A Resource-Centric Architecture for Service-Oriented Cyber Physical System

Kaiyu Wan[1] and Vangalur Alagar[2]

[1] Xi'an Jiaotong-Liverpool University, Suzhou, PRC
Kaiyu.Wan@xjtlu.edu.cn
[2] Concordia University, Montreal, Canada
alagar@cse.concordia.ca

Abstract. The strategic application domains of Cyber Physical Systems (CPS) [7,6] include health care, transportation, managing large-scale physical infrastructures, and defense systems (avionics). In all these applications there is a need to acquire reliable resources in order to provide trustworthy services at every service request context. Hence we view CPS as a large distributed highway for services and supply chain management. In traditional service-oriented systems service, but not resource, is a first class entity in the architecture model and resources are assumed to be available at run time to provide services. However resource quality and availability are determining factors for timeliness and trustworthiness of CPS services, especially during emergencies. So in the service-oriented view of CPS discussed in this paper we place services around resources, because resource constrain service quality. We investigate a *resource-centric*, and *context-dependent* model for *service-oriented* CPS and discuss *3-tiered* architecture for *service-oriented* CPS in this paper.

Keywords: Resource, Service-oriented Architecture, Service Model, Cyber Physical System.

1 Introduction

The NSF program description [7] states that CPS initiative [2] is "to transform our world with systems that respond more quickly, are more precise, work in dangerous and inaccessible environments, and provide *large scale distributed services*." This paper is a contribution to specify resources and resource-centric services. The term *resource* is used in a generic sense to denote an entity that is relevant in either producing or consuming a service. In CPS, physical devices are resources, which are hence first class entities. Services may be either generated or consumed by physical devices, which might in turn be consumed by cyber computational resources, such as communication protocols. Software services may be generated by the computational resources that reside either in a static or dynamic host computer in CPS network and may be consumed by other physical devices (actuators) to make changes in the environment. In general, a CPS resource might offer many services, a CPS service might require several resources, a CPS resource might *use* other resources, and a CPS (complex) service may be produced by combining several services and resources. Thus the service-oriented view of

J.J. Park et al. (Eds.): GPC 2013, LNCS 7861, pp. 686–693, 2013.

CPS is more complex than the service-oriented view required for traditional business applications, as discussed in SOC literature [1].

In this paper we regard the three conceptual layers of CPS resources as *physical*, *logical*, and *process*. Selic [9] uses the term resource to denote *any runtime entity for which the services can be quantified by one or more quality of service characteristics*. Thus the UML resource model of Selic [9] is restricted to the run-time (process) environment of a specific application in a centralized real-time system. The Resource-Explicit Service Model (RESM) proposed by Huang uses Entity Relationship diagrams to model resources and services as equal citizens [4]. It is possible to refine a physical layer model to a process layer model by adding more details. In doing so the complexity of the diagram will increase. Expressing logical dependencies between resources in this modeling notation is hard. RDF [10] is meant to describe web resources, which according to our classification are *virtual resources*. Resource models at physical and logical layers can use RDF. The Resource Space Model (RSM) describes the resource space and logical relationship between resources, for a specific application domain. This model will have multiple descriptions of one resource when that resource is used in different applications involving different resources. So, this approach does not support modeling the physical layer and to model resources at process level will be quite complex. In all these models context information, and QoS properties (such as reliability and availability) are absent.

CPS applications in areas such as health care, flood monitoring, and emergency evacuation require timely services, which in turn depend upon *availability* and *reliability* of resources. Both quantity and quality must be negotiated as often as necessary, and with as many resource providers as possible within the time limit set to complete the requested service. The absence of availability of reliable resources, and the emergence of severe competition for resources among services might cause the deadline not be met. For strict real-time applications, such as emergency evacuation, such situations are unsafe. Even when reliable resources are available in sufficient quantity, their distribution and cyber communication to service requesters may fail causing the deadline to fault. Consequently quality properties, attributes, context of use and availability constraints for resources must be published by resource producers in advance in order to enable service providers repeatedly discover resources required for providing services. Such discovery of resources maximizes the creation of services in advance and minimizes non-availability of resources at run-time. This is the motivation why we investigate resource-centric service model for Cyber Physical Systems in this paper.

