuration. This experiment is a deterministic simulation which is instrumented to allow the properties and configuration of the system to be read and altered. The second part discusses the simplifications used in the
experiment, describes the issues which need to be considered in order to improve the fidelity of the experiment, and presents a set of research issues for further development of the resilience approach in terms of the metadata, provenance and policy definitions which make up the framework.

The key results from the experiment efforts are the understanding of the properties that would be required of a simulation platform to gain a realistic confidence in the approach. The main requirements are a high fidelity simulation of a system-of-systems (SoS), with a valid model of the component failures. This would allow the effectiveness of the transfer of metadata and the decisions made upon it to be assessed.

A set of six properties is provided that may be used to guide the definition of metadata for a component or SoS. The main research issues are the generation of ontologies of metadata by examination of existing systems, and the exploration of protocols used to ensure that components within the SoS get the metadata they need in a timely manner.

The means for generating policies is seen as the biggest research issue in that particular area, with the policies presented in an earlier deliverable been programmatic and too fine grained for use in a battlespace situation. An approach using natural language terms is suggested and related work from the security community is discussed.

The final research issue relates to provenance metadata. Here the provenance structure used during in the experiment is presented and shown to be of use for detecting data incest and data repudiation.

The work reported in this document remains at an early stage and is still relatively low Technology Readiness Level (TRL). The research issues section addresses the developments in the approach which would be required in order to increase its maturity.

1 Introduction

This is the final report from a task that has investigated the support for dynamic reconfiguration of software-intensive Systems-of-Systems (SoS) to allow a level of resilience to be maintained. We use the term metadata to refer to data which describes functional and non-functional properties of components: this distinguishes metadata from ‘normal’ data which is used by components as part of their normal operation [1]. Resilience refers to the ability of a system to maintain a predictable level of dependability in the
face of changes in the environment.

The Task has produced two earlier written reports:

1. A survey of the means to acquire and assess dependability metadata (CS-TR-1232, [2]). This presented methods for acquiring primary metadata (metadata measured from the system itself) and for evaluating a system based on this data.

2. A more detailed case study description described using MODAF [3] views (CS-TR-1248, [4]), illustrating how metadata and policies could be used to govern the system configuration, using some illustrative scenarios and two policy languages, PDL (Policy Description Language) and T-R (Teleo-Reactive programming).

The reconfiguration framework described in the above reports is divided into two parts. An *internal* part concerns the software configuration of a specific asset; the *external* part is focussed on the emergent configuration of all visible assets. Both parts share an abstract structure, featuring a goal policy that uses available health monitoring and sensor metadata to determine if there is a need to reconfigure, and what the goal of that reconfiguration is. The external part of the framework corresponds to an SoS view, where we are dealing with the use of services provided by other assets in the battlespace and the provision of services to other assets. Such a view has been the focus of work described in the third deliverable and in this report.

This report forms the final deliverable for the Task, presenting the development of prototype tool support to demonstrate metadata-based dynamic resilience, and a set of research issues to outline further work to improve the fidelity of the tool support and mature the resilience framework. The report is in two main parts. The first part (Section 2) describes the experiment which is used to illustrate how metadata-based reconfiguration can be more resilient to change than a system with a static configuration. This experiment is a deterministic simulation which is instrumented to allow the properties and configuration of the system to be read and altered. The second part (Section 3), discusses the limitations of the experiment performed at Newcastle, presenting these along with other issues as properties that should be considered when designing further experiments. It also presents a set of research issues for further development of the resilience approach in terms of the metadata, provenance and policy definitions which make up the framework.
The work reported in this document remains at an early stage and is still relatively low technology readiness level (TRL). In the conclusion (Section 4) we sum up the achievements and limitations of the task.

2 Experiment/Demonstration

The purpose of the experiment is to investigate whether using metadata about components and data to govern the configuration of an SoS can make the system more resilient to change than a system with a static configuration. The experiment takes the form of a deterministic simulation of a scenario, where the simulator is instrumented to allow the properties and configuration of the SoS formed by friendly forces to be both read and altered.

2.1 Simulator

The experiment is performed using a simulation tool developed as part of the task. The tool takes six input parameters that define the simulation to be run and where the results are to be stored. Those inputs are described in Sections 2.1.1 to 2.1.7.

2.1.1 Actors Input File

In the simulation all mobile units are called actors. There are two main types of actor: friendlies and hostiles. A friendly actor may provide some function for the SoS being simulated, while a hostile just exists to provide something that can be detected. An actor’s input file can contain an arbitrary number of actors, where each is essentially defined by three properties:

ID A unique identifier for the actor in the simulation.

Class defines what type of actor this is. The options are currently Unmanned Aerial Vehicle (UAV), Analysis, and Command for friendly forces and Hostile for a hostile actor. The details of these types and services are described later in Section 2.2 where the logical system being simulated is explained.

Waypoint(s) define the points the actor visits in the world, the speed it moves between those points and how long it lingers when reaching those points.
2.1.2 Capabilities Input File

The capabilities file defines the initial configuration of the SoS being simulated in terms of which actors are providing which services to which other actors. The simulation has a concept of a capability instance, which is a specific set of actors providing services to each other that results in an instance of a capability in the SoS. A capability instance is defined by a set of actor instructions, each of which describes the role an actor performs in providing that capability. It contains the following:

**Actor ID**  the actor that will provide the service.

**Service ID**  the service that will be provided.

**Received Message**  the message data (token), channel and originating actor ID for any messages the service should receive.

**Sent Message**  the message data (token), channel and target actor ID for any message the service should send.

The capabilities file contains zero or more capability instances, each of which contains one or more actor instructions.

2.1.3 World Input File

The world file defines the fixed objects that exist in the world of the scenario. These are currently limited to buildings or hills. The purpose of these fixed objects is to act as barriers to message propagation between actors, to allow scenarios where point-to-point connectivity between specific actors may not always be possible, for example UAVs separated by a range of mountains.

The world file contains zero or more world features, each of which is defined by the following properties.

**ID**  the id of the object

**Type**  type of the feature, either hill or building

**Vertices**  depending on the type there will be a number of named vertices defining the points making up the three dimensional object using x,y,z coordinates. Each world object type expects a specific set of vertices, an example of the hill object type is shown in Figure 1.
Figure 1: The vertices expected by the simulator to define the pyramidal hill world object.

2.1.4 Events Input File

The events file allows the simulation designer to define a sequence of changes to actor properties that will occur during the simulation. It is by changing properties that component failures can be introduced into the system. As an example, a transmitter failure could be simulated by reducing the transmitter power to zero. Such an event could be followed after some time by another event restoring the original transmitter power, thereby allowing transient faults to be simulated. The file contains zero or more events each of which has the following properties.

**Time** the simulation time at which the property change will occur.

**Actor ID** the actor that will be affected.

**Component ID** the component within the actor that will be affected.

**Property ID** the property of that component that will be affected.

**New Value** the new value of the property.

2.1.5 Simulation Run Time

This sets the number of seconds the world should be simulated for.

2.1.6 Output Folder

Each time an event occurs in the simulation, the state of all actors being simulated is written to an XML file to allow post simulation analysis. The detailed contents of this file are too verbose to include here, but in summary it includes a description of the event:
**Event time** the time that the event occurred

**Event type** the type of event, e.g. a message exchanged between actors or an actor reaching a waypoint

**Special Tags** any identifying event tags associated with this event (these will be described below in Section 2.1.7)

The file also includes the state of each actor:

**Actor Identification** its ID and type

**Actor Navigation** its current actual location, its navigation type (Global Positioning System (GPS)/Inertial) and estimate of error and its waypoints

**Actor Services** the services the actor can provide, any instances of the services that are running and their processing state

**Actor Memory** the information known to that actor including its provenance, metadata and any special tags attached to it (Metadata used in the simulation is discussed in Section 2.2.1)

**Actor Ports** the state of the communication ports used by the actor, e.g. 2.4Ghz radio transmitter

**Actor Sensors** the state of any sensors attached to the actor, e.g. visual sensor (camera)

Each file is named by the time at which the event occurred and an event sequence number such that they are in order.

