Semantic Management of Enterprise Integration Patterns: A Use Case in Smart Grids

Om P. Patri*, Anand V. Panangadan†, Vikrambhai S. Sorathia‡, Viktor K. Prasanna†
*Department of Computer Science, University of Southern California, Los Angeles, CA
patri@usc.edu
†Ming-Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA
{anandvp, prasanna}@usc.edu
‡Kensemble Tech Labs LLP, Gandhinagar, India
vsorathia@gmail.com

***Abstract*—Enterprise Integration Patterns are a set of design patterns for linking multiple systems using asynchronous messaging interfaces. This approach to system integration is increasingly popular due to its relatively simple loose coupling requirement. Implementations of these patterns are available in current integration frameworks but these are not semantic in nature. This paper introduces the concept of automatic management of messaging resources in an integration application via the use of a semantic representation of the Enterprise Integration Patterns. We have developed semantic representations of some of the commonly used integration patterns, which include a description of the expected resource requirements for each pattern. We then demonstrate this approach by considering the design of an application to connect mobile customers to Smart Power Grid companies (for the purpose of near real-time regulation of electricity usage). We illustrate potential savings in messaging resources and automatic lifecycle management using real-world sensor data collected in a Smart Grid project.**

I. Introduction

Many real-world applications rely on largely independent systems working together to achieve their full functionality. As an increasing number of systems interface with other systems and end users, the scale of the system integration task has grown tremendously. While careful analysis of system interfaces is possible for small-scale systems, this becomes increasingly infeasible for enterprise-scale projects (due to the high coordination needed). Recent analysis of integration efforts in the enterprise shows that approximately 70% of integration projects fail due to management issues rather than any fundamental technical issues[1].

A set of integration design patterns, collectively known as "Enterprise Integration Patterns" (EIPs), have been proposed by Hohpe and Woolf [1] that use asynchronous messaging as the predominant framework for system integration. The chief advantage of an asynchronous message-oriented interface for integration is that it requires only a loose coupling between the systems. Thus, relatively few assumptions need to be made regarding the functional capabilities and implementations of each system. The EIPs proposed are a well-defined set of typical integration scenarios and the functional requirements of a message-oriented interface for each of these cases. These integration patterns have become popular and are natively supported by commercial as well as open-source integration frameworks and enterprise service bus (ESB) products such as Apache Camel, Mule ESB[2], and WSO2 Carbon[3].

Current implementations of EIPs are not semantic. In typical usage, an application would make use of multiple integration patterns connected together. We believe that a semantic wrapper around each of these integration patterns that specifies the message requirements for each pattern would (1) ease the design of a complex integrated system (by formalizing the representation of the integration design), and (2) enable automatic management of the underlying message passing resources. Specifically, if the systems to be integrated are event-driven [2], then a semantic implementation of the asynchronous interfaces would enable the systems to automatically manage the lifecycle of the underlying message channels.

In this paper, we introduce the concept of semantic management of messaging interfaces for system integration. In our work, we selected a few common EIPs and designed an ontology to represent these selected patterns. This ontology includes the most relevant messaging properties for each pattern. We then demonstrate the value of this approach by showing how it can be applied to a particular application linking mobile customers to the demand-response outputs of smart electricity grid operators. The primary contributions of this paper are 1) introduction of a semantic representation of commonly used integration patterns, 2) the concept of messaging resource management within the integration (semantic) middleware, and 3) demonstration of its potential by showing how it can be used in a real-world application.

II. Motivational Scenario

Traditional power grids are now moving to smart grids [3] for more efficient, reliable and sustainable production and distribution of electricity. Huge amounts of data are generated by the swarm of sensors in buildings to record electricity consumption. The data are integrated and analyzed to implement dynamic scaling of resources or alternative pricing may be

