Journal of Theoretical and Applied Information Technology
15th June 2018. Vol.96. No 11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SERVICE COMPOSITION ALGORITHMS IN CYBER PHYSICAL SYSTEMS

[1]SWATI NIKAM , [2] RAJESH INGLE

[1]Research Scholar, DIT, Pune, Maharashtra, India

[2]Professor, PICT, Pune, Maharashtra, India

E-mail: [1]swatinikam3@gmail.com, [2]ingle.ieee@org

ABSTRACT

Service composition in Cyber Physical Systems (CPS) means integrating individual services used for different purposes which individually cannot accomplish the goal, but if integrated then it can achieve a specific goal. So there is need of combining various services into one composite service to satisfy complete requirement. CPS is an emerging field in which cyber and physical world interact closely. By nature CPS is application oriented so the need of composing the existing services arises frequently. Hence understanding and resolving the service composition issues in the context of CPS becomes very important. Service composition is very well studied in Web service, Cloud Computing, Grid Computing and Wireless Sensor Network domain. Service composition work has initiated in CPS domain also, but still it lacks in maturity as compared to other domains. Service composition in CPS becomes critical firstly because of the dynamic and unpredictable nature of CPS which comes from the involvement of cyber and physical domain. Secondly lot of heterogeneity is observed in CPS components which range from simple sensors, actuators to high end computing devices. Thirdly resources also needs to be considered in the process of service composition and last but not the least, while looking at the practical applications of CPS, it needs to be considered in networked CPS context. Selecting best individual services for service composition is the main problem which is addressed in this paper. A middleware is designed for performing service composition and also phase wise algorithms for service composition are presented. Two significant methods of Multi Attribute Decision Making (MADM) methods are used to solve the service selection problem. Algorithms are tested in simulated environment with different scenarios to check suitability of MADM methods for service selection problem. The observation is few significant methods of MADM like PSI can be used to select best service for service composition.

Keywords: *Cyber Physical System, Service Composition, Quality of Service (QoS)*

1. INTRODUCTION

Cyber-Physical Systems have emerged in recent years as a new technological revolution to support a collection of resources in the execution of physical processes. The National Science Foundation (NSF) CPS Summit[1] defines CPS as "Physical and engineered systems whose operations are monitored, coordinated, controlled and integrated by a computing and communication core". Various research challenges in CPS are presented in [2,3]. Detailed study of three major research challenges namely service composition, resource provisioning and autonomics is given in [4]. Service composition is well studied in various domains like Web service [5], Pervasive computing [6], Opportunistic network [7], Cloud computing [8], Internet of things [9] and many more, but CPS service composition lacks in maturity. As CPS service consists of lot of resources which are present in the physical world, so service composition problem cannot be solved only by considering the service, but it has to be considered in the context of resources as well. Our work focuses on phase wise service composition method in the context of CPS where numerous resources like sensors and actuators are also considered. So when service composition is done efficiently, the required services can be combined as per the requirement of user. Section 2 gives an overview of related work of service composition in the field of CPS. Section 3 gives an idea about the background of service composition. Section 4 gives detailed problem formulation along with execution flow and algorithms. Section 5 elaborates experimental setup followed by results and analysis in Section 6 whereas section 7 is comparison with the work

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

done by other researchers followed by concluding remarks in section 8.

2. RELATED WORK

Many researchers have worked towards service composition problem [10-16]. The related work found can be categorized as per the service composition phases. Some of them have worked on individual phases like service discovery, service composition, service deployment and service execution. But very limited literature is present in the context of CPS which has considered phases of service composition. Yajing Zhao[10] have discussed about collaboration problem amongst multiple CPS in which they have extended OWL-S framework to address the functionalities. They have used the abstract service and concrete service representation. Hell Brucke [11] has given details about name centric service architecture for CPS. Tao Wang [12] has presented efficient context-aware service composition framework along with algorithm of atomic service filtering algorithm. Service composition problem is discussed as two phase context sensitive service composition optimization problem[13] which is solved using particle swarm optimization method. They have also proposed PE-SOA ontology model for the same along with case study discussion of traffic accident rescuing task. It talks about implementation results but comparison with other algorithms or results is not present. J.Huang[14] have extended the conventional SOA model, where PE-ontology is used to connect different physical entities based on their capabilities (services they can provide) and PE-SOA model helps in service specifications that are suitable for physical entities. Thus it simplifies PE service specifications and reduces the complexity of the reasoning procedure. Jian Huang[15] has discussed context-sensitive resource-explicit service model and for service composition, an AI planning technique is used. They have enhanced graph plan algorithm for context consideration. The discussion is limited to framework and a case study, but there is no discussion on implementation and results. S. Wang[16] has discussed service composition problem in cyber physical social systems using mixed integer programming approach but the service characteristics are like a web service. As discussed above, many of the researchers have worked either on framework or given theoretical discussion on ontology. Also very few have captured CPS characteristics and discussed all phases of service composition. While choosing optimal services also many of the researchers have considered either one or two attributes. Using either skyline operator or dominance relation they have minimised search space. Whereas optimal service selection based on all QoS attributes is not considered. In this work the focus is on phase wise service composition as well as solving optimal service selection problem where all the attributes of CPS service are considered.