**2.1.7 Data and Event Tags**

To reduce complexity, the simulation abstracts away from detailed implementations of the services provided by actors in two ways. The first abstraction is that data in the system is represented by a string token, for example “imageData”. The second abstraction is that no actual processing of the data takes place and service instances take a fixed number of cpu cycles to act upon received data. So when a service receives an item of data, either from another service or from a sensor, that data enters a queue and is processed
in strict order. The amount of time taken to process the data is a function of the number of processor cycles required, the speed of the processor on the actor and the number of active services the actor has running.

This abstraction does not however facilitate the tracking of data through the system. To solve this issue the simulation creates uniquely identifiable event tags when data is sensed by the system and these tags are appended to any messages that are derived from the sensed data. In this way it is possible to determine if the contents of a message may include data about a particular sensor event. It should be noted that the presence of an event tag does not say that the data is useful, this can only be determined from the metadata associated with the message (Metadata used in the simulation is discussed in Section 2.2.1). So for example, if an event tag associated with the possible detection of a single foot soldier is attached to a message, but the resolution of the data in that message is such that only objects of 10m or larger may be resolved, then we would say that the message does not contain useful data about the foot soldier.

It should be noted that the event tags are purely used so the simulation has a means for tracking which messages may contain useful information. As such they are not metadata, and should not be used by either the policy or actors to make decisions as they would not exist in a real system.

2.2 Anti Guerilla Operations Scenario

![Figure 2: OV1 [3] view of the AGO scenario](image)

Figure 2: OV1 [3] view of the AGO scenario
The scenario that will be used in the simulation considers a fictitious operation where an allied force is attempting to detect and neutralise a guerilla threat [5]. The allied forces are employing UAVs to perform the surveillance part of the operation, while theatre command processes the intelligence before passing it on to the ground troops, transport helicopters and artillery so that they may perform their parts of the operation safely, Figure 2.

In our simulation we concentrate on the intelligence gathering part of the operation, which we have termed the target tracking capability, shown in Figure 3. The capability consists of a chain of three services as follows:

**Visual Sensor** This service controls a visual sensor and streams image data messages to any service requiring them.

**Image Analysis** The analysis service has the job of processing image data to locate any hostile actors within that data. It outputs location data messages containing identifying information and estimated location of the actors found.

**Location Collator** The location collator takes location data messages and combines them with any past messages to form tracks for identified actors.

![Figure 3: SV10c [3] view of the target tracking capability](image)

Alongside these there is a fourth service defined in the simulation, the data relay service. This allows an actor to receive any message from another actor and forward it on to another without altering the message beyond adding itself into the provenance tree.
The simulation contains three actor types as shown in Table 1. The UAV contains the service and sensor required to capture images, and supports the data relay service. The Analysis station supports the image analysis service and the Theatre command actor type supports the location collator service. They all are able to use both the 2.4Ghz and 5.8Ghz radio bands for communications.

<table>
<thead>
<tr>
<th></th>
<th>Service</th>
<th>Sensor</th>
<th>Ports</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Visual Sensor</td>
<td>Image Analysis</td>
<td>Location Collator</td>
</tr>
<tr>
<td>UAV</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Analysis Station</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Theatre Command</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Services, Sensors and Ports attached to each actor type in the simulation

### 2.2.1 Metadata

The simulation employs several metadata associated with the data shared during message exchanges. These are as follows:

**Image Data Message**

- **Image quality** the image quality is represented using the National Imagery Interpretation Rating Scale (NIIRS) [6] scale, where 0 represent unusable, 9 represents the highest quality and, for example, 6 would allow the spare tyre on a medium sized truck to be identified. Its value is a function of the visual sensor resolution, field of view and altitude.

- **Positional accuracy** the positional accuracy is represented on a scale similar to NIIRS where 0 indicates a potential error of greater than 10km and 9 indicates an error of less than 0.1m. The value
is a function of the navigational means employed (GPS or Inertial Navigation), the time since calibration and geographic position of the actor. The positional accuracy value directly represents the estimated error for the actor carrying the image sensor, therefore the coordinates associated with that image or any part of it are subject to the same potential error.

**Location Data Message:**

**Identification accuracy** the identification accuracy uses the same scale as NIIRS and indicates what is the smallest actor that could be identified in this message. It is a function of the image data quality and also any data loss during transmission (described later in this section).

**Positional accuracy** the positional accuracy is a direct copy of the positional accuracy of the incoming message.

The above represents the metadata included in the messages sent out by their respective services. There is another metadata that is recorded by the receiving service, the *message readability*. A message is sent using the maximum power allowed by the transmitter for that channel on the actor, and the simulation then uses various physical properties to determine the signal power actually received by the target. These are the positions of the sending and receiving actors and the presence of any obstructions on the straight line between them\(^1\). Alongside the received signal power the simulation also allows a simple form of error correction to be built into a message, in the form of a redundancy factor. The readability of a received message then is a function of the received signal power, the sensitivity of the receiving antenna and the message redundancy factor. A readability of one or more indicates that the original message can be read perfectly and so is suitable for automatic or manual processing. A readability of less than one indicates that even with the error correction there is still some data loss. In this situation we assume that automatic processing is not possible, but it may be possible to manually process the data albeit with some loss.

In the above capability the image analysis service is manually processed, so messages received with a readability of less than one can be tolerated but

\(^1\)The signal propagation model simply assumes a line-of-sight communication path and ignores factors such as multi-path interference.
the identification accuracy may need to be reduced depending on the degree of data loss. The location collator service is considered to be automatic and as such a readability of less than one results in the data being considered corrupt and unusable.

2.3 Experimental Measurement

The measures chosen to assess the effectiveness of the simple policy used in this experiment attempt to evaluate the SoS in terms of both its timeliness and quality of output. The timeliness measure utilises the event tags described previously in Section 2.1.7. Every five seconds the hostile actor in the scenario generates an event tag which, if in view, is detected by the visual sensor. This event tag is then attached to the next image data message sent by the UAV and after that it is attached to the next location data message sent out by the image analysis station to the location collator at theatre command. The timeliness is calculated by comparing the time at which the event was generated with the time it is eventually received at the theatre command, Figure 4. The other timings are also recorded for completeness.

![Figure 4: The three timing points available in the experiment. For the purpose of this experiment, only t0 and t2 are used.](image)

The other aspect, quality, relates to the utility of the data that is delivered to theatre command. For this we use the NIIRS image rating metadata that the services in the simulation generate automatically. The original NIIRS value is generated by the visual sensor when it generates image data and is derived from the UAV altitude, camera resolution and field of view. This is the baseline value and can not be improved upon for any one image, it can however be degraded due to data loss during message transfer. It is the differential between the rating of the initially captured image and the identification accuracy finally delivered to theatre command that forms the quality measure for the SoS. The event tags are once again used to provide a traceability between the original and finally delivered data.
2.4 Experiment Details

The experiment consists of two simulation runs which are identical except for the policy applied to govern the configuration. The simulations contain a total of five actors. There are two UAVs, one analysis station, one theatre command and a single hostile actor. The analysis station and theatre command are both static while the other three actors are mobile. Of the two UAVs, one is following a simple search pattern covering a forward area, while the second is patrolling the forward point of the controlled ground space to dissuade any attempted guerilla movements onto friendly territory. These flight areas and relative positions of the analysis station and theatre command are illustrated in Figure 5.

![Diagram of flight areas and positions of friendly actors](image.png)

Figure 5: Flight areas and positions of friendly actors.

