# A Model-Driven Approach to Interoperability and Integration in Systems of Systems

Gareth Tyson[1], Adel Taweel[1], Steffen Zschaler[1], Tjeerd Van Staa[2], and Brendan Delaney[1]

[1] King's College London, UK
[2] General Pratice Research Database, UK

**Abstract.** Increasingly, systems are being constructed from sets of pre-existing sub-systems, which are developed and owned by managerially independent organisations. Such systems are termed Systems of Systems (SoS) due to their composite nature. The integration of multiple heterogeneous and autonomous systems, however, can be a complicated and resource-consuming task. For instance, differences between service interfaces, business processes, data formats and underlying technologies can make interoperation and integration difficult. To investigate this, we present an SoS called ePCRN-IDEA, which utilises model-oriented techniques to improve the integration process. Through this case-study, a number of issues and challenges are discovered and presented. To address these limitations, a model-driven conceptual architecture is then described in the context of designing the next phase of ePCRN-IDEA.

## 1 Introduction

Nowadays, computing systems are often built from many existing sub-systems, which collectively offer the services and data required to fulfil some goal. Such systems are called Systems of Systems (SoS) because they combine the services of each (sub-)system to collectively build a new system [1]. The integration of multiple heterogeneous and autonomous systems (managed by different organisations), however, can be a complicated and resource-consuming task. Common examples of problems include issues with system assumptions, data compatibility, programming interfaces and security. This is further exacerbated when considering the potential diversity of developers and organisations responsible for the construction, maintenance and management of these sub-systems. Consequently, these organisations will often have varying goals and priorities, making integration a complex business process. As a foundation for this it is therefore vital that divergent systems are able to interoperate. This paper investigates the use of model-oriented techniques for addressing service and data interoperability within an SoS. According to [2], interoperability can be separated into three categories,

- *Technical Interoperability:* This refers to the compatibility of the underlying technologies used to perform interactions (e.g. protocols).

– *Semantic Interoperability:* This refers to the ability of each party to understand and interpret the data of others (e.g. data formats).
– *Process Interoperability:* This refers to the compatibility of the different processes undertaken by each party (e.g. Task A should be performed after Task B).

These different interoperability challenges make it difficult to build and integrate composite systems scalably and in a reliable way, especially when attempting to evolve such systems in a consistent and coordinated manner. For instance, modifying the operation of one sub-system can heavily affect the operation of another [3]. However, the decentralised and autonomous nature of an SoS means that this is a problem that can easily occur.

To gain a better understanding of these challenges, this paper performs an investigation into using model-oriented techniques within an SoS. Specifically, a case-study is presented of an SoS that has been developed to improve clinical trial recruitment. This SoS, *ePCRN-IDEA*, brings together a number of systems and organisations to integrate the necessary services and data required to identify and recruit patients for clinical trials. Using this case-study, a number of issues and research challenges are identified. A model-driven conceptual architecture is then proposed to address these challenges. It is suggested that the key to improving the integration of an SoS is to better enable communication between the different organisations. To enable this, we therefore propose the use of specialised infrastructure to support the required forms of communications through shared, evolvable runtime models.

The rest of the paper is structured as follows. Section 2 provides a background to the problem, alongside related work. Section 3 then details a case-study SoS, ePCRN-IDEA, which has been developed using a model-oriented approach. Following this, Section 4 discusses a number of issues and research challenges that have been discovered during the development process. Using the lessons learnt, Section 5 then details a model-driven conceptual architecture to addresses the issues found, discussing the next phase of ePCRN-IDEA.