3. BACKGROUND OF SERVICE COMPOSITION

The lifecycle of CPS service composition is different than a web service composition lifecycle [17] because web service gives only software service whereas CPS service consists of resources as well. Similarly CPS service composition phases can be written as shown in figure 1.

Phase1: Service Definition: Service is defined along with their input and output.

Phase 2: Service Discovery: Services are discovered by matching input and output of the service.

Phase 3 : Service Selection : It is likely that more than one candidate service will meet the requirement which are same in functionality but are different in QoS. So from this huge set of services an optimal service is selected.

Phase 4: Service Dependency Resolution and Scheduling: Here all the service dependencies are resolved and the priorities of services are decided.

Phase 5: Service Deployment: In this phase constructed composite service is deployed to allow its instantiation and invocation. For deployment the optimal resource system is selected. The output of this phase is called as executable composite service.

Phase 6: Service Execution: In this phase, the composite service instance will be created and executed.

This phase wise detailing of service composition helps us to understand how these different phases impact overall service composition process. The execution flow of the service composition process is shown in figure 2

Step 1: User input request is submitted for service composition request in the form of task list along with the requirement on quality of service.

Step 2: The input request may or may not contain dependencies. If the task dependencies are present then they are resolved.

Step 3: The runtime CPS model is generated which consists of all the runtime information of existing CPS domains along with resource systems registered under each CPS systems and resource nodes registered under each resource systems.

Journal of Theoretical and Applied Information Technology
15th June 2018. Vol.96. No 11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

