Distributed Service Integration for Disaster Monitoring Sensor Systems

Lu Liu[1], Nick Antonopoulos[1], Jie Xu[2], David Webster[2], Kaigui Wu[3]
[1]School of Computing and Mathematics, University of Derby, Derby, DE22 1GB, U.K.
[2]School of Computing, University of Leeds, Leeds, LS2 9JT, U.K.
[3]College of Computer Science, Chongqing University, Chongqing, China
l.liu@mdx.ac.uk

Abstract

Sensor networks have the potential to revolutionize the capture, processing and communication of critical data for use of disaster rescue and relief. In order to provide a dependable rescue capability through dynamically integrating newly developed and legacy sensor systems with other systems and computing, new methodologies are required for the dependable integration of services in heterogeneous environments. In this paper, we present a new architectural model which can proactively self-adapt to changes and evolution occurring in the provision of search and rescue capabilities in a dynamic environment. This performance and reliability of the approach has been evaluated using simulations in a dynamic environment and demonstrated through developing and testing a demonstration system for a scenario of disaster area monitoring.

Keywords – *service oriented architecture, sensor networks, disaster relief*

1. Introduction

A number of large earthquakes happened in 2010 (e.g. 8.8-magnitude Chile earthquake, 7.0-magnitude Haidi Earthquake and 6.9-magnitude China Yushu Earthquake) which significantly impacted the lives of thousands of people in different countries. Real-time monitoring data and information is crucial for the success of disaster relief efforts from international communities. In order to provide a dependable rescue capability through dynamically integrating systems and computing, new methodologies are required for the dependable integration of services in heterogeneous environments. A rescue capability is the operation of integrated services to fulfil a rescue mission objective.

Sensor networks have the potential to revolutionize the capture, processing and communication of critical data for use of disaster rescue and relief [1]. The functions of sensors need to be integrated to provide a joint capability to meet different search and rescue requirements. Different types of services across different platforms [2] need to be integrated to provide a joint rescue capability. For example, sensors for detecting survivors could employ any of laser, visible spectra, heat (infrared), radar or sonar sensors and these heterogeneous sensor services can be deployed on recue helicopters, unmanned vehicles and search and rescue personnel. The sensors should provide interoperable service interfaces to the integrating applications taking into account quality of services (QoS) parameters to deliver dependable capabilities in highly critical situations.

For the provision of a dependable rescue capability in a dynamic and unpredictable disaster area, the networked sensors should have the ability to autonomously support and co-operate with each other to quickly configure any services available on the disaster area to deliver a real-time capability, self-adaptability to modify their behaviours to deliver a sustainable capability according to environmental changes, and ability to share information, generate access and protect information throughout the network.

In this paper, we present a new architectural model which can proactively self-adapt to changes and evolution occurring in the provision of search and rescue capabilities in a dynamic environment. The rest of the paper is organised as follows. Section 2 discusses related work on service-oriented architecture (SOA) and service composition. The proposed architectural model of distributed service integration for disaster monitoring sensor systems is discussed in Section 3. The reliability of provision of rescue capability is evaluated and discussed in Section 4. The demonstration system for disaster area monitoring to demonstrate the use of the architectural model for sensor systems is introduced in Section 5. In Section 6, conclusions are drawn and future work is described.

2. Related Work

Service-oriented architecture is concerned with the structure of service provision and consumption and the infrastructure to support the interactions. The architecture is made of service suppliers and consumers, with suppliers advertising through registries or brokers for consumers to discover. Loose coupling is one of the key architectural principles of SOA. This enables services to maintain a relationship that minimises dependencies and only requires maintaining an awareness of each other. The loose coupling of SOA enables service implementations to be inter-changed and modified. Loose coupling [3] is the main benefit resulting from operating in this architecture. Loose coupling is obtained by abstracting the description of service provision from the implementation of service provision, thereby allowing different implementations to offer interchangeable services. This can then enable dynamic binding, where service implementations can be selected before service consumption by composing applications from suitable services based on their service descriptions.

Web services are a de facto implementation of SOA. In the last decade, a number of Web Service composition frameworks and applications have been developed. Alonso et.al [5] described six different dimensions of web service composition models which can make different assumptions of the types of components considered. The disadvantages of composition models makes composition work more involved because of the heterogeneity of the components. The Web Services Composite Application Framework (WS-CAF) [6] is an open framework developed by the OASIS group. The purpose of the OASIS WS-CAF is to define a generic and open framework for applications that contain multiple services used in combination.

3. Distributed Service Integration

Service-oriented architecture is designed for changes. The loose coupling of SOA enables service implementations to be inter-changed and modified where integration interfaces are developed with minimal assumptions between the sending/receiving parties, thus reducing the risk that a change in one service will force a change in another service. However, service integration is dependent on service interface definitions and requires management of workflow definitions to minimise impact on provision of rescue capability. The changes of requirements and workflow definitions could still affect the dependability and sustainability of provision for a rescue capability through service integration.