It is not the object of the experiment to assess the effectiveness of the search pattern employed and so this aspect has been removed by giving the hostile actor an identical, but lower, flight path to the UAV performing the
search. This ensures that every hostile event is detected and so any lost data would be solely due to the configuration of the friendly system, not the search pattern.

The initial SoS configuration is; UAV1 provides the visual sensor service used by the image analysis service of Analysis and Command hosts the location collator service. UAV2 does not form any part of the initial logical SoS. This is shown in Figure 6.

![Figure 6: Initial configuration of the experiment SoS.](image)

The two experiments differ only in the policy that was used to adjust the configuration. The first experiment used a null policy, one that takes no actions, the intention being to provide a base line representing a statically configured system. The second experiment utilised a deliberately simple policy consisting of a single rule, the intention being to both demonstrate the principle and allow the experiment to be conducted in a timely manner as policy evaluation is performed manually at this time.

The rule has the event:condition:action structure associated with the PDL (Policy Description Language), one of the languages discussed in deliverable 3 of this task [4]. Essentially, rules of this kind state that if a specific event occurs, and if the associated condition is true then perform the stated action.

The policy rule used is as follows

**event** A message is exchanged between two components

**condition** The readability of the received message is less than 1

**action** Search for an actor that provides a data relay service located closer to the sender and receiver than the current distance between the sender and receiver

The policy is evaluated immediately after the receipt of each message during the simulation run. The required metadata is readily available being part of the provenance tree for each message and so can be accessed by querying the actor’s memory via the user interface of the simulator.
The simulation runs are 325 seconds long, this allows for 64 hostile events to be generated and also for the UAV performing the search to complete one complete circuit of the the flight path.

2.5 Experiment Results

2.5.1 Story

The experiment was set up such that the initial configuration was suitable at the start of the simulation, would be stressed during the middle portion and would return to suitability at the very end. There were a total of 21 messages sent from the searching UAV to the analysis station, at 15 second intervals and we will now discuss the readabilities recorded for each message and its effect on the configuration.

Figure 7 shows the readability metadata recorded for the null policy experiment. It can be seen that the first four measurements are in the target zone of greater than one, however from the fifth to the 19th messages the readability scores are all below one. This was expected as the distance between UAV1 and the analysis station increases for the first half of the experiment and then decreases during the second half. The null policy does not permit changes to the configuration in response to this so there will be a degree of data loss as a result. This is discussed in the following section.

Figure 7: Null policy, readability of data by message sent from UAV 1

The readabilities for the simple policy experiment are shown in Figure 8.
Here we see the same pattern for the first five message as seen in the previous experiment, however this time the policy does permit a reaction to the fifth message having a readability of less than one. The action requires that an additional data hop be added between the sending and receiving component and in the case of this experiment there is only one choice for this and that is to utilise the data relay service of UAV 2. The resulting logical configuration is shown in Figure 9.

Figure 8: Simple policy, readability of data by message sent from UAV 1

![Figure 8](image1.png)

Figure 9: Result of configuration change by the simple policy

The next six messages sent from UAV 1 to UAV 2 and then from UAV 2 to the analysis station all have a readability of greater than one, however the seventh message post configuration change see the readability from UAV 1 to UAV 2 drop just below one. This triggers the reconfiguration rule for a second time, however on this occasion there is no component that may be used to bridge the gap between the UAVs without introducing individual transmission distances that are longer than the current one. Therefore the action fails to find a potentially better configuration and no change is performed. After this message UAV 1 begins to return to its original location,
closing the distance between the two UAVs, resulting in an ever increasing readability score.

2.5.2 Performance Measures

The first measure we consider is the latency between hostile event tags being generated as part of the simulation and their eventual receipt at theatre command. Figure 10 shows that there are three distinct latencies recorded for the null policy experiment. The three times arise from the hostile events occurring every 5 seconds while both the visual sensor service and the image analysis service operate on a 15 second cycle, this results in three hostile events being contained in each message. The important result from the graph is that all events had a latency of 15 to 25 seconds.

![Figure 10: Null policy, Time for each event tag to reach theatre command](image)

The results from the simple policy experiment are shown in Figure 11. Here we see that there are two groups of latencies. The group with the lowest count represents the system before the configuration change and so have the same latency range of 15 to 25 seconds as observed for the null policy. The group with the larger count represent the system performance post configuration change and have a latency range of 17 to 27 seconds. This increase is consistent with the two second lag added by the receipt and re-transmission time of the data relay service.
The second performance metric considered is the quality of data received by theatre command, the measure used is the *Identification at NIIRS* metadata included in the location data messages output by the image analysis service. This metric is directly influenced by the original NIIRS metric of the image sent from the UAV and the readability of that message.

Figure 12 shows that for the null policy experiment, the quality metric starts off with a value of six, the optimal value as capture by the visual sensor, but drops down by three levels to three by the middle of the experiment and then gradually returns to the original value at the end. If this were a real system that would indicate an initial image resolution allowing objects of 0.4-0.75m to be identified to a resolution requiring objects to be 2.5-4.5m for identification.

The results from the null policy experiment bear this out, Figure 12.

The quality results for the simple policy experiment are shown in Figure 13. Here we see that apart from the contents of two messages, all data is received with the optimum NIIRS value of six. The first drop to a value of five relates to the below one readability of the fifth message, the subsequent return of the quality to six can be attributed to the reconfiguration. The second drop relates to UAV 1 reaching its furthest point from UAV 2 and the readability of a message dropping as a result. The return of the quality to optimal levels after this is attributed to UAV 1 turning to return to its original
position and therefore closing the gap between the two UAVs. As discussed previously, at that point there was no suitable alternate configuration so the policy itself was unable to maintain optimal performance.

![Simple policy, quality of data by time](image)

Figure 13: Simple policy, quality of data by time

We can now address the actual result of the experiment, and ask whether the simple policy performs better than the null policy. The answer depends on the goals of the situation.
If timeliness is paramount and a NIIRS of three is acceptable then the null configuration performed better. If on the other hand the extra few seconds of latency do not hinder the operation significantly while the degraded level of identification accuracy would then the simple policy performed better. Such goals are only likely to be known during the planning stages of an operation and as such the personnel planning those missions need support to define the correct policies to be used. This concept is discussed later in Section 3.3.

The experiment did show that even a simple policy is able to produce a marked improvement to the effectiveness of a configuration employing actions that could be automated, specifically the routing of data that do not infringe on what could be considered tactical decisions (such as the actual course flown by a UAV).

3 Research Issues

The aim of the research issues section is to set out the technical challenges which remain in order to achieve a technically plausible approach to metadata-based resilient reconfiguration. At the outset of this work, we posited that such an approach would require three aspects:

1. a statement of the dependability properties which can be defined and stated explicitly
2. policies and mechanisms for reconfiguration in order to maintain a level of resilience
3. a monitoring and reasoning framework for maintenance and exploitation of metadata.

Milestone report D3 [4] addressed the first two of these aspects, by proposing some of the properties which can be defined and stated explicitly and investigating some potential policy languages. The demonstration and experiment described in Section 2 includes some of the features required for a monitoring and reasoning framework for metadata exploitation. However, there are some clear limitations in the experiment simulation described above.

Section 3.1 reviews the limitations of the demonstrator and highlights means for it to be improved. The following Sections (3.2 to 3.4) describe extensions to metadata types, the types of policies that we would benefit...
from in an SoS, and the provenance structure used in the simulation. We then consider what is required in order to mature the metadata approach.