Throughout the paper we suggest the underlying formalism without being formal. In Section 2, the resource-centric abstract service model is specified. In Section 3 we introduce the basics of context formalism necessary to understand how satisfaction relation is to be evaluated in a context. In Section 4 we use three-tiered approach to specify resource-centric service-oriented architecture for Cyber Physical Systems. We conclude the paper in Section 5 with a brief summary of its significance and our ongoing work.

2 Abstract Service Model

Abstractly, the three major stakeholders in CSP are *Resource Producer* (RP), *Service Provider* (SP), and *Service Requester* (SR). A SP may interact with one or more RPs

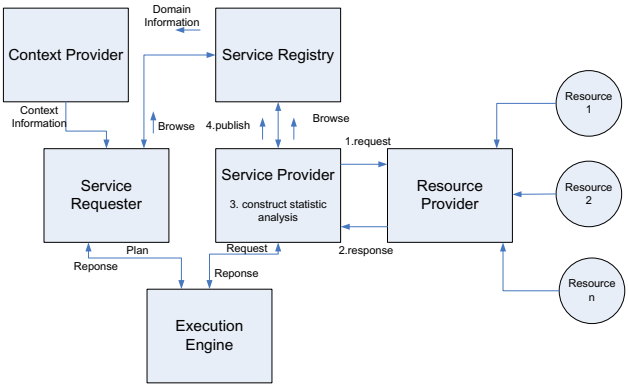

Fig. 1. Resource-Centric Abstract Service Model

and one or more SRs. A RP may not be *directly visible* to any SR in the system. So, a SR gets to know about resources used for service composition and delivery only from the service descriptions posted by the SPs. In this abstract CPS model shown Figure 1, every RP creates a resource model for each resource in its ownership and publishes it to all SPs who subscribe to its services. Thus, it is a comprehensive description of the physical, logical, and process layer needs. This specification will enable the SPs conduct a static analysis of published resource descriptions and request their distribution across CPS nodes in a demand-driven fashion. That is, they may acquire the resources and create their services well before service execution times. We regard *reliability* and *availability* as fundamental attributes for resource acquisition. A reliable resource is one that adds *economic value* for the client who uses it, by *satisfying the QoS characteristics of the client*. That is, the QoS characteristics *provided by* the resource satisfies the QoS characteristics *required by* the client. So, reliability is part of the QoS *contract* between the client and the resource used by it. In order that the client may use the resource to its full advantage, the resource must be *available* in sufficient quantity and when required by the client. So, availability has both a quantitative and temporal dimension. Consequently, both reliability and availability are made part of resource model. Once the resource model is published by a RP, the SPs who are clients of RPs will have an opportunity to independently verify the claims made in service descriptions before selecting it for use in the services created by them.

A SP creates service descriptions for services provided by it. A service description includes the functionality of the service, its non-functional properties, a list of resources used in creating and delivering the service, and a service contract. A SP publishes service descriptions and make them available to SRs who subscribe to its services. The SP guarantees the quality of service through a list of *claims*, which should be validated by the SP when challenged by the SRs. A SR creates a demand model of service. This model is very much dependent upon the application. It may be as simple as the 'quality of result specification'. Examples include (1) 'the cost should not exceed $50', and (2) 'the service should be delivered within 2 hours from the time the contract is signed'. Once the SR presents its model, after choosing a service type, the SP is expected to

deliver a service whose quality attributes satisfy the quality attributes in the model presented by the SR.

