Specifying Resource-Centric Services in Cyber Physical Systems

KaiyuWan, VangalurAlagar and Yuji Dong

Abstract A service-oriented view of Cyber-Physical Systems (CPS) is a good platform for managing global supply chain management, service acquisition and service provision. A necessary condition for complex service delivery is that resources required for complex services are of high quality and are available at service execution times. Therefore in a resource-centric service model, both resource quality and service quality using that resource are explicitly stated. In this paper a cascaded specification approach is discussed for describing resource types, services offered by resource, and a cyber configured service (CCS) that package physical services. Energy support from internal-combustion engine is regarded as a resource-centric complex service and discussed as a case study to illustrate our specifying and modeling approach.

Keywords Cyber physical systems · Resource · Resource description · Resource management · Resource specifying · Service model

K. Wan (✉)
Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China
e-mail: kaiyu.wan@xjtlu.edu.cn

V. Alagar
Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada
e-mail: alagar@cs.concordia.ca

Y. Dong
Research Assistant at Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China
e-mail: wildfire1106@gmail.com

G.-C. Yang et al. (eds.), *Transactions on Engineering Technologies*,
Lecture Notes in Electrical Engineering 275, DOI: 10.1007/978-94-007-7684-5_7,

1 Introduction

Cyber-Physical Systems (CPS) [1] 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 [2].

In [3] we proposed the three conceptual layers of CPS resources as *physical*, *logical*, and *process* through three-tiered approach, as illustrated in Fig. 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

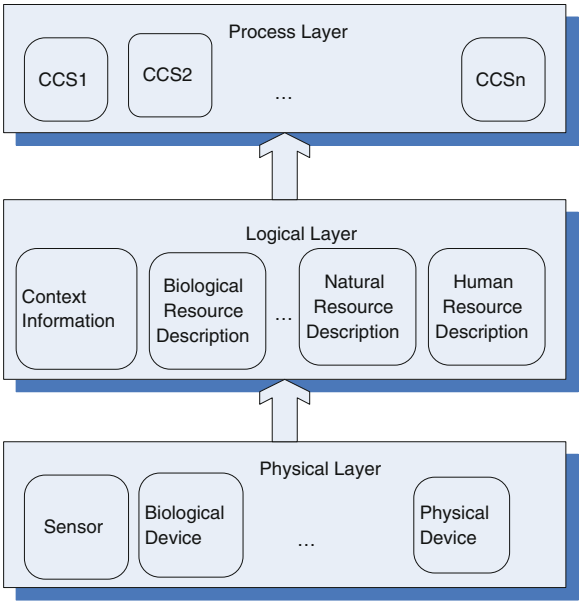

Fig. 1 Three-tiered architecture for CPS

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.

Throughout the paper we suggest the underlying formalism without being formal. In Sect. 2 we explain our view on service model. In Sect. 3 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 Sect. 4 we give resource class specifications, that resemble Larch [4] 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 Sect. 5 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 Sect. 7 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 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 Fig. 2, 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.

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

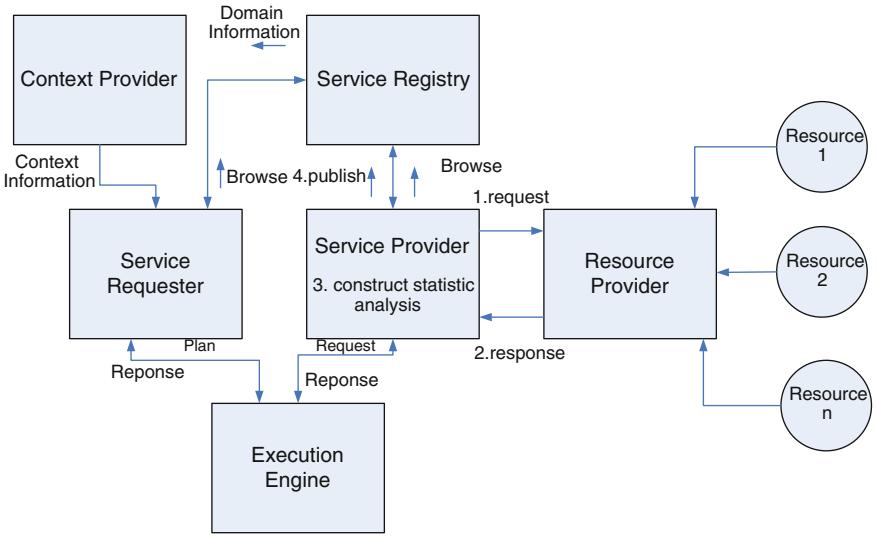

Fig. 2 Resource-centric abstract service model

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.

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.

3 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* [3].

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 1. 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 1 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.

We have explained the semantics in details in [5], therefore below we explain the semantics briefly.

Table 1 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,temporal 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*

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 may include *human resources, biological resources, natural resources,man made resources, and virtual resources*.
3. The *Properties* section might include *physical* properties, *chemical* properties, *temporal* properties (persistent or change with time), and *trustworthiness* properties.
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.
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. *Contingent* is an estimate (both amount and time period) of getting the reserves (this is a certainty). *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.
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.
12. The intent of *Side Effects* section is to list the impact and interference effects with environment.