In order to ensure the reliability of provision of rescue capabilities, a systematic approach needs to be developed which would lead to flexible architectures for through-life evolution. Figure 1 illustrates service-oriented architecture for the delivery of rescue capabilities. In this architecture, each peer node provides a number of services, each service performs a set of functions, and these can be integrated to form a higher level of functionality to deliver a rescue capability. Dynamic binding allows common functions to be identified in different implementations. The architecture enables functions from different services across sensors to be integrated to provide rescue capabilities in a loosely coupled manner. For example, a remote sensing sensor on a rescue helicopter provides services, such as surveillance, target recognition and target tracking services. The surveillance service includes functions for metrological surveillance and situation surveillance. The metrological surveillance function may be combined with other functions in a workflow to form a higher level weather service that contributes to search-and-rescue missions.

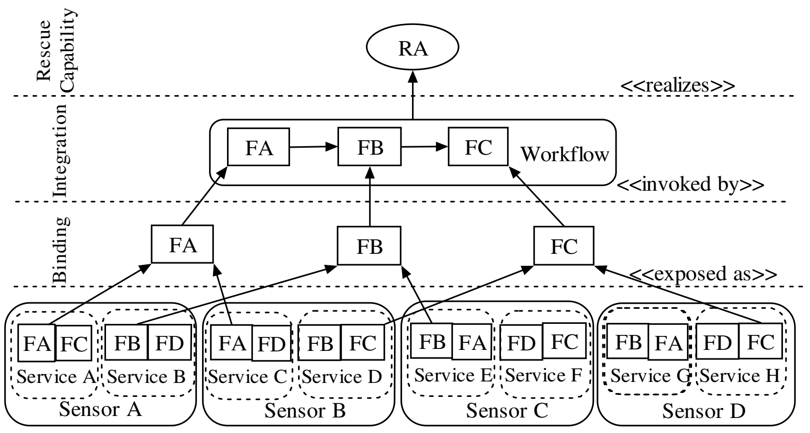


Figure 1. Redundant service binding

3.1. Dynamic Service Discovery

For provision of rescue capabilities, a set of services need to be integrated in a workflow to fulfil a mission objective. However, the benefits of integration of services cannot be ground unless we have an efficient way to discover all required services. In the traditional implementations of SOA, a centralised UDDI (Universal Description Discovery and Integration) registry provides the functionalities to advertise and discover services, which centralises all responsibilities for publishing and handling enquiries about services. In disaster management systems, the centralised registry can be considered a single point of failure. Once the service registry failed, all the information about registered services is unavailable. The problem of single-point-of-failure could be more significant in this case, since the service registry is also threatened by disaster attacks. Moreover, data links for access to information on remote repository cannot always be guaranteed in the unknown and dynamic disaster area. Additionally, the frequent remote communication could generate too much traffic to the network and the connection to remote registry could be overloaded if too many requests are received at the same time.

In order to dependably integrate services for disaster monitoring and relief, the worst conditions must be considered as the failure of the centralised registry or the failure of the connection to the remote centralised registry. To reduce the problem of single-point-of-failure, each network node should have the capability to efficiently discover desirable resources by interacting with connected nodes. For example, a search and rescue vehicle which carries a disaster rescue system drives in a disaster area. In the places where the communicating infrastructures have been broken and sensors have to communicate through low bandwidth ad hoc networks, the disaster rescue system located on the vehicle can choose the services which can only provide data with a low update frequency.

An adaptive and efficient peer-to-peer search (AEPS) algorithm [7] has been utilized for distributed service discovery for dependable service integration based on a number of social behaviour patterns. Using AEPS, efficient resource discovery could be performed by interacting with locally reachable nodes based on whatever local information is available. Each node can automatically detect potential interests of other nodes in the network according to their previous behaviours and preferentially links to the nodes that have similar interests. Peer communities are formed and maintained spontaneously, where nodes are self-organised based on their daily intercommunications.

3.2. Workflows for Disaster Relief

In this section, a disaster area monitoring scenario is used as an exemplar to demonstrate the evolution of rescue capabilities and to analyse the potential impact of evolution.

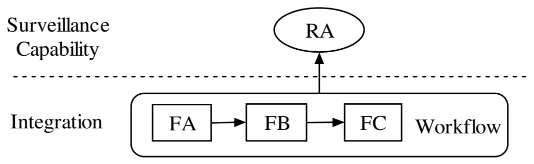


Figure 2. Workflow of service integration for delivery of rescue capabilities

In a network of sensors, a number of radar sensors supply data through services. The network of radar sensors is modelled conveniently as a dynamic network of services, facilitating ongoing changes. In the modelled system, a surveillance user can submit real-time requests to the system for information of Points of Interest (POIs) in a specified region. A sequence of services (such as “Get map information” (*FA*) and “Get sensor reading” (*FB*), “Display targets on map” (*FC*) can be operated in a workflow in order to provide a regional surveillance capability. The service integration can be abstracted by using workflow patterns as shown in Figure 2, where function *FA* represents the service of getting map information, *FB* represents the service of getting radar sensor reading and *FC* represents the service of displaying targets on map.

