Modeling Resource-centric Services in Cyber Physical Systems

Kaiyu Wan and Vangalur Alagar

***Abstract*—A service-oriented view of CPS is taken in this paper, because it is a good platform for managing global supply chain management, service acquisition and service provision. Complex services are enabled by a strong influence between computational and physical components that might be globally distributed. A necessary condition for such service delivery is that resources required for complex services are of high quality and are available at service execution times. In a resource-centric service model, both resource quality and service quality using that resource are explicitly stated. In order to make resource quality visible, resource providers will publish a faithful description of the resources and service providers will publish trustworthy service descriptions which explicitly mention the resources used by them. In this paper a cascaded specification approach is discussed for describing resource types, services offered by resource, and a cyber configured service that package physical services.**

***Index Terms*—Resource, Resource Modeling, Resource Description, Cyber Physical Systems**

I. Introduction

Cyber-Physical Systems (CPS) [2] is a new research area with a grand vision. This paper is a contribution to formally specify resources and resource-centric services for CPS. 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 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 CPS is more complex than the service-oriented view required for traditional business applications, as discussed in SOC literature [1].

In [8] we proposed the three conceptual layers of CPS resources as *physical*, *logical*, and *process* through three-tiered approach, as illustrated in Figure 1. In this paper, we continue our research and strive for notations that have semantic consistency across these layers. That is, we design the description language for resources, and services. The three important characteristics of the language are: (1) The published resource or service description, intended for clients, has information completeness, consistency, and correctness. (2) It should be possible to create precise formal descriptions of published service descriptions. Formalized service descriptions are not for public consumption, and are used by the service provider only for the purpose of validating the published descriptions and the *demand-response model* (DRM) (when their behavior models become available). (3) The service descriptions are modular, and declarative. Complex descriptions are assembled by putting together simpler descriptions, supported by strict semantics. This specification approach imposes some uniformity of resource description across CPS sites.

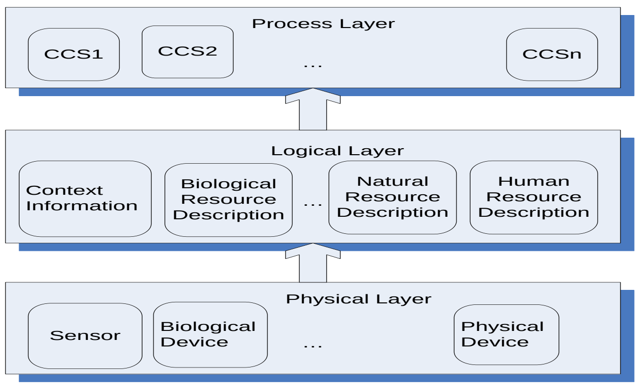

Fig. 1. Three-tiered architecture for CPS

Throughout the paper we suggest the underlying formalism without being formal. In Section II we discuss resource types, and a generic resource description template. This notation is suggested for modeling resources at the physical layer. The merits in our approach are brought out through a brief comparison with other resource modeling approaches. In Section III we give resource class specifications, that resemble Larch [5] specifications. A resource class specification will include a resource type description, and it is extensible. This notation is suggested for modeling resources at the logical layer. In Section IV we give a template for service description, which will include the resource description classes for all resources used by the service. We explain the significance of the service description template, and how it can be analyzed for quality claims. This notation is suggested for modeling resources at the process layer. We conclude the paper in Section V with a brief summary of its significance and our ongoing work.

Manuscript received January 11, 2013.

Kaiyu Wan is with the Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, China. Email: kaiyu.wan@xjtlu.edu.cn

Vangalur Alagar is with the Department of Computer Science and Software Engineering, Concordia University, Canada. Email: alagar@cs.concordia.ca

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.

