From ODRL-S to Low-level DSL: A Case Study Based on License Compliance in Service Oriented Systems

Soudip RoyChowdhury[1], G.R. Gangadharan[2], Patricia Silveira[1], and Vincenzo D'Andrea[1]

[1] University of Trento
Via Sommarive, 14, Trento, 38100 Italy
{rchowdhury,silveira,dandrea}@disi.unitn.it
[2] Politecnico di Milano
Piazza Leonardo Da Vinci, 32 , Milan 20133 Italy
gangadharan@elet.polimi.it

Abstract. In this paper, we present a case study in the framework of COMPAS, a research project focused on supporting compliance monitoring and verification in service based systems. In this paper, we also illustrate how we translate high-level service licenses (specified in Open Digital Rights Language for Services (ODRL-S)) to low-level rules for verifying the compliance requirements at runtime. We validate our approach by architecting a compliance driven service oriented system, where at runtime business processes are monitored for compliance.

1 Introduction

Licensing is the way for exploiting intellectual rights into software. Licenses serve for developers to hold the control over consumers on the way they use the software content. Consequently, developers rely on the contracts in the form of license agreements to protect software from unauthorized consumption. Thus, software licensing is considered to include all transactions between the licensor and the licensee, in which the licensor agrees to grant the licensee the right to use some specific software or contents of information for a specific tenure under predefined terms and contracts [1].

Service Oriented Computing (SOC) allows the software-as-a-service concept to expand to include the delivery of complex business process and transactions as a service. It also allows applications to be constructed on the fly and services to be reused everywhere and by anybody [2]. While software serves as a standalone application, the rationale behind services is making network accessible operations available anywhere and anytime. A consequence is that while software is separately installed and executed using the computing resources internal to an organization, a service is often executed using external resources. In addition, while software is designed with particular use in mind, services are designed to facilitate potential reuse as well as use in several contexts. The design of

services supports loose coupling, wherein a service acquires knowledge of another services, still remaining independent, and requests the execution of operations to the other services. Software is designed to incorporate a set of specific functions and usually not allowed to be integrated with other software chosen at runtime, giving rise to interoperability issues. Further, software could be restricted by the organizational boundaries and could not communicate with other software crossing the boundaries. The fundamental of service orientation is to design services to encourage composition. Thus, the said distinguishing characteristics and nature of service prevent service directly to adopt the licensing models of software.

The objectives of a service license are as follows [3].

- To define the extent to which the service can be used, on the basis that any use outside the terms of the license would constitute an infringement.
- To have a remedy against the consumer where the circumstances are such that the acts complained of do not constitute an infringement of copyrights.
- To limit the liability of service providers in case of failure of the service.

Optionally, a service license will also specify information on service delivery, acceptance, and payment. We have formalized service license clauses and proposed a language ODRL-S for defining service licenses in machine interpretable way [3]. The anatomy of a service license includes clauses on Subject, Scope of Rights, Financial Terms, Warranties, Indemnities, and Limitation of liabilities (WIL), and Evolution (Figure 1) as detailed in [4].

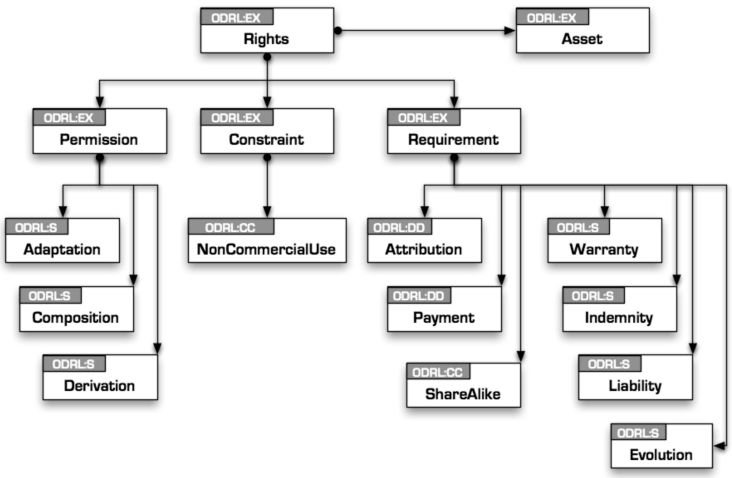

Fig. 1. ODRL-S Licesne Model

ODRL-S Profile defines three new permissions that are directly related to the types of uses of services- Adaptation, Composition and Derivation. The ODRL-

S Profile incorporates the NonCommercialUse constraint from the ODRL/CC Profile and Payment Requirement from the ODRL Core Profile. The ODRL-S Profile defines WIL model and also Evolution model (defined by ODRL-S as new requirements).