## 2 Background and Related Work

Increasingly, systems are being constructed from multiple divergent sub-systems, which each constitute independent entities in their own right. Frequently, these are owned, developed and maintained by separate organisations, which are brought together by a common goal. Due to their composite nature, these are often called Systems of Systems (SoS) [1]. Lock et. al. describe an SoS as,

> "A collection of systems both technical and socio-technical which pool their abilities to present a more complex system, whilst retaining their individual autonomy." [4]

An example of an SoS is NPfiT [5], which attempts to support seamless data sharing across a range of different healthcare systems in the UK. Currently, a

number of sub-systems exist within the UK, each handling various tasks, e.g. patient record storage, decision support, test result management, data collection etc. The various sub-systems are maintained by a variety of organisations, including public, private and research bodies. Consequently, the processes of interoperation and data integration are restricted by many technological, ethical and organisational issues. For example, issues include inconsistencies that arise during sub-system evolution [4]; the limitations of semantic data interoperability [2]; reliability problems caused by inter-dependencies [6]; and error tracing [7].

To overcome these restrictions, model-oriented approaches have been proposed. These involve formally modelling an SoS to fuel the design, development and integration process. MODAF [8], for example, offers a comprehensive framework for modelling and specifying an SoS. TOGAF [9] is another example, which provides an approach for designing and implementing enterprise systems. These allow developers to model an SoS at various levels; TOGAF, for instance, defines four 'architectural domains' allowing business processes, system interactions, data assets and technological aspects to be formally specified. Similarly, software, such as HP's Universal CMDB [10], attempt to allow developers to manage and evolve an SoS, whilst maintaining a consistent view of inter-dependencies. [4] describes an approach for modelling a socio-technical SoS using inter-system and environmental dependencies, alongside system capabilities; in this context, a capability is considered as the ability to offer some expertise that is beneficial to the SoS. Consequently, the SoS is viewed as containing a set of independent modules, which possess capabilities and rely on dependencies. Various other research similarly focusses on ensuring that such models can be realised in a reliable way; for instance, through component contracts [6]. In contrast, other approaches attempt to model an SoS through requirement specifications [11], thereby assisting in verification and validation of the SoS. More generally, a range of work has also been performed in the field of large-scale systems [12].

Despite this research, there is still little support for handling the true level of (evolving) heterogeneity within a large-scale SoS. To explore these limitations, the paper now extends this background work by describing a prototype of an SoS called ePCRN-IDEA, which has been built in a model-oriented manner. Through this prototype, we identify a set of issues and challenges, alongside investigating the potential of model-oriented engineering to address them.

## 3 Case Study: ePCRN-IDEA Recruitment System

This section provides a detailed overview of the ePCRN-IDEA system. It is a distributed SoS used for detecting eligible patients for clinical trials within a consultation (i.e. in real-time). Its correct operation relies on a number of sub-systems, developed and managed by a range of organisations. Consequently, we present this as a case-study to illustrate a number of issues and research challenges in the field.

### 3.1 Overview of ePCRN-IDEA

The ePCRN-IDEA system intends to improve the recruitment procedures of clinical trials by enabling real-time identification of patient eligibility. In essence, whenever a patient enters a clinic, their details are compared against a set of eligibility criteria for trials that are currently active; generally, eligibility requires inclusion on a pre-computed list. If they are found to be eligible, a notification is presented to the practitioner, who can then inquire about the patient's interest. If the patient would like to be recruited, this can be performed immediately through a website.

Fig. 1 shows a UML deployment diagram of the systems involved in the recruitment of patients through ePCRN-IDEA. Providing a detailed description of each process is beyond the scope of this paper, however, the core processes are as follows. When a research body wishes to create a new clinical trial, they can inject it through a service called the Central Control Service (CCS), which is hosted at King's College London (KCL). The CCS stores trials within a large database in a pre-defined format that all researchers must adhere to. Associated with each trial is a list of potentially eligible patients; currently, these lists are generated by the General Practice Research Database (GPRD), which operates a large data warehouse containing over 12 million patient records. Following this, the trials and their eligibility lists are distributed to software agents (called LEPIS agents) that operate on clinicians' PCs at each participating clinic. LEPIS then listens to the interactions between the practitioner and their local Electronic Health Record (EHR) database, which is used to store information about patients (e.g. diagnoses, treatments, demographic data etc.). Therefore, during consultations, LEPIS compares the patient information against the eligibility lists of all known trials. If a patient is found to be eligible for a trial, a graphical pop-up is generated to notify the practitioner. If the patient is interested, the pop-up offers the ability to load a Random Clinical Trial (RCT) website provided by the research body responsible for the trial, which allows the patient's recruitment to be completed.