Redundant service binding is a technique to improve the reliability of the provision of rescue capabilities [2]. In order to deliver a reliable rescue capability, the required functions need to be provided by multiple services allocated to different peer nodes. The reconfiguration algorithm can switch to one of backup services in case of failure of initial service. The distributed recovery block (DRB) scheme [8] is applied to minimise the recovery time of integration. Figure 1 shows an example of redundant service binding. As shown in Figure 1, when a failure occurs in service *A*, the required function provided by the backup service *C* can still work for the provision of rescue capabilities.

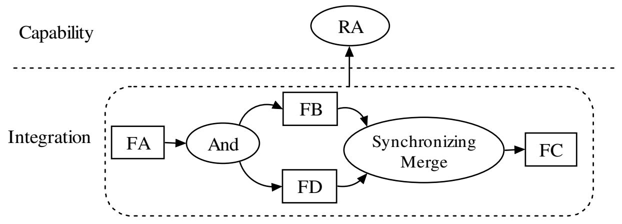


Figure 3. A possible evolution

Apart from failures of services, rescue capability evolution could cause potential problems affecting the reliability of provision of rescue capabilities. Operational requirements often change during a rescue and search operation, in accordance with changes of situation. The rescue capability could be evolved according to changes of environment and users' needs. A possible evolution is illustrated as an example: it is decided that some POIs are more important and a more aggressive monitoring service needs to be established. A number of rescue helicopters with remote sensing sensors are launched to provide a mixed monitoring service in conjunction with the deployed radar sensors. The rescue capability is evolved with the additional requirement to a new version of capability defined with a parallel split workflow pattern and synchronizing merge workflow pattern [9] illustrated in Figure 3. The parallel split pattern represents a point in the workflow process where a single thread of control splits into multiple threads of control which can be executed in parallel, while synchronizing merge pattern represents a point in the workflow process where multiple paths converge into one single thread; Synchronization needs to take place if more than one path is taken [9]. The new capability could be delivered only in case that both the services: *FB* and *FD* are implemented successfully.

3.3. Self-adaptability

The reliability of provision of service integration could be affected by the evolution of workflows in the case mentioned above. The proposed reconfiguration algorithm is able to proactively self-diagnose the evolution, evaluate the impact of evolution and self-configure services to adapt to capability evolution. By configuring two services which are independent with each other for performing a required function as shown in Figure 1, the required function is delivered if either service is implemented successfully. In order to investigate the impact of the evolution of capabilities, p_{FA}^{1} and p_{FA}^{2} are defined as the probability of failing to connect these two respective services for a required function *FA*. The probability of failure of the delivery of a required function for service integration is $p_{FA}^{1} \cdot p_{FA}^{2}$. In the original case without evolution, three functions are integrated in a workflow to deliver a rescue capability. Since all three functions, *FA*, *FB*, and *FC*, are necessary for the provision of a rescue capability, the probability of successful service integration is $\left(1-p_{FA}^{1}\cdot p_{FA}^{2}\right)\left(1-p_{FB}^{1}\cdot p_{FB}^{2}\right)\left(1-p_{FC}^{1}\cdot p_{FC}^{2}\right)$ in the original example.