In this paper, we validate the compliance of service licenses expressed in ODRL-S, illustrated by a case study of COMPAS[3]. We have described a mechanism that interprets ODRL-S licenses and transforms to low-level Esper[4] rules. Following this, we present a framework for license compliance verification during runtime of a service-oriented system.

The remainder of the paper is organized as follows: In Section 2, we present the case study in detail. Section 3 represents licenses of the case study in ODRL-S, as an unambiguous representation of service licenses. Section 4 represents the overview of validation of our approach in COMPAS project. Finally, Section 5 concludes and describes future work.

2 WatchMe Case Study Overview

In COMPAS project, our case study focuses on advanced telecom services offered by a mobile virtual network operator (MVNO), named WatchMe. WatchMe procures video and audio streams from different video and audio providers, composes them in-house, and broadcasts them to end-users in-terms of service. In this scenario, we have identified three compliance requirements regarding licensing concerns (See Table 1), which need to be monitored during the composition and acquisition of media streams. Such requirements regard only the services invoked between providers and WatchMe.

Any non-compliant invocations could lead to costly penalties to a company, in terms of money as well as reputation loss. Mitigating the risk of license violations is one of the concerns today services Industry is trying to address. Being motivated by this fact, in our approach we tackle service licensing problem in two ways. Firstly, we describe the service license in ODRL-S specification, so that the domain experts (legal advisor or business analyst), who understand the legal terms and condition as well as the business logic, can express service constraints and requirements in a more formal way which can be consumed by IT eco-system. Secondly, we demonstrate how these license concerns can be linked with the runtime system and translated to low-level conditions checking. This translation enables runtime system to be automatically monitored in order to identify potential violations, which may lead to high business risk.

3 Solution Overview

3.1 License Representation in WatchMe Scenario

Following the ODRL-S license model, we represent the license for WatchMe scenario as follows. In our license representation, we visualize a service license

[3] http://www.compas-ict.eu
[4] http://Esper.codehaus.org

Compliance Requirements	**Description**	**Compliance Risks**	**Rationale**	**Control**
Time-based plan	When the WatchMe company subscribes for the Time-based plan, it acquires any number of times any possible streams in a certain period, based on the amount paid to the media supplier.	If WatchMe uses streams outside the 30 days period, this leads to violation of the license. This could lead to claims and reputation damage.	When WatchMe company subscribes for the time-based plan, it has to pay 89.90 Euros first and then receive an unlimited number of times any available stream from the media supplier in a 30 days period starting from the contract start date.	Is Date before end of period?
Pay-per-view plan	When the WatchMe company subscribes for the Pay-per-view plan, it acquires a limited number of streams based on the amount paid to the media supplier.	If WatchMe uses too many streams, then this leads to violation of the license. This could lead to claims and reputation damage. If WatchMe uses too few streams with respect to the acquired number of streams, this leads to (unnecessary) costs.	When WatchMe company subscribes for the Pay-per-view plan, it has to pay 29.90 Euros first and then receive 300 streams from the media supplier.	Number of streams (before next payment has to be made).
Composition plan	Only pre-defined combinations of video and audio providers are allowed due to the licenses specified by the video provider.	When a wrong combination of video and audio is used, this leads to violation of the license. This could lead to claims and reputation damage.	Video Stream VideoTube can only have audios streams from AudioTube or QuickAudio. QuickVideo can only have audio streams from QuickAudio.	Audio-Video combination

Table 1. WatchMe License Requirements

as a set of offers indicated by <o-ex:offer> tag. A license contains declaration of resource, identified by unique asset id, on which the license would be applied to (line 2). This is followed by declaration of the fine grained process/service endpoint description, followed by the information of its version (line 4 and 6).

