

## Organisational Failures in Dependable Collaborative Enterprise Systems

**Dr. Panayiotis Periorellis, Prof. John E. Dobson**

University of Newcastle Upon Tyne, Department of Computing Science, Centre for Software Reliability, NE1 3RU

### Abstract

In this paper we attempt to classify organizational failures that can occur during the integration process of autonomous systems. The paper reports on ongoing research in the area of systems integration. Our purpose is to provide a taxonomy of organizational failures, so that there is a basis for analyzing some of the possible organizational failure modes resulting from putting together two organizational systems, each with its own purpose in its own organizational context, to make a new Dependable System of Systems. It thus provides a way of examining some new emergent failure modes. It also points out possible failure modes that, because they are organizational, cannot be prevented or tolerated at the level of the individual technical systems or the technical system that brings them together.

## 1 INTRODUCTION: THE CONCEPT OF SYSTEM OF SYSTEMS

A DSoS is a dependable system composed of independent autonomous systems. The purpose of a SoS is to provide a set of enhanced or improved “emergent” services, based on some or all of the services provided by the participating components systems. The provision of these emergent services requires co-operation between the systems. Since the component systems are autonomous, putting them together involves a number of challenges. Certain aspects that pose a number of interesting technical as well as non-technical problems are a) crossing organizational boundaries [1] b) providing fault tolerance across the spectrum of the entire SoS and [2] c) making sure that goals embedded in the design and execution of component systems does not bring the services of the SoS into conflict [3].

In outline, the approach will be to take two simple conceptual models of an organization, models that are relevant to the idea that a system has a role to play in an organizational context, and use these models to describe a number of organizational

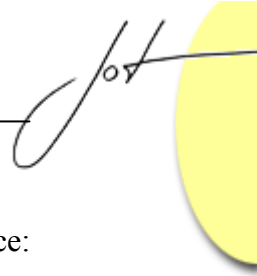
failure in which a single system can be implicated. By organizational failure here, we mean to exclude technical failures of the system itself, i.e. failures to deliver – for whatever reason – the specified service, but to include those many cases where although the service delivered may be considered in accordance with some specification, it fails to fulfill some organizational purpose it was intended to fulfill. We shall extend this discussion to look at problems emergent from a composition of two organizations. The conclusions and results presented in this paper have been derived from a travel agent case study in which we attempted to integrate autonomous booking services (hotel, flight, car and insurance booking services) in order to provide a full trip booking service comprising flight, accommodation, car rental and insurance.

### Simple Models

The two simple models we shall present show two different but related aspects of an organization: structure and process. Briefly, the structure model is a standard one of dividing the organization into responsibilities for direction, for management and for execution. The actual structure of any particular organization is determined –to a greater or lesser degree – by how these responsibilities are mapped onto individual role holders and individuals in the organization; this can obviously vary from one organization to another. What is invariant across organizations is the existence of these three types of responsibility, and the fact that they are mapped onto roles and individuals. The process model divides organizational processes into three types: scoping the business (i.e. deciding what the organization is about), resourcing the business (i.e. procuring and managing the resources needed for the organization to do whatever the scoping process determines it should do) and delivering the business (i.e. the actual performance). It is important to realise that this process model is not just a re-articulation of the structure model. An organisation that simply combined direction with scoping, management with resourcing and execution with delivery would be very naive and not very effective. At the very least, each of the scoping, resourcing and delivery processes would have its own internal D/M/E structure within it, but the actual relationships in practice show a wide variety of configuration possibilities.

### Organisational Structure

As indicated earlier, we classify the responsibilities that exist in an organization into direction, management and execution responsibilities. Direction responsibilities are for deciding on desired future states of the organization, for enunciating strategies for achieving those states, and for allocating generic resources (e.g. overall budgets) to enable the achievement. Management responsibilities are for turning policy objectives and strategies into plans, for transformation of the generic budget into actual resource instances and allocating and deploying them. And of course there are required back channels of reports and accounts. Similarly execution consumes the resources in fulfilling (or not) the plans and reporting back. It is already easy to enumerate a number of failure modes i.e. wrong strategy, wrong plans, wrong execution. When we compose two



organisational structures, there are two levels at which the composition can take place: shared management or shared execution.