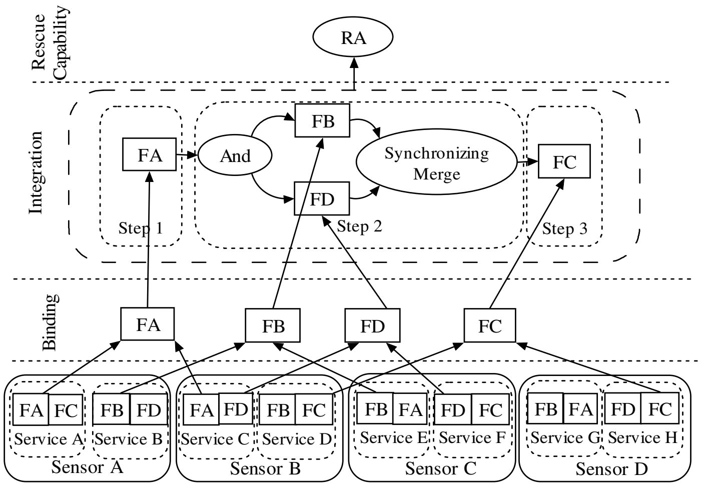


Figure 4. Evolution of rescue capability

In this scenario, the rescue capability could be delivered only when both *FB* and *FD* are available. The probability of successful provision of rescue capability is $\left(1-p_{FA}^{1}\cdot p_{FA}^{2}\right)\left(1-p_{FB}^{1}\cdot p_{FB}^{2}\right)\left(1-p_{FD}^{1}\cdot p_{FD}^{2}\right)\left(1-p_{FC}^{1}\cdot p_{FC}^{2}\right)$. Since $\left(1-p_{FA}^{1}\cdot p_{FA}^{2}\right)\left(1-p_{FB}^{1}\cdot p_{FB}^{2}\right)\left(1-p_{FC}^{1}\cdot p_{FC}^{2}\right)\leq\left(1-p_{FA}^{1}\cdot p_{FA}^{2}\right)\left(1-p_{FB}^{1}\cdot p_{FB}^{2}\right)\left(1-p_{FD}^{1}\cdot p_{FD}^{2}\right)\left(1-p_{FC}^{1}\cdot p_{FC}^{2}\right)$ $(0\leq p\leq 1)$, the reliability of rescue capability is decreased. For a better understanding of impact of evolution, the workflow is divided into three steps as shown in Figure 4. The theoretical success probabilities of the three steps are: $\left(1-p_{FA}^{1}\cdot p_{FA}^{2}\right)$, $\left(1-p_{FB}^{1}\cdot p_{FB}^{2}\right)\left(1-p_{FD}^{1}\cdot p_{FD}^{2}\right)$, and $\left(1-p_{FC}^{1}\cdot p_{FC}^{2}\right)$, respectively in this case. The step two is the weak point in the workflow. Its success probability is lower than the original success probability prior to evolution: $P_{evo}=\left(1-p_{FB}^{1}\cdot p_{FB}^{2}\right)\left(1-p_{FD}^{1}\cdot p_{FD}^{2}\right)\geq\ P_{org}=1-p_{FB}^{1}\cdot p_{FB}^{2}\ (0\leq p\leq 1)$. To address this issue, the reconfiguration algorithm has the self-adaptability to proactively modify its behaviour to deliver a sustainable rescue capability according to the environmental changes. The redundancy of service binding that is the number of services providing a required function is a key parameter and designated as *R*. The redundancy needs to be dynamically self-justified, in order to adapt to capability evolution. For sustainable provision of rescue capability, more services ($R>2$) need to be added and configured to provide each function *FB* and *FD* to enable its Cumulative Distribution Function (CDF) value smaller than the original CDF prior to the evolution.

3.4. Example

In this section, the self-adaptive configuration model is illustrated using a simple example of self-adaptive service binding, where p is defined as the probability of failing to connect a service for integration ($p=p_{FB}^{1}=p_{FB}^{2}=p_{FD}^{1}=p_{FD}^{2}$). The success probability of the second step is evaluated as $P_{org}=1-p^{2}$. As discussed in Section 3.3, the redundancy *R* needs to be dynamically self-justified to maintain the reliability after evaluation $P_{evo} = \left(1 - p^R\right)^2$. The Cumulative Distribution Function (CDF) value needs to be smaller than the original CDF prior to the evolution: $\int_0^1 \left(1 - x^R\right)^2 dx \geq \int_0^1 (1 - x^2) dx$. This inequality is satisfied only in the case of $R \geq 4$. The minimum value $R = 4$ is adopted to minimise the cost of the sustainable provision of rescue capability as illustrated in Figure 5.

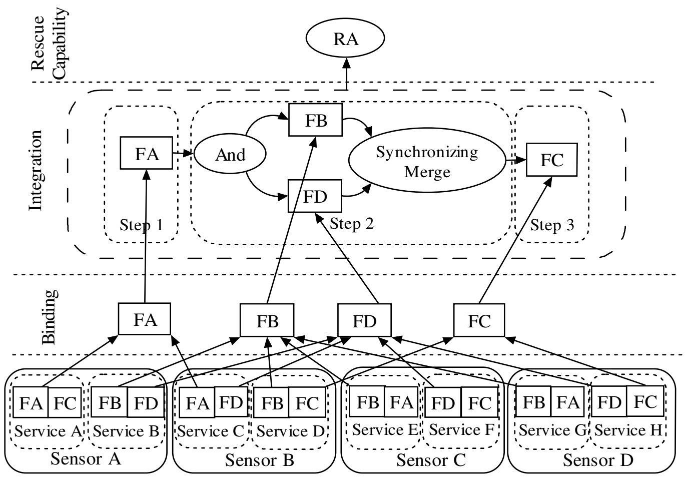


Figure 5. Adaptive service reconfiguration

4. Simulations

In this section, the reliability of provision of rescue capability is evaluated using simulations to see whether the self-adaptive configuration model described above can achieve enhanced reliability for the provision of rescue capability. The simulation system was developed using the Java programming language.

Different systems have different topologies. But many of them are evolving from random topology [10]. The simulation starts from random platform composition and topology to see the influence of capability evolution. In the simulation, a large scale network was first setup containing 1000 nodes. 100 different high level functions were generated, and each service performed 3 functions. Each peer node provided 5 services which were randomly selected from a pool of 500 services. Each peer node randomly connected to 4 other peer nodes bi-directionally and formed a random topology.