We compare our RDT with the UML modeling approach [6], RDF [7], the Resource Space Model (RSM) [8], the entity-relationship model [9], and Service-Oriented Middleware Architecture (WebMed) [10].

- 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 [9] is similar to an Entity Relationship (ER) 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.
- WebMed is a early conceptual middleware designed with a service-oriented view point to support CPS applications. It enables access to the underlying smart devices and integration of its device specific functionality with other software services through five components such as WebMed node, Web service enabler, service repository, engine, and application development. WebMed tries to provide an easy interface with physical devices. However it didn't support resource modeling and management.

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 2 we show the RDT model for *gasoline 97* resource, which is considered a resource for energy support from internal-combustion engine at cars.

4 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.

Table 2 RDT for gasoline 97

Resource: ⟨*description of unleaded gasoline 97*⟩
- Type: ⟨*man made resource* ⟩
- Attribute:
  - *Opet product*
  - *refuelled at XYZ gas station*
  - *passed ISO 14064-1 quality management system*
- Properties:
  - Appearance: Clear and bright
  - Density(@15°C): 770–775 kg/m^3
  - Vaporization percentage @100°C: 46–71 % volume
  - Vaporization percentage @150°C: 75 % volume
  - Final boiling point: 210°C
  - Distillation residue: 2 % volume
  - Oxidation stability: 360 min
  - Research octane number RON: 97,0
  - Motor octane number MON: 86,0
  - Lead: 5 mg/L
  - Sulfur: 10 mg/kg
- Utility:
  - ⟨*combustion in engine to supply heat, 1*⟩
- Cost: $1.43/litre
- Availability: gas stations supported by Opet
- Sustainability: 100 % in reasonable years
- Renewability: *NO*
- Reuse: *automobile engine*
- Recycling: *NO*
- Legal Rules for Supply: *URI to a web site*
- Other Resources in the Context of Use:
  - *needs related equipments to store and transfer gasoline*
  - *needs an engine to convert chemical energy to kinetic energy and electrical energy*
- Side Effects: *gases from combustion, CO^2 for greenhouse effect, oxide from Lead, Sulfur, etc. for air pollution*

Table 3 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
$\{\sigma_1, \ldots, \sigma_m\}$

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 3, resembles a Larch trait [4]. 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, therefore, 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

Table 4 Gasoline 97 resource class specification

Resource Class Gasoline97Class
includes *RDT gasoline97* (Table 2)
requires
{*RDT Storage*; *RDT Transfer*; *RDT Filtered_Air*}
consumed-by
{ *Mix_With_Air*, *Compression_stroke*, *Power_stroke*, *Exhaust_stroke* }
constraints
{
resource constraints:
high quality requirement (industrial standard)
correct mix ratio with clean air
compatibility constraints:
⟨*Mix_With_Air*, *Compression_stroke*, *Power_stroke*, *Exhaust_stroke*⟩
dependency constraints:
Filtered_Air Class $\xleftarrow{combustion}$ *Gasoline_97 Class*
}

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 2) is shown in Table 4.

5 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 *CyberConfigured-Service* (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. 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

Table 5 Engine CCS

Service	Function:	*Name:* Gasoline_97 CCS *Pre:* check(quality) $\wedge$ check(quantity) check(equipments_specify) $\wedge$ burn(gas,air,energy) *Post:* Mix_with_FilterAir $\wedge$ MechanicalEnergy
	Resource:	*Resource Class:* Gasoline_97 CCS *Provider Data:* Company Name: Opet
	Non functional:	*Gasoline Cost:* \$1.43/litre *emission:* CO^2
Contract	Trust attributes:	*Resource:* *Safety:* industrial standard *Security:* amount calculating and error monitor *Reliability:* no record of malfunction *Availability:* 99.9999% *Service:* *Security:* intelligent control system *Availability:* 99.99% *Provider:* Consumer rating: $\frac{4.1}{5}$ Organization rating: 5 $\star$ recommendation awarded by BBB
	Legal exceptions:	*Liability insurance:* not covered for intentional injury or Non-Countervailable accident *Renewal of Contract:* not automatically renewable *Maintaining:* must be maintained at the official authorizing place *Refund:* damages and depreciations considered
	Context	*Context Info:* Provider: [LOCATION:Shanghai] Execution: [Time:contract time(date)] *Context Rule (Situations):* Consumer Related: related manufacture must compatible to this engine Delivery Related: free shipping for places within 100 kms from Shanghai for other places the shipping charge should be paid by the consumer

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 Engine CCS is illustrated in Table 5.