[1] www.ebizq.net/topics/int_sbp/features/3463.html

[2] http://mulesoft.org

[3] http://wso2.com/products/carbon/

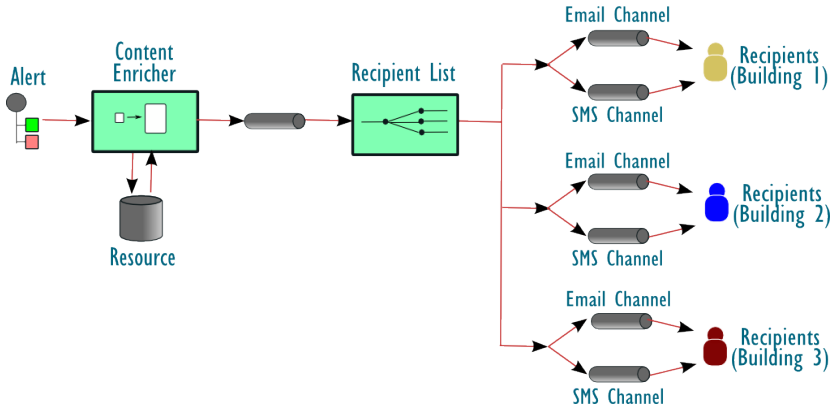

Fig. 1. Demand-response scenario in Smart Grid

implemented. In smart energy grids, this is known as *demand-response*. Demand-response further requires communicating with multiple data sources, detecting events and anomalies and evaluation of possible responses in realtime. As listed by Wagner et al. [4], supporting diverse participants, using a flexible data schema, and performing complex event processing are all open challenge areas for Smart Grid research. Semantic representations of concepts and middleware used in Smart Grid systems can enable these requirements through a generic integration architecture.

Using EIPs from Hohpe and Woolf [1], we consider a typical demand-response scenario for the Smart Grid in Figure 1. The figure shows an alert message, usually generated by an event stream processing system in response to a certain event, such as high electricity consumption in a building. Once the alert is generated, it needs to be enriched with contextual information as well as its intended recipients (e.g. the occupants of the building). When this information is obtained, the alert is sent over an appropriate message channel (such as email or SMS) to the recipients. The message content of the alert may be to suggest a reduction in usage or send information about excessive charges, if needed. The challenge then is to manage the creation and deletion of these message channels, and to decide between types of channels (email or SMS) in response to failed messages. Traditionally, this functionality would be explicitly programmed when creating the integration application. We propose that this functionality be included within the implementation of the EIPs and expose this capability using Semantic Web standards.

III. Related Work

Integration Patterns: Enterprise integration patterns [1] have been used for various integration architectures to connect messaging-based systems and perform enterprise application integration [5]. Umapathy et al. [6] integrate EIPs with web services to build conversation policies for enterprise integration tasks. DSL Route [7] implements EIPs for building a domain specific language to improve messaging. Chen et al. [8] provide an overview of more enterprise integration methods and Vernadat [9] focuses on laying out principles for interoperability in enterprise systems. Such interoperability can be achieved by using semantic and ontology-based approaches. Design patterns other than EIPs have been proposed for integration of disparate systems. Not all of these patterns are restricted to message-oriented architectures.

Semantic Middleware: Semantic web methods have been used in several enterprise integration approaches, such as semantic business process management [10] and semantic middleware for the internet of things [11]. The semantic representation for EIPs proposed by us is not stand-alone, and the power of semantic web and linked data enables us to find other ontologies which can be integrated with the proposed approach to provide greater value. Such ontologies include the semantic sensor network ontology and the enterprise ontology [12]. The sensor network ontology can be used to represent sensor data measurements, such as those from smart meters, and the Enterprise ontology can represent organizations, people, groups and various other associated enterprise components. Even though there are other related efforts using EIPs and semantic middleware there is no existing approach which brings them together or provides a generic semantic representation of EIPs for use in ontology-based approaches. It is precisely this sort of representation that we aimed at providing in this work.

Smart Grids: An overview of challenges and opportunities for using semantics in the Smart Grid is provided by [4]. A model-based semantic method is used for energy management and distribution in the Smart Grid by [13]. Information integration approaches for the Smart Grid are proposed in [14] through semantic complex event processing and data mining over streams. A cloud-based semantic software architecture for the Smart Grid is proposed by Simmhan et al. [15] to perform big data analytics, demand-response forecasting and secure, storage of data on the cloud. However, EIPs are not investigated in any of the Smart Grid approaches.