### 3.2 Composite Sub-Systems

The ePCRN-IDEA system requires a number of sub-systems to cooperate to provide the necessary data and computation to locate eligible patients for trials. The required computation and data detailed above is therefore spread over a range of systems, which are owned, developed and managed by a number of organisations. Fig. 2 shows the organisations responsible for each sub-system; these are as follows,

- *Vision EHR:* This is the database system used to store patient Electronic Healthcare Records (EHR). This is used by practitioners to enter information about patients during consultations, e.g. treatment codes, diagnoses. Currently, only one EHR system is supported (Vision), however, there are also a number of alternate vendors, e.g. EMIS. Vision is developed and managed by a private company called InPS.

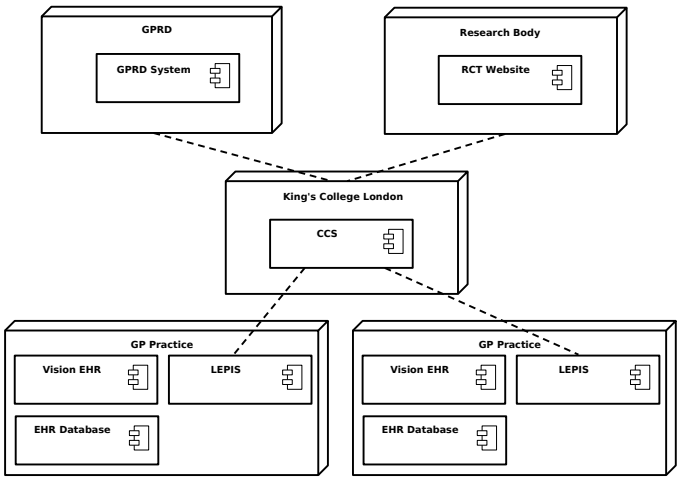


**Fig. 1.** Overview of ePCRN-IDEA's Recruitment System

- *General Practice Research Database (GPRD):* This is a centralised data repository for performing and assisting with medical research. It collects and indexes information from ≈590 general practices across the UK. It is used to pre-compute lists of eligible patients for each trial based on complex search criteria. These lists are then associated with individual trials and passed to the CCS. This is managed by a governmental body.
- *LEPIS:* This is a software agent, which operates alongside Vision on a practitioner's local PC. It inspects any patient information entered to ascertain if they are eligible for a trial (e.g. on an eligible patient list). If they are, LEPIS is responsible for generating a pop-up notification. LEPIS is developed and managed by KCL but resides on practitioners' computers.
- *Central Control Service (CCS):* This is a service, which accepts the registration of new trials including eligible patient lists and recruitment details. It distributes this information to LEPIS instances on practitioners' PCs. This is hosted and managed by KCL.
- *Random Clinical Trial (RCT) Website:* This is a website, which can be used by authorised practitioners to register patients for trials. Currently, a prototype has been developed by a private company, Red Ant; however any research bodies wishing to use the system are likely to utilise their own RCT website.

The above sub-systems exist in various organisations, including private, governmental and research establishments, as shown in Fig. 2. The development process has involved all of these organisations either building new software or modifying their own. Importantly, this software operates on a huge number of hosts, with ≈9,000 clinics operating in the UK alone. There are four key organisations involved in the system; (*i*) KCL, which has developed the CCS and

LEPIS, (*ii*) GPRD, which is used to compile lists of eligible patients before passing them to the CCS, (*iii*) InPS, which owns and manages the Vision EHR, and (*iv*) the RCT website, which is managed by the research body responsible for the trial (currently a prototype developed by a private company, Red Ant, is being used).