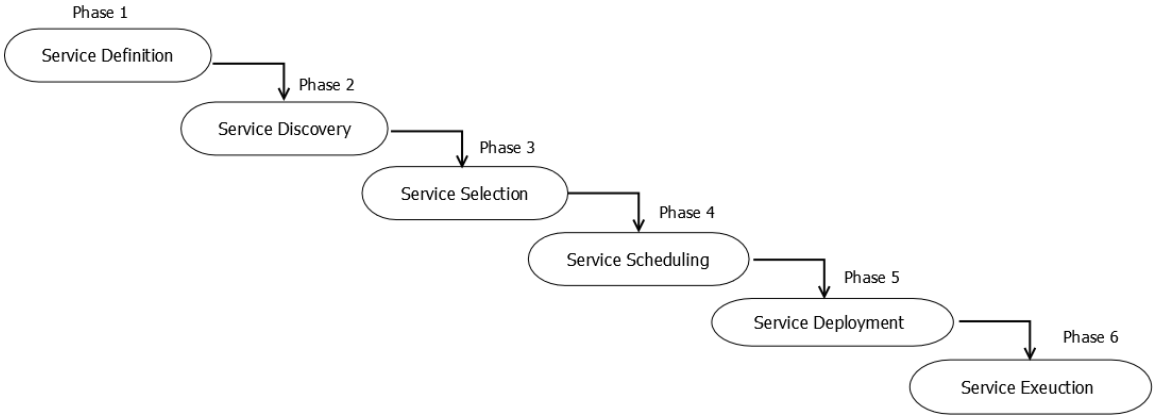

Figure 1. Service Composition phases

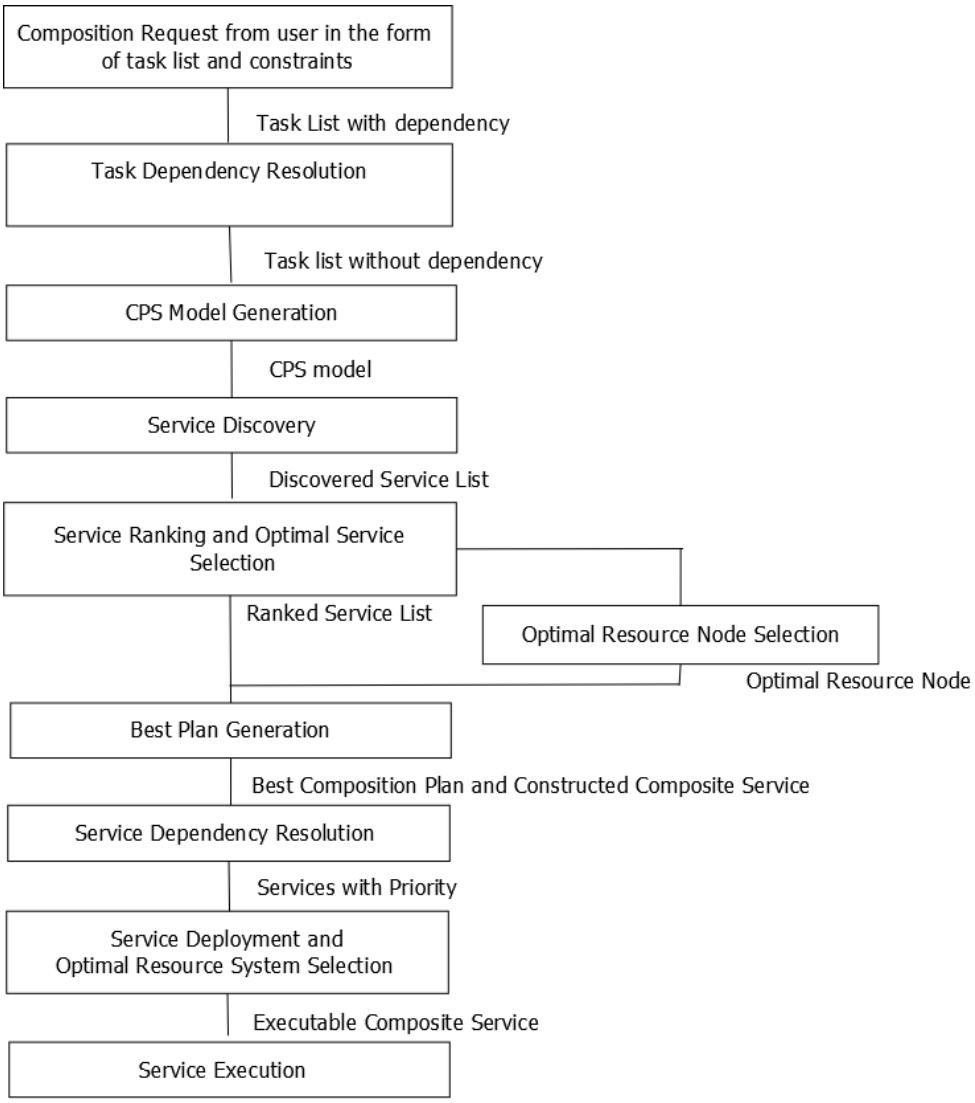

Figure 2. Service Composition Flow

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step 4: The resources may get impacted by the context in which it is present, hence only the currently available resources and its corresponding services should be reflected while generating the CPS model. So at resource system, service filtering based on availability of resource node is done i.e. the resources which are currently not available are not considered for service composition and hence its corresponding services can not be considered as a candidate services for service composition. Those services will be filtered in the CPS model. Later on when the resources becomes available, then it can be considered for service composition. So all resource systems hold a list of services which are the candidate service at that instant whose corresponding resources are available. So these systems are discovered in service discovery phase and form a discovered service list.

Step 5 : While composing services, for a particular task many candidate service with the same functionality but different quality of service attributes may be found in step 4. All the discovered services many not prove suitable as a candidate service for service composition process, hence they need to be ranked as per their quality of service attributes which is done in this step. Now the service list contains the list of services which are arranged in ranked order as per their quality of service. The topmost service is now the best service.

Step 6: But only choosing the best service for service composition may not be sufficient as these services are given by resources and hence the corresponding resource should be provisioned. So the optimal resource is selected for resource provisioning.

Step 7: The services chosen for each input task are the elements of composition plan. So the ranked services are selected and a constructed composite service list is formed. The aggregated attributes are calculated considering all the elements of this constructed composite service. Then the plan is evaluated to see that this combination of services in constructed composite service list is best or not using best plan generation algorithm and as an output now the best plan is generated.