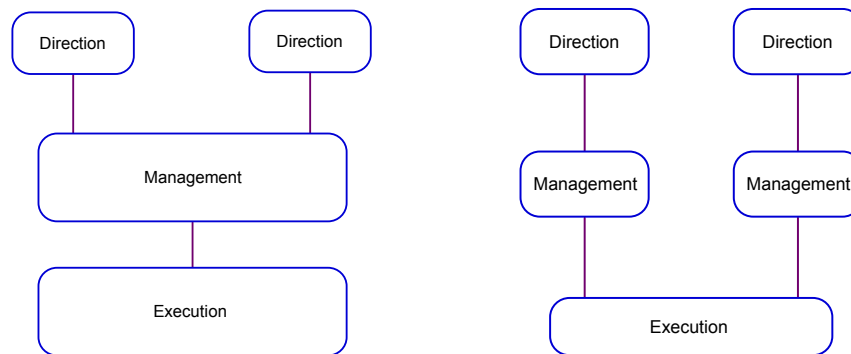
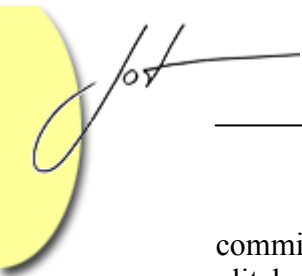


Figure 1. Collaboration Modes

We can now start enumerating different failure modes. In the shared execution case, different plans can conflict or interfere, different reports and accounts can be passed upwards on the two different channels, execution failure can result in different recovery actions at the management level, management can disagree on the allocation of responsibilities, and so on. Similarly, in the case of shared management, there are opportunities for conflicting policies, differing reports and accounts, arguments over managerial responsibilities and so on. The point of these pictures is to provide a simple pictorial representation so that when a particular failure is analysed, it is clear at what level in the organization exception signaling and recovery mechanisms can be placed.

## 2 ORGANISATIONAL FAILURES DURING SYSTEMS INTEGRATION

Although the scoping/resourcing/delivery model describes an organisation in terms of processes, these processes do not always (or indeed hardly ever) employ distinct mechanisms. Thus the delivery mechanism will embody aspects of the (results of the) scoping process; and so on all the way round. And, as explained previously, in an effective value-adding network, the scoping policies of individual enterprises may not be independent. This means that we cannot assume that we can provide a failure -proof travel agent simply by connecting together delivery mechanisms from component suppliers. Examples of scoping mismatch include marketing policies (e.g. different booking systems can assume it is the customer, or an agent, who is interacting with the system), systems with differing models of trust (e.g. credit card authentication required before the transaction can begin, as opposed to authentication when the transaction is



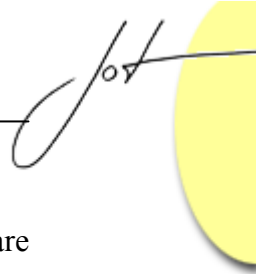
committed), and so on. These policy mismatches should not be seen as mere technical glitches to be overcome by ingenious Java programming in the travel agent system. Rather, they are policy mismatches which constitute a fault which may result in a failure of the travel agent to deliver an adequate service to the customer, and recovery management needs to be addressed at that level. Two particular sources of problem arise from (a) post-transaction failures and (b) post-transaction changes.

If one component organization or service in a brokered package of services fails, then under some but not all circumstances it is the responsibility of the service provider to make alternative arrangements, or compensate, for the loss. Where there is a lack of transparency, which is particularly true of the travel industry, it is understandable that a travel agent may not wish to make it clear to the customer in advance what the possible failures are and how they might be recovered from. In the event of airline failure (whether lack of aircraft or cessation of trading), for example, some –but not all– airlines will themselves try to rebook passengers on other flights. Some –but not all– hotels will seek to re-accommodate travelers if booked accommodation is not available. Some –but not all– travel agents have an emergency number which clients can phone for assistance in the preceding cases. There are three major strategies which can be used to deal with these post-transaction organizational failures. They are:

Forward recovery, Alleviation, Compensation. An example of forward recovery is rebooking, either by the failed airline or the travel agent, on to an alternative carrier. An example of compensation is leaving the responsibility for alternative arrangements up to the traveler who can then claim on some insurance policy — either traveler's or the travel agent's. An example of alleviation is the facility offered by some charge cards that under circumstances of failure, a certain amount may be charged to the card which will not be recharged to the cardholder (usually provided the original charge was made on the card). Fault-tolerance does not seem to be an option.