Tag sets <o-ex:permission>, <sl:composition/>, and </o-ex:permission> represent that the specified service is composable with other services (line 10).

```
  <!-- Namespace declarations go here-->
1. <o-ex:offer>
2.    <o-ex:asset o-ex:id="watchMe-video-service">
3.       <o-ex:context>
4.            <o-dd:uid>
                  urn: watchMe:service: watchMe-Provider1TimeBased_service
5.             </o-dd:uid>
6.             <o-dd:version> 1.0 </o-dd:version>
7.        </o-ex:context>
8.    </o-ex:asset>
9.    <o-ex:permission>
10.    <sl:composition/>
11.   </o-ex:permission>
```

We define Financial Terms as a set of permission, requirement and constraint tags. These tags are attached with some actions tag e.g., <o:dd-play>, which implies the requirement and the constraint terms would be associated with a specific action of the service (line 2). In our example, we have also defined the corresponding plan associated with a license. As ORDL-S V1.0 does not have the option to define plan type, we implement by creating WatchMe specific name space, by which we can specify specific tags like plan type (line 5-6). Line 3 shows how in a service license, IT experts can specify which event is required to be monitored for verifying the specific license requirements and constraints.

```
1.   <o-ex:permission>
2.           <o-dd:play>
3.              <wm:event name="WatchMeGetVideoStreamEvent">
4.            <o-ex:requirement>
5.                <wm:plan>
6.                 <wm:type>TimeBasedplan</wm:type>   </wm:plan>
7.    <o-dd:prepay>
8. <o-dd:payment>
9.              <o-dd:amount o-dd:currency="EUR">89.90</o-dd:amount>
10.   </o-dd:payment>
11. </o-dd:prepay>
12. </o-ex:requirement>
13. <o-ex:constraint>
14.     .
15.     .
16. </o-ex:constraint>
17.  </wm:event>
18.  </o-dd:play>
```

To check the control constraints in our scenario as specified in Table 1, we use <o-ex:constraint> tag as defined in ODRL.

For Time Based plan we define the constraints as follows. <o-dd:datetime>, <o-dd:start>, and <o-dd:end> tags are used to specify the start and end time for the validity of the license.

```
1. <o-dd:play>
2.   .
3.   .
4. <o-ex:constraint>
5. <o-dd:datetime>
6. <o-dd:start>2008-01-01T12:30:00</o-dd:start>
7. <o-dd:end>2008-01-31T12:30:00</o-dd:end>
8. </o-dd:datetime>
9. </o-ex:constraint>
10. </o-dd:play>
```

For Pay-per-View plan we define the constraints as follows. <o-dd:unit>, <o-ex:type>, and <o-dd:count> tags are used to specify the unit and type of the constraints based upon which the counter would be run to check the validity of the license. Here, if the number of video stream exceeds 300 count, then the license terms are violated.

```
1. <o-dd:play>
2.    .
3.    .
4. <o-ex:constraint>
5. <o-dd:unit o-ex:type="watchMe:NumberOfStreams" />
6.   <o-dd:count>300</o-dd:count>
7.  </o-ex:constraint>
8. </o-dd:play>
```

For Composition plan, we define the constraints as follows. This license definition specifies that AudioTube having uid A1 and QuickAudio having uid A2 are considered as the ApprovedAudioProviders to be composed in this specific service context.

```
1. <o-dd:play>
2. <wm:event name="WatchMeGetAudioStreamEvent">
3.  <o-ex:requirement>
4.        <wm:combinations>
5.         <wm:type>ApprovedAudioProviderOnly</wm:type>
6.        </wm:combinations>
7. </o-ex:requirement>
8. <o-ex:constraint>
9.  <o-ex:context>
10.        <o-dd:name>ApprovedAudioProviders</o-dd:name>
11.  <o-ex:constraint>
```

```
12.  <o-ex:context>
13.        <o-dd:name>AudioTube</o-dd:name>
14.        <o-dd:uid>A1</o-dd:uid>
15. </o-ex:context>
16.  <o-ex:context>
17.       <o-dd:name>QuickAudio  </o-dd:name>
18.       <o-dd:uid>A2</o-dd:uid>
19.  </o-ex:context>
20.  </o-ex:constraint>
21.  </o-ex:context>
22.   </o-ex:constraint>
23. </wm:event>
24. </o-dd:play>
```