Complex CCS Representation We take the energy support from internal-combustion engine as an example and illustrate the CCS specification accordingly. In reality the energy support system from internal-combustion engine in vehicle is a extremely complex service. For simplicity, we model the service involving fuel supply, air supply, working engine to convert energy, electrical system to help engine and get

Table 6 Syntax for creating energy support from internal-combustion engine CCS

Service Name: *EnergySuportMission CCS*

includes *CyberConFiguredServices, FuelCCS, EngineCCS, SensorCCS AirCCS, ElectricalCCS, AgentCCS, TransferCCS, ProtectionCCS*
extensions {
⋮
}
modifications{
⋮
}

energy from engine, related manufactures to transfer fuel, air, heat and electricity, sensors and agent system to measure and make decision, and protection system to deal with error. So, there are eight services required for the energy support. The SP who offers the energy support service creates the eight configured service specifications including *FuelCCS*, *AirCCS*, *EngineCCS*, *SensorCCS*, *ElectricalCCS*, *AgentCCS*, *TransferCCS* and *ProtectionCCS* and puts them together as shown in Table 6. The semantics of the specification in Table 6 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 6 is intended to be used by SPs in the service execution layer.

We decide to develop them separately. One rationale is *reuse* potential, in that each CCS can be used individually in other service creations. For example, there are many different kinds of fuels used in different kinds of environments while the fuel supply service might be quite similar. The service structures can be used in same way but different places by just relying on different resources. Second, they can be combined in many ways dynamically, as and when a service provision context arises. In the case of the energy support system in vehicle from internal-combustion engine, all the eight CCSs are required. In another situation when energy support system is required at aircraft perhaps these eight CCSs are not sufficient. The SP may need to add more modules such as *BalancingCCS* and *CoolCCS* into the entire energy support system. Thus the SP can create a complex service using the two additional *BalancingCCS* and *CoolCCS* together with the eight CCSs, by modifying the contract part of the structure. Furthermore, *EngineCCS*, *SensorCCS*, *ElectricalCCS*, *AgentCCS*, *TransferCCS*, *ProtectionCCS* and other specific CCSs can be included to model and specify some other CPS systems like intelligent irrigation system, robots rescue systems, intelligent Manufacturing systems etc.

6 Toolkit Implementation

The semantic basis is 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.

According to the formal definition and semantics for resources that we proposed, we implemented a graphical user interface (GUI) at each CPS site for humans and systems to interact, share, and yet securely manage resources across CPS [11]. The rationale for GUI is that humans who manage resource will find it user-friendly, and the mechanism that we build behind RDT will faithfully transform resource information in languages that can be shared and communicated securely across the CPS. A Framework for GUI which supports RDT has been discussed in [11]. However Resource management is a multi-step activity. The implementation is only the first step. We may consider Resource Discovery, Resource Acquisition, Resource Modeling, Resource Publication, and Resource Allocation as the distinct layers in resource management. Therefore a complete GUI should be implemented to fulfill resource management.

Some of the benefits and key features in the design of GUI are as follows:

1. Comprehensive and complete visibility of resource availability, resource requests, and resource allocation is possible for the business enterprise.
2. Local and global pool of resources can be assessed for a project, regardless of physical location.
3. Within GUI, it is possible to view and manage resource utilization by graphical plug-ins.
4. Modifications to resource bookings can be monitored and dealt with by real-time reallocation of resources to other projects.
5. Security settings of key aspects of resource knowledge can be distributed across the multiple layer.

7 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 [3]. 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 energy consumption in a vehicle would need an analysis based on the mechanics behind the design of the energy support system. 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.

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. C. S. Group (2008) Cyber-physical systems: executive summary. Report, http://varma.ece.cmu.edu/summit/CPS-Executive-Summary.pdf
2. Georgakopolous D, Papazoglou MP (2008) Service-oriented vomputing. The MIT Press, Cambridge
3. Wan K, Alagar V (2013) A resource-centric architecture for service-oriented cyber physical system. In: Proceedings of the 8th international conference on grid and pervasive computing (GPC 2013), May 2013, Korea
4. Guttag JV, Horning JJ, Wing JM (1985) The Larch family of specification languages. IEEE Softw 2(5):24–36
5. Wan K, Alagar V (2013) Modeling resource-centric services in cyber physical systems. In: Proceedings of the international multi conference of engineers and computer scientists 2013, Lecture notes in engineering and computer science, 13–15 Mar 2013, Hong Kong, pp 716–721
6. Selic B (2000) A generic framework for modeling resources with uml. IEEE Comput 33(6):64–69
7. W3C. W3c recommendation. Technical report
8. Zhuge YH, Shi P (2008) Resource space model, owl and database: mapping and integration. ACM Trans. Internet Technol 8(4):20
9. Yen IL, Huang J, Bastani F, Jeng JJ (2009) Toward a smart cyber-physical space: a context-sensitive resource-explicit service model. In: Proceedings of 33rd annual IEEE international computer software and applications conference, IEEE Press

10. Hoang DD, Paik HY, Kim CK (2012) Service-oriented middleware architectures for cyber-physical systems. Int J Comput Sci Netw Secur 12(1):79–87
11. Wan K, Alagar V, Wei B (2013) Intelligent graphical user interface for managing resource knowledge in cyber physical systems, KSEM 2013, LNCS 8041. Dalian, China