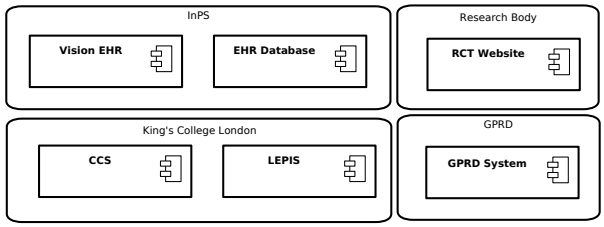


**Fig. 2.** Organisations Responsible for Each Sub-System

### 3.3 Integration of Models

To enable the above sub-systems to interoperate and integrate their data, it was necessary to build a set of shared data models. These were used to marshal incompatible data representations into a single standard. The four models are as follows,

- *Trial Description:* This model holds a description of a trial, including the title, overview, funding body etc. It allows practitioners to be presented with details of the trial during the recruitment phase.
- *Eligibility Criteria:* This model represents a formal computable set of criteria for patient eligibility. It defines the characteristics required of any participating patients. Currently, the most predominant form of criteria is eligibility lists, which store a set of patient identifiers that are potentially eligible for each trial.
- *Recruitment Model:* This model stores information about the recruitment process. It allows target recruitment numbers to be set, as well as detailing the RCT website that should be used for registering recruitment.
- *Consultation Model:* This model stores information about a given consultation that can be exposed by the EHR to allow LEPIS to compute eligibility. It currently holds the patient identifier, date of birth, staff identifier, staff role and any data entered (e.g. treatment codes).

These models were defined through design-time collaboration between all parties. Following this, they were realised as XML schemas and distributed to each organisation. KCL managed the first three models, whilst the fourth model was managed by InPS. These models were then reified internally by each individual organisation into a form that could be used; specifically, they were converted

into Java classes using JAXB. Following this, any data exchanged between the sub-systems was marshalled into the XML standard from these Java classes.

## 4 Issues and Research Challenges

Within ePCRN-IDEA, a model-oriented approach was taken, using a set of data models to encapsulate the required data exchanges that must take place. Through this, a number of problems and challenges have been identified. These challenges will be discussed in the following section and used to drive the next phase of ePCRN-IDEA.

**Data Integration from Heterogeneous Sources** When integrating the sub-systems in ePCRN-IDEA, we found it necessary to bridge the multiple formats and storage mechanisms used within each organisation. Through this, we discovered a number of challenges.

First, the sheer quantity of data, as well as the diversity of organisations involved, made it unscalable to insist that all sub-systems begin to store data in the new shared format. Consequently, each sub-system maintained its own data storage mechanisms and then performed runtime translations. This therefore made it difficult to optimise the underlying storage mechanisms, as they were shielded from the true needs of the external cooperating subs-systems. Similarly, this heterogeneity made it difficult to fully integrate the different data sets into one shared repository as each sub-system maintained different information, using different schemas.

A second related problem that emerged is the complexity of integrating further sub-systems. Currently, ePCRN-IDEA only uses one EHR vendor, Vision; however, in the next phase, further vendors are to be added. The model-oriented approach taken assists in extending the SoS, however, there are remaining issues caused by the limitations of design-time models, which often cannot take into account future evolution. For instance, the Consultation Model currently uses Vision-specific identifiers, making it necessary to extend it to support alternate schemes. Beyond this, it would also be necessary for new vendors to modify their software to reflect the model. We therefore found it necessary to increase the automation of such processes, e.g. through code generation.

To allow the system to be rolled out across the UK, the cost and complexity for new sub-systems to integrate therefore must be controlled. We believe this can be done in a number of ways; clearly, performance must be considered an integral requirement. However, it is also important to support seamless integration of multiple data sources (e.g. in terms of data semantics). To enable this, it is important for model designers to gain a deep insight into a wide range of storage formats and technologies in the domain before proposing any models. Evidently, beyond this, it is also important to select the correct level of abstraction to ensure that sub-system specific attributes are not integrated into shared models that might need to be used with other (existing or future) sub-systems. Beyond

this, we learnt that where possible code generation should also be supported to better enable convenient and low-cost integration.

**Sub-System Process Changes** During the development of ePCRN-IDEA, only a small number of members from each organisation cooperated together. As organisations often operate with a range of aims and priorities, it was obvious that sub-system process changes could take place without necessarily informing all other organisations (or perhaps even other members of the same organisation). This therefore led to a new challenge, as an internal change within an organisation can further modify the interaction needs of another. For instance, as the data storage of the CCS developed over time, the required information from the RCT website similarly changed. Originally, when a patient was recruited, the RCT website passed the information to GPRD, which would then remove the patient from the pre-compiled list of eligible patients, before issuing an update to the CCS. However, as more complex eligibility criteria were developed (e.g. based on age ranges, diagnoses, gender), this became impossible as the pre-compiled lists weren't always used. This created the need for a new interaction between the RCT website and the CCS, thereby resulting in a process change. Such process changes might often then require new infrastructure, security protocols, data policies etc.

To enable practical deployment of the the system, we have therefore found it important to extend the shared data models to also include interaction models. These models must then contractually oblige each sub-system to adhere to what has been defined. Further, modifications must be captured and published in a formal manner, thereby ensuring that all sub-systems remain informed about potential changes. Beyond this, we learnt that it is also important to put mechanisms in place to support communication between the different members of each organisation to improve such process changes.

**Model Evolution** As the requirements, specifications and sub-systems of ePCRN-IDEA evolved, the related models also needed to evolve to represent this. Statically defining a model would prevent this and make a system limited in its ability to reflect new changes. We found this, however, highly challenging as the management and versioning of such models can be difficult, as well as the necessary translations that must be performed between different versions. Within ePCRN-IDEA, all models were defined and distributed in a static manner after the necessary collaboration between the organisations. Consequently, no mechanisms were put in place to manage the subsequent evolution of such models. This meant that changes to one model must manually be discussed with all organisations before sending (e.g. via email) the updated model to all parties. It was then necessary for each organisation to manually modify their system to utilise the new model. This could consist of a number of steps but generally involved first reifying the model into code, replacing the existing code, then performing any changes to the other parts of the system. For instance, the original Trial Description model contained the URL of the RCT website; later, however, this

was moved into the Recruitment Model. The above steps therefore had to be performed to notify the related parties, creating a significant overhead.

To allow the system to grow, we therefore believe it is necessary to define a scheme by which models can elegantly evolve in a secure and trusted manner. To enable this, we consider it necessary to (*i*) offer an infrastructure to store, manage and distribute models, (*ii*) define a versioning mechanism by which old models can co-exist with new models, and (*iii*) support model translation to allow sub-systems using previous models to continue interoperation with sub-systems using newer models.

**System-Wide Consistency** Within ePCRN-IDEA the above issues of heterogeneous data integration, process change and model evolution created a number of problems in terms of system-wide consistency. This occurs when different sub-systems begin to hold inconsistent views of the wider system as a whole. For instance, if a new EHR vendor were integrated into ePCRN-IDEA, it would use a different type of patient identifier. This would have a wider impact because the shared models attempt to abstract away from the underlying implementations of the various sub-systems. Consequently, data passed from the EHR to LEPIS (and then subsequently to the RCT website, the CCS and GPRD) could easily be mis-interpreted; for example, LEPIS might attempt to compare an eligibility list of Vision identifiers against an alternate type of patient identifier (e.g. EMIS). These semantic inconsistencies extend to various other aspects such as the coding of clinical concepts (e.g. diagnoses, treatments), which are represented in many different ways. Unfortunately, however, such inconsistencies are easy to propagate.

To address these forms of inconsistencies, we have found it necessary to begin building detailed meta-information around the models (and data) used within the system. For instance, sub-systems should explicitly state the models (and versions) they support, before allowing other sub-systems to interconnect. Further, all data exchanged between the different sub-systems should be checked and validated to limit the propagation of possible data inconsistencies.