3.1 Experimental Fidelity

3.1.1 Current Simulation

The simulation performed during this proof-of-concept work was based upon a low fidelity representation of the real world. This was due to the amount of time available to develop the simulation environment by personnel outside of DSTL/MOD with access to only unclassified information. A list of the main simplifications that exist in the simulation are as follows:

**Actor movement** the actor objects follow a fixed straight line path between waypoints and are able to pass through world objects

**Actor message propagation** actor messages trace a straight line path between sender and receiver and ignore issues such as multipath interference. This reduces the confidence of scenarios where multipath interference could be an issue, such as a built up urban situation.

**Actor message placement** messages are effectively sent instantaneously, meaning that there can be no interference between competing transmissions and there is no cost associated with message redundancy.

**Actor services: data** the actors share only data tokens not actual data, meaning that software failures of the SHARD [7] types coarse value failure and subtle value failure can not be included in the experiment.

**Actor services: time** the services in the simulation take a fixed number of cpu cycles to perform. Other than when other service instances are created on an actor there is no variation in processing time. This greatly reduces the potential to consider the SHARD issues of early and late service provision.

**Actor metadata dissemination** the simulation assumed that an unspecified body had perfect access to the metadata contained in each actor. There was no opportunity to consider the issues of imperfect information due to propagation time, message loss or message corruption.
Actor configuration dissemination the simulation assumed that instructions changing the configuration of the system propagated instantly and perfectly, again this does not allow for issues of lag, message loss or message corruption.

No human factors the image analysis service is thought of as being partly manual, however factors such as the fatigue of the operator and slips and lapses are not considered.

Manual policy evaluation the manual evaluation of policies, while not directly reducing the fidelity of the simulation did affect the number of simulation runs possible and in doing so reduced the variety of scenarios and failure sequences that could be tested.

Fixed failures a decision was made to employ the event file method for defining a fixed sequence of failures for a simulation run. This has both pros and cons. The benefit is that it can lead to certainty that the only variation between two simulation runs is the policy employed or the paths taken by the actors, with the small number of simulation runs. The downside to this approach is that a policy designer may unknowingly design a sequence of failures that favoured one policy over another, thereby biasing the results of an experiment.

Security issues security issues were not considered in the simulation.

The above list of issues should be seen as an overview of significant areas of potential improvement for the experimental aspects of this work.

3.1.2 Experimental Value

The purpose of the previous section was to highlight the relative immaturity of the platform developed and used at Newcastle University with respect to the needs of this work. While it is fair to say that the demonstration performed is not an unexpected result, the simplicity stems from the developmental state of the platform. The list presented in the previous section represents a set of features that should be both added to the Newcastle platform and be considered as relevant to inform future work on this subject within DSTL/MOD.

The value from both the Newcastle platform and any internal DSTL/MOD platforms will be obtained with the simulation of more complex scenarios
than the current state of the Newcastle platform allowed. A list of the complexities that would lead to less obvious and more valuable results are as follows:

**More complex SoS** The SoS in the earlier demonstration is trivial, consisting of only four actors and a single capability instance. This leads to the experimental results being unsurprising. The goal of these simulations, and the original intention for the Newcastle platform, should be to simulate more complex SoS, where the complexity stems from there being multiple logical capabilities sharing the limited resources that a battle space scenario may provide. For example the AGO scenario would need to include services providing the shared situational awareness that is key to NEC. These would disseminate the target tracks to the front line actors, while also sharing the current and future locations of those actors. With the addition of voice and other communications services we would arrive at a situation where the act of altering the configuration to solve an issue with one capability could negatively impact the performance of another.

**Metadata Transport** Metadata is likely to be generated by all actors within the SoS and, unless a self organising approach is adopted then it will need to be transported to one or more actors charged with controlling the configuration of the SoS in the self adaptive system\(^2\). Depending the protocol used there will be different latencies between the generation of metadata and its arrival at the decision making actor. The simulation of different protocols would allow their impact in terms of the accuracy of information held by the decision maker and the communications bandwidth consumed to be assessed.

**Policy Framework** In our third deliverable [4] we discussed the decomposition of the problem of metadata based configuration management into a number of sub-components. For example there are components charged with evaluating the SoS, deciding if any configuration changes are required and searching for alternative configurations. This is only one decomposition of the tasks required and is not demonstrated as

\(^2\)A self organising system [8] is one where actors are able to collaborate to make decisions about the local configuration of the SoS. A self adaptive system [9] will take a top down approach to managing the configuration where one or more actors are charged with governing the configuration of the whole system.
being effective. A simulation environment would allow this layout, and others, to be tested and evaluated.

**SoS Failures** The Newcastle platform as described can model complete/partial/transient failures of specific items of hardware associated with an actor. The scripted nature of their definitions means that they are necessarily limited by the imagination of the script designer. A valuable development of the simulation work, both at Newcastle and within DSTL/MOD, would be the inclusion of a stochastic model for the failures, such that failures are generated by the simulation itself. Such a model could be populated either with realistic component failure rates or, if the policy approach is to be stress tested, with exaggerated failure rates. This will allow both a more varied range of failure scenarios to be explored and also remove the possibility of a subconscious tailoring by the designer of a failure script to suit a particular policy. The coverage of failure types within the SoS should also be expanded. The Newcastle platform essentially only allows the SHARD failure type *omission* to be simulated, while SHARD liststells us of five other types that are significant, Table 2.

<table>
<thead>
<tr>
<th>Group</th>
<th>Failure Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service provision</td>
<td>Omission, Commission</td>
</tr>
<tr>
<td>Service timing</td>
<td>Early, Late</td>
</tr>
<tr>
<td>Service value</td>
<td>Coarse (detectably) incorrect, Subtle (undetectably) incorrect</td>
</tr>
</tbody>
</table>

Table 2: Table of SHARD failure classifications, replicated from Pumfrey [7]

The purpose of the experiments are to gain confidence and knowledge about the application of metadata/policy based configuration management in scope of NEC systems. Simulation is one form of testing which is itself only one form of verification. More formal techniques include model checking and formal proof, both of which can give a very high degree of confidence if a suitable model with the right level of abstraction is defined. The issue, as discussed by Payne [10], is that while it is possible to prove certain properties
of a policy language using such formal techniques, model checking the behaviour of a system using a policy in that language is not practical. To give meaningful results the model being checked would need to be a valid representation of the actors being governed by the policy, their failure behaviour and the environment in which they exist potentially including malicious attempts by hostiles to subvert the operation of the SoS. Simulation is therefore seen as the most practical means by which a reasonable confidence in the use of metadata and policies can be gained. The main drawback of the use of simulation and other non-formal testing techniques is that it is not possible to simulate all possible executions, and so it is necessary to define a strategy for choosing which simulations to run. This is discussed in the following section.

3.1.3 Experimental range

As discussed in the previous section, policy evaluation in the simulation environment is performed manually. This limited the number of experiments that could be performed and in doing so reduced the potential confidence in the results. After discussion with Ralph Mansson of DSTL, the following guidance from the R Statistical project\(^3\) was brought to our attention.

Computer experiments with quantitative factors require special types of experimental designs: it is often possible to include many different levels of the factors, and replication will usually not be beneficial. Also, the experimental region is often too large to assume that a linear or quadratic model adequately represents the phenomenon under investigation. Consequently, it is desirable to fill the experimental space with points as well as possible (space-filling designs) in such a way that each run provides additional information even if some factors turn out to be irrelevant.

While it is not possible to state at this point how many experiments will be possible as this work is developed, it is possible to provide guidance on the three main axes that define the experimental space.

**Mission** the mission defines the number and types of actors that will take part on the simulation, along with their initial configuration, paths and goals. It also includes the physical world in which the mission takes

\(^3\)http://www.r-project.org/
place, for example mountainous terrain versus flat plains versus built up urban environments.

**Policy** the policy describes what changes, if any, are made to the configuration of the SoS in response to different values of the metadata considered.