TABLE I
RESOURCE DESCRIPTION TEMPLATE

Resource: ⟨*generic description of resource name*⟩
Type: ⟨ *resource type*: T⟩
Attribute: ⟨*producer, production facility profile, quality attributes*⟩
Properties: {*physical properties, chemical properties,trustworthiness properties*}
Utility: $\{\langle a_1, u_1\rangle, \langle a_2, u_2\rangle, \ldots, \langle a_k, u_k\rangle\}$
Cost: *cost per unit*
Availability: *available for shipment to all parts of the world or state constraints*
Sustainability: *ratio of demand to supply for the next x years*
Renewability: Reliable period of resource supply
Reuse: *list of applications for reuse of this resource*
Recycling: method names and technology used
Legal Rules for Supply: *URI to a web site*
Other Resources in the Context of Use: *a set of contexts suggesting resource dependencies*
Side Effects: *health and environmental protections*

II. 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* [8].

In general, let $\mathcal{RT}$ be a finite set of resource types. The semantics for each resource type is to be understood from its domain. A resource type $T \in \mathcal{RT}$ is a finite collection of resources of that type. As an example, *Metal* is a resource type, and $\{gold, platinum, iron, copper, zinc\}$ are resources of type *Metal*. The description of one resource r_T of type T is a RDT whose structure is shown in Table I. The RDT table may be extended by adding more element descriptions. We are suggesting that a *model-based formal specification* may be attempted, given the RDT structure, the choice of its description parameters and their types. The tabular RDT format shown in Table I is meant for human agents. An XML version of the RDT is automatically generated from the RDT and is used for resource propagation across CPS processing sites. CPS sites will subscribe to the sites of Resource Providers (RP) in order to receive their RDTs and their periodic updates. Published resource types can be searched at service execution times, in order to get the most recent resource that best fits the service requirements.

Below we explain the semantics related to the RDT shown in Table I.

1) The *Type* of a resource is the *resource category*, as classified earlier or given in industries. We can include more resource types, such as Health (Medical) Resources.
2) The *Attribute* section is used to provide the identity and contact information of resource producer. A general yet concise description of the resource may also be included. Some examples are the following.
(1) *human resources:* If the RDT is to describe the resource physician, an attribute might be 'GP' or 'SURGEON', or 'PSYCHIATRIST'.
(2) *biological resources:* If the RDT is to describe a vegetation, an attribute might be a statement on the specific vegetation type and the geographical region.
(3) *natural resources:* If the RDT is to describe a metal, an attribute might be 'PRECIOUS' or 'RARE EARTH'.
(4) *man made resources:* If the RDT is to describe a transportation vehicle, an attribute might be 'TRUCK' or 'TRAIN'.
(5) *virtual resources:* If the RDT is to describe a software resource, an attribute might be 'PROGRAM' or 'PROTOCOL' or 'MODELING TOOL' or 'SEMAPHORE' or 'BUFFER'.
3) The *Properties* section might include *physical* properties, *chemical* properties, *temporal* properties (persistent or change with time), and *trustworthiness* properties. Below are a few examples.
(1) *human resources:* Academic and professional skills, as will be presented in a Curriculum Vita, constitute the properties of the human resource of interest. If the RDT describes administrator in a company, then the status, academic qualifications, professional expertise, and history of experience are properties.
(2) *biological resources:* If the RDT describes vegetation, then a description of irrigated pasture lands, non-native annual grasslands including the vegetated sand mound, seasonal wetlands, sloughs, and drainage ditches may be included. Additional information on how these are used by a variety of wildlife and its potential for occurrence of special-status species may be included.
(3) *natural resources:* For a metal typical properties include *physical properties* (such as physical state: solid, or liquid, luster, hardness, density, electro positivity, and ductility), *chemical properties* (such as reaction during heating, reaction with water, reaction with acid), and *temporal properties:* (such as distribution characteristics in space (air) and time (seasonal)).
Trustworthiness properties are of three kinds. These are (1) *trustworthiness of the vendor*, expressed in terms of business policies, contractual obligations, and recommendations of peers; (2) *trustworthiness of the utility*, expressed in terms of safety guaranteed for workers, and environmental safety; and (3) *trustworthiness of the product*, expressed as a combination of quality guarantees, and rules for secure delivery of the product.
4) 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 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.