Step 8: If any service dependencies are present then they are resolved and services are scheduled.

Step 9 : In this phase the constructed composite service is deployed to allow its instantiation and invocation by end users. The result of this phase is the executable composite service.

Step 10 : Service is executed and a composite service result is delivered to the requester.

4. PROBLEM FORMULATION

4.1 Problem Description

Many times the available services are not sufficient to satisfy the task requirement. So there is a need of combining various services into one composite service to satisfy complete task requirement. This is called as service composition problem. Different phases and its detailed flow of service composition process is described in section 3. The scope of this paper is limited to phases of service composition. Hence resource provisioning (step 6) is not considered for this paper rather it is assumed that with the help of resource provisioning algorithms appropriate resources are provisioned. The focus is on detailing of the main phases of service composition process including task dependency resolution, CPS model generation, service discovery, service ranking & optimal service selection, best plan generation and finally for service dependency resolution and scheduling. Various algorithms for these phases are designed.

4.2 Solution Approach

To solve this service composition problem middleware approach is adopted. Nikam, Ingle[18] have presented detailed study of existing middleware present in CPS and the middleware for service composition & resource provisioning along with its working is discussed. The following CPS model is assumed which considers that CPS systems of different domains like Transport CPS, Medical CPS, Environmental CPS, Water Distribution CPS are already in existence. Under each CPS system, multiple resource systems are present who will take care of various resource nodes and services which are present in resource system. The resource nodes can be sensors, actuators and processing nodes. Resource system is responsible for keeping the updated list of resource nodes and services. After reviewing the literature of service composition, the observation was that while solving service selection problem either one or two attributes of service were taken into consideration[15] and no solution approach focuses on all the attributes of service. One more challenge is that, the attributes are combination of positive and negative attributes which needs to be considered simultaneously. The literature on literature on Multi Attribute Decision Making methods(MADM) is reviewed as a potential solution approach including methods like SAW(Simple Additive Weighting Method), WPM (Weighted Product Method), TOPSIS (Technique for Order Preference by Similarity to Ideal

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Solution), PSI (Preference Selection Index), AHP (Analytic Hierarchy Process), PROMETHEE (Preference ranking organization method for Enrichment Evaluation Method, VIKOR, Entropy, Fuzzy Based methods, WEBDA(Weighted Euclidean distance based approach), Grey Relational Analysis, ANP(Analytical Network Process) etc. Few MADM methods are also applied in web service, cloud computing, wireless network domain for solving optimal selection problem. A study on comparative analysis on few significant methods of MADM is presented by Nikam, Ingle [19]. So the service selection sub problem is solved using MADM methods.

4.3 Definitions

Before formulating the problem of SC, it is important to understand the basic definitions.

Definition 1 : Task is defined as the functionality to be performed by service.

Task = {Functionality, Task UUID, Domain Id, Input Set, Output Set, Input Task List, Output Task List, Node-Type Id, Sub-node Type Id }

where functionality is the name of functionality to be performed by service.

UUID Stands for Universal Unique Identifier. It is the unique identity of each task.

Domain Id is a integer number represents the unique id of domain defined by middleware system such as Medical, Electricity, Water etc.

Input Set = Set of inputs required to task.

Output Set = Set of outputs generated by task.

Input Task List = List of tasks from where the inputs are available. If task is independent then this list is blank and if task is dependent, then task dependencies are resolved and this list is updated

Output Task List = List of task to which generated output of current task is given.

Node-Type Id = If task depends on sensor or actuator then node-Type-Id will contain the unique id of sensor or actuator.

Sub-Node-Type Id= Represents the unique id of sub types of sensor or actuator.

Definition 2 : Service is a logical implementation of functionality of task and they are used to encapsulate every functionality of resource node.

Service ={ Service UUID, Task Id, Node-Type Id, Sub-Node-Type Id, QoS attributes Q= { EC, R, RT, F , SER ,CC , A }}

i)Execution Cost (EC) : Is the fee that users must pay for invoking a service.

ii) Reputation (R) : It is the aggregate of ratings of that service by other principals.

iii) Response Time (RT) : It is the delay between service invocation and the result is obtained.

iv) Frequency (F) : Measures the number of times the service is requested for execution.

v) Successful Execution Rate (SER) : It measures the number of times the service is successfully executed. SER is the ratio of number of times the service is requested to number of times the service is executed.

vi) Communication Cost (CC) : It is the cost of communication between the service provider and service responder.

vii) Availability (A) : It is the probability of accessing the service where its domain is [0.1].