### Case Study Observations

The structure model of the TA reveals its scope, resources and delivery system. The TA provides full trips consisting of separately chosen accommodation, flight and vehicle to holiday makers. This automatically sets the market which the TA targets. After the market has been identified there are certain assumptions and decisions that need to be made. Bearing in mind that the scope determines the type of resource, we have selected a number of booking systems which are considered appropriate for the TA in the sense that their scope is compatible – we are, for example, excluding package holiday providers. Within the scope we have defined a number of policies regarding the operation of the TA, assumptions about the clientele, the interaction process and the overall responsibility held by the TA. The selection of the type of resource is based on the assumption that the booking systems comply with the scope we have determined for our system. The delivering system provides the service determined by the scope using the resources. Again, the type of delivery system is determined more by the scooping decisions we have



made than by the resources brokered. Design decisions, and implementation schemes are based on this.

The following diagram illustrates what we have discussed so far.

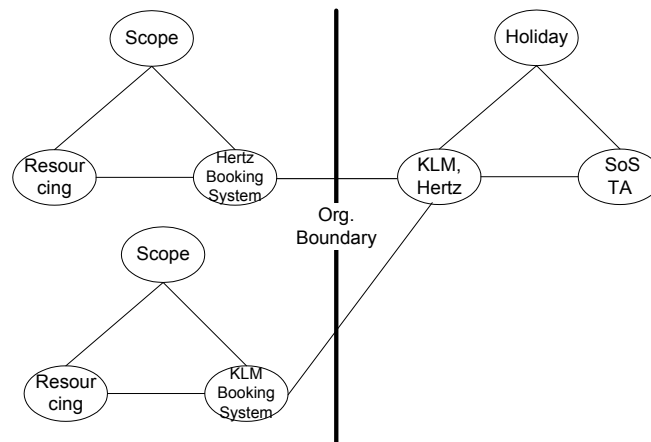


Figure 2. Travel Agent Case Study

Notice that the connections between the resources and the TA are taking place via the linking interfaces of the delivering systems. The booking systems are assumed to be autonomous organisational structures that provide a particular service. We obtain their services via a linking interface. The thick black line indicates an organisational boundary which prevents us viewing the scope of the participating systems. We only have access to the service i.e. the delivery system. Observing a system from a call interface only provides a limited view about the system's services and in particular the policies and scope of those services. Given that the scope of the TA defines its market target and the interaction with its clients it is wise to obtain resources (i.e. booking systems) whose scope complies with the scope of the TA. The scope however cannot be seen via the call interface. We could indeed observe and test a LIF to improve fault tolerance of the overall system but we cannot obtain information about the scope of the service (e.g. who it is intended for). We can obtain partial information on how the service is delivered by looking at the protocol and the relevant transactions that take place but we could not for example obtain information about the policies regarding a protocol, authentication, levels of trust etc. This limitation can be accommodated within a static system because decisions can be made prior to setting up a TA delivery mechanism. What cannot so easily be done is to detect, and respond to, changes in the scoping decisions of suppliers. Some typical failures would arise in the following scenarios. We have assumed that the TA policy has defined a way which the TA interacts with customers and processes their requests. Such policy assumes, for example, that the TA does not require registration and

shows a high level of trust in its clients. We also assume the TA chooses to consider a journey as an end-to-end arrangement.

**a) Booking System A requires customer registration.**

This is in direct conflict with the protocol of the TA since the latter does not require registration. Although protocol differences like this can be observed at the LIF, additional policies regarding the levels of trust (which the registration process is related to) cannot be observed at the LIF. This can be regarded as a mismatch between policies of organisational trust.

**b) Company A is a ‘no-frills’ airline (customer base).**

Such airlines typically take no responsibility for knock-on consequences of delayed or cancelled flights, even when the next leg is one of their own operations. This is an example of market target clash between the TA and the supplying system. Obviously this cannot be observed at the LIF and additional monitoring mechanisms at the level of the TA would be needed to resolve these issues. This is in direct contrast with the scope of the TA which targets independent holiday makers traveling on full-service airlines.

There could be many examples like the ones above that illustrate the point that LIFs do not offer adequate information for including an autonomous system as a resource of a SoS. We count these as examples of organisational failure since recovery, if it is to be achieved at all, has to be done at the level of the organisation. Let us view another category of failures that is raised during the actual service delivery. Remember that we have assumed that systems are autonomous in the sense that they operate outside the scope of the SoS.

**c) Inaccuracy of information**

The quality of information produced by the booking system could hinder the overall service offered by the TA. Questions such as ‘is the information up-to-date’ and ‘is the source reliable’ cannot be answered by only looking at the LIF. The main question that we need to consider when we compose a service from various sources is whether the quality of data adheres to the quality expected by the TA and its customer base. It is a sad fact that some operators publish incorrect information about their services on their own websites. Here again it is a policy decision to be made by the TA how much effort they are prepared to spend in dealing with recovery from failures experienced by their clients due to misinformation outside the control of the TA.