Satisfaction Criteria
Therefore, in order to have matched CPS services the two essential conditions are

- *Provided-by*(RP_q) *SAT Required-by*(SP_q)
- *Provided-by*(SP_q) *SAT Required-by*(SR_q)

where *Provided-by*(X_q) means the 'quality attributes provided by the entity X', *Required-by*(Y_q) means the 'quality attributes required by the entity Y', and *SAT* is the 'satisfaction relation'. So we posit that the resource model should include *Provided-by*(RP_q), and the service model should include *Required-by*(SP_q), *Provided-by*(SP_q), and *Required-by*(SR_q). We assume that a SP, by whichever *Required-by*(SP_q) model it has, will select the resources in order to satisfy the relation *Provided-by*(RP_q) *SAT Required-by*(SP_q). We assume that a SR, by whichever *Required-by*(SR_q) model it has, will select the services in order to satisfy the relation *Provided-by*(SP_q) *SAT Required-by*(SR_q). Thus, the resource description should enable a formal execution of the *SAT* relation. Typical *SAT* relations are *implies* ($\rightarrow$), and *includes* (subset relation $\subset$)). These are resolved using Logic and Set Theory provers. We discuss in Section 3 a method to resolve situation constraints in different contexts.

3 Context-Dependence

In service-oriented systems and in particular for CPS, service contracts are usually context-dependent. Resource availability must be assessed from a combination of several factors ranging from rarity of the resource to legal implications in delivering it. An ubiquitous resource, such as water, may not be sold by a RP to a SP who is located in a zone Z either because the RP is not permitted to supply water in Zone Z or the water quality does not meet the standards of zone Z. In many countries strict environmental laws might forbid or restrict the use of certain types of energy resources. These examples are to motivate the necessity to include context information as part of resource and service descriptions. In order that such descriptions be formalized we need a formal representation of context. We use the formal notation of context and context toolkit developed by Wan [11,12] in order to formalize context information. A *context space* is defined for an application in a domain and contexts are constructed within that space. A context space includes a finite set of *dimensions* and a *type* associated with each dimension. The typed values are called *tags* along each dimension. A RP may define a context space with (1) *who* needs the resources? (2) *what* resource types are available? (3) *where* a resource can be delivered? (4) *when* the resource will be available? and (5) *why* the resource might be required? Contexts are constructed from the knowledge collected in the five dimensions *who*, *what*, *where*, *when*, and *why*. A RP can construct contexts that include all or only a subset or a superset of these dimensions. In a similar way a SP can construct contexts related to service provision. In the notation of Wan [11] a context is represented as $c = [WHERE : \text{Chicago}, WHEN : 04/07/2012, WHO : XYZ, WHAT : \text{EPR2}]$.

The interpretation is that c defines the setting in which the RP XYZ at $Chicago$ has the resource $EPR2$ at time $04/07/2012$.

A context in itself is not useful, unless it is associated with *events* or *situations* that are of interest in the context. The context formalism [11,12] allows evaluating situations at a context. A situation is encoded as a logical formula p on the dimension names and other variables. In order to check that a situation p is true in a context c, the dimension names in p are bound to the tag values in the definition of context c and p is evaluated. An example situation is the predicate $can\_deliver == (\mid x - WHERE \mid < 100) \wedge (d_2 < 10 + WHEN)$, where $\mid \ldots \mid$ denotes the distance expression and $(10 + WHEN)$, meaning within 10 days of specified time. When evaluated at c we will get the expression $(\mid x - Chicago \mid < 100) \wedge (d_2 < 14/07/2012)$. Once the values for the location variable x and date variable d_2 are known this expression can be evaluated to either true or false. This approach is used to resolve the *SAT* relation involving context situation constraints.

4 A Three-Tiered Architecture for Service-Oriented Cyber Physical System

In this section we put forth a *resource-centric*, and *context-dependent* model for *service-oriented* CPS. A *3-tiered* approach is shown in Figure 2. Tier-1 is the physical layer in which the attributes and properties of a resource are specified together with legal and contextual constraints. Tier-2 is the logical layer which imports specifications from Tier-1, introduces dependencies and constraints and lists possible ways to utilize the imported resource in services. Tier-3 imports resource class specifications from Tier-2 and specifies configured services by adding QoS properties of created service.