Now QoS attributes are classified as Positive and Negative attributes.

Positive attributes: They denote the higher value with higher user utility. Eg. Availability, Frequency, Reputation, Successful Execution Rate.

Negative attributes: They denote the lower the utility with higher values . Eg. Negative QoS Attributes like Response Time, Execution Cost and Communication Cost.

Definition 3 : Feature count is the number of best attributes in comparison of first service with another service .

Definition 4: Score is defined as the measure used to compare the goodness of service. It is calculated when the feature count of both the services is same.

For positive attribute, calculate score as follows

$$\text{Score}=\frac{FL1(i)-FL2(i)}{Max(FL1(i),FL2(i))} \quad (1)$$

Where FL1 and FL2 are feature lists.

For negative attributes calculate score as follows

$$\text{Score}=\frac{1/FL1(i)-1/FL2(i)}{Max(1/FL1(i),\ 1/FL2(i))} \quad (2)$$

Calculate final score as

$$\text{Calculate final Score} = \sum_{i=1}^{n} Score(i) \quad (3)$$

where n > 0

Definition 5 : Resource System is defined as a representative computing node which is coordinating the resource nodes connected to it. It has following attributes.

Resource system={UUID, Communication cost, Computation cost, Communication speed, Computation speed, No of cores, No of deployed service}

Definition 6: Resource Node can be sensor and actuator node along with their sub types known as sub node types

Journal of Theoretical and Applied Information Technology
15th June 2018. Vol.96. No 11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Resource Node = {UUID, Node-type Id , Node sub-type Id, Description, Geo-location{latitude, longitude eg, Node Attributes{ Cost, Availability, Status of Battery, Service allocation count} }

Definition 7 : Aggregated Attributes of composite service are defined as the aggregation of all the individual services. Aggregated attributes of the n elements (where n>0) of the Constructed Composite service (CCS) list are calculated as follows.

AEC = $\sum_{i=1}^{n} Execution\ Cost(CCS(i))$ (4)

ART = $\sum_{i=1}^{n} Response\ Time(CCS(i))$ (5)

ACC = $\sum_{i=1}^{n} Communication\ Cost$ (CCS(i)) (6)

AR = $\frac{1}{n} \sum_{i=1}^{n} Reputation\ (CCS(i))$ (7)

ASER = $\frac{1}{n} \sum_{i=1}^{n} Service\ Execution\ rate\ (CCS(i))$ (8)

4.4 Problem Formulation

Let us consider that user gives the input in the form of a task list to compose a service.

Let T = $\{t_1, t_2, t_3, \ldots\ldots\ldots\ldots t_n\}$ is a list of tasks to be performed and individual task is represented by t, n > 0. Each task is fulfilled by a its corresponding service.

Let CS= $\{S_1, S_2, S_3, \ldots\ldots S_i\}$ be an abstract composite service that consists of m abstract service, where S_i denotes i^{th} abstract service of S.

Let an abstract atomic Service Si ={ C_{i1} , C_{i2}, C_{i3}……..C_{ij}) } consists of j concrete services where C_{ij} denotes j^{th} concrete service of i^{th} abstract service. It denotes that they all are similar in functionality but differ in QoS.

Let Q = (q_1, q_2…… q_k) is set of QoS attributes of C_{ij}, where q_k is the value of k^{th} attribute of C_{ij}.

Let CO = { c_{o1}, c_{o2}, c_{o3}……..c_{om}} be a set of constraints given by user where each co_m is a constraint on q_k. If q_k is positive attribute then co_m imposes lower bound and if q_k is negative then co_m imposes upper bound.

Let W ={ w_1, w_2, w_3……., w_p} be a set of weights given by users. Each w_p (1<= p<= n) corresponds to each QoS property . For each q_k user assigns a weight w_p such that all weights satisfy

$\sum_{i=1}^{n} Wp = 1 \ and\ (0 <= wp <= 1)$ (9)

Equal weights to all the attributes are given so as to give equal importance to all the attributes which can be changed to as per equation (9) to give importance to the specific attribute.

Service Composition problem is represented as SC= {T, C, RS, S, OP {SD,SRSC}}

Where

T = Task list

C= CPS Systems

RS= Resource Systems

S= CPS service and OP is a set of operations defined as OP= { DR, SD,SRSC, PG} where

where

DR = Dependency Resolution,

SD = Service Discovery,

SRSC = Service Ranking and Selection and

PG = Composite Plan Generation.