**d) Service offered differs from the service promised or advertised**

This is closely related to the first point and again unless performance records are maintained this type of information cannot be found at the level of the LIF. However, this raises the possible need for an additional interface to the TA which allows clients to submit reports to the TA on the services brokered by the TA. This is a facility already offered by conventional travel agents, particularly those serving the business travel community.



### **e) Time semantics**

It would be wrong to assume that component systems operate on the same time semantics regarding the handling of requests. In fact it is likely that component systems will operate according to their own semantics which are embedded into the delivery system and hidden from view. The TA needs to be aware of these prior to making any requests. Consider the following example. The TA has a 30 second timeout rule. This implies that a reply for any request has to be received within 30 seconds. If this does not happen the TA throws a timeout. The booking system however operates a queue which due to its nature and the number of requests received operates a 45 second timeout rule. This as one can imagine can lead to the booking system actually making a booking i.e. handling the request successfully while the TA thinks otherwise. Can we observe time semantics over the call interface?. Unfortunately these are set as part of the operation policies of each system and are hidden from view.

We have shown so far that call interfaces (LIFs) can only offer some indication regarding to the services of the booking system. In fact we have shown that although they can show how a service is delivered (protocol) they do not indicate the operation policies of the service. Additional interfaces are necessary to do this. Composing an emerging service out of services obtained from autonomous systems can lead to failure due to a number of errors. We summarise these below.

### **a) Market**

The service may not be intended for the same market base. These targets are set as part of the scope of the organization and are therefore invisible at the interface level.

### **b) Protocol**

Although the protocol is visible and to an extent it can be manipulated, we cannot always assume that we can compose the trip using any component system. Although wrapping would allow us to hide some of the incompatibilities regarding i.e. requirements expressed in additional requests, they cannot hide certain aspects of the interface that are part of a wider policy i.e. authentication, user registration etc.

### **c) Reliability of Service**

As we have mentioned earlier this cannot be assumed and additional mechanisms would be to be in place to ensure that the same quality as described by the scope of the TA would be maintained throughout.

### **d) Responsibilities**

There are a number of responsibilities that need to be assigned on certain roles in order to avoid failure. Consider the following questions: Who is responsible for informing the user of changes? Who is responsible for compensating the user? Can the user cancel the trip and within what time scales? All these responsibilities need to be assigned roles in order to avoid failure.

### 3 ENHANCING DEPENDABILITY

Some of the organizational failures can be prevented by adding additional layers of exception handling, maintaining additional information about each component system, and keeping track of performance records. Some of the following could solve some of the errors that give rise to failures.

#### **a) Accessing additional interfaces.**

Although this may not always be possible it is desirable to obtain some information (such as public information) about the scope and policies of the systems. An interface between the scopes of the two systems would eliminate failure raised from clashing policies. It would also provide a better idea as to whether it is feasible to include a particular system as a resource. However, to be effective, this would require support at the management level of the separate organisations.

#### **b) Metadata**

Metadata information could be used to maintain certain policies in data structures. While not all policies can be represented in data structures, keeping metadata about customer base, protocols and authentication would allow the TA to reason about the composition of the emerging service (i.e. suggesting that a,d,f compose better than a,b,d). Since operation policies can change metadata would also need to be changed. Additionally we would need to ensure a reliable interface for obtaining such information.

#### **c) Model of Responsibilities**

While maintaining the brokerage model of operation, providing the user of a model of responsibilities (i.e. who is responsible for what) would help the TA to assign roles for every responsibility (cancellation policy or changes in trip details). As we mentioned earlier post transaction management needs to be dealt at the level of the SoS and all responsibilities derived by it need to be assigned roles.

#### **d) Composing according to user requirements**

Being able to compose an emerging service according to user's requirements would allow the TA to avoid failures regarding clashes between certain policies (the user is a backpacker while the trip is for business class travelers). This of course implies certain technical implementation in order to discreetly obtain data about the type of client and address that particular client using the appropriate service.

#### **e) Maintaining performance records**

Maintaining records on different compositions and users would help the TA to assess the compatibilities between the services offered by its component systems and the type of users it services. It would also allow the TA to evolve its services according to the evolution of the component system. Additionally they could provide an indication about changing policies, scope, operations etc.