5) The semantics of *Cost* is the price per unit, where the unit definition might vary with resource type. For example, for natural gas the unit may be 'cubic feet', for petrol the unit may be 'barrel or liter'.
6) The semantics of *Availability* is information under the three categories (1) Measured (provable), (2) Indicated (probable), and (3) Inferred (not certain).
7) The semantics of *Sustainability* is related to *Reserves*, *Contingent*, and *Prospective*. *Reserves* expresses a comparison between the measured amount of resource with the current demand. Possible ways to express this are: (1) sufficient: $currentdemand \leq measured$; (2) low: $currentdemand = measured$; (3) high: $currentdemand << measured$. *Contingent* is an estimate (both amount and time period) of getting the reserves (this is a certainty). Is this a 'high' or 'low' estimate? Can the reserve meet the demand during this time period? How soon the 'Indicated' amount of resources will go into production? Possible ways to represent this are: (1) x amount within y days, and $demand$ after y days will satisfy $demand << measured + x$. (2) similar inequalities to express sufficiency, low capacity. *Prospective* specifies the resource quantity determined, and an approximate time scale for its availability.
8) The semantics of *Renewability* is related to the 'perpetual' or 'migratory' nature of the resource. For example, 'solar power' resource can be labeled 'perpetual'; however 'ground water' resource may not be available for ever.
9) The terms *Reuse and Recycling* are well understood both in technology and in environmental applications. A resource r_1 may be used in an application for a certain period of time. During this period the resource may decay, as in the case of iron used in producing trucks or aluminum used to manufacture aeroplanes. The recycling process re-produces the resource with its original specification. The reuse of a resource refers to the use of the resource in multiple applications. A robot might be used to act as a fire fighter, or just to flush away contaminated water. When a resource r_1 is used to produce a resource r_2, and in turn the resource r_2 is used to produce another resource r_3 we also have reuse of r_1.
10) The semantics of *Legal Rules* include the business rules of the RP, the government regulations governing the distribution of resources, and international rules regarding quality of resources.
11) The meaning of *Other Resources in the Context of Use* is to express 'resource dependency'. Examples of dependencies may be expressed using *before*, *during*, and *following* temporal operators. As an example, to extract coal from a mine, some natural resources such as power and other man-made resources are required *before* production commences. As another example, if a robot is used to pick the debris from a hazardous environment, an autonomous vehicle is required immediately *following* it in order to place the debris and move it.
12) The intent of *Side Effects* section is to list the impact and interference effects with environment.

Comparison

We compare our RDT with the UML modeling approach [6], RDF [7], the Resource Space Model (RSM) [3], and the entity-relationship model [4].

- Modeling resources with UML is the first method proposed with respect to modeling run-time resources for real-time systems. Resource properties and resource dependencies are not part of the model, however resources required for a service can be modeled. The model is service-centric and not resource centric. It might be possible to develop resource-centric UML models at all levels, but we have not attempted this. Given the distributed nature of the resources in CPS it might be hard to manage and use UML models.
- RDF is meant to describe Web resources, which according to our classification are *Virtual resources*. Since RDT is meant for all types of CPS resources we expect that all Web resources can be represented as RDTs. We have not come across RDF examples which include all RDT aspects. In particular, it is not clear as to how Availability, Reuse, Legal Rules for Supply, and Context of Use can be specified in RDF.
- The RSM method considers the resource space as multi-dimensional where each dimension is a resource type. So, essentially RSM produces a model at the logical layer, however it is oriented towards an application. RSM is not resource-centric and its models require a centralized management in order to avoid inconsistencies.
- The Resource-Explicit Service Model (RESM) proposed in [4] is similar to an ER (Entity Relationship) diagram. They consider physical devices as resources, and model resources, the services offered by them, and the service contexts as a bundle in a single ER diagram. This approach suggests that resources and services should be modeled together although the emphasis is on resources, and services offered by the resources always match with the services required by a consumer. A soft real-time application in CPS requires an open market approach. An open CPS network is a loosely coupled system, and it is best to avoid tight coupling between resource and service models. A service provider in CPS should be free to choose the best resources in order to fulfill a service request.

Our modeling approach emphasizes separation of concerns and modularity. An RDT is created by a RP independent of a service that might be created by a Service Provider (SP). A modification to a RDT produces a new RDT which is published by the RP and can be acquired by SPs in the CPS network. The RDT notation is suitable for the physical layer. The logical and process layer modeling include the RDTs, and thus the resource specifications are modular. The Reuse section in RDF adds one more level flexibility by explicitly stating the alternate uses of a resource. The RDT notation is richer than other notations, because it allows 'user defined types' to be introduced with their semantics and operations. In Table II we show the RDT model for *robot* resource, one of the three resources required for a rescue mission discussed in [4]. This RDT description has more information on resources than the ER diagram of the RESM model.

TABLE II
RDT FOR ROBOT

Resource: ⟨*description of robot used in a rescue mission*⟩ Type: ⟨ *man made resource* ⟩
Attribute: *XYZ Manufactures* *link to production facility profile* *6 degrees of freedom, mobility in mountain terrain, forests, and in hazardous places*
Properties: Weight: 30 Kg Height: 1.2 meters Arms: 2 Vision: *fitted with camera for night vision* Payload: *can lift and move 50 Kgs load at a time*
Utility: ⟨*navigating hazardous environment, critical*} ⟨*forest fire containment, critical*} ⟨*emergency road service, recommended*} ⟨*auto assembly plant, essential*}
Cost: \$5,000 *per hour of rental - not for sale* Availability: *available for shipment to all parts of China* Sustainability: 100% Renewability: *YES*
Reuse: *surveillance in forests* *clean atomic waste*
Recycling: *YES* Legal Rules for Supply: *URI to a web site*
Other Resources in the Context of Use: *needs a separate vehicle for transporting to the work site* *needs a truck for storing the debris* *needs an ambulance to place survivors*
Side Effects: *NIL*

III. LOGICAL LAYER DESCRIPTION

For the resource-centric CPS model we need to follow the resource-centric service approach. In our approach, the activities in the service are ordered, and the list of activities per single resource are handled taking into account resource dependencies. 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.

The structure of RCS, shown in Table III, resembles a Larch trait [5]. The semantics of Larch is adapted to give semantics to the different clauses in a RCS. Thus, the meaning of the different clauses in it are as follows: (1) The clause *Resource Class* introduces the name RC of the specification. (2) The **includes** clause states that the RDT defining resource r is specified in RC. The effect is that all the information in the included RDT is exported to this specification. (3) The **requires** clause specifies a list of the resources that are *packaged* together with r. These resources are necessary to make r operational. This list may be empty, in which case the resource r is self-sufficient and will be exported to service execution phase. If the list is not empty, the resource r together with all the resources included in this list will be exported as a package to a service. Note that, the resources included in the section "Other Resources in the Context of Use" of the RDT r are required for a service that requires r, in addition to the resources listed in the **requires** clause. (4) The clause **consumed-by** lists the tasks or resources for which r may be needed. Each task listed in this clause is an atomic activity belonging to at least one application domain listed in the Utility section of the RDT r. (5) The **constraints** clause lists *resource* constraints, *compatibility* constraints, and *dependency* constraints. Resource constraints are dependent upon the type of resource r and the context of its use. They may include *minimum* and *maximum* units of resource r that will be available in specific contexts, and a list of byproducts arising from the use of resource r. The compatibility constraint is a relationship between the resource r and the tasks consumed by it. That is, resource r is compatible with two tasks τ_1 and τ_2 if they can share the resource, that is, both these tasks can be concurrently processed. The dependency constraint can be a relationship between two resources listed in **requires** clause or it can be a relationship between two resource class specifications. In the former case we include the dependency constraints, written $\tau_i \ll \tau_j$, in the **constraints** clause. To describe the later case let us assume that RC_1 and RC_2 are resource class specifications for resources r_1 and r_2. Suppose there exists a context c in which the resource r_1 should be used *before* resource r_2 is used, then the class RC_2 is dependent on class RC_1. That is, all tasks listed in RC_1 must be completed before starting the tasks in RC_1. We use the notation $RC_1 \overset{c}{\Leftarrow} RC_2$ to show class dependency and include it in **constraints** clause of RC_1. Class dependencies are local to a site where resources are produced. A class specification for robot RDT (Table II) is shown in Table IV.

TABLE III
SYNTAX FOR RCS

Resource Class RC
    includes *RDT* r
    requires $\{$*RDT* $r_1, \ldots,$ *RDT* $r_k\}$
    consumed-by
    $\{\tau_1, \ldots, \tau_n\}$
    constraints
    $\{\sigma_1, \ldots, \sigma_m\}$

Flexible RCS: Modification and Extension

It is impossible for a RP to know a *complete* list of tasks for which a resource may be required. So necessarily the published resource information may only be incomplete. Also, the RCS for a resource is independent of the *actual* set of tasks demanded by CPS services. Consequently the RCS for a resource should adapt to changes in the set of tasks attributable to a resource, resource dependencies and constraints, and changes in contextual information. To meet

TABLE IV
ROBOT RESOURCE CLASS SPECIFICATION

```
Resource Class RobotClass
        includes RDT robot (Table II)
        requires {RDT ElectricPower}
        consumed-by
        { Load, Pick, Place, Assemble, Recognize }
        constraints
        {
                resource constraints:
                  limited visibility in unknown territory
                  not suitable for underwater exploration
                compatibility constraints:
                  ⟨Load, Place⟩, ⟨Pick, Place⟩
                dependency constraints:
                  LightEquipment Class $\xleftarrow{rescue}$ Robot Class
        }
```

TABLE V
SYNTAX FOR RESOURCE CLASS REFINEMENT

```
RCS Refinement NRCS
        includes Resource Class RCS
        extended-by
                ⟨ new tasks ⟩

        modified-as
        {
                new constraints
                modified constraints
        }
```

this requirement we introduce the structure and semantics of a *flexible* RCS. The flexibility of RCS is that it is both *extendable* and *modifiable*.

The class specification for a resource r can be modified in order that new tasks and new constraints are added, or existing constraints are modified. Note that existing tasks in a RCS are not permitted to be deleted or modified. The syntax for class modification is shown in Table V. The name of the new class is $NRCS$. The syntax to create $NRCS$ has three clauses. The **includes** clause lists the RCS to be modified. The clause **extended-by** lists the information to be added to the **consumed-by** clause of the included RCS. The effect of **extended-by** clause is to add new tasks to the list of tasks in the class RS, without deleting or modifying any of the listed tasks in RS. The clause **modified-as** lists the information which will overwrite (replace) the existing information in the **constraints** section of the class RS included RCS. The syntax is a short-hand notation to create a new RCS. A RCS can be viewed as a table and its XML version is for CPS processors. The RCS notation is employed to bring in conciseness and semantics.

IV. PROCESS LAYER MODEL

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

TABLE VI
ROBOT CCS

Service	Function:	*Name:* Robot CCS *Pre:* valid(credit_card) $\wedge$ valid(professional_license) *Post:* Deliver
	Resource:	*Resource Class:* Robot CCS *Provider Data:* Company Name: XYZ-Rent-A-Robot
	Non Functional:	*Normal Rental Cost:* 5000$ per day *Deposit:* 5000$
Contract:	Trust Attributes:	*Resource:* *Safety:* no damage to environment *Security:* auto shut in case of error *Reliability:* no record of malfunction *Availability:* for business customers only *Service:* *Security:* encrypted transactions *Availability:* 24 hours everyday *Provider:* Consumer rating: $\frac{4.1}{5}$ Organization rating: 5 $\star$ recommendation awarded by BBB
	Legal: Exceptions:	*Liability insurance:* not covered for personal injury *Zonal Violations:* must be paid by the renter *Renewal of Contract:* not automatically renewable *Return:* must be returned to the same location *Refund:* damages recovered from the deposit
	Context	*Context Info:* Provider: [LOCATION:Shanghai] Execution: [Time:contract time(date)] *Context Rule (Situations):* Consumer Related: business rating must be at least AA Delivery Related: free shipping for places within 100 kms from Shanghai for other places the shipping charge should be paid by the renter

included. Such configured services are published in the site of the SP. 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. The structure of CCS for Robot CCS is illustrated in Table VI.

Complex CCS Representation

TABLE VII
SYNTAX FOR CREATING RESCUE MISSION CCS

Service Name: *RescueMission CCS*
  includes *CyberConFiguredServices*
    RobotCCS, AmbulanceCCS,
    TruckCCS, LightEquipmentCCS