IV. Proposed Approach

Our approach assumes a semantic workflow for both the data being exchanged between systems and for the implementation of the messaging interface between the systems. Figure 2

shows the main components in the proposed approach. If the data that is to be exchanged between the systems is not in semantic format, it is first transformed into some appropriate semantic representation, such as triples in RDF[4]. Events are then detected from this data stream, typically using a (semantic) event processing engine [16]. The system architect is then expected to integrate the systems with appropriate integration patterns using a semantic implementation of the integration middleware. The detected events are inputs to the integration middleware. Each integration pattern implemented in the middleware then creates and destroys its required messaging resources (channels, filters, routers, etc.) in response to the event stream.

In our semantic representation of the integration patterns, each integration pattern is an OWL class and the messaging resources are properties of this class representing the different types of messaging components used in the patterns. Not only the EIPs, but also the data records coming from the sensors themselves can be represented semantically. The semantic sensor network ontology [5] can be used to effectively represent such data records and observations. The ontology of EIPs is described in detail in the following subsection.

A. Enterprise Integration Patterns Ontology

We represent each EIP as an OWL class and refer to the ontology of all patterns as the enterprise integration patterns ontology (represented by the prefix `eip:`). Each EIP (class) has its own set of properties linking it to other classes and specifying its messaging resource requirements, as described by Hohpe and Woolf [1]. Three example patterns are described here with their semantic representation.

An *aggregator* pattern (`eip:Aggregator`) is shown in Figure 3. Based on the definition in [1], an aggregator is used to combine related messages to output one aggregated message. As shown in the figure, aggregator is a type of `eip:EI_Pattern`, the class of all integration patterns. `eip:Aggregator` has two properties specifying its input and output messaging resource requirements: (i) `eip:givesOutputMsg`, which maps to `eip:Message`, provides the aggregated output message and has a one-one property (i.e. each aggregator gives exactly one output message), and (ii) `eip:takesInputMsg` which also has `eip:Message` as its range but is a one-many mapping (i.e. each aggregator takes multiple input messages). The `eip:Message` class, in turn, has object-type properties specifying the message content, sender and receiver as well as a data-type property recording the timestamp of the message.

The *dead letter channel* pattern (`eip:Dead_Letter_Channel`), which is a more complex pattern, is shown in Figure 4. The dead letter channel pattern is a type of message channel with the additional restriction that all messages in this channel must be dead messages (`eip:Dead_Message`).

[4]http://www.w3.org/RDF/
[5]http://purl.oclc.org/NET/ssnx/ssn

Figure 5 shows a composite pattern - the *dynamic router* (`eip:Dynamic_Router`) which is composed of a message router (`eip:Message_Router`) and an external rule-base. The rule-base (`eip:Rulebase`) is a type of resource (`eip:Resource`) which other patterns may also use and is a sibling class of `eip:EI_Pattern`. Similarly, other EIPs are implemented in the ontology following their description in [1] as closely as possible.