3.2 Translation of ODRL-S Based License to Low-Level (Esper) Rules

In our case study, we have developed our system based on Event driven architecture where business applications communicate with each other via common Enterprise Service Bus (ESB). Communications happen through underlying enterprise-messaging service. In our implementation, we have used apache activemq message queue[5] for the messaging part. In COMPAS, we have used an open source complex event processing engine Esper, which processes event streams and discovers complex patterns among multiple streams of event data. Event processing server listens to the message queue for event traces, processes those events stream data and reacts to them as per the rules/condition specified in the form of Esper rule. In our implementation, we translate license concerns to corresponding Esper rules to verify compliance of the business processes at runtime against event traces. Transforming high-level license concerns (written in ODRL-S), to low-level domain specific rules (Esper rules) was not an easy challenge to solve. The first challenge we had to tackle, was to associate the license concerns with the corresponding events, so that Esper engine can verify those constraints against related event data at runtime. To solve this challenge, we have introduced IT-experts into the loop during the process of writing the license, as explained in the previous section. The second challenge was to transform the licenses to corresponding Esper rules. To solve this challenge, we have introduced a License Translator component (as shown in Figure 2) into our architecture. License translator translates ODRL-S licenses to Esper rules by using pattern identification and template based translation.

Figure 2 shows that Domain experts and IT experts feed the system with artifacts like Licenses as shown in the previous diagrams and event/process related information during design time. At the runtime system, License Translator reads those inputs and produce low-level rules like Esper rules in our case. These low level rules are consumed by the event processing engine, which checks and

[5] http://activemq.apache.org/

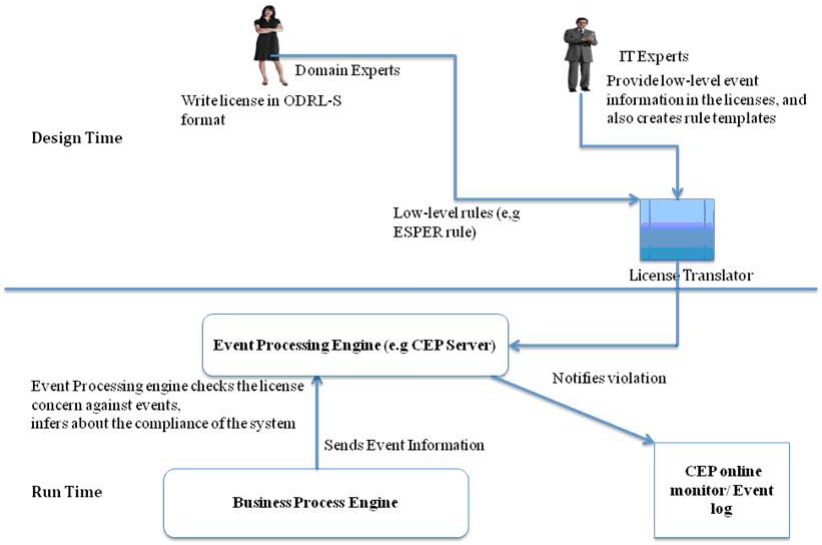

Fig. 2. Design time (ODRL-S) to runtime (Esper rules) conversion system

monitors the usage and legal concerns against the events generated by the runtime system. Incase of violation, it takes corrective measures or reports to the concerned person / reporting tool or to the compliance-monitoring dashboard (see Section 4).

3.3 License Translator

License Translator takes ODRL-S licenses as input and finds the patterns in it. In COMPAS, we deal with 3 types of patterns corresponding to the license concerns as mentioned in Table 1. For example, the pattern corresponding to Pay-per-view plan is the constraints on the number of streams.

```
1. <o-ex:constraint>
2. <o-dd:unit o-ex:type="watchMe:NumberOfStreams" />
3.  <o-dd:count>300</o-dd:count>
4.  </o-ex:constraint>
```

Similarly for Time-based plan, the pattern corresponds to constraints on date-time attribute.

```
1.  <o-ex:constraint>
2. <o-dd:datetime>
3. <o-dd:start>2008-01-01T12:30:00</o-dd:start>
```

```
4. <o-dd:end>2008-01-31T12:30:00</o-dd:end>
5. </o-dd:datetime>
```