  extensions {
      ⋮
  }

  modifications{
      ⋮
  }

We illustrate the CCS specification for the rescue mission example discussed by Huang et al. [4]. The rescue mission requires a *complex* service, involving services of a robot for rescuing survivors, ambulance services to transport people to hospitals, services of a truck to remove debris, and services of a lighting equipment. So, there are four resource types required for the rescue mission. For each resource type we assume that the RDT and the RCS have been developed by the respective RPs. The SP who offers the emergency rescue service creates separately the four configured service specifications *RobotCCS*, *AmbulanceCCS*, *TruckCCS*, and *LightEquipmentCCS*. The rationale for their separate development is two fold. One is *reuse* potential, in that each CCS can be used individually in other service creations. Second, they can be combined in many ways dynamically, as and when a service provision context arises. In the case of the rescue mission example, all the four CCSs are required. In case there is a demand for emergency roadside assistance perhaps the Truck and LightEquipment services are sufficient, and so the SP can create a complex service using *TruckCCS* and *LightEquipmentCCS*, modifying the contract part following the structure in Table V. For the emergency rescue mission example, the SP puts together *RobotCCS*, *AmbulanceCCS*, *TruckCCS*, and *LightEquipmentCCS* as in Table VII. The semantics of the specification in Table VII is the following: In the **includes** clause the CCSs that are necessary for the complex service are listed. The **extensions** clause will include additions to the non-functional and trust attributes of the included CCSs. The **modifications** clause will list changes and additions to the contract part of the included CCSs. We emphasize that no change will be made to the functionality of the included configured services and the resources used to produce them. In essence, the syntax in Table VII is intended to be used by SPs in the service execution layer.

V. CONCLUSION

In this paper we proposed a model-based language to specify resource and resource-centric services through three-tiered approach. The RDT table structure, given its semantics, can be turned into a lightweight formal description. We import the RDT specifications within resource class specifications which are written in Larch style. Cyber configured service specifications are also declaratively written in Larch style. Thus RDT, RCS, and CCS all have set theory and logic semantic basis. Moreover context formalism is also founded on relational semantics. Therefore, the semantic basis in a tier is consistent with the semantic basis in all tiers below. Consequently formal validation of service claims are possible if they are stated in first order logic. Thus a claim verified in a tier is not contradicted in higher tiers. We have suggested rigorous methods for evaluating three kinds of SAT relations [8]. We are still working on this aspect. To validate the quality claims made for resources themselves we would need scientific evidence and engineering analysis of their respective resource types. For example, the precision and accuracy of gripping operation in a robotic device would need an analysis based on the mechanics behind the design of the robot. So, validation issues are hard to tackle; however, the specifications suggested in this paper will enable the claims to be stated formally, a first step towards validation. Clearly, verifying resource claims is a broader challenge which needs investigation by domain experts. Since all the quality claims of resources may not verifiable using software, a tight coupling exists between what experts can do and what machines can be made to do.

The semantic basis is also necessary for developing tools. We are currently developing a *Graphical Resource Editor* (GRE). The goals are (1) to provide assistance to developers in creating the specifications at the three layers, (2) to automatically generate XML files that can be shared by CPS nodes, and (3) to enable a formal resolution of SAT claims by providing links to other verification tools. The GRE tool that we have completed enables the creation of RDTs and their XML versions. Protecting resources, assuring confidentiality in service provision, and privacy of CPS clients are the three challenges to be faced in making CPS survive attacks. In the three-tired architecture that we have proposed these three issues can be addressed separately at each tier. Importing a secure lower layer into the next higher layer enables security verification compositional. As a prerequisite to service layer confidentiality, resource models must be protected. As a simple first step solution the tool enforces access control rights for RPs and SPs. The intent is to ensure the integrity of resource information. SPs can use, but not modify RDTs, and resources allocated to the service bought by a SP are assured to be the resources included in the CCSs viewed by the clients. Thus, deception attacks can be detected, if not prevented, at source. Currently we are working on resource protection issues for other layers.

REFERENCES

[1] D. Georgakopolous and M. P. Papazoglou. *Service-oriented Computing*. The MIT Press, 2008.
[2] C. S. Group. Cyber-physical systems: Executive summary. Report, http://varma.ece.cmu.edu/summit/CPS-Executive-Summary.pdf., 2008.
[3] Y. H. Zhuge and P.Shi. Resource space model, owl and database: Mapping and integration. *ACM Transactions on Internet Technology*, 8(4), 2008.
[4] I.-L. Y. Jian Huang, Farokh Bastani and J.-J. Jeng. 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] J. J. H. John V. Guttag and J. M. Wing. The larch family of specification languages. *IEEE Transactions on Software Engineering*.
[6] B. Selic. A generic framework for modeling resources with uml. *IEEE Computer*, 33(6):64–69, 2000.
[7] W3C. W3c recommendation. Technical report.
[8] K. Wan, Vangulur Alagar *A Resource-centric Architecture for Service-oriented Cyber Physical System*. The 8th International Conference on Grid and Pervasive Computing (GPC 2013), May 2013, Korea.