OP is set of operations including service discovery, service ranking and selection.

To evaluate SC, the objective function is mapped with the QoS attributes mainly response time.

$T_{sc} = T_{dr} + T_{sd} + T_{srsc} + T_{pg}$ (10)

Where

T_{sc} = Service composition time, T_{dr} = Task dependency resolution time, T_{sd} = Service discovery time, T_{srsc} = Service ranking and selection time, T_{pg} = Best plan generation time

Table 1: Notations

NOTATIONS	MEANING
T	Input task list
CS	Abstract Composite Service
S	Abstract atomic service
CCS	Constructed composite service
ECS	Executable composite service
C_{ij}	Concrete Service
Q	Set of QoS attributes
CO	Set of constraints
W	Set of weights of QoS attribute
C	CPS Systems
C_n	Number of CPS
RS	Resource System
RS_n	Number of Resource Systems
S_n	Number of services deployed on each Resource Systems
R_n	Number of nodes registered on each Resource Systems
N	Number of Discovered Services
C_t	Number of Attributes
Sl	Selection Limit
P_n	Total number of generated plan
AEC	Aggregated Execution cost

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ART	Aggregated Response time
ACC	Aggregated Communication Cost
AR	Aggregated Reputation
ASER	Aggregated Service Exec. rate

Our objective is to select an optimal service C_{ij} from the set of similar functionality service list subject to constraints that T_{sc} is minimum.

4.5 Assumptions

Following are the assumptions made.

i) Task is defined by administrator in middleware system which is already decomposed into task list.
ii) Administrator can define N number of task for each domain.
iii) For one task there can be one or more than one services but the input and output of each service is same as defined in task but with different QoS and they may reside on different physical node.
iv) Each invocation to a service implementation will materialize into a unique task in a physical node.

4.6 Algorithms

Algorithms 1 to 13 are designed for different phases of service composition. Algorithm 1 (Service Composition) as an input takes task list and ranking limit and generates the optimal composition plan. It is the main algorithm of service composition. In step 2 it resolves task dependency and in step 3 it generates CPS model. In step 4 , it ranks the discovered services and thus compositeServiceList is formed by appending atomic services. Then calculates aggregated values of attributes in step 19 and generates optimal composition plan in step 22.

Algorithm 1 : Service Composition
Input : Task list, ranking limit
Output : Optimal composition plan
1. Tasklist = **Task_Dependency_Resolution (inputtasklist)** // Resolve task dependency using Task Dependency Resolution
2. map = **CPS Model Generation** // Generate a CPS model to access the corresponding information
3. discoveredServiceList=Service Ranking(tasklist, map) // To rank the Discovered Services.
4. bestPlan=NULL
5. newPlan=NULL
6. set serviceRankingLimit = n // where n > 0
7. while TRUE
8. compositeServiceList=0
9. for i=0 to taskListsize
10. task=taskList[i]
11. serviceList= map[task]
12. selectedService= Service Selection(task, serviceList,serviceRankingLimit)
13. if selectedService ≠ NULL then
14. compositeServiceList.add(selectedService) //Select best services and thus keep appending all the best atomic services to composite service list
15. end if
16. end for
17. aggregatedQoSAttribute=Aggregated_QoS_Attributes(compositeServiceList)
18. newPlan.service=compositeServiceList
19. newPlan.attributes= aggregatedQoSAttribute
20 bestPlan=**Optimal Composition Plan(plan, newPlan)** // And keep counter on number of plans generated
21. Repeat steps 19 to 22 if (counter < threshold) // generate z= threshold composition plans.
22. else stop generating composition plans
23. end if
24. end while

Algorithm 2 : Task Dependency (input_Task_List)
Input : Inputtasklist.
Output : Finaltasklist.
1. Create empty list called as finalTaskList

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2. For each task in InputTaskList do
3. CALL Execute(task, finalTaskList)
4. End for
5. Return finalTaskList

Procedure Execute (task list, final task list)
1. Begin
2. Status= CALL IsAvaliable(task)
3. If status= FALSE then
4. finalTaskList.add(task)
5. inputTaskList=task.inputTaskList
6. CALL Addtask(inputTaskList,finalTaskList)
7. Endif
8. End Procedure