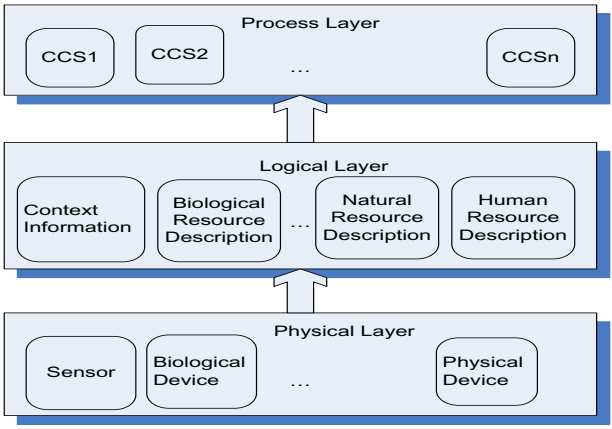

Fig. 2. Three-tiered architecture for CPS

4.1 Physical Description Layer

In this section we discuss the attributes for modeling resources in the physical layer. The model that we create is called *Resource Description Template* (RDT). We may assume that CPS resources are categorized so that all resources in a category are of the same *type*. One such classification is *human resources*, *biological resources*, *natural resources*, *man made resources*, and *virtual resources*. Human resources are well understood and many human resources management systems are available today. Resources required by a living being for survival and growth are biological resources. Examples include water and food. Natural resources are derived from the environment. Examples include trees, minerals, metals, gas, oil, and some fertilizers. Biological resource type is a subtype of natural resource type. Man made resources include physical entities such as bricks or mortar, books and journals for learning, and machineries. Any virtual component of limited availability in a computer is a virtual resource. Examples of virtual resource are *virtual memory*, *CPU time*, and the whole collection of Java resources [8].

A RP and its experts determine the essential features and properties to be specified in a resource model. The main attributes of resources, especially when it comes to their *adaptation* for providing services, are *utility*, *availability*, *cost*, *sustainability*, *renewability*, *reuse*. The utility factor for a resource defines its relevance, and often expressed either as a numerical value u, $0 < u < 1$, or as an enumerated set of values $\{$*critical*, *essential*, *recommended* $\}$. In the former case, a value closer to 1 is regarded as critical. In the later case the values are listed in decreasing order of relevance. A *Resource Producer* (RP) may choose the representation $\{\langle a_1, u_1\rangle, \langle a_2, u_2\rangle, \ldots, \langle a_k, u_k\rangle\}$ showing the utility factor u_i for the resource in application area a_i for each resource produced by it. The utility factors published by a RP are to be regarded as recommendations based on some scientific study and engineering analysis of the resources conducted by the experts at the RP sites. Cost might depend upon duration of supply (as in power supply) or extent of use (as in gas supply), or in required measure (as in the supply of minerals). Dependency between resources can often be expressed as situations, in which predicate names are resources.

4.2 Logical Layer Description

For the resource-centric CPS model we need to follow the resource-centric service approach, which is somewhat similar to the order-centric approach [13]. The activities in the service are ordered, and the list of activities per single resource are handled taking into account resource dependencies. This calls for a specification for each resource in which the dependencies on other resources and the tasks that can be done with that resource are listed. This is the logical view and we call this specification a *Resource Class Specification* (RCS). To realize the resource-centric model of CPS it is necessary that every CPS site publishes the RDTs of resources owned (or produced) by it as well as the RDTs acquired from other RPs, develop a mechanism for allocating resources in different service request contexts, and create a RCS.

4.3 Process Layer Model

The process layer for resource-centric CPS should model how services are configured, discovered, composed, and optimized. Among these process layer activities only service configuration activity requires a language description, the other activities require algorithms. So we restrict to service configuration description below.

In a service-oriented model the center piece is service and resources are not fully addressed within service model. On the other hand, in a resource-centric model, such as in [3], the center piece is resource class specifications and service model is ignored. In our resource-centric service model, resource class specifications are included in configuring and composing service specifications. The first step for SP is browsing the sites of those RPs, examining the RDTs published by them, and then selecting the RCSs published by them. The second step is that the SP selects the RPs from whom the RCSs can be bought. The final step for SP is to create services that can be provided by putting together the atomic tasks in the RCSs. We introduce the *CyberConfiguredService* (CCS) notation for this purpose. In CCS the service with its contract, quality assurances, and other legal rules for transacting business are included. Such configured services are published in the site of the SP.