**Failures** the failures define which components in the system experience what type of problem and when.

Ideally the experiments performed would fill the experimental space as described above, however it may prove difficult to have confidence that this goal has been achieved as none of the axes are strictly linear or easily bounded. Taking the Mission as an example, there are a great many different operations that a force may undertake, for example peace keeping, search & rescue, ground support etc. These operations may require different numbers and types of units following different paths through the environment. Also the missions will take place in different physical environments, e.g urban, mountainous, coastal etc, all of which may pose different problems for the control and communications of units such as who is in control of units during a beach landing. The definition and justification of the ‘points’ on this axis is outside our area of expertise and would be better approached by domain experts within MOD/DSTL. The policy axis is similar in that there is no natural ordering or progression between policies.

The possible exception to this lack of ordering, where points on a numerical axis may be drawn, is the failures. It is conceivable that a progression from a zero probability of any component failing up to presumably somewhere short of a 100% chance of component failures could form an interesting basis for a series of experiments.

To summarise then, we see both the mission and policy axes as being enumerations, where sets of experiments may be grouped by similar characteristics, e.g. all search & rescue operations in a mountainous region using policy A while the failures axis can be considered partially numerical where the number of points off the line is chosen to suit the number of experiments that are possible.
3.1.4 Alternative Platform

A possible alternate platform for continuing the experimental part of this work is the Joint Intelligence Model (J2M)\(^4\) model which uses the SIMUL8 simulation software\(^5\). The J2M model is:

J2M is a high level stochastic simulation model of the [Direction, Collection, Processing and Dissemination] (DCPD) process, developed in SIMUL8. It is being developed incrementally to allow the model to evolve as studies extend their requirements. It enables the user to construct a range of [Intelligence, Surveillance, Target Acquisition and Reconnaissance] (ISTAR) architectures based on formally approved scenarios. It is a data driven model and therefore allows the user to evaluate the impact of alternative architectures on the delivery of ISTAR capability.

The J2M model appears to be well suited to the needs of simulating the logical SoS part of this work as that is essentially what it was designed to do. To perform the sort of experiments we require, a number of functionalities would be required, specifically these are:

**Metadata generation** the components within the simulation must have the ability to generate metadata either about their own performance or their experience of interacting with other components.

**Metadata access** to perform the experiments an external policy evaluation engine would require access to the metadata stored by each component

**Configuration adjustment** the policy engine would need to be able to make changes to the configuration of the system being simulated either while the simulation is paused or while it is running

Having discussed these requirements with Sarah Druon (SIMUL8) the answers received suggest that the programmatic facility afforded by the visual logic\(^6\) within SIMUL8 could be used to support the above requirements.


We are grateful to Paul Elrick of DSTL for this suggestion

\(^5\)http://www.simul8.com/

\(^6\)http://www.simul8.com/products/features/vl.htm
For example, while SIMUL8 does not directly support configuration change during a simulation, if a completely connected system model was constructed in SIMUL8 then the visual logic could make specific connections and components either available or unavailable at simulation time. This would support an ‘emulation’ of reconfiguration.

3.2 Metadata

The first deliverable [2] for this task contained a set of five properties by which metadata may be characterised, these were as follows.

**Quantitative / Qualitative:** quantitative metadata can be represented numerically. This applies to the vast majority of metadata found during the survey such as reliability, availability. Qualitative metadata are descriptive properties such as a list of the failure modes of a component;

**Functional / Non Functional:** functional metadata describe the service provided, for example, the semantics of service or its metrological properties and pre and post conditions on using a service. Non-functional metadata may describe Quality of Service (QoS) properties such as the expected response time or the cost of the service;

**Internal / External:** internal metadata describe the components within the system under control, these are metadata that we can affect. External metadata describe components and properties of the environment in which the system operates, such as the traffic on the wide area network (WAN) between two system components. These metadata can only be reacted to, not directly affected;

**Predictable / Unpredictable:** some metadata can be predicted, within some degree of confidence, before a change happens. For example the load experienced by bank web server may alter predictably. Other metadata may be less predictable such as the availability of a previously unknown service provider;

**Controllable / Uncontrollable:** closely related to the above but not exactly similar. Controllable metadata might include the power applied to the transmitter of a portable network device, while the presence of hostile counter measures against that transmission is uncontrollable.
These properties still stand, however a sixth property has been added to the list in the light of further consideration on the subject.

**Historical / Current / Future:** at any point in time there may be multiple versions of a particular metadatum. The historical values are those that have been recorded in the past. The current value is the latest value recorded though there may well be a time limit on how long a particular value may be considered current before it is given historical status even if there is no replacement value. The future values are the best estimates based on our current / historical knowledge of the system at hand.

The simulation included a number of metadata including the readability of messages received, image quality, positional accuracy and identification accuracy, all of which were described previously in Section 2.2.1 on page 10.

The above categories could guide the developer of a service or some hardware with respect to what metadata *could* be generated or required by it, but it does not provide guidance regarding what metadata *should* be generated or required. This task has been performed with the concepts of Network Enabled Capability (NEC) [11] in mind, paying close attention to two of the main principles, which are ‘agile mission groups’ and ‘shared situational awareness’. The first of these principles tells us that we do not know in what ways our current assets will be used or indeed what assets they will be used with, the second tells us, amongst other things, that units may not know beforehand what information they will need to perform their part of the mission. The common factor is that when units are designed the environment in which they will be used will only be partially known.

In terms of metadata this would advocate an approach where all metadata that can be shared about a component or the data it exchanges, should be made available. By doing so we allow the components that are interacted with to choose which metadata they utilise, rather than not having the information required to make a decision. The cost of doing so is, of course, the extra communications bandwidth and the additional processing resources required to produce, transmit and process this extra data. If we consider a front line situation in comparison to an actor’s home base, then we can imagine that the internal computational power of each actor in the battlespace is unchanged, however the bandwidth and reliability of communications between them may be considerably reduced. In such a situation we can imagine
that the attachment of all available metadata to each message could increase communication latencies and reduce the effectiveness of the deployed forces. Therefore a means for selecting which metadata to send would be required. While this aspect has not been significantly explored as part of this task, we suggest two potential means by which such a decision could be made.

The first means assumes that components are able to monitor the communications bandwidth available while potentially having little knowledge about the actual mission taking place. Here we propose two further classifications for each metadata available, as described below:

**Dynamic / Static:** a dynamic metadatum would be one whose value changes within the time bands [12] that are significant within the current mission. A static metadatum is one whose value would not change within such a period.

**Platform / Data:** platform metadata would relate to the process performed on the data. Examples are the name of the person who performed the process, and the type of instruments used to measure/process the information. Data metadata describes the information itself and how it may be interpreted.

Examples of the type of metadata these two classifications may include are shown below in Table 3.

<table>
<thead>
<tr>
<th></th>
<th>Dynamic</th>
<th>Static</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Platform</strong></td>
<td>Standard ‘ilities’, such as reliability and availability.</td>
<td>Sensor types, Services and capabilities offered, configuration constraints</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>Quality and provenance of each data item.</td>
<td>Semantics of the data</td>
</tr>
</tbody>
</table>

Table 3: The four types of metadata considering dynamism and target of the data only.

The frequency with which an item of metadata is included in communications between actors in the battle space could be adjusted according to which class of metadata it belongs to. If these classifications were the only means by which an actor could determine what metadata to share with the SoS then it would need to contain a policy defining a priority order for the metadata.
classifications and the metadata within. When bandwidth is restricted this policy would determine what metadata to exchange and when.

The second method proposed is based upon an extension of the current MODAF OV3 view [3]. This view defines the services and needlines required to provide a capability. Our suggestion is that this view could also be annotated with the metadata that would be required to assess the quality of data and general performance of an instance of that capability. This means the same architectural knowledge used to define the capability could also define the data needed to assess it.