Procedure isAvailable(task)
1.Begin
2. uuid=task1.uuid
3. for each task in finalTaskList
4. if task.uuid=uuid
5. then Return TRUE
6.End for
7.Return FALSE
8.End procedure

Procedure AddTask(taskList,finalTaskList)
1.Begin
2.For each task in tasklist do
3. CALL Execute(task,finalTaskList)
4. End for
5.End Procedure

Algorithm 2(Task Dependency Resolution) takes inputtasklist and finds the list of all dependent tasks from the given task list which is finaltasklist. It calls procedures Execute, isAvilable and AddTask to resolve the dependency. The best case time complexity is Ω (Tn-1) and worst case time complexity is O(Tn-1 * Tn) where Ti is input task and Tn is total number of tasks.

Algorithm 3(CPS Model Generation)
Output : map // Map = HashMap containing list of CPS System Model against the UUID of CPS System, UUID is key and CPS System Model is value.
1. CPSList = get List of all registered CPS from middleware.
2. For (i=0 to CPSlist.size-1)
3. url= CPSlist[i]. url // for each CPS System get URL
4. uuid= CPSlist[i].uuid // for each CPS System get UUID
5. CPSSystemModel = Connect(url) // connect with CPS system of given URL and get detailed information from Resource Systems in CPS System Model.
6. Map[uuid] = CPSSystemModel // Store it against UUID in map.

Algorithm 3(CPS Model Generation) collects list of services of all resource system connected with different CPS systems and generates a model to hold information of service along with CPS and resource system's id. This algorithm executes in middleware. The best case time complexity is Ω(Cn*Rsn* (Sn+Rn)) and worst case time complexity is O(Cn*Rsn* (Sn+Rn)) where Cn is number of CPS, Rsn is number of resource System, Sn is number of services deployed on each resource system and Rn is number of nodes registered on each resource system..

Algorithm 4 : Service Discovery (task, map)
Input : task, map
Output : discoveredSservicelist

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
1. domainid = get domain details from task uuid          // for a given task against which an atomic
                                                            service is to be found.
2. For (i=0 to map.size)          // Find matching CPS system by matching Domain details and for
                                     each CPS system which is found, search its Resource System List
3.      cpsSystemNode = map[i].cpsSystemNode
4.      If (domainid= cpsSystemNode.domainid) then
5.      resourceSystemList= cps.SystemNode. resourceSystemList
6.      For(j=0 to resourceSystemList.size)
7.              resourceSystemNode=resourceSystemList[i]
8.                      for (k=0 to serviceList.size)
9.                              if (serviceList[k]. taskUUID= taskUUID) then
10.                                     discoveredService=(cpsID, resourceSystemID, service)
11.                                     discoveredList.add(discoveredService)
12.                             end if
13.                     end for
14.     end for
15.     end if
16.  end for
17. return discoveredList
```

For a given task, algorithm 4(Service Discovery) is called by Service Ranking algorithm. It takes task and map as input and generates discoverdServiceList as an output. It first gets the domain details for the given task and then searches its resource system and service list. The best case time complexity is Ω((Tn *Cn) and worst case time complexity is O(Tn* Cn* Rsn* Sn) where Cn is number of CPS, Rsn is number of resource System, Sn is number of services deployed on each resource system and Rn is number of nodes registered on each resource system and Tn is total number of tasks.

Algorithm 5 : Service Ranking(task list, map)

```
Input : tasklist, map.
Output : ranked service list.
1.for ( i=0 to taskList.size)
2.      discoveredList= Service Discovery( task, map)
3.      Apply merge sort on discoveredList by injecting Service Comparator.
3.      Iterate all services of discoveredList
4.      Result = Service Comparator( DS1, DS2)  // where DS1 and DS2 are discovered services.
5. end iteration
6. end for
7. return discoveredList
```

Algorithm 5(Service Ranking(task list, map)) ranks all the discovered services using Service Comparator algorithm with merge sort. The best case time complexity is Ω (Tn * N* log {N}*Ct) and worst case time complexity is O(Tn* N*log {N} *(Ct+Ct/2)) where Tn is total number of tasks and N*log {N} is complexity of sorting algorithm where Ct is Criteria.

Algorithm 6 : Service Comparator(DS1, DS2)