As noted above, ongoing changes could be caused by adding and removing services from peer nodes. To simulate the evolution of peer nodes, one peer node was randomly selected and upgraded to provide one extra service to the network and updated one peer node to remove one previously provided service from the network every hour in the simulation. In the simulations, two services ($R = 2$) providing a required function were bound and configured as illustrated in Figure 4, if no other setting is mentioned.

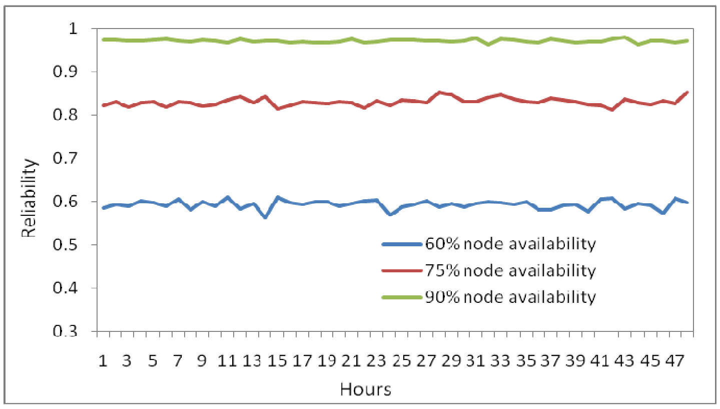


Figure 6. Reliability of capability provision with different availability of nodes

Due to the complexity of disaster environments, there are many combinations of parameters to experiment with and lots of scenarios to test which could generate far too many graphs to analyse. In this section, we only present an analysis of simulation results from the most pertinent experiments as we see.

In the first experiment, we examine the reliability of capability provision with different availability of nodes. The availability of each peer node was set at 60%, 75% and 90%, respectively. As shown in the results, the availability of peer nodes significantly affects the reliability of the rescue capability provision. The highest reliability is achieved in the network with a highest availability of peer nodes.

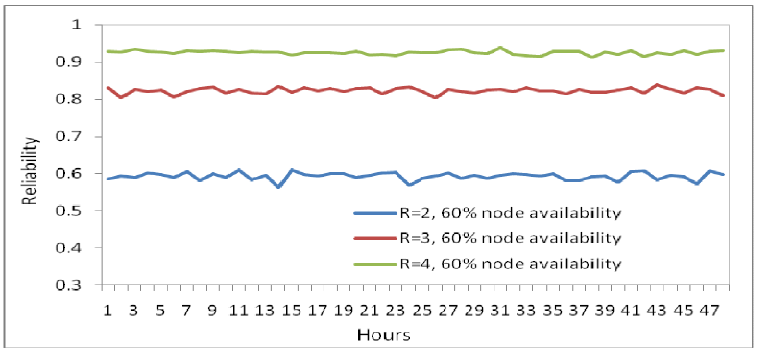


Figure 7. Reliability of capability provision with differen redundancy

As shown in Figure 6, the reliability of capability provision is keeping low in the dynamic network with low availability. To address this problem, the required functions need to be provided by multiple services allocated to different peer nodes. In the second experiment, we examine the reliability of rescue capability provision where two, three and four services providing a required function are bound and configured for the provision of a capability, respectively. As shown in Figure 7, additional redundant services increase the reliability of the provision of capability. The capability configured with the highest redundancy ($R = 4$) achieves the highest reliability and sustainable provision of capability.

From the results shown in Figure 7, the reliability increases by 41% by changing R from 2 to 3. However, reliability grows by only about 10% by modifying R from 3 to 4. Since the highest redundancy achieves the highest reliability, multiple services providing each required function need to be configured to deliver a critical capability with high assurance requirements especially in the dynamic network with the low node availability. But the provision of more services could lead to higher cost and affect affordability.

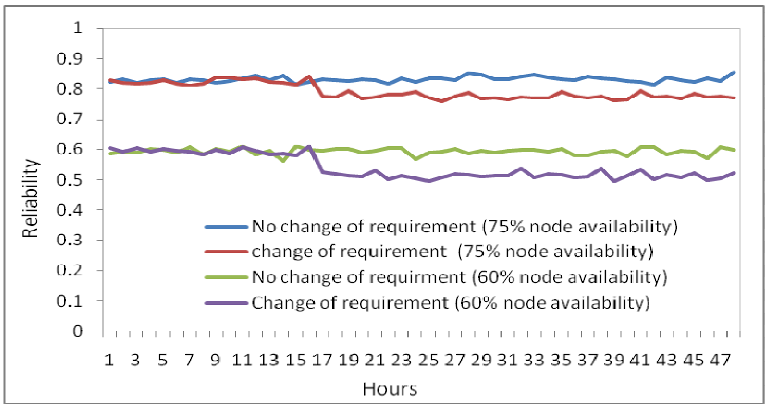


Figure 8. Rescue capability evolution

In the third experiment, simulations were carried out to show the effect of sudden change of requirements. The rescue capability evolution as discussed in Section 3.2 has been injected into the simulations at the 16th hour. Figure 8 shows the reliability of provision of capability with and without the change of requirement with different availability of network node. As shown in Figure 8, the evolution of capability negatively affects the reliability as we analysed in Section 3.3. The capability evolution affects the network with lower availability than the network with higher availability.