#### **f) Involving the user**

Finally involving the user in certain decisions would allow the TA to drop certain responsibilities. The user can be involved in selecting a particular configuration of a particular trip that addresses his type. The TA additionally could make suggestions about certain configurations and maintain track of users' preferences. This would help resolve a number of issues regarding targeting the right customer with the right service. Involving the user in this process would help the TA to resolve issues raised by quality of service (user can select a service based on past experience), reliability, accuracy etc.

## **4 CONCLUSIONS**

In this paper we showed that integrating enterprise systems is not only dependent on implementing the right protocols, transactional models and providing a sound exception handling strategy. System interfaces do not provide sufficient information to guarantee dependability across the sos. This lack of information can lead to organisational failures. Some failures stem from incompatibilities in the scope of the two systems and they may or not manifest themselves as technical faults. We attempted to classify these failures according to their impact and provide some ideas for solutions to prevent them.

## **REFERENCES**

- [Perio2001] P. Periorellis, J.E. Dobson. Case Study Problem Analysis. The Travel Agency Problem. Technical Deliverable. Dependable Systems of Systems Project (IST-1999-11585). University of Newcastle upon Tyne. 2001. 37 p. [www.newcastle.research.ec.org/dsos/](http://www.newcastle.research.ec.org/dsos/)
- [Ro2002] A.Romavonsky, P.Periorellis. On Structuring Integrated Internet Applications for fault tolerance. Technical Report March 2002, <http://ouston.ncl.ac.uk/main.htm>
- [PP2002] P.Periorellis, A.T. Lawrie, Goal Oriented Composition of Dependable Systems for Systems, Technical Report February 2002, <http://ouston.ncl.ac.uk/main.htm>
- [JED2002] Dobson John, Vithida Chongsuphajaisiddhi June 1999. 'Enterprise modelling as a way of describing the context of use and evaluation of Enterprise Modelling' 4th International workshop on evaluation of modelling methods in systems analysis and design. Heidelberg Germany

- [PP2000] Periorellis Panayiotis. July 2000 'Dynamic Enterprise Modelling'  
International Conference on CAD/CAM, Robotics & Factories of the future,  
CARS&FOF 2000, Vol 2, pp 705-712
- [PP2000] Periorellis Panayiotis. September 1999. 'Supply Chain Modelling Using  
Object Orientation: Advantages and Pitfalls' TOOLS99' ASIA Nanjing  
China
- [PP1999] Periorellis Panayiotis & Dobson John. June 1999. 'Enterprise Thesaurus  
IEMC99' International Enterprise Modelling Conference, Verdaal, Norway

Further Work, Technical reports and a prototype of the case study can be  
found at <http://ouston.ncl.ac.uk/main.htm>

## 5 ACKNOWLEDGEMENT

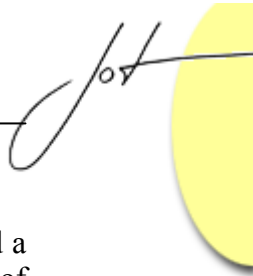
This work is supported by European IST DSOS project (IST-1999-11585)

### About the authors



**Dr. Periorellis Panayiotis** is a research associate at the Centre for Software Reliability, Dept of Computing Science at Newcastle University. He carried out his Ph.D. research in the area of enterprise modelling between 1997 and May 2000. Since June 2000 he has been working in the area of dependable systems with particular emphasis in the composition of systems from autonomous components. He has published a number of papers in the area of enterprise modelling and information retrieval. He is investigating organisational implications and failures when integrating largely autonomous systems. E-mail: [panayiotis.periorellis@ncl.ac.uk](mailto:panayiotis.periorellis@ncl.ac.uk)

**Prof. John E. Dobson** John Dobson obtained a M.A. in mathematics and philosophy from Cambridge in 1962 and subsequently researched into mechanical translation. In 1968 he joined Rolls-Royce as computing instructor, leaving in 1971 to teach computing at Newcastle Polytechnic. From 1974 to 1980 he was employed at the University of



Newcastle developing the University's computing systems and network. In 1980 he and a few colleagues set up their own R & D company, where he assumed the position of Technical Director of a company which grew to employ over 120 staff. In 1986 he returned to the University, carrying out research in computer security and in information technology and organisational change. In August 1996 he was appointed Professor of Information Management at the University of Newcastle. His current research projects are DIRC, of which he is Technical Director, a large multi-site interdisciplinary project investigating dependability of systems which involve both people and computers; AMASE, a project which is looking at systems to support the integration of public sector service delivery; and County Durham and Darlington EHR, a project which is looking at a pilot implementation of the proposed new NHS electronic health record.

For Further information <http://www.csr.ncl.ac.uk>