```
Input : DS1, DS2          // DS1 is first discovered service and // DS2 is second discovered service.
Output : case 1: -1 when DS1 > DS2 ,
         case 2: 1 when DS1 > DS2 ,
         case 3: 0 when DS1 = DS2
1. get Attribute List of both discovered services.
2. Call Multi Attribute Comparator Algorithm for attribute wise comparison.
3. Return result..
```

Algorithm 6 (Service Comparator) takes as an input DS1 and DS2 and returns output as 1,-1 or 0. It is used to compare two discovered services on the basis of their respective QoS attributes. It calls Multi Attribute Comparator which is called with service attributes, so it compares two services.

Algorithm 7 : Multi Attribute Comparator(attribute List1, attribute List2)

```
Input :   attributeList1 = attributes List of first entity
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

attributeList2 = attributes List of second entity
Output : Total Score
1. For given two entities compare all positive and all negative attributes by calculating feature count and prepare feature list.
2. If Feature count (first entity) > Feature count(second entity)
3. then first entity is best
4. Else if Feature count(first entity) < Feature count(second entity)
5. then Second entity is best
6. else if Feature Count(first entity) = Feature Count(second entity) then
7. Calculate Score of positive and negative attributes of both the entities using equations (1) ,(2) and (3)
8. end if
9. end if
10. if score1 > score 2 then first entity is best
11. else second entity is best.
12. end if
Algorithm 7 (Multi Attribute Comparator) is a generic comparator who compares two entities considering all positive and negative attributes by considering feature count and score in case of positive and negative attributes as defined in definition 3 and 4.

Algorithm 8 : Service Selection (task, servicelist, serviceRankinglimit)
Input : task , servicelist, serviceRankinglimit
Output : rankedServicelist
1. if (discoveredServiceList < serviceRankingLimit)
2. then serviceRankingLimit= discoveredServicelist.size
3. Within serviceRankinglimit choose services randomly from rankedservicelist for each task and form plan1 and plan2.
4. With plan1 and plan2, call **Optimal Composition Plan**
5. Repeat step 5 and 6 for all the Services of rankedServiceList.
Algorithm 8 (Service Selection) selects services within service ranking limit from the ranked service list for a given task and calculates optimal composition plan by calling Optimal Composition Plan algorithm.

Algorithm 9 : Optimal Composition Plan(plan1, plan2)
Input: plan1, plan 2
Output : optimalPlan
1.QoSAttributes.Attributes1=plan1.Attributes //take Aggregated QoS attributes by Aggregated QoS Attributes algorithm for plan 1 and plan2
2. QoSAttributes1.Attributes = plan2.Attributes
3. code=**Multi Attribute Comparator(plan1, plan2)**
4. If code= -1 then
5. return plan1
6. else if code= 1 then
7. return plan2
8. else return plan1
9. end if
10. end if
Algorithm 9 (Optimal Composition Plan) generates optimal composition plan by comparing plan1 and plan2 using Multi Attribute Comparator which is now called with aggregated attributes of composition plan1 and plan2 and returns best plan. The best case time complexity is Ω (3*Tss) and worst case time complexity is O(Pn*Tss) where Pn is total number of generated plans.

Algorithm 10 : Aggregated QoS Attributes(compositeServiceList)
Input: compositeServiceList
Output: AggregatedQoSAttributes
1. For the given constructed composite service list calculate Aggregated QoS attributes using equation (3),(4),(5),(6),(7).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Algorithm 10 (Aggregated QoS Attributes) calculates aggregated attributes of all the elements of constructed composite service List.

Algorithm 11 : Service Deployment(task)
Input : task
Output : executable composite service
1. At resource system, filter resource System list on the basis of QoS parameters. i.e. the availability of the resource node.
2. Get all resource system list in current CPS system
3. For all members in the resource system list, connect with resource system.
4. Calculate resource system parameters using algorithm Resource Parameter Calculation(CPS Model)
5.Compare resource systems using algorithm Multiattribute Comparator and Score .
6.Return optimal resource system to deploy the service.
Algorithm 11(Service Deployment) deploy the constructed composite service to allow instantiation. While deploying the service against the task, it chooses optimal Resource System using algorithm Resource System Comparator and it also takes into consideration the count of already deployed services which is maintained by Resource System. The best case time complexity is Ω (Rsn * log {Rsn}*Ct) and worst case time complexity is O(Rsn* log {Rsn} *(Ct+Ct/2)) where Pn is total number of generated plans.

Algorithm 12 : Resource System Parameters(CPS Model)
Input : CPS Model
Output: Parameters P
1. cs = CALL CommunicationSpeed(cps)
2. cc = CALL CommunicationