A Service Oriented Approach to Critical Infrastructure Modeling

Ebrahim Bagheri
Faculty of Computer Science,
University of New Brunswick
Fredericton, N.B., Canada
E.Bagheri@unb.ca

Ali A. Ghorbani
Faculty of Computer Science,
University of New Brunswick
Fredericton, N.B., Canada
Ghorbani@unb.ca

ABSTRACT

Due to their extraordinary behavior caused by their subtle and obscure interdependencies, and their vital role in the society, modeling the structure and performance of critical infrastructures is a challenging task. In this paper we investigate the possibility of applying different aspects of service oriented architecture to the problem of modeling the dynamic, heterogeneous, and loosely coupled service provisioning paradigm of the critical infrastructures. We intend to describe the opportunities of employing current state of the art SOA technology for modeling the dynamism in the behavior of a critical infrastructure. This model may later complement our previously designed UML-CI profile. The two models can create a solid framework for capturing both the dynamic and static parts of a critical infrastructure and can serve as an appropriate basis for modeling and further simulating the outcomes of a critical infrastructure operation under different conditions.

Categories and Subject Descriptors

*J.7 [**COMPUTERS IN OTHER SYSTEMS**]: Command and control,Consumer products,Industrial control,Military,Process control,Publishing,Real time*

General Terms

Design, Standardization, Verification

Keywords

Critical Infrastructure Modeling, Service Oriented Architecture and Design Methodology.

1. INTRODUCTION

Service centric analysis is a promising new design approach that focuses on the interface description and interaction models of computing entities, instead of concentrating on the intricacies of the internal design issues [1]. This attitude towards modeling and designing complex systems eases the modeling process and change management; however it may complicate later analysis and understanding [5]. Service centric approaches have received great attention due to their alignment with regular system processes.

Service oriented computing is a pattern that utilizes services as its fundamental elements [4, 16]. It encompasses the wide spectrum of service based models ranging from service oriented architecture [3] which is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains, to service oriented enterprises [9] which address all different layers of an enterprise through layered service provisioning.

From a design perspective, service oriented architecture provides diverse features that can assist in crafting a highly complex and dynamic system. The major drawback of such an approach is related to its preservation issues. As dynamic systems rapidly change with the progression of time, tracking system functionality becomes very hard or even impracticable in such architecture. For this reason it would be sensible to split the design skeleton of highly dynamic systems into two main segments. With this approach, the first segment will focus on designing the stationary elements of the system. Having designed the static structure in the first segment, it would then be both easier and more tractable to design service interfaces and their respective interaction models.

One of the very well known examples of highly complex systems is critical infrastructures. Critical infrastructures are networks of extremely complex systems that can be classified as socio-technical organisms with hidden consciousness [23]. They influence the normal lives of every citizen through providing them with their basic and most essential needs. Electricity as an example among the other prominent critical infrastructures has a very vital influence on the correct operation of the social and industrial aspects of every day life. It is well known that a disruption in the electricity infrastructure can disrupt the operation of the telecommunication networks. Internet as only one of the off springs of the telecommunication infrastructure would become nonfunctional due to this disruption. All other catastrophic snowball effects caused by this rather small failure can be easily realized.

As it can be inferred from the previous example, the complexity of the operation of critical infrastructures is due to their high connectivity and mutual interdependence. All of their functionalities, either the ones which are providing their main products, or the ones that have a supportive role are both provided through each others constituent systems. For example the required electricity for the operation of the data networks of the telecommunication infrastructure is provided through the generators of the electricity infrastructure. The functionality of a critical infrastructure is provided through its many heterogeneous systems. This allows us to classify these complex evolving organisms as complex adaptive systems [24]. Their dynamic nature and highly vibrant and real-time decision making process brings about an extremely complicated scenario for modeling and simulation.

One of the main features that has to be carefully examined while modeling a critical infrastructure is that although its different processes and decision making procedures are highly dynamic, but their core skeleton (which mostly constitutes its technical and mechanical elements) are to a great extent permanent and unchanging. The reason for this is the very expensive critical infrastructure constituent element construction costs. The high dynamism of such systems is therefore not

related to their re-construction, but is extremely dependent on business oriented decisions, process re-engineering and their corresponding directions. Therefore, it is logical to follow the bi-segmented approach introduced earlier in the paper towards modeling critical infrastructures. In the first phase UML class diagrams can be employed to create the firm static vision of the internal structure of each critical infrastructure. The main difficulty towards such a model in the absence of a metamodel is that different designers with diverse perspectives may end up designing dissimilar structures. For this reason we have tackled the problem from a model driven architecture perspective in our earlier studies and have designed a UML profile. The UML profile named UML-CI [23] provides the base and most essential constituent elements of a critical infrastructure which enhances the critical infrastructure design process by offering a set of unique metaclasses. Many different aspects of a critical infrastructure related to their static features can be modeled through this UML profile.

The main goal of modeling and simulating the behavior of different inter-related critical infrastructures is to elucidate their unforeseeable behavior caused be their intricate and obscure interdependencies. This goal would be most likely attained if both static and dynamic features of a critical infrastructure are precisely modeled.

From a static point of view and in the UML-CI profile, each *Infrastructure* is built from many different *Systems*, each of which perform certain predefined *Tasks*. Each task carries out some internal action on some of its parent system *Assets*. For example a nuclear power plant can only access and convert the nuclear fuel owned by its owners. The assets of a critical infrastructure can be classified into three different categories: *Non-Physical Asset, Physical Asset,* and *Cyber Asset.* Physical Assets themselves can be further categorized into *Transformable* and *Non-Transformable*. Fuel and water are two Transformable assets among a wide range of others as opposed to pipes, and roads which are examples of Non-Transformable assets.

To clarify the managerial and visibility issues of a critical infrastructure, S*take holder, Government, and Government territory* metaclasses have been further defined in the UML-CI profile. The government territory metaclass is composed of a collection of G*eographical bound* metaclasses that provides the basis for modeling visibility and operational access features on assets and tasks. Regulatory operations such as policy making operations have been incorporated in the profile through *Policy maker and Regulation* metaclasses. One of the major features of any critical infrastructure is the *Hazards* that it faces throughout the process of its operation. These hazards can be any *Threat, Risk or Vulnerability* that threatens the operation of an infrastructure. These hazards may well influence the operation of any task and hence endanger the operation of its respective system and infrastructure. Figure 1 shows a partial view of the abstract format of the symbolic critical infrastructure pattern that has been depicted to clarify the UML-CI profile structure. Interested readers are encouraged to visit [23] for a more detailed description of the UML-CI profile.

In this paper we will investigate the possibility of applying service oriented architecture to complete our bi-segmented structure through modeling the dynamic parts of a critical infrastructure as services. We believe that due to many common characteristics such as loosely coupling, heterogeneity and dynamic upgrade of constituent components, the operational aspects of a critical infrastructure can match with the basic principals underlying the service oriented architecture. Features such as dynamic asset provisioning, contract establishment between two infrastructures, policy enforcement are some of the dynamic features of critical infrastructures that can be modeled through the exploitation of services. Assets, tasks, systems, regulations, and even the infrastructure itself introduced in the UML-CI may take the form of services to make use of the benefits of service oriented architecture.

From the service provisioning perspective, infrastructures are currently pursuing a service oriented operation in which most of their functionality is broken up into many smaller constituent service provisioning units. Each of these elements is responsible for providing a set of services or assets. The electricity infrastructure for example, is composed of many constituent elements such as generators, SCADA systems, electricity transmission lines, and fuel provisioning services. Generators can be looked upon as services (that may contain many other smaller services themselves) that are responsible for catering required electric energy.

Although many of the high level services do not directly map onto a tangible real world service, but they can act as a mechanism to encapsulate different services under a single interface and represent a meta-service. This approach eases the possibility of creating critical infrastructure models as holonic systems [25].

As the high complication of applying a service oriented architecture to the problem of modeling critical infrastructures is anticipated, we will propose a four layered architecture (see Figure 2) that attempts to properly classify different infrastructure services into appropriate service abstraction layers as the first step.

The next section of the paper will give a brief explanation of our perception of service oriented architecture. Section 3 will review some of the available technologies related to the service oriented architecture in other applications such as web services. Based on this investigation we will then propose a four layered architecture for a service oriented critical infrastructure modeling framework in Section 4. The paper will then conclude in Section 5.

2. SERVICE ORIENTED ARCHITECTURE

Services are self describing, platform agnostic domains of control that support rapid, and low cost composition or provisioning of distributed applications and functionality [4]. The main differences between an object oriented and service oriented design refer to their different perspective on interaction models and design patterns [10]. In an object oriented approach, an object describes what it is regardless of the functionalities that it provides; however on the other hand, services tend to represent themselves as entities providing functionality obscuring their internal structure [1].

Service oriented architecture is fundamentally based on three main concepts: Visibility Scope, Interaction Model, and Effect [2]. A service should first of all be introduced (advertised) in the environment that it is supposed to operate in. The simplest form of service advertisement is achieved through its interface publication. In this way services will understand how to communicate with each other from a syntactical perspective. Publishing a service in the environment is not the only functionality needed. Discovery operations [22] are also required for services to find the best service matching their requirements. Advertisement and discovery create the basic operations required for service awareness. Besides awareness, a service should understand under what conditions other services are available and reachable. Having reached a conclusion as to which service is suitable to have interaction with, discovering the binding willingness of that service is the last condition that should be satisfied by the requestor. Therefore awareness, reachability and willingness are the most decisive factors in defining the visibility of a service from another service's point of view.

Interaction models define the information and behavioral schemas that should be conformed to while having a transaction with a service. A service information schema defines structural and ontological

structures that should be employed in the transaction. On the other hand, the behavioral schema focuses on the characterization of the set of actions that can be performed against the service and clarifies its temporal relationships through choreography and orchestration [19]. Every service will undoubtedly have some corresponding results based on the received inputs. The results are often explicit messages or output values; however a service can also alter a shared state instead of releasing an explicit response [21].

To view things from a higher standpoint, it can be stated that service oriented architecture is simply based on a "Publish-Discover-Bind" paradigm [7]. Services have an opaque nature by which their internal functionality is hidden from the requestor. This provides the basis for the collaboration of heterogeneous services that result in agility, adaptability, distribution, transparency, and much more flexibility [6, 11].

Apart from the very interesting notion of service orientation that eases the design through loosely coupled interfaces, the service oriented architecture itself does not provide direct solutions to many of the available sophistications that arise while exploiting such architecture. It is purely within the able hands of the designer to complement such architecture based on the target application. Policy management, contract establishment, transaction handling, failure detection and automatic recovery are some of the features that are not directly addressed within the service oriented paradigm [4].

Web services are the most well established and prevalent applications of the service oriented architecture. They have exploited the guidelines of SOA to create a distributed program execution model based on dynamic coupling built on top of the web infrastructure. Due to their role as one of the pioneers of exploring the service oriented architecture, there have been tremendous amounts of research carried out on their different aspects. Various standardization bodies have proposed several standards for a variety of web service requirements such as WS-Policy, Ws-Agreement, and WS-Transaction. In the next section we are after spotting the features that fall beyond the concept of service oriented architecture but are required for its functionality and have been addressed in other contexts such as web services. This browse will enable us to identify suitable features in other applications that can be either directly applied to our model of critical infrastructures or can serve as a guideline. We anticipate that due to the differences between critical infrastructures and other applications such as web services or grid services, many of these features should be modified to match our requirements. A very simple example of the differences between the services in a critical infrastructure and a web service is their address types. The address of a web service is defined as the hostname (IP address) and the related port number; however the address of the service in a critical infrastructure is related to a geographical coordination.

3. RELATED RESEARCH

There have been different approaches to model or simulate the behavior of a critical infrastructure [29, 30]. These attempts can be classified under either *global analysis* or *system analysis* [31]. Global analysis endeavors to model infrastructures through the employment of pure mathematical models using differential and algebraic differential equations. The second class exploits the use of different types of analysis under simulation conditions. None of the previous models have used service oriented architecture to create a model or simulation environment. In this section, we will review the most important advancements in the field of service oriented architecture that can be applied to critical infrastructure model. We will classify these developments and will provide a proper mapping of critical infrastructure services onto a service oriented architecture in the next section. The concepts visited in this section are an eclectic set selected from the wide range of features available in SOA research that are more closely related to the notion of infrastructures.

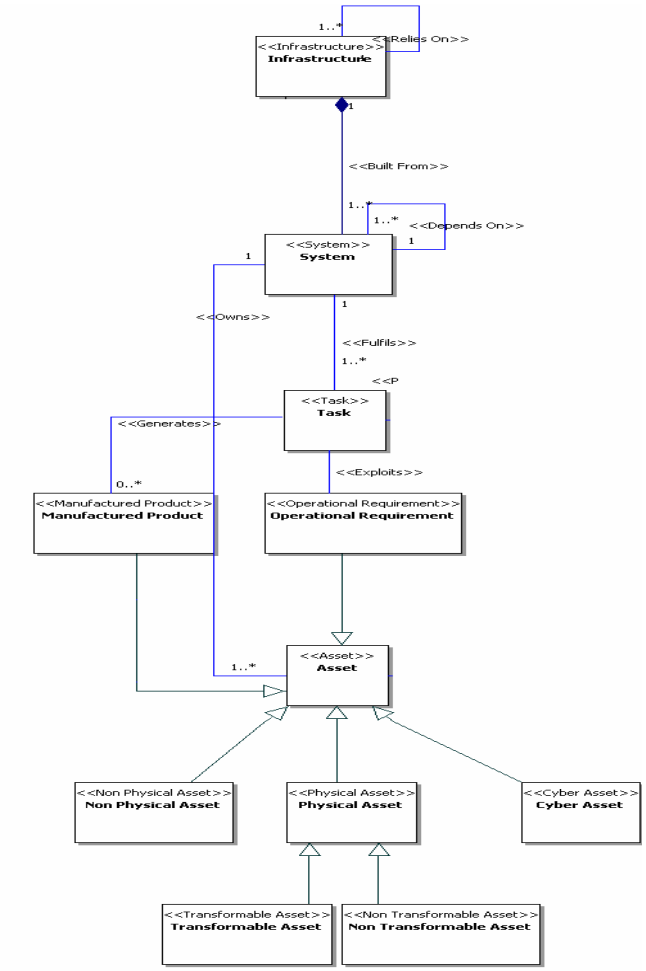

Figure 1: A Partial View of the Symbolic Critical Infrastructure Pattern based on UML-CI [23]

Service oriented architectural concepts can be basically divided into two base categories. The first set of concepts is related to the service itself. However the second class of these concepts is associated with the context in which the service is functioning [18]. Services can also be categorized based on their granularity. Each service may exploit other services functionality as a part of its operation. We name these services meta-services as opposed to atomic services which fully depend on their internal functionality for their operation.

3.1. Service Related Concepts

Concepts directly associated with services mainly focus on the specification of a service itself. Interface description is one of the major roles of the service. This is seriously required since it serves as the possibility for introducing a service and hence creating a chance for service interaction. Web services as the major technology currently using service oriented architecture make use of a standard description language called WSDL to describe their functionality and interfaces

[26]. It is not only the syntactical interfaces that should be defined but the semantic meaning of the interfaces should also be clarified through a domain specific ontology [17]. The ontology should cover objects, events, states, or anything that can be conceived or exchanged [20]. Through the exploitation of a stable ontology, semantic services can be created. This provides the basis for improved service discovery, reuse, simpler change management, and semi-automatic service composition [6, 13].

3.2.Context Related Concepts

From the context perspective, there are more issues to consider. Contract management, policy enforcement, awareness mechanisms, failure detection, automatic recovery, mediators, load balancing, and dynamic provisioning are some of the major concerns that need to be addressed.

3.2.1. Contracts

Contract establishment is a concept that requires a specific protocol definition such as the 3-way handshake used in network protocols [27]. Providing different services the opportunity to create a bilateral relationship with each other necessitates the existence of a contract monitoring system. Such monitoring element should be able to enforce contract specifications and at the same time resolve any type of conflict or dispute between the involved services.