While the almost modular nature of the second option is attractive, if followed strictly it would not be conducive to the agility suggested by NEC. The reason for this is that as new actors enter the battle space the unit in control of the SoS configuration needs to be aware of the services that actor can provide to the SoS. The OV3 approach would likely not include this data as it is unlikely it would be of direct importance to a specific capability instance.

The recommendation for the first steps in this area is as follows:

**Assess current capabilities** currently understood logical capabilities should be assessed according to the characteristics presented early in this section. This will generate metadata that is significant to these capabilities

**Generate ontology** when a number of capabilities have been assessed the resulting metadata can be brought together to form the basis of a metadata ontology, facilitating a common understanding of the semantics associated with each metadatum name

**OV3 extension** the OV3 view of the assessed capabilities should be extended to include the metadata that would be expected to be exchanged along with the working data across the needlines

**Protocol** a protocol allowing the communication of metadata not directly included in the OV3 views would then be required to allow the SoS to know the remaining platform and static metadata required to take advantage of newly appeared actors in a scenario

### 3.3 Policies

We have previously described a number of policies and policy languages in our third deliverable [4]. To briefly recap, the two policy languages used were
Teleo-Reactive programming (T-R) and Policy Description Language (PDL). The languages differ in their structure and approach.

T-R uses rules employing a condition:action pairing, where the action is performed whenever the condition is found to be true. The important part of the T-R approach is the order of the rules. Whenever a T-R program is evaluated the algorithm starts at the top of the list and works downwards until it finds a condition that is true, it then proceeds to execute the associated action, which could be another T-R program. The paradigm is that the goal condition for a system should be represented by the very first rule in a T-R program and then that rule should be associated with a null action and therefore make no changes to the system. Each rule after that should describe situations that are increasingly far away from the goal condition, and their associated actions should be targeted at moving the system back towards the goal.

PDL rules adopt a event:condition:action structure, as used in the experiment. In this approach the policy engine requires some knowledge of the events that have occurred in the system, such as a message being exchanged as in the experiment. When the triggering events for one or more rules are found, the conditions for those rules are tested and any rules where for which their condition is true have their actions performed. In this language then the order in which rules are described is not significant. A complicating factor is that the sequence in which rules are evaluated and their actions performed may not be deterministic from the policy description alone and may be a function of the policy engine implementation.

The policies presented in D3 were described by one reviewer as having a programmatic feel to them. This was accepted as fair comment as they contain many fine grained rules considering multiple metadata simultaneously. Further, while the policies can be read relatively intuitively by anyone with a programming background, they are far from trivial to construct. The main difficulty results from the requirement for the policy to cover the entire state space of the metadata used, such that the policy can give an indication of the action required, even if that action is ‘no action required’. Following this feedback, the main recommendation for development of the policies is research into usable policies.

We suggest that the types of policies described in deliverable D3 are only applicable for use by domain experts and during the development of new capabilities and platforms; they are far too detailed to be practically and safely constructed in an operational context. A usable policy would abstract
away the underlying details of the policies produced by a domain expert, replacing them with a statement of preference using natural language terms. These terms would be taken from a list that describe different performance levels for a capability using language that would be significant to operational staff. For example terms such as ‘optimal’, ‘minimal’ and ‘degraded’ would be related to relevant performance benchmarks accepted by the military. A mission policy could then reference, for different time periods within the mission, the required levels of performance for each capability using these terms. The act of including a reference to a capability and a performance level term would then automatically include the ‘lower level’ policy developed by the domain experts.

Policy creation is not a new concept. A number of outputs of the United States and United Kingdom International Technology Alliance (USUKITA) have been directed towards this area. They include approaches using tools based on templates to construct rules [13] and tool support allowing selected natural language terms to be translated into machine processable rules [14]. While the main focus of these outputs has been the field of security and access control instead of configuration management as in this work, they share the same goal of making the policies more manageable. The most significant difference between our suggestion above and these works is that the majority of these outputs are still considering defining policies at the level of individual rules. A significantly different approach is suggested by Lim et al. [15]. Their approach takes the standpoint that people are better at dealing with specific examples than generating general rules. The work involves the use of an algorithm to attempt to infer policy rules from example decisions. Similar to our proposal above, this line of work attempts to allow the users of a policy system to define a policy via a means that they may be more familiar with, rather than simply using fine grained predicate based rules.

The second research challenge relating to policies arises at the point that policies are imported: this is an issue of compositionality. To have confidence in the behaviour of the SoS would require an understanding of the emergent behaviour of the combined policy rules.

### 3.4 Provenance

Provenance is a class of metadata that relates to the source of an object. In the case of a SoS this is taken to mean the source and potential route taken by the information used within that system. In this work we have identified
a simple structure that has been able to represent all of the limited set of scenarios it has been presented with. The structure consists of three basic elements:

**Provenance Link** the simplest of the three elements, it is used to describe a transfer of data from one node in a system to another. This element should include descriptions of which units sent and received the data, when the data was transferred and the means by which it was sent.

**Data Transform** used to represent some process that takes some data as a source, processes it and outputs one or more different data items. The data output may also include the original data. This element should include a description of the process that took place to transform the data, the identifiers of operators involved and timing.

**Provenance Join** used when data from two or more sources are brought together in some way. This may include corroborating sources such as in a voting system or the generation of new data from temporally separated data items, amongst others. A join results in new information in some way and so, as with the data transform above, must include a description of the process that took place.

The provenance of an item of data can be described by chaining together these elements to form a tree representing the sources, transformations and routes taken by the data from which it is constructed. The target tracking capability as discussed in the experiment (Section 2) can be used as an example here. The form of the capability used in the experiment stops at the point where the theatre command receives target location data, but the originally intended form went one step further and constructed target track data items by combining individual target location data items. The provenance for data generated by such a capability can be represented as follows:

**Provenance Link** The visual sensor shares the image with the visual sensor service

**Provenance Link** The visual sensor shared the image data with the image analysis service

**Data Transform** The image analysis takes the image data and outputs target location data
Provenance Link  The image analysis service shared the location data with
the location collator service

Provenance Join  The location collator brings together any past target lo-
cation messages relating to a particular target with the one received to
form a target track

In this way the data upon which a target track is formed can be traced
right back to the sensors taking the image, Figures 14 & 15.

Figure 14: SV10c view of the data exchanges and the provenance elements
attached to them

Provenance is significant as it relates directly to the trust that may be
associated with a particular data item. The two main cases that can be used
to illustrate this are data incest and data repudiation.

3.4.1 Data Incest

Data incest is the term used to describe a situation where corroborating data
appears to come from two or more sources when there is in fact a common root
Figure 15: Provenance tree for a single target track based upon two separate images

Figure 16: SV10c view of the data exchanges leading to a data incest situation for both sources, Figure 16. In the earlier experiment this situation would have occurred had there been a single UAV supplying identical image data
to two or more analysis stations. If those analysis stations then fed into the same command station then a false level of confidence may be applied to the target locations supplied if the provenance of those tracks is not considered. Data incest can be detected by comparison of the provenance tree and looking for identical roots for any of the branches, Figure 17.

While data incest can lead to common cause failures it does not always preclude an increase in confidence in the combined data. If the scenario depicted in Figure 16 included three separate analysis stations instead of two and those analysis stations used diverse means for analysing images then we have an N-Version voting programming situation [16]. The joining of these three data analysis paths could then include an adjudicating element where the value is selected based upon a voting mechanism. In this situation there can be an increased confidence in the results of processing the single source image when compared to a single image with a single processing path.

### 3.4.2 Data Repudiation

Data repudiation describes a situation where some property of a source or path of data is found to be untrue. An example of this based around the experiment would be the emergence of information that an analysis station, used as part of the target tracking, had been compromised by hostile forces and was falsifying results. In such a situation any data that is dependent on
the output of that analysis station needs to be reassessed.