License translator parses the xml files corresponding to the licenses and discovers these patterns. Once it finds a specific pattern, License translator then searches through the Esper rules templates as shown below, to find the corresponding low level rule template.

```
1. Count pattern =
    <rule1>
      create window
        PayPerViewWindow.win:keepall().std:unique(SessionID)
      as
        select SessionID, RequesterID from<event_name>
      </rule1>
      <rule2>
         select count(*) from PayPerViewWindow
       </rule2>
2. Date pattern=
    <rule1>
       create window
         TimebasedWindow.win:keepall().std:unique(SessionID)
        as
         select SessionID, RequesterID from WatchMeGetVideoStreamEvent
            where (<start_Time> > current_timestamp()) or
                  (current_timestamp() ><end_Time>)
     </rule1>
     <rule2>
        select count(*) from TimebasedWindow
     </rule2>
```

In our approach, IT experts analyze service licenses and create these low level rules templates for each category of license. License translator binds the constraint data retrieved from service licenses to the template and produces the low level rules (as shown below), which are then consumed by Esper event processing engine to check the compliance at runtime.

```
3. <?xml version="1.0" encoding="UTF-8" ?>
4. <license>
5. <ServiceUID>
     urn: watchMe:service: watchMe-Provider1-PerUse_service
   </ServiceUID>
6. <PlanType>Pay-per-view plan</PlanType>
7. <amount>29.90</amount>
8. <unit>watchMe:NumberOfStreams</unit>
9. <count>300</count>
10. <Esper>
11. <rule1>
        create window
```

```
          PayPerViewWindow.win:keepall().std:unique(SessionID)
        as
          select SessionID, RequesterID from WatchMeGetVideoStreamEvent
      </rule1>
12. <rule2>
        select count(*) from PayPerViewWindow
      </rule2>
13. </Esper>
14. </license>
```

Processing server monitors the event logs and based upon these rules, it ascertains whether any violation has occured. In the above rule example, `<rule1>` asks processing server to create a virtual window based upon unique SessionID of the events. `<rule2>` specifies to count the number of entries in the window. Esper server parses these rule files and finds the data and the low level rules. During runtime, it finds the violation occurrences with respect to the constraints specified in the licenses.

In COMPAS scenario, our approach is able to detect the compliance violations successfully at runtime against the license concerns related to audio-video data broadcasting perspective. However, our approach can be extended to support all other kind of license concerns, which can arise in other business scenarios. In future, we will extend our solution by making it more generalized to support for diverse licensing scenarios.

4 Validation

Assessing whether a company's business practices conform to laws and regulations and follow standards and best practices, i.e., compliance governance, is a complex and costly task. As a result, today it is very hard for any CFO or CIO to answer questions such as: Which rules does my company have to comply with? Which processes should obey which rules? Which processes are following compliance sources? Where do violations occur? Which processes do we have under control? [5]. Even more, it is hard to do so from a perspective that not only satisfies the company but also the company's auditors, which is crucial as the auditors are the ones that certify compliance. In COMPAS project we show how to address these issues, which concepts and models underlie the problem, and, eventually, how IT can effectively support compliance analysis in Service Oriented Architectures (SOAs) keeping compliance experts in the loop [6].

To address the compliance requirements in COMPAS, we have proposed a conceptual model (See Figure 3) for compliance along with a compliance governance Dashboard [6] architecture, which is targeted for several classes of users: chief officers of a company, line of business managers, internal auditors, and external auditors (certification agencies). The aim of dashboard is to report on compliance, to create an awareness of possible problems or violations, and to facilitate the identification of root-causes for non-compliant situations.

In COMPAS, we have developed an event driven architecture where events generated by Business process engine are published in the ESB, which afterwards

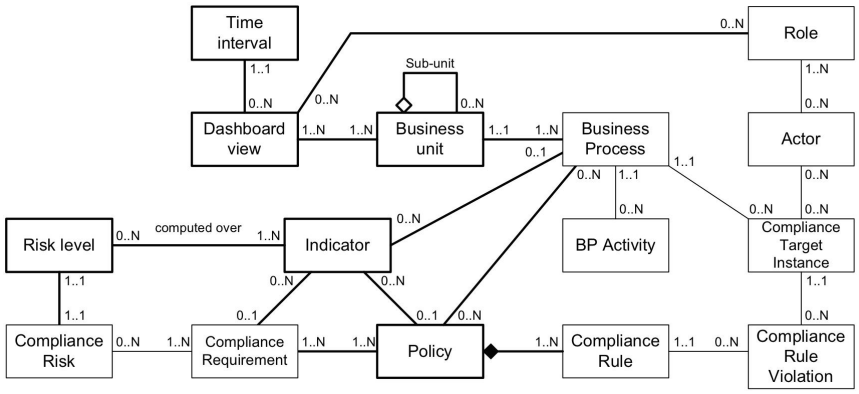

Fig. 3. Conceptual model for compliance requirements in COMPAS

are stored in the Event log. After that, an extraction, transformation, and load (ETL) routine is executed to capture those raw data events and to store them into the Data warehouse (DW). These events structured according to a data model can be used as input for the analyzing of compliance requirements and computing Key Compliance Indicators (KCIs). Figure 4 shows the high level view of COMPAS runtime architecture.