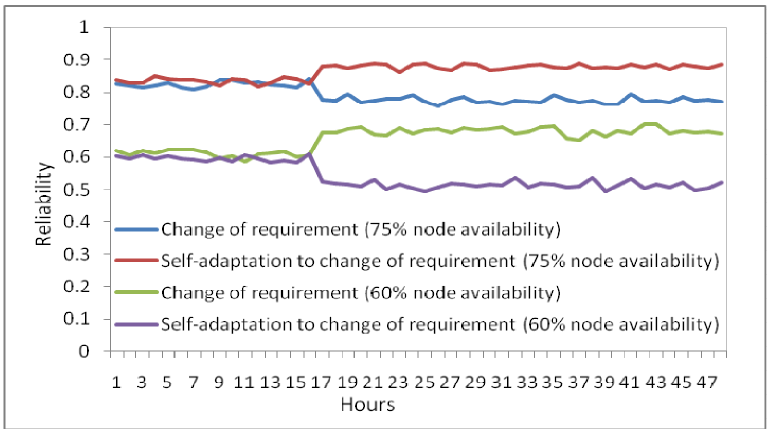


Figure 9. Self-adaptation to change of requirement

Figure 9 shows the reliability of provision of capability with proactive reconfiguration introduced in Section 3.4. Once the evolution of capability occurred at the 16th hour, the reconfiguration algorithm of the new model can autonomously identify the impact of the evolution and self-adapt the redundancy of *F2* and *F4* to $R = 4$ accordingly. From the results shown in Figure 9, the negative impact of evolution has been coped by the re-configuration algorithm and the reliability of provision of capability has been improved.

5. Demonstration System Development

A demonstration system has been developed to demonstrate the use of the architectural model for the dynamic service integration of a network of sensors on a disaster area and to provide a search and rescue capability. The core of the approach is the process of mapping high-level requirements for capability onto the invocation of actual services, allowing the establishment of a dynamic workflow of service composition and integration, and dynamic search for services and on the fly planning through dynamic integration of services. The competitive advantage, such as timeliness, reliability and fault tolerance, can be achieved through the dynamic service discovery, composition and integration.

The software demonstrator was developed in NetBeans 6.1 IDE on a GlassFish application server, which enabled our research group members to write, deploy, test and debug SOA applications using the Extensible Markup Language (XML), Business Process Execution Language (BPEL) and Java Web Services. The Software Demonstrator consists of the following main modules:

- Web Services to simulate capabilities of different systems;
- A dynamic workflow module for dynamic Web Services selection and composition for dependable provision of rescue capability;
- A client interface with sensor information display and user input.

The workflow dynamically discovers and integrates the sensor services and is implemented in a Java written Web Service. The Map Client allows the user to define the coordinate pairs for the ROI along with required response time. The user interface displays a map of the ROI utilizing Google Maps and overlaying the POI. The Map Client user interface uses the AJAX pattern (Asynchronous JavaScript And XML) to connect to an intermediary through a simple XML schema passing region and QoS parameters to the Sensor Workflow and return POI information including feature attributes and the sensors status. The features were then overlaid into a Google Maps display of showing the ROI.

The scenario aim of the software demonstrator was to model region surveillance using dynamic service integration of sensor networks on a disaster area. The intent is to demonstrate the architectural approach to engineering. The main concepts are:

- Use of SOA for disaster relief enhanced with other architectural styles and patterns;
- Integration of distributed systems in a dynamic environment;
- Coping with changes in availability of distributed components.

5.1 Exposing a Legacy Sensor Application to an SOA Network

As shown in Section 3.2, a legacy sensor application is exposed as a Web Service in order to allow it be integrated into an SOA enabled workflow for the provision of a regional surveillance capability. This system model can be summarized with UML as in Figure 10. In this model a “Sensor Service Provider” and client “Sensor Client” are defined.

The Web Service ("Sensor Wrapper") server is hosted within the "Sensor Service Provider" along with the legacy application ("readsensor") that communicates with the physical sensor hardware. "Sensor Wrapper" acts as both an external facing Web Service and a wrapper to the "readsensor" application. The wrapping implementation of the "readsensor" application to a Web Service was shown in Program 1.