Abstractly viewed, a service is a function. In business, a service not only has functionality but also has non-functional properties, legal issues for providing the service, and context information for service delivery. These are bundled together by the SP in a configured service. We define a *CyberConfiguredService* (CCS) is a service package that includes all the information necessary that a service requester in CPS needs to know in order to use that service. It will include (1) service functionality, (2) a list of resources used to create the service, together with resource specifications, (3) nonfunctional attributes of service, (4) quality attributes of the service, and (5) contract details. Legal rules, context information on service availability and service delivery, and privacy guarantees are part of contract details. The service and contract parts are integrated in CCS, and consequently no service exists in our model without a contract. The contract part in CCS includes QoS contract *Provided-by*(SP_q) as well as the QoS contract *Provided-by*(RP_q). These contracts must be resolved at service discovery and service execution times using methods explained in Section 3.

5 Conclusion

In this paper we have put forth a *resource-centric*, and *context-dependent* model for *service-oriented* CPS. Our contribution is a *3-tiered* approach. Tier-1 is the physical layer in which the attributes and properties of a resource are specified together with legal and contextual constraints. The attributes are typed, properties and legal rules can be formulated in logic, and context has a relational semantics [12]. As such Tier-1 specification has a semantic basis. Tier-2 is the logical layer which imports specifications from Tier-1, introduces dependencies and constraints and lists possible ways to utilize the imported resource in services. Tier-3 imports resource class specifications from Tier-2 and specifies configured services by adding QoS properties of created service. A specification from a lower tier can be included in more than one specification in the next higher tier. Modifications to a higher tier specification do not affect their constituent lower tier

specifications. This 3-tier approach has the advantages of separation of concerns and modularity, the essential software engineering principles for developing large systems. In the near future we will continue our work on resource modeling, investigate formal notation for describing resource-centric services, and illustrate our ideas through proof-of-concept case studies.

Acknowledgement. This research is supported by Research Grants from National Natural Science Foundation of China (Project Number 61103029), Natural Science Foundation of Jiangsu Province, China (Project Number BK2011351) , and Natural Sciences and Engineering Research Council, Canada.

References

1. Georgakopolous, D., Papazoglou, M.P.: Service-oriented Computing. The MIT Press (2008)
2. C.S. Group. Cyber-physical systems: Executive summary. Report (2008), http://varma.ece.cmu.edu/summit/CPS-Executive-Summary.pdf
3. Zhuge, Y.H., Shi, P.: Resource space model, owl and database: Mapping and integration. ACM Transactions on Internet Technology 8(4) (2008)
4. Jian Huang, I.-L.Y., Bastani, F., Jeng, J.-J.: Toward a smart cyber-physical space: A context-sensitive resource-explicit service model. In: 33rd Annual IEEE International Computer Software and Applications Conference. IEEE Press (2009)
5. John, J.J.H., Guttag, V., Wing, J.M.: The larch family of specification languages. IEEE Transactions on Software Engineering
6. Networking, I.T. Research, and D. Program. High-confidence medical devices: Cyber-physical systems for 21st century health care. Technical report, NITRD (2009)
7. NSF. Usa nsf program solicitation, nsf-08-611. Report, NSF (2008)
8. Oracle. Java resources. Web report, Oracle (2003)
9. Selic, B.: A generic framework for modeling resources with uml. IEEE Computer 33(6), 64–69 (2000)
10. W3C. W3c recommendation. Technical report
11. Wan, K.: Lucx: Lucid Enriched with Context. Phd thesis, Concordia University, Montreal, Canada (2006)
12. Wan, K.: A brief history of context. International Journal of Computer Science Issues 6(2) (2009)
13. zur Muehlen, M.: Resource modeling in workflow applications. In: Proceedings of Workflow Management Conference