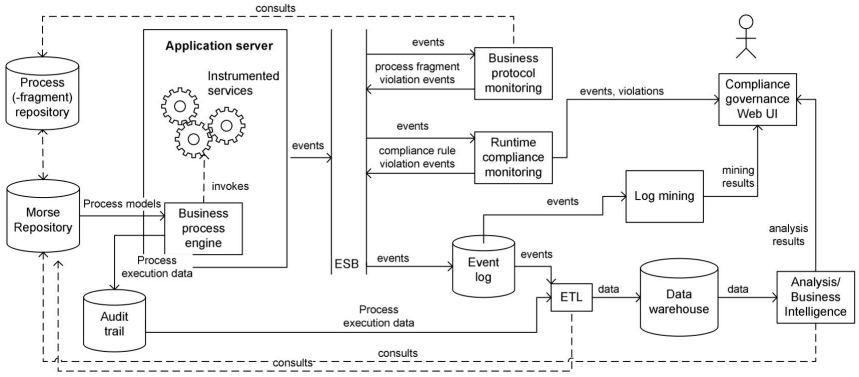

Fig. 4. Runtime Architecture of COMPAS Compliance governance

For checking the license concerns at runtime, in COMPAS we have developed an interface between Runtime compliance monitoring (which checks the compliance concerns against generated events) and License translator. License translator translates high level ODRL-S licenses (e.g. pay-per-view plan, Time-based plan) to corresponding low-level Esper rules. Esper event processing en-

gine, which is also a part of runtime compliance monitoring (Figure 4), takes those low level Esper rules as input and check against the events associated with those rules. Incase of occurrence of any violation of requirements as specified in the license, it reports to the compliance governance dashboard[6].

5 Conclusions

In this paper, we described the compliance requirements in the COMPAS project from the licensing perspective by architecting an event driven IT system. Based on a case study, we presented how service licenses in ODRL-S can be translated to a low level (Esper rules) DSL so that license compliance of the system can be validated at runtime. In our future work, we are planning to develop an user interface framework for domain experts to specify licenses in ODRL-S format in a more user friendly manner rather than writing in xml files.

Acknowledgments

This work is carried out under European commission project COMPAS and is supported by funds from the European Commission (contract N 215175 for the FP7-ICT-2007-1 project COMPAS).

References

1. Classen, W.: Fundamentals of Software Licensing. IDEA: The Journal of Law and Technology 37(1) (1996)
2. Papazoglou, M.P.: Web Services: Principles and Technology. Pearson, Prentice Hall (2008)
3. Gangadharan, G.R., D'Andrea, V.: Licensing Services: Formal Analysis and Implementation. In: Proceedings of the Fourth International Conference on Service Oriented Computing (ICSOC'06), Chicago, USA. (2006) 365–377
4. Gangadharan, G.R., D'Andrea, V., Iannella, R., Weiss, M.: ODRL Service Licensing Profile (ODRL-S). In: Virtual Goods: Technology, Economy, and Legal Aspects. Nova Publishers, USA (2008)
5. Bellamy, R.K.E., Erickson, T., Fuller, B., Kellogg, W.A., Rosenbaum, R., Thomas, J.C., Wolf, T.V.: Seeing is believing: designing visualizations for managing risk and compliance. IBM Syst. J. 46(2) (2007) 205–218
6. Silveira,P., Rodríguez,C., Casati,F., Daniel,F., D'Andrea,V., Worledge,C., Taheri, Z.: On the Design of Compliance Governance Dashboards for Effective Compliance and Audit Management. In: Proceedings of NFPSLAM-SOC'09. (2009)

[6] http://compas.disi.unitn.it:8080/CGDs/dashboard.jsp