Figure 18: SV10c view of the data exchanges before and after confidence in the analysis station has been questioned

Provenance data serves two purposes in this situation. The first use of the provenance data is to identify any data that may be affected, in this case that would be any data that had been processed by the compromised analysis station since it had been infiltrated. The second use of the provenance structure is to record that the associated data has been reassessed. The reassessment record should consist of two parts, firstly the nature of the reassessment, i.e. in what way(s) did the data change and secondly why was the change performed. To solve both of these a provenance join element may be employed. This element allows the process used to manipulate the data to be recorded, and the provenance of the information indicating that the analysis station had been compromised to be joined to the data’s own provenance tree. The benefit of using the provenance join in this way is that the reasoning behind the updated data is understood and there is the record of when the data was changed. The latter part allows any decisions made
using the data before it was reassessed to be justified.

3.5 Further development

In order to consider what is required in order to mature the metadata approach, the steps involved in the reconfiguration process are reconsidered here.

- Monitoring. The system is monitored by gathering metadata relevant to its goals, either actively or passively.

- Detection. Gathered metadata is compared with the goal policy to determine whether or not it is acceptable, and the goal of any necessary reconfiguration are determined.

- Generation. A candidate configuration is obtained which meets the reconfiguration goals.
• Selection. A choice of candidate configurations is made

• Execution. The change is implemented (and monitoring continues)

Section 3.3 has introduced the concept of usable policies to describe an improvement to the policy language which is less programmatic and could be used by mission planners to define policies appropriate to the mission priorities, in a higher-level language. Research is necessary here to identify the relevant terms (optimal, minimal, degraded etc) and the appropriate value ranges.

Further research should also be conducted into the definition of capability functions which would be used to assess a given configuration and aid the selection: the simulation results have been manually analysed to show that a given reconfiguration is an improvement over a static configuration. A further step in defining capability functions is to extract and generalise the evaluations that characterises the improvement.

4 Conclusions

This report presented the results of a simple experiment performed on a simulation platform developed during the task at Newcastle University. It is fair to say that the results of the experiment are intuitively predictable in that the system allowed to change its configuration in response to failing communications performs better than one that has a static configuration. The simplicity of the experiment performed stems from the simulator still being at an early stage of development when the experiment was performed.

The value from the simulation work performed is in the form of the insights into the properties a more valuable simulation would possess. These properties relate to the complexity of SoS that can be simulated, the range of failures supported and the ability to emulate the behaviour of the framework employed to manage the SoS configuration. This latter point is given weight as we can imagine that if perfect information regarding the state of each actor in the simulation were available then policies would be able to determine the best configuration at each point in time. If we consider situations where metadata about the actors is lost, delayed or corrupted then the ability of a policy engine to make the best decisions could be questioned.

The experiment and discussions around it have resulted in guidance regarding the experimental space that should be explored, including the range
of missions performed, reconfiguration policies and failures experienced by the SoS and its constituent parts.

The report also presented a number of research issues that should be addressed if this work is to reach a higher TRL. The first of these focused on the metadata that would be used to govern configuration. Here we provide six properties that can be used to guide the identification of metadata relating to an SoS or component within it. This was followed by a discussion of the question of what metadata a component should provide and when.

The second issue concerns the policies presented in this deliverable and our earlier technical report, CS-TR-1248. These policies are very programmatic in nature and so are considered to be impractical for use in anything like front line conditions. They are better suited for use during the development and maintenance stages for a component as the engineering personnel involved may have both the time and the skills necessary to develop such rule sets. The issue of developing policies that are more applicable to battlespace use was then touched upon; we propose using policies that are described using agreed natural language terms more easily related to the operation. Related work from the security sector includes work in which an operator is presented with scenarios and a policy is inferred from the decisions made in those scenarios. Certainly some approach other than the construction of fine grained rules is required if policies are to be generated in response to particular mission needs.

The final issue concerns the structure used to represent provenance in the simulation. The structure consists of three basic elements representing transfers of data, manipulation of data and the joining of multiple data sources. This structure is sufficient for the needs of the simple demonstration performed and was also shown to be of use for detecting the issues of data incest and data repudiation.

We believe that, given the drive for more dynamic systems as implied by NEC, SoS controllers will need support in managing those systems. Whether policies are employed to autonomously change the configuration or simply to suggest changes to the controllers, they will need access to the metadata so that SoS performance can be assessed. We therefore suggest that there is merit in continuing this line of research using a range of simulation experiments as the tool to allow the emergent behaviours of systems to be explored.
5 Acknowledgements

The work described here was funded under the UK Software Systems Engineering Initiative (SSEI) within Task 7 on Dependability Explicit Metadata. The authors are grateful to colleagues in SSEI, DSTL, BAESystems, MBDA, SIMUL8 and the Centre for Software Reliability at Newcastle for their helpful input: John McDermid, Neil Bowman, Jane Fenn, Tim Kelly, Colin McDonald, Richard Payne, John Fitzgerald, Paul Glover, Sarah Druon, Oliver Macfarlane and Paul Elrick.