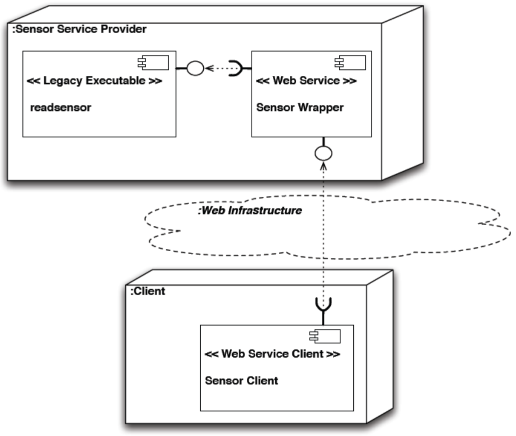


Figure 10. UML diagram of wrapped sensor system

Program 1 Simple Sensor Wrapper program in Java

```
/**
* Simple Sensor Wrapper
*/
@WebMethod(operationName = "getSensorImage")
public String getSensorImage()
{
    /**
    * Launch 'readsensor' program and read the output
    * represented by plain text with base64 encoding.
    */
    Runtime runtime = Runtime.getRuntime();
    Process proc;
    StringBuffer programOutput
    = new StringBuffer();
    try
    {
            proc = runtime.exec("readsensor");
            InputStream inputstream = proc.getInputStream();
            InputStreamReader inputstreamreader
               = new InputStreamReader(inputstream);
            BufferedReader bufferedreader
               = new BufferedReader(inputstreamreader);
            // read the program output
            String programOutputLine;
              while (
              (programOutputLine = bufferedreader.readLine())!= null )
              {
              programOutput.append(programOutputLine);
              }
    } catch (Exception ex) { return null; }
    /**
    * Return the wrapped program output to the
    * Web Service client.
    */
    return programOutput.toString();
}
```

5.2 Dynamic Service Integration

The surveillance was based on Points of Interest, PoIs and physical features within a geographical area that are detected by a group of simulated sensors. The sensor data is then processed to eliminate duplicates and points outside the region of interest and the detected feature positions are displayed on a map. The workflow can be illustrated at a high level in Figure 11. This diagram can be directly mapped onto the SOA integration model shown in Figure 11 and illustrates the implementation technology used in the lower boxes. The workflow integrates Web Services which were the chosen demonstration implementation technology for SOA. The implementation of region surveillance used Google Maps to display the results from the feature detection workflow.

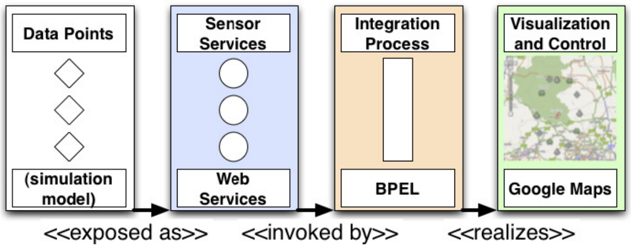
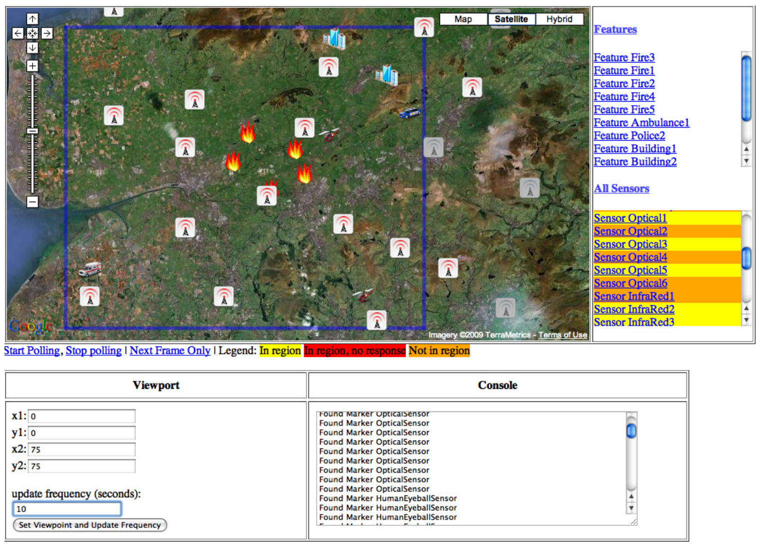


Figure 11. A high-level overview of the service integration workflow


Map
Satellite
Hybrid
Imagery ©2009 TerraMetrics - Terms of Use
Viewport
Console
Set Viewpoint and Update Frequency

Features

Feature Fire3
Feature Fire1
Feature Fire2
Feature Fire4
Feature Fire5
Feature Ambulance1
Feature Police2
Feature Building1
Feature Building2

All Sensors

Sensor Optical1
Sensor Optical2
Sensor Optical3
Sensor Optical4
Sensor Optical5
Sensor Optical6
Sensor InfraRed1
Sensor InfraRed2
Sensor InfraRed3

Start Polling, Stop polling | Next Frame Only | Legend: In region In region, no response Not in region

x1: 0
y1: 0
x2: 75
y2: 75

update frequency (seconds): 10

```
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker OpticalSensor
Found Marker HumanEyeballSensor
Found Marker HumanEyeballSensor
Found Marker HumanEyeballSensor
```

Figure 12. Disaster monitoring showing PoIs

A screenshot from this interface can be shown in Figure 12. In the system, a disaster relief volunteer can submit real-time requests for information of Points of interest, POIs, in a specified region. The system will return the related information about the POIs within that region, e.g., current locations of those POIs (Figure 12). The blue rectangle is the region of interest. The lists on the right of the image show the detected features and the sensors that have been accessed in the workflow. The system is built on a dynamic and changing environment, where sensor services in region may fail to respond with information about the POIs as shown in Figure 12. By using the approach proposed in Section 3, multiple sensor services are contacted to receive the data about POIs in the requested region.

The demonstration system incorporates the following innovations to achieve competitive advantage:

- *Information-Rich Information Services*: provide description of services, composition templates with candidate composed services, application workflows, architectural patterns, application patterns, evaluation information [11].
- *Evolving Ontology*: ontology available for dependability, capability, system assessment [12].
- *Service Interoperability*: advanced techniques for dynamic authentication and run-time negotiation [13].
- *Optimisation for On-the-Fly Planning*: based on a tool [13] that supports the use of a variety of optimization techniques and their combination.

6. Conclusion and Future Work

In this paper, an architectural model of distributed service integration for disaster monitoring sensor systems has been presented, using the concepts of dynamic workflow management to enable dynamic service integration for reliable and sustainable provision of rescue capabilities. This approach is able to identify evolution, evaluate the impact of evolution, dynamically discover services with a distributed peer-to-peer approach and self-configure services to adapt to evolution.

The reliability of provision of rescue capability is evaluated using simulations in a dynamic environment. The impact of node availability, redundancy of service binding and change of requirements has been evaluated and analysed. The simulation results show that the proposed self-adaptive model can achieve enhanced performance for the provision of rescue capability. This architectural model has been demonstrated through developing and testing the demonstration system for a scenario of disaster area monitoring. The wrapping of legacy sensor systems to a SOA network has been discussed. The development of the demonstration system has been used to ascertain that the implementation the architecture is fit for use. Further development of the demonstration system, in consideration of weather conditions, landform and aftershocks, will be used for further evaluation of the performance and reliability of proposed systems in a dynamic environment.

Acknowledgment

The work reported in this paper has been supported by Major Program of the National Natural Science Foundation of China, under contact No.90818028.

References

[1] Cayirci, E., and Coplu, T.: ‘SENDROM: Sensor networks for disaster relief operations management’, Wireless Networks, 2007, 13, (3), pp. 409-423

[2] Liu, L., Russell, D., Webster, D., Luo, Z., Venters, C., Xu, J., and Davies, J., "Delivering Sustainable Capability on Evolutionary Service-oriented Architecture," in *IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2009)*, Tokyo, Japan, 2009, pp. 12-19

[3] Papazoglou, M., and Dubray, J.-J., "A Survey of Web Service Technologies," Available from: http://eprints.biblio.unitn.it/archive/00000586/

[4] Szyperski, C., Gruntz, D., and Murer, S.: 'Component Software: beyond Object-Oriented Programming' (ACM Press: Addison-Wesley. 2002)

[5] Alonso, G., Casati, F., Kuno, H., and Machiraju, V.: 'Web Services Concepts, Architectures and Applications' (Springer Verlag. 2004)

[6] OASIS, "Web Services Composite Application Framework (WS-CAF)," Available from: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-caf

[7] Liu, L., Webster, D., Xu, J., and Wu, K.: 'Adaptive Service Discovery on Service-Oriented and Spontaneous Sensor Systems', Special Issue on User-oriented, Service-oriented and Spontaneous Wireless Ad-hoc Networks, Ad Hoc & Sensor Wireless Networks, Under Review, 2010

[8] Kim, K.H., and Welch, H.: 'Distributed Execution of Recovery Blocks: An Approach for Uniform Treatment of Hardware and Software Faults in Real-Time Applications', IEEE Transactions on Computers, 1989, 38, pp. 626-636

[9] Aalst, W.v.d., Hofstede, A.t., Kiepuszewski, B., and Barros, A.: 'Workflow Pattern', Distributed and Parallel Databases, 2003, 14, (3), pp. 5-51

[10] Watts, D.: 'The Dynamic of Networks Between Order and Randomness' (Princeton University Press. 2005).

[11] Tsai, W.-T., Zhou, X., Chen, Y., and Bai, X.: 'On Testing and Evaluating Service-Oriented Software', IEEE Computer, 2008, 41, (8), pp. 40-46

[12] Webster, D., Looker, N., Russell, D., Liu, L., and Xu, J., "An Ontology for Evaluation of Network Enabled Capability," in *Realising Network Enabled Capability (RNEC'08)*, Leeds, United Kingdom, 2008

[13] Townend, P., Huai, J., Xu, J., Looker, N., Zhang, D., Li, J., and Zhong, L.: 'CROWN-C: A High-Assurance Service-Oriented Grid Middleware System', IEEE Computer, 2008, 41, (8), pp. 33-38