## 5 A Conceptual Architecture: ePCRN-IDEA v2

This section presents a conceptual architectural for addressing the above issues and challenges that we discovered. Consequently, this section provides an overview of the next phase of ePCRN-IDEA, which is currently under development.

### 5.1 Requirements

The previous discussion has resulted in the compilation of a small set of requirements. In essence, we believe many of the problems with ePCRN-IDEA can be addressed by improving and formalising the communication between the different organisations. Specifically, we believe our approach should be extended to

make greater use of model-driven engineering. Importantly, it has been learned that models should be evolvable to reflect changing needs; similarly, it must be possible to distribute new models amongst the different sub-systems during runtime. Clearly, the models must therefore be extended beyond their current use in ePCRN-IDEA. First, models must not only define shared data formats but also the underlying processes that enable their exchange, including the interconnections between the sub-systems and the data models supported. Similarly, we believe the use of these models must be extended into runtime, allowing models to change to reflect new requirements. Therefore, to enable this, a secure infrastructure must be put in place that can assist this process by allowing controlled modification of models, as well as their subsequent distribution. Finally, to support the progressive evolution of the system, mechanisms must be provided to allow model translation to occur, so that sub-systems using different versions can continue to interoperate.

### 5.2 A Dynamic Model-Driven Framework

To address the above requirements, we propose a dynamic model-driven framework, as a mechanism to improve and formalise the communication between sub-systems and organisations. Within our approach, an SoS is required to formally model all sub-systems, as well as their interactions (including service interfaces and data formats). Such models can then be stored and made publicly available during runtime, allowing any constituent sub-system to understand the capabilities and data of other sub-systems, as well as what is required of itself. The purpose of this is three-fold, (*i*) it allows all systems to understand the data and services provided/consumed by all other systems, (*ii*) it reduces development costs by allowing auto code generation from the models, and (*iii*) the evolution of the system can be computed, allowing model translation to enable more seamless interoperation between the different systems where possible (e.g. generating mappings between different data formats). Therefore, by utilising formal system modelling, sub-systems can be bound and evolved at runtime under the constraints of the overall system model. To enable the use of such models, a number of architectural entities are proposed,

- *Service Repository:* Each sub-system must centrally register the service interfaces and data models it consumes and produces. This allows sub-systems to gain references to each other, alongside discovering the data models they are capable of interpreting.
- *Model Repository:* Each model must be centrally registered and made accessible to the sub-systems (potentially at runtime). This allows models to be centrally managed and updated, before being distributed to other sub-systems. This can be separated into Local Model Repositories (LMR) and the Central Model Repository (CMR).
- *Terminology Service:* A look-up service must be provided that can offer mappings between different terminologies used by the sub-systems. This allows data types to be given further (shared) meaning, e.g. 'diabetes' is coded differently in the UK and USA.

Fig. 3 provides a basic overview of the model-driven framework, as integrated into the ePCRN-IDEA recruitment system. It can be separated into two main groups, (*i*) the model-driven framework, which handles the storage, version control and distribution of service and data model information, and (*ii*) sub-systems, which consist of an underlying service implementation, as well as a Local Model Repository (as part of the framework).

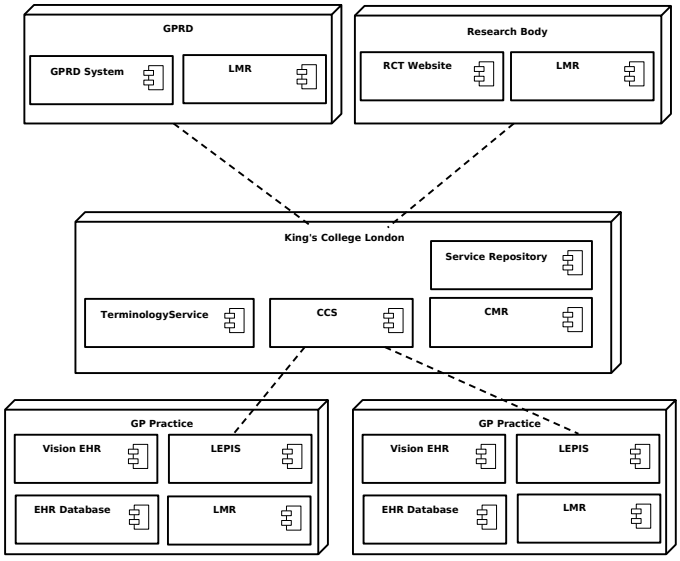


**Fig. 3.** Overview of Model-Driven Framework and ePCRN-IDEA's Recruitment System

A sub-system would first register itself with the Service Repository, stating which service interfaces it supports. Importantly, when services are registered, they would also be associated with the versioned data models they understand, thereby allowing other sub-systems to ascertain interoperability. This would be done be validating that all inter-connected sub-systems utilise matching service interfaces and data models, or, alternatively, that mapping functions exist to convert between any incompatible interfaces or data. After this, the sub-system would attempt to retrieve its required models; first through its Local Model Repository, before using the Central Model Repository if it is not locally available. Once the models have been retrieved, they can then be reified into a concrete form (e.g. Java objects). Service interface models would first be used to validate the compatibility of the sub-system, whilst data models would (usually) be used to generate the necessary code. If discrepancies existed between the various consumed and provided models (e.g. two sub-systems utilise different versions of the same service interface), the necessary mapping functions would

then also be acquired to translate the models. This would allow the system specification to be evolved and distributed progressively, alongside these mappings.

## 6 Conclusion and Future Work

This paper has investigated the potential issues and challenges when faced with integrating Systems of Systems (SoS). To study this, a model-oriented development project has been used as a case-study; this has detailed the ePCRN-IDEA recruitment system, which uses model-oriented engineering to better support data and service integration between a number of organisations. The key outcome of this study has been a set of issues that have been discovered during the development phase. These include (*i*) the complexity and cost of performing data mappings, (*ii*) the problems that can emerge during process changes within each organisation, (*iii*) the difficulties of handling model evolution within the system as a whole, and (*iv*) the risks of losing system-wide consistency during any of the above processes. To address these challenges, a conceptual architecture has been proposed that utilises a new infrastructure to extend model definition and control into post-deployment. The next stage in this work is therefore realising this design to deploy the next phase of ePCRN-IDEA.

## References


1. M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, Feb 1998.
2. T. Benson, *Principles of Health Interoperability HL7 and SNOMED*. Springer, 2009.
3. B. Smith and T. Tolman, "Can we talk? Public safety and the interoperability challenge," *National Institute of Justice Journal*, April 2000.
4. R. Lock and I. Sommerville, "Modelling and analysis of socio-technical system of systems," in *IEEE Intl. Conference on Engineering of Complex Computer Systems*, 2010.
5. A. Mark, "Modernising healthcare: Is the NPfIT for purpose?," *Journal of Information Technology*, vol. 22, Feb 2007.
6. A. Beugnard, J.-M. Jezequel, N. Plouzeau, and D. Watkins, "Making components contract aware," *Computer*, vol. 32, July 1999.
7. M. Yabandeh, N. Knezevic, D. Kostic, and V. Kuncak, "Crystalball: predicting and preventing inconsistencies in deployed distributed systems," in *USENIX Symposium on Networked Systems Design and Implementation*, 2009.
8. B. Biggs, "Ministry of defence architectural framework (MODAF)," *IEE Digest*, vol. 2005, Feb 2005.
9. A. Josey, *TOGAF Version 9: A Pocket Guide*. Van Haren Publishing, 2009.
10. "An insider's view to the HP universal CMDB." HP Technical White Paper, 2008.
11. S. A. Naqvi, R. Chitchyan, S. Zschaler, A. Rashid, and M. Südholt, "Cross-document dependency analysis for system-of-system integration," in *Foundations of Computer Software. Future Trends and Techniques for Development*, 2010.
12. S. Bullock and D. Cliff, "Complexity and emergent behaviour in ICT systems." HP Technical Report HPL-2004-187, 2004.