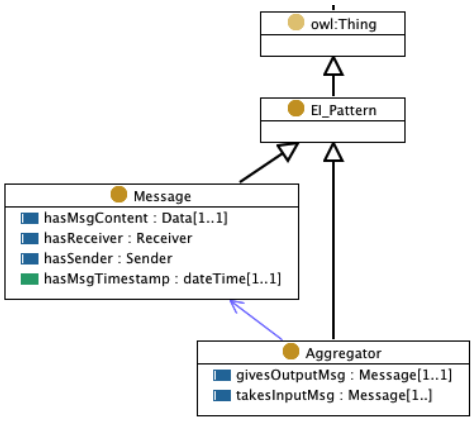

Fig. 3. Aggregator

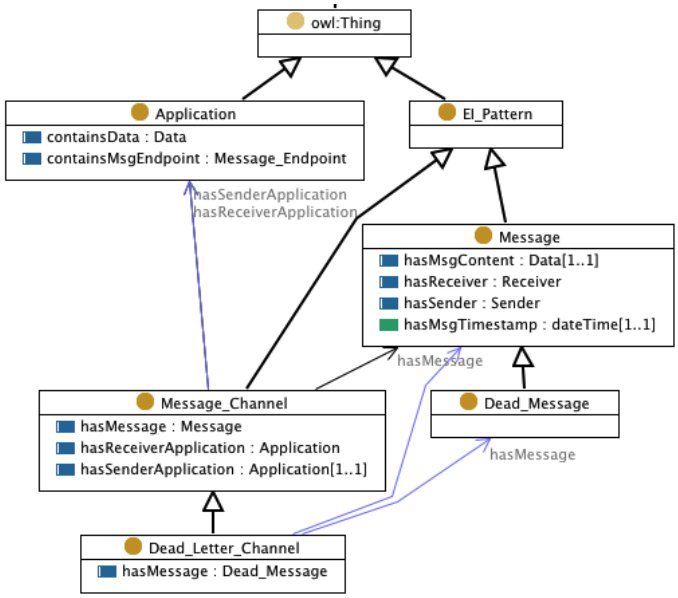

Fig. 4. Dead Letter Channel

V. Evaluation and Discussion

In Section 2, we described a motivational use-case of demand-response in the Smart Grid. In this section, we describe how semantic EIP can be applied to this scenario to reduce its messaging resources.

In the original integration method in Figure 1, we observe three buildings whose power consumption is monitored and

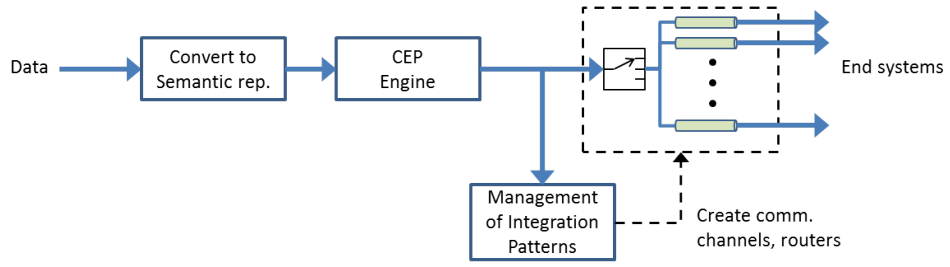

Fig. 2. Overview of the proposed approach

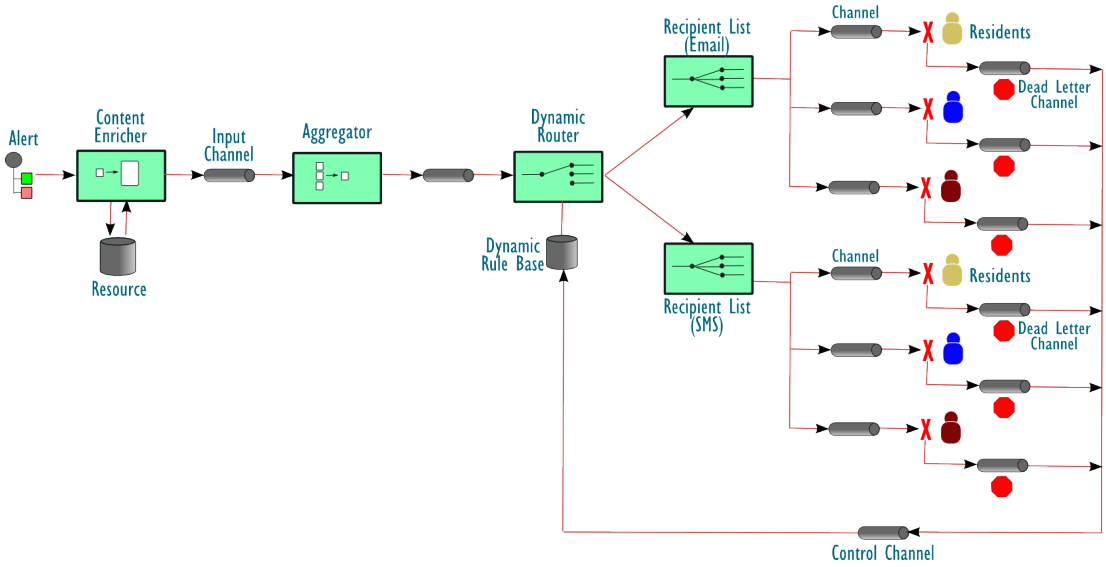

Fig. 6. Smart Grid demand-response scenario after optimizations

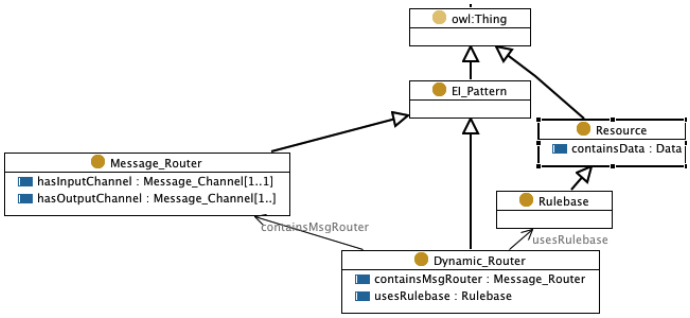

Fig. 5. Dynamic Router

their recipients notified in case the consumption is high. The recipients for each building include any person who might be associated with consuming electricity in that building. Since it may not be possible to track individuals and their locations, it is useful to include all possible persons who are expected to be present in the building at any time. For instance, in a classroom, this would include everyone registered for the class whether they show up for the class or not. In a Smart Grid scenario such as the USC campus micro grid, on which we base our evaluation, a key observation is that several recipients are common to two or more buildings. The same student may reside in a dormitory building, attend lectures in a lecture hall (in a separate building) and work at a third building, all on the same day. As per the scenario in Figure 1, such a student would receive three alerts simultaneously if all three buildings have a case of high electricity consumption on a particular day. Further, each individual may be contacted over multiple messaging channels (such as SMS or email) further increasing the number of messages in the system. If we include the number of message re-transmissions due to unavailable subscribers or faulty email/SMS addresses, the total number of alerts is even higher. There is thus a need for efficient message filtering and aggregation so that alert messages are not duplicated with the growth in number of buildings and subscribers.

To optimize the original scenario, we start by adding an aggregator to the input message channel after its content has been enriched with the information about the destination recipient. The aggregator pattern can efficiently combine multiple messages routed to the same recipient and provide exactly one output message for each recipient, irrespective of the number of buildings the recipient is associated with. Instead of

a separate alert for each building, the user would now receive exactly one message with information about all the buildings he/she is associated with.

Next, a dynamic router (which is composed of a message router and a dynamic rule base) is added to decide (dynamically) through which messaging channel (e.g. email or SMS) the alert should be delivered. The content or destination of the message may also be modified based on other rules from the rule base. To optimize further, we propose to keep track of which recipients are not actively receiving messages (for instance, due to incorrect email/SMS addresses). This information can be stored in the dynamic rule base associated with the dynamic router and can be used to make future decisions about sending messages to those subscribers (e.g., remove them from the recipient list, or retry using alternative messaging channels). The resulting optimized scenario in Figure 6 eliminates redundant messages from the original scenario, thus saving messaging resources.

We next demonstrate the potential savings in messages achieved by the use of EIPs in this scenario. To quantify such savings, we compare the number of alert messages sent.

A. Dataset

We use power consumption data from campus buildings in the University of Southern California (USC) campus micro grid. This data is recorded through smart meters located in each campus building. The dataset contains energy consumption data recorded at 15 minute intervals for approximately three years (from January 1, 2008 to November 21, 2010). We aggregate the 15 minute data for each building to obtain a cumulative daily total of power consumption and use this value in our simulations. To demonstrate our proposed approach in various scenarios, we choose three buildings of different categories for analysis. One of the buildings is a dormitory/residential building, the second building has laboratories/lecture rooms, and the third has academic/administrative offices. Daily power consumption for each of these three buildings is shown in Figure 7.

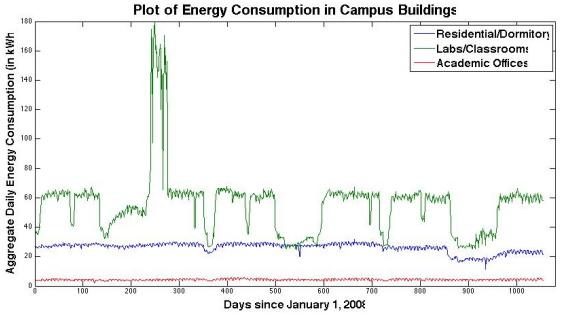

Fig. 7. Energy consumption plots for three buildings in the campus

B. Evaluation Methodology

For our evaluation setup, we use OpenRDF Sesame[6] as our triple store, Python for data processing and scripting, and SPARQL as the semantic query language.

The generation of alerts is typically done by an event processing engine which processes the incoming sensor data streams and detects interesting events, which are linked with alerts. A common alert is to find out whether energy consumption for a day exceeds a certain threshold obtained from historical data, such as a moving average. In our scenario, we implemented a simple query to generate an alert for a particular day when the aggregate daily power consumption for a building on that day exceeds the aggregate power consumption for the same building on the same date in the previous year. The analysis was performed on 1055 days between Jan 1, 2008 and Nov 21, 2010. Out of these, we report results for each day in 2009 and 2010 since there is no previous data to compare against the 2008 values. We simulated the list of occupants of each building using a realistic estimate of the occupancy or association of persons to each building. The simulated occupant list was used as the recipient list for messages about particular buildings. These numbers are shown in Figure 8.

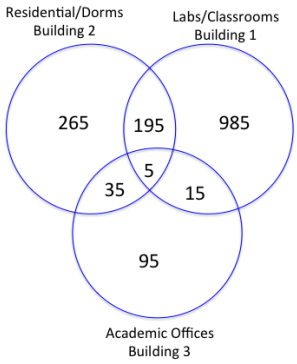

Fig. 8. Number of subscribers in each building

Since our concept representations and data representations are semantic, the query to generate alerts must also be semantic. The SPARQL representation for the query used is listed below. This query lists all records which satisfy the condition of having a higher cumulative daily total energy use compared to the same day on the previous year for the same building. We query over records of the same class type (building) and retain only those records whose timestamp is exactly 365 days ahead of the previous record and energy consumption value is higher than the previous value. Based upon the output results of this query, semantic representations for alerts are created. The results of such a query can also be used to create, destroy and modify the instances of EIP classes themselves (via SPARQL UPDATE and CONSTRUCT statements), thus, leading to overall semantic-driven lifecycle management of enterprise integration patterns.

[6]www.openrdf.org

```
SELECT ?record
WHERE {
    ?record a ?type;
        eip:hasValue ?val ;
        eip:hasTimestamp ?time .
    ?rec_old a ?type;
        eip:hasTimestamp ?time_old ;
        eip:hasValue ?val_old.
    FILTER (?time=?time_old+365 &&
     ?val>?val_old)
}
```

C. Results

We perform the described experiment to compare the number of alert messages generated in the workflow in Figure 1 and compare it to the number of alerts in the optimized workflow in Figure 6. The resulting total number of difference in alerts (for all three buildings) is shown in Figure 9. As expected, optimizing the demand-response scenario with the aggregator and dynamic router EIPs has reduced the number of redundant alerts. Over the complete dataset, 469,100 alerts are generated in the basic workflow, and 441,155 alerts are generated by using the optimized workflow with enriched EIPs, thus, leading to 27,945 less messages being sent in the optimized scenario. This is an improvement of 5.96% over the original number of alerts.

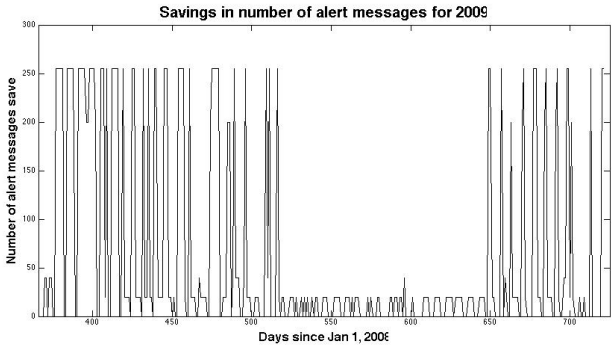

Fig. 9. Savings in the number of alert messages sent in 2009

VI. Conclusions and Future Work

Enterprise integration patterns are used to loosely couple systems with messaging interfaces. We introduced the concept of a semantic implementation of these integration patterns and described how such a representation can encapsulate automatic management of messaging resources. This concept was demonstrated in the context of a Smart Grid demand-response communication application. Our simulations using data from this application showed the potential savings of communication messages in the system.