In real world cases services announce their availability decorated by a specific SLA [13]. SLAs can be added to services and hence accompany any contract to allow service level agreement monitoring in the contract management module. However if the contract agreements are to some extent violated, it is not always the case that the involved parties will take legal actions against each other. There are always flexible toleration levels that should be observed. ContractLog is one of the available frameworks for representing contracts through different logical formalisms such as Deontic logic, Event calculus, Horn logic and ECA rules [12]. It also utilizes Abductive logic to add proactivity to its monitoring process. It provides a formal representation of contracts, separates contract rules from the service management application, and the contract rules can be automatically linked and executed.

Contracts should basically have a formal representation language that can be agreed upon between different services in the modeling environment [15]. The representation should also conform to the previously defined ontological definitions. Automatic contract writing-negotiation is another field for comprehensive research.

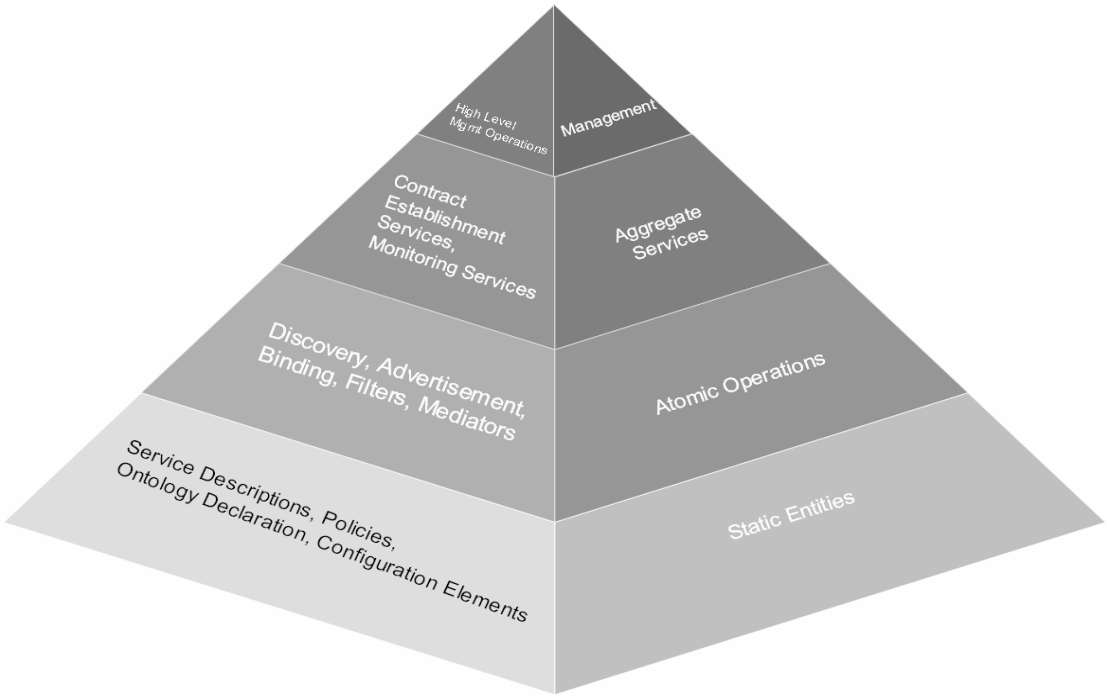

Figure 2: The Proposed Four Layered Architecture for a Service Oriented Critical Infrastructure Model

3.2.2. Policies

Each service has a set of regulations, governing its resource consumption/production or task accomplishment procedures. These regulations may be enforced for business support or quality assurance [14]. The regulations have been named policies in the service oriented architecture. Such policies can provide the basis for three different types of monitoring: Permission, Obligation, and Prohibition [12]. Permission based policies grant the proper rights for accessing a certain asset or performing a specific task by a person, group of people or a service. Obligations state that a certain task must be performed by a subject. Prohibition is a kind of restriction condition. Policies can be enforced to monitor tasks or assets from both visibility and access points of view. Access restrictions can be imposed on time, location or a group of people or services. Similar to contracts, policy assertion should also be accomplished through a policy specification language (PSL). The type of policy ascription models used in service oriented architecture can vary based on the different conditions. Service policy ascription can follow any of the One-to-One, Many-to-One, One-to-Many, and Many-to-Many models.

Policy enforcement itself can take two forms. It can be ascribed to a service (task) or asset as a decoration (in the form of some additional PSL based XML document). With this feature, every access or

manipulation of that service or asset will be checked for compliance against the given policy. The other approach defines the policy enforcement as an external service itself. When a service or asset is manipulated that service would be automatically invoked too. Policy can be enforced through filtering illegitimate messages. If the message violates the policies related to resource consumption or task manipulation, it should be filtered out.

One of the very complicated features of policy ascriptions is related to its distributed nature. As the policies are dispersed among different assets or services, it is not possible to detect contradicting policies at design time and hence runtime solutions should be sought.

WS-Policy is a standard for service decoration, proposed by an industrial coalition which attempts to define a framework for attaching attributes to services [10]. It is a flexible and extensible grammar for expressing the capabilities, requirements, and general characteristics of entities in an XML format used for web service policy specification.

As is the case for contracts, Policy violations should also be tolerated to a defined extent; however the penalties and punishments for violating policies should be clearly explicated in the policy specification document. Policies can have three major types: Post-Conditions, Pre-Conditions, and Invariants. Pre-Conditions and Post-Conditions are applied instantly before and after the execution of a service or access to an asset respectively. However invariants apply all the time and should always hold regardless of the time or location.

Policies and contracts are similar in their nature. Both concepts assert a type of constraint or limitation on the activity of two or more subjects. Their main difference is in the assertion models. A policy is created by one party, but a contract should be mutually approved.

One of the major features that should be incorporated into the policy and contract description languages is the feature for specifying the third party which will be in charge of policy violation detection (policy enforcement) or dispute resolution. The third party itself can be a service that is operating on a standalone basis.

Platform for Privacy Preferences (P3P) is a proposal for describing a policy language for expressing the privacy rules adhered to by an organization in a machine readable and human understandable format [14]. P3P can also be used as a successful blueprint for designing the required policy specification language. The policy specification language should also concentrate on devising a clear policy lifecycle consisting of its definition, maintenance, application, versioning and control facilities.

3.3.Service Granularity

Any of the operations related to the service or the service context can be either atomic or hybrid. Atomic services have no external dependency on other services for their operation; however meta-services (hybrid) are composed of many other atomic (or meta-) services whose aggregate behavior forms the final functionality of the meta-service.

3.3.1. Atomic Services

From the perspective of atomic services related to the context of the service oriented architecture, the most important services are awareness mechanisms, filters and mediators. Awareness mechanisms are achieved through pull, push or subscribe-notification methods. In the pull mechanism, services should always query service repositories for the availability of new services, however in a push mechanism a service is notified automatically by the repository. In a subscribe-notification method, services can announce their willingness to receive notifications if a specific type of service is registered at a repository.

Mediators are the other aspect of a service oriented platform that enable interoperability between services with heterogeneous message formats [17]. Mediators act as translators that convert different message types and allow an even more heterogeneous and loosely coupled service provisioning environment.

3.3.2. Meta-Services

Meta-services are created based on the composition of several other atomic services. Functionalities among the context related services such as load balancing, dynamic provisioning, failure detection, automated recovery, contract management, and policy enforcement are some of these services.

Load balancing refers to the ability of a service oriented platform to evenly spread the requests for a specific service to all available services of that type. In infrastructure management, load balancing is also a very important issue. If a provider of a specific task is overwhelmed with the amount of pressure, the requests can be evenly distributed to other service providers. Dynamic provisioning is one of the solutions that complements load balancing. This model is not always possible since it might not be possible to create systems or instances of a task in a real world scenario. However in a web service context real time dynamic provisioning is feasible.

Failure detection is one of the other demanding tasks [5]. The first major obstacle for reasoning about failure is failure tracking. This is due to the dynamic real time coupling nature of services. If an error occurs the scenario cannot be re-pictured since couplings have been dynamically made and in a new situation the service selection model might not be consistent with the previous one, and hence reasoning about a failure that has previously occurred, will be irrelevant. A comprehensive logging feature would be required to track previous activity and allow a complete understanding of previous couplings. The other problem with failure detection goes back to the opaque character of a service. Since the internal actions of a service is hidden from outside services, and each service might itself be a meta-service consisting of other services, it would logically be impossible to mark out the service coupling procedure. In this sense, each service is unable to track occurring failures in its peers. Specific failure detection and tracking services would be required to reveal and spot a malfunction.

4. MAPPING CI SERVICES ONTO SOA

The proposed service oriented architecture for a typical critical infrastructure has been depicted in Figure 3. The architecture consists of four major layers:

- Static Entities
- Automatic Operations
- Aggregate Services
- Management

The first layer consists of all static entities that are more or less descriptive elements or fine grained service related concepts. Fine grained services along with their description documents, the ontology declaration, configuration elements, and policy specifications are all placed in this layer. Going back to the electricity infrastructure, electricity transmission lines and nuclear power plants used for electricity transmission and electricity production respectively are a few of the static entities that can be placed in the first layer(along with their descriptions for example maximum line capacity, and line-miles). Since transmission lines and nuclear power plants are examples of services in a critical infrastructure context, they can be compared with a web

service. As web services publish their description files through WSDL, these entities also need to publish their descriptions.

Policies that govern the electricity transmission over a certain transmission line such as their maximum capacity in specific geographical area are also placed in this layer. This can be compared to a WS-Policy expression that states that a specific web service only accepts X.509 certificates and Kerberos tickets.

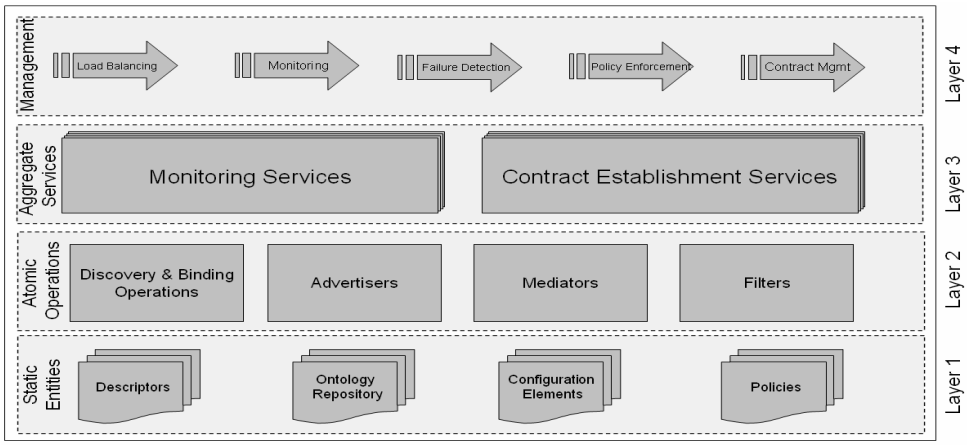

Figure 3: The Architecture consists of Static Entities, Atomic Operations, Aggregate Operations, and Management layers. Each Layer is Responsible for Provisioning the Higher Level Layer with Suitable Services.

The atomic operation layer comprehends a wide range of different services related to the context that provide the basic required operations for manipulating a layer one service. The major feature of the available services in this layer is their atomic nature. They are atomic in the sense that their functionality is not provided through delegation. Service discovery, mutual service selection and binding, advertisers, mediators, and filters are the most important services of this layer.

The third layer, Aggregate Services, encapsulates the services provided in the atomic operations layer into two major functional services: Monitoring Services, and Contract Establishment Services. The management layer finally creates proper interfaces along with the required logic to govern the activities of a sample critical infrastructure.

Let's consider a very small subset of the electricity infrastructure as an example. Electricity generators are undoubtedly one of the most important productive elements of the electricity infrastructure. These entities, the policies that control them and other descriptive elements along with other similar services or resources encapsulated into services will be placed in the first layer of the proposed architecture.

The second layer of the architecture is responsible for providing proper context related services that provide the higher level services with appropriate functionality such as discovery operations. The discovery operation in an electricity infrastructure may be in charge of finding the most suitable supplier of fuel for the input of electricity generators. The discovery operation will be hence in charge of contacting different fuel provisioning services to find the most appropriate service with the optimal SLA.

Monitoring the operation of the services in the first and second layers is handled in third layer. An infrastructure is also able to establish electricity maintenance contracts with other infrastructures or even end consumers at the third layer; however contract management would be applied from the last layer. For example the infrastructure stake holders may decide to sign an electricity provisioning contract with a multinational food company. This decision will be made in the fourth layer, but the actual contract is established in the third layer. To more clearly define the difference between the third and the fourth layers we consider a simple case of the SCADA systems of both electricity and the telecommunication infrastructure. In the case of the electricity infrastructure SCADA system, the system will establish an electricity provisioning contract with one of the power plants of the same infrastructure. This can be handled in the third layer. However if the same kind of SCADA system but that of the telecommunication infrastructure requires electricity for the operation of its services, it should create a higher level contract at the fourth layer with the electricity infrastructure. The contract should first be established between the two infrastructures and then the actual service provisioning process takes place at the third layer.

It is also possible to apply load balancing controls on lower layer services such as nuclear power plants from the fourth layer. With this feature, if any of the power plants are overloaded, some of the electricity demand on that specific power plant can be transferred onto some of the other possible power plants or electricity generators.

Many of the low level services placed in the Static Entities layer can themselves form a four layered structure. A nuclear power plant for example, has a four layered architecture itself. Entities such as Reactors, Steam Generators, Pumps, Turbines, Cooling Towers, Control Rods and other static entities that form a nuclear power plant are placed in its first layer. The base operations required for the functionality of these entities are located in the second layer. The third and final layers are responsible for the correct and normal operation of the nuclear power plant, enforcing policies on the nuclear power plant services and managing any possible contracts.

Figure 4 clarifies the interaction of two critical infrastructures in the proposed four layered service oriented architecture. The process follows the Open Systems Interconnection (OSI) standard [28]. To establish a contract between two different infrastructures (electricity and food) a high level, conceptual binding should occur at the fourth layer. All of the high level decisions in a critical infrastructure are made in the fourth layer (which acts like the brain of the system) because of the high level vision that is available at this layer. The contract will be managed by the contract management service that provides a façade for the contract establishment services in the third layer. In order to find out if the contract should be confirmed, the service discovery operations and filters in the second layer would be negotiated. These operations will then query the services that have encapsulated the description files for

the right information in layer one. It is likely in this process that service descriptions would be checked to see if any suitable service is available, policies are reviewed so that no violation exists, and finally previous contracts are visited to check that the new contract will not violate the previous ones. Other high level services in the fourth layer would have a similar procedure. The specification of each individual high level service process is out of the scope of this paper and hence we do not go into further details.

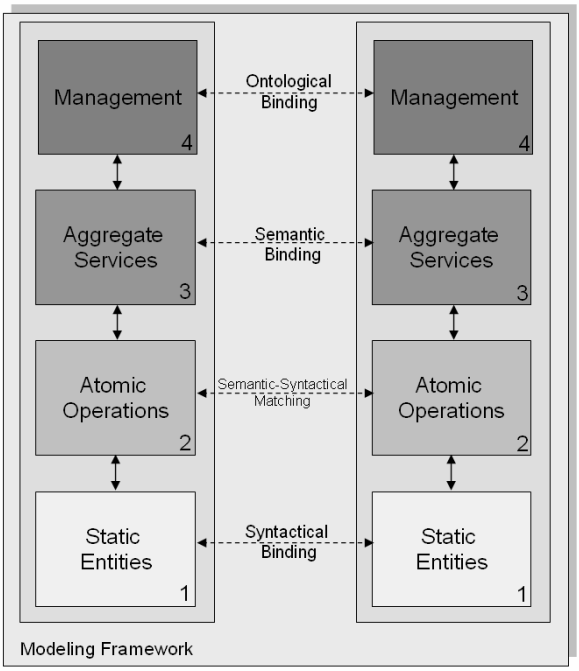

Figure 4: The Interaction Model Consists of Four Different Conceptual levels

The important point that should be noted is that in our model, infrastructures establish a very high level conceptual binding at the fourth layer, although most of the interactions between low level services are grounded on syntactical operations. We therefore name the different levels of binding as: Conceptual Binding, Semantic Binding, Semantic-Syntactical Matching, and Syntactical Binding respectively.

Since infrastructure services are mainly centered on four core actors namely Asset, Task, System, and Infrastructure, in both the UML-CI and the proposed four layered service oriented architecture, we have created a UML use case model to show how different services are spread out between these four elements which are shown in Figure 5. Assets are the most fundamental elements of an infrastructure that correspond to the static entities layer in our architecture. The operation of any infrastructure is heavily reliant on the presence of these entities. Roads for example in the transport infrastructure are an example of physical assets. The behavior of these assets is encapsulated in services. For example a road has a degree of erosion which is reflected through the self adjustment of the asset encapsulated as a service.

It is also worth noting that not only infrastructure assets are placed in this layer, but more complex entities such as a generator can be placed in this layer too. The generator itself will have a four layered architecture similar to the one introduced in Figure 3. In the generator case, elements such as turbines, steam engines or even the fuel that the generator uses are examples of the generator assets and are represented by appropriate services.

Tasks are responsible for performing a very specific mission. For example a task may be in charge of finding the most suitable route for transferring a postal packet in the post infrastructure. These tasks are the services that are placed in the second layer of the proposed architecture. Systems as the other major component of an infrastructure are complex actors that encapsulate various tasks and coordinate the actions of these tasks. Air transport in the transport infrastructure is an example of a system. It encapsulates the activities of many lower level tasks and provides the other systems of the same infrastructure or other infrastructures a set of high level services. An infrastructure is finally shaped by the aggregation of different systems. The transport infrastructure for example is developed by the composition of air transport, naval transport and land transport.

Although the four layered architecture provides a complementing dynamic model for the previously devised static UML-CI profile, but still the granularity of the services that will be further incorporated into each layer to satisfy the critical infrastructure related operations should be more specifically studied in our future research.

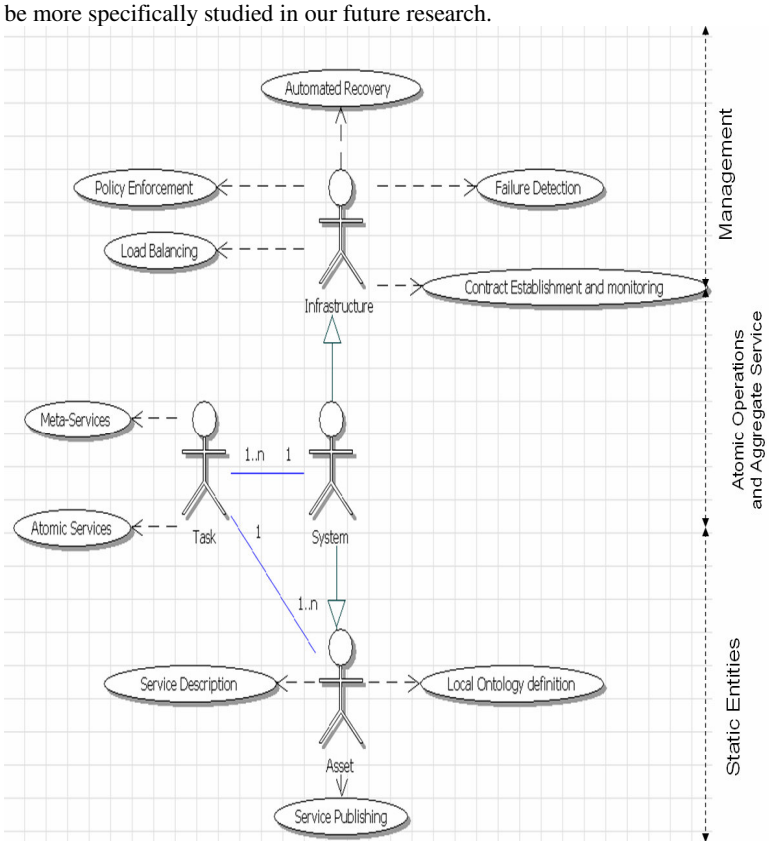

Figure 5: The UML Use Case Model Depicting the Actors and Their Responsibilities in a Sample SOA-CI Model

5. CONCLUSION

Critical infrastructures are networks of independent, mostly privately owned, man made systems and processes that function collaboratively and synergistically to produce and distribute a continuous flow of essential goods. Although their internal structure is independent of any other infrastructure but they have many subtle and obscure interdependencies. A very small failure in one system may ripple through to the others causing immeasurable damages. We are after creating a critical infrastructure modeling and simulation environment in our long term project. UML-CI [23] was designed in the first step to

capture the base and structural elements of an infrastructure. In this paper we have further completed our design by proposing a four layered service oriented model. We believe that the UML-CI profile along with the service oriented model described in this paper can create a suitable framework for modeling an infrastructure with its dynamic interdependencies. We intend to further refine each layer of our proposed architecture and explicitly specify how the services are going to interact in each of the layers in our future research.

6. REFERENCES

[1] Jones, S.: Towards an Acceptable Definition of Service. In: IEEE Software. IEEE Computer Society, 2005.

[2] Reference Model for Service Oriented Architecture, Committee Draft 1.0, OASIS group, 2006.

[3] A Methodology For Service Oriented Architecture, OASIS Draft, OASIS Group, 2005.

[4] Papazoglou, M. P., Service-Oriented Computing: Concepts, Characteristics and Directions, 4th International Conference on Web Information Systems Engineering (WISE'03) , Rome, Italy, 2003.

[5] Gold, N., Knight, C., Mohan, A., Munro, M, Understanding Service-Oriented Software, IEEE Software, vol. 21, no. 2, pp. 71-77, Mar/Apr, 2004.

[6] Stal, M., Using Architectural Patterns and Blueprints for Service-Oriented Architecture, Software, IEEE, vol.23, no.2pp. 54- 61, March-April 2006.

[7] Jiang, Y., Xing, C., he, W., Yang, J., On Procedure Strategy of Constructing SOA's Modeling Language, *sose*, pp. 79-84, IEEE International Workshop on Service-Oriented System Engineering (SOSE'05), 2005.

[8] Wong-Bushby, I., Egan, R., Isaacson, C., A Case Study in SOA and Re-architecture at Company ABC, Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) Track 8, 2006.

[9] Tsai, W.T., Service-Oriented System Engineering: A New Paradigm, IEEE International Workshop on Service-Oriented System Engineering (SOSE'05), 2005.

[10] Baresi, L., Heckel, R., Thöne, S., and Varró, D., Modeling and validation of service-oriented architectures: application vs. style. SIGSOFT Softw. Eng. Notes 28, 2003.

[11] Jammes, F.; Smit, H., Service-Oriented Paradigms in Industrial Automation, *Industrial Informatics, IEEE Transactions on* , vol.1, no.1pp. 62- 70, Feb. 2005.

[12] Paschke, A., Bichler, M., SLA representation, management and enforcement, e-Technology, e-Commerce and e-Service, EEE '05. , 2005.

[13] Furmento, N., Hau, J., Lee, W., Newhouse, S., Darlington, J., Implementations of a Service-Oriented Architecture on Top of Jini, JXTA and OGSI, Lecture Notes in Computer Science, Volume 3165, 2004.

[14] Gleason, T., Minder, K., Pavlik, G. Policy Management and Web Services, Proc. at Policy Management for the Web Workshop at the IWWW Conference, Japan, 2005.

[15] Rouached, M., Perrin, O., and Godart, C., A Contract Layered Architecture for regulating cross-organisational Business Processes, Third International Conference on Business Process Management, Nancy, 2005.

[16] Tsai, W. T., Paul, R., Bayne, J., The Impact of SOA Policy-Based C2 Interoperation and Computing, Proceedings of the 10th International Command and Control, 2005.

[17] Duke, A., Davies, J., and Richardson, M., Enabling a scalable service-oriented architecture with semantic Web Services. BT Technology Journal 23, 3, 2005.

[18] Gu, T., Pung, H. K., and Zhang, D. Q., A service-oriented middleware for building context-aware services. J. Netw. Comput. Appl. 28, 1, 2005.

[19] Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C., WSMX - A Semantic Service-Oriented Architecture, International Conference on Web Services (ICWS 2005), 12-15 July, 2005.

[20] Vetere, G., Lenzerini, M., Models for semantic interoperability in service-oriented architectures, IBM Systems Journal, 2005.

[21] Cox, E., Kereger, H., Management of the service-oriented-architecture life cycle, IBM Systems Journal, 2005.

[22] Bagheri, E., Naghibzadeh, M., Kahani, M., Ensan, F.,. A Novel Resource Advertisement and Discovery Model for Ubiquitous Computing Environments using Mobile Agents, Proceedings of the 10th IEEE Tencon-2005, Melbourne, Australia, November 21-24,2005.

[23] Bagheri, E., Ghorbani, A., Towards an MDA-Oriented UML Profile for Critical Infrastructure Modeling, submitted to PST'06.

[24] Rinaldi, S.M., Peerenboom, J.P., Kelly, T.K., Identifying, understanding and analyzing critical infrastructure dependencies, IEEE Control Systems Magazine, vol. 21, no 6, pp. 11-25, 2001.

[25] Fletcher, M., Garcia-Herreros, E., Christensen, J.H., Deen, S.m., Mittmann, R., An Open Architecture for Holonic Cooperation and Autonomy, 11th International Workshop on Database and Expert Systems Applications (DEXA'00), 2000.

[26] www.w3.org/TR/wsdl.

[27] Cohen, E., Kaplan, H. and Oldham, J., Managing TCP Connections under Persistent HTTP, Proceedings of the Eighth International World Wide Web Conference, 1999.

[28] Zimmermann, H., OSI Reference model - The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communications, COM-28(4):425–432, April 1980.

[29] Basu, N., Pryor, R. J., Quint, T., Arnold, T., Aspen: A Microsimulation Model of the Economy, SAND Report, 1996.

[30] Panzieri S., Setola R., Ulivi G.: An Approach to Model Complex Interdependent Infrastructures, 16th IFAC World Congress 2005, Praga, 4-8 Luglio, 2005.

[31] Panzieri, S., Setola, R., Ulivi, G., An Approach for Modeling Heterogeneous and Interdependent Critical Infrastructures, Submitted to IEEE System, Man and Cybernetics, 2006.