6 List of References

References


7 List of Abbreviations

**DPCD:** Direction, Collection, Processing and Dissemination

**DSTL:** Defence Science and Technology Laboratory

**FPS:** Frames Per Second

**GPS:** Global Positioning System

**ISTAR:** Intelligence, Surveillance, Target Acquisition and Reconnaissance

**J2M:** Joint Intelligence Model

**MoDAF:** Ministry of Defence Architecture Framework

**NEC:** Network Enabled Capability

**NIIRS:** National Imagery Interpretation Rating Scale[6]

**QoS:** Quality of Service

**PDL:** Policy Description Language (Actual language name, not a reference to the class of languages)

**SHARD:** Software Hazard Analysis and Resolution in Design[7]

**SIG:** Special Interest Group

**SoS:** Systems of Systems

**TRL:** Technology Readiness Level

**T-R:** Teleo-Reactive

**UAV:** Unmanned Aerial Vehicle

**USUKITA:** United States and United Kingdom International Technology Alliance

**UML:** Unified Modelling Language

**WAN:** Wide Area Network
8 Annexes

8.1 Results By Experiment

Tables 4 and 5 below contain the complete timing and quality data from the two experiments performed.
<table>
<thead>
<tr>
<th>Time event generated</th>
<th>Event Tag Number</th>
<th>Time at UAV1</th>
<th>Time at UAV2</th>
<th>Time at Analysis</th>
<th>Time Theatre Command</th>
<th>Units identified at NIIRS level</th>
<th>Time to Theatre Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 0 5</td>
<td>15 30 6 25</td>
<td>205 40 205 210 210 212 227 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1 10</td>
<td>15 30 6 20</td>
<td>210 41 210 210 212 227 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 2 15</td>
<td>15 30 6 15</td>
<td>215 42 215 225 227 242 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20 3 20</td>
<td>30 45 6 25</td>
<td>220 43 220 225 227 242 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25 4 25</td>
<td>30 45 6 20</td>
<td>225 44 225 227 242 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30 5 30</td>
<td>30 45 6 15</td>
<td>230 45 230 240 212 257 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35 6 35</td>
<td>45 60 6 25</td>
<td>235 46 235 240 242 257 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 7 40</td>
<td>45 60 6 20</td>
<td>240 47 240 240 242 257 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45 8 45</td>
<td>45 60 6 15</td>
<td>245 48 245 255 257 272 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 9 50</td>
<td>60 75 6 25</td>
<td>250 49 250 255 257 272 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55 10 55</td>
<td>60 75 6 20</td>
<td>255 50 255 255 257 272 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60 11 60</td>
<td>60 75 6 15</td>
<td>260 51 260 270 272 287 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>65 12 65</td>
<td>75 90 5 25</td>
<td>265 52 265 270 272 287 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70 13 70</td>
<td>75 90 5 20</td>
<td>270 53 270 270 272 287 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>75 14 75</td>
<td>75 90 5 15</td>
<td>275 54 275 285 287 302 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80 15 80</td>
<td>90 90 107 6 27</td>
<td>280 55 280 285 287 302 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>85 16 85</td>
<td>90 92 107 6 22</td>
<td>285 56 285 285 287 302 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90 17 90</td>
<td>90 92 107 6 17</td>
<td>290 57 290 300 302 317 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95 18 95</td>
<td>105 107 122 6 27</td>
<td>295 58 295 300 302 317 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 19 100</td>
<td>105 107 122 6 22</td>
<td>300 59 300 300 302 317 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>105 20 105</td>
<td>105 107 122 6 17</td>
<td>305 60 305 315 317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>110 21 110</td>
<td>120 122 137 6 27</td>
<td>310 61 310 315 317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>115 22 115</td>
<td>120 122 137 6 22</td>
<td>315 62 315 315 317</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120 23 120</td>
<td>120 122 137 6 17</td>
<td>320 63 320</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>125 24 125</td>
<td>135 137 152 6 27</td>
<td>325 64 325</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>130 25 130</td>
<td>135 137 152 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>135 26 135</td>
<td>135 137 152 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140 27 140</td>
<td>150 152 167 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>145 28 145</td>
<td>150 152 167 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150 29 150</td>
<td>150 152 167 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>155 30 155</td>
<td>165 167 182 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160 31 160</td>
<td>165 167 182 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>165 32 165</td>
<td>165 167 182 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>170 33 170</td>
<td>180 182 197 5 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>175 34 175</td>
<td>180 182 197 5 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>180 35 180</td>
<td>180 182 197 5 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>185 36 185</td>
<td>185 197 212 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>190 37 190</td>
<td>195 197 212 6 22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>195 38 195</td>
<td>195 197 212 6 17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 39 200</td>
<td>210 212 227 6 27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Timing and Quality results from the null policy experiment.
<table>
<thead>
<tr>
<th>Time event generated</th>
<th>Event Tag Number</th>
<th>Time at UAV1</th>
<th>Time at Analysis</th>
<th>Time Theatre Command</th>
<th>Units identified at NIIRS level</th>
<th>Time to Theatre Command</th>
<th>Time event generated</th>
<th>Event Tag Number</th>
<th>Time at UAV1</th>
<th>Time at Analysis</th>
<th>Time Theatre Command</th>
<th>Units identified at NIIRS level</th>
<th>Time to Theatre Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 0</td>
<td>5 15</td>
<td>30 6 25</td>
<td>205 40</td>
<td>210 225</td>
<td>4 20</td>
<td></td>
<td></td>
<td>10 15</td>
<td>30 6 20</td>
<td>210 210</td>
<td>225 4 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 2</td>
<td>15 30</td>
<td>6 15</td>
<td>215 42</td>
<td>215 240</td>
<td>4 25</td>
<td></td>
<td></td>
<td>20 30</td>
<td>45 6 20</td>
<td>220 225</td>
<td>240 4 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25 4</td>
<td>25 30</td>
<td>45 6 20</td>
<td>225 44</td>
<td>225 240</td>
<td>4 15</td>
<td></td>
<td></td>
<td>30 50</td>
<td>45 6 20</td>
<td>230 240</td>
<td>255 5 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35 6</td>
<td>35 60</td>
<td>6 25</td>
<td>235 46</td>
<td>235 240</td>
<td>5 20</td>
<td></td>
<td></td>
<td>40 70</td>
<td>60 6 20</td>
<td>240 240</td>
<td>255 5 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45 8</td>
<td>45 60</td>
<td>6 15</td>
<td>245 48</td>
<td>245 255</td>
<td>5 25</td>
<td></td>
<td></td>
<td>50 90</td>
<td>75 6 20</td>
<td>250 250</td>
<td>270 5 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55 10</td>
<td>55 75</td>
<td>6 20</td>
<td>255 50</td>
<td>255 270</td>
<td>5 15</td>
<td></td>
<td></td>
<td>60 110</td>
<td>75 6 15</td>
<td>260 260</td>
<td>285 5 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65 12</td>
<td>65 90</td>
<td>5 25</td>
<td>265 52</td>
<td>265 285</td>
<td>5 20</td>
<td></td>
<td></td>
<td>70 130</td>
<td>90 5 20</td>
<td>270 270</td>
<td>295 5 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75 14</td>
<td>75 90</td>
<td>5 15</td>
<td>275 54</td>
<td>275 295</td>
<td>5 25</td>
<td></td>
<td></td>
<td>80 150</td>
<td>105 5 25</td>
<td>280 280</td>
<td>300 5 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85 16</td>
<td>85 90</td>
<td>105 5 20</td>
<td>285 56</td>
<td>285 300</td>
<td>5 20</td>
<td></td>
<td></td>
<td>90 170</td>
<td>90 5 15</td>
<td>290 290</td>
<td>315 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95 18</td>
<td>95 105</td>
<td>120 4 25</td>
<td>295 58</td>
<td>295 315</td>
<td>6 20</td>
<td></td>
<td></td>
<td>100 200</td>
<td>105 4 20</td>
<td>300 300</td>
<td>315 6 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105 20</td>
<td>105 120</td>
<td>4 15</td>
<td>305 60</td>
<td>305 315</td>
<td>6 15</td>
<td></td>
<td></td>
<td>110 210</td>
<td>120 5 25</td>
<td>310 310</td>
<td>315 6 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110 21</td>
<td>110 135</td>
<td>5 25</td>
<td>315 62</td>
<td>315 315</td>
<td>6 15</td>
<td></td>
<td></td>
<td>120 220</td>
<td>120 5 20</td>
<td>320 320</td>
<td>325 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120 22</td>
<td>120 135</td>
<td>5 15</td>
<td>325 64</td>
<td>325 325</td>
<td>6 25</td>
<td></td>
<td></td>
<td>125 225</td>
<td>135 4 25</td>
<td>330 330</td>
<td>335 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>140 27</td>
<td>140 150</td>
<td>165 4 25</td>
<td>350 66</td>
<td>350 360</td>
<td>6 25</td>
<td></td>
<td></td>
<td>145 280</td>
<td>150 4 20</td>
<td>360 360</td>
<td>375 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150 29</td>
<td>150 150</td>
<td>165 4 15</td>
<td>360 67</td>
<td>360 375</td>
<td>6 25</td>
<td></td>
<td></td>
<td>155 300</td>
<td>155 3 25</td>
<td>370 370</td>
<td>385 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160 31</td>
<td>160 165</td>
<td>180 3 20</td>
<td>380 68</td>
<td>380 390</td>
<td>6 25</td>
<td></td>
<td></td>
<td>165 320</td>
<td>165 3 15</td>
<td>390 390</td>
<td>405 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>170 33</td>
<td>170 180</td>
<td>195 4 25</td>
<td>400 69</td>
<td>400 410</td>
<td>6 25</td>
<td></td>
<td></td>
<td>175 340</td>
<td>180 4 20</td>
<td>410 410</td>
<td>425 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>180 34</td>
<td>180 195</td>
<td>195 4 20</td>
<td>420 70</td>
<td>420 430</td>
<td>6 25</td>
<td></td>
<td></td>
<td>185 350</td>
<td>195 4 15</td>
<td>430 430</td>
<td>445 6 25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>190 35</td>
<td>190 210</td>
<td>210 4 25</td>
<td>440 71</td>
<td>440 450</td>
<td>6 25</td>
<td></td>
<td></td>
<td>195 360</td>
<td>195 4 20</td>
<td>450 450</td>
<td>465 6 25</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Timing and Quality results from the simple policy experiment.