Future work would include developing semantic representations of all EIPs. In order to easily realize the potential benefits of managing the messaging interfaces using semantic representations, the EIP patterns in existing middleware suites have to be wrapped with a semantic layer; this layer would implement the lifecycle of the messaging resources (such as messaging channels) via the specific middleware implementation.

Acknowledgments

This work is supported by Chevron Corp. under the joint project, Center for Interactive Smart Oilfield Technologies (CiSoft), at the University of Southern California. We thank Facilities Management Services at University of Southern California for providing us with the energy consumption data of campus buildings.

References

[1] G. Hohpe and B. Woolf, *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004.

[2] H. Taylor, A. Yochem, L. Phillips, and F. Martinez, *Event-driven architecture: How SOA enables the real-time enterprise*. Addison-Wesley Professional, 2009.

[3] S. Massoud Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *Power and Energy Magazine, IEEE*, vol. 3, no. 5, pp. 34–41, 2005.

[4] A. Wagner, S. Speiser, and A. Harth, "Semantic web technologies for a smart energy grid: Requirements and challenges," in *In proceedings of 9th International Semantic Web Conference (ISWC2010)*, 2010, pp. 33–37.

[5] D. S. Linthicum, *Enterprise application integration*. Addison-Wesley Professional, 2000.

[6] K. Umapathy and S. Purao, "Designing enterprise solutions with web services and integration patterns," in *Services Computing, 2006. SCC'06. IEEE International Conference on*. IEEE, 2006, pp. 111–118.

[7] X. Tang, X. Luo, X. Mi, X. Yuan, and D. Chen, "Dsl route: An efficient integration solution for message routing," in *Semantics, Knowledge and Grid, 2009. SKG 2009. Fifth International Conference on*. IEEE, 2009, pp. 436–437.

[8] D. Chen, G. Doumeingts, and F. Vernadat, "Architectures for enterprise integration and interoperability: Past, present and future," *Computers in industry*, vol. 59, no. 7, pp. 647–659, 2008.

[9] F. Vernadat, "Interoperable enterprise systems: Principles, concepts, and methods," *Annual Reviews in Control*, vol. 31, no. 1, pp. 137 – 145, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578807000132

[10] C. Pedrinaci, J. Domingue, C. Brelage, T. Van Lessen, D. Karastoyanova, and F. Leymann, "Semantic business process management: Scaling up the management of business processes," in *Semantic Computing, 2008 IEEE International Conference on*. IEEE, 2008, pp. 546–553.

[11] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Terziyan, "Smart semantic middleware for the internet of things," in *Proceedings of the 5-th International Conference on Informatics in Control, Automation and Robotics*, 2008, pp. 11–15.

[12] M. Uschold, M. King, S. Moralee, and Y. Zorgios, "The enterprise ontology," *The knowledge engineering review*, vol. 13, no. 01, pp. 31–89, 1998.

[13] S. Rohjans, M. Uslar, and H. Juergen Appelrath, "Opc ua and cim: Semantics for the smart grid," in *Transmission and Distribution Conference and Exposition, 2010 IEEE PES*. IEEE, 2010, pp. 1–8.

[14] Y. Simmhan, Q. Zhou, and V. Prasanna, "Semantic information integration for smart grid applications," in *Green IT: Technologies and Applications*. Springer, 2011, pp. 361–380.

[15] Y. Simmhan, V. Prasanna, S. Aman, A. Kumbhare, R. Liu, S. Stevens, and Q. Zhao, "Cloud-based software platform for big data analytics in smart grids," *Computing in Science and Engineering*, vol. 99, no. PrePrints, p. 1, 2013.

[16] K. M. Chandy, O. Etzion, and R. von Ammon, "10201 Executive Summary and Manifesto – Event Processing," in *Event Processing*, ser. Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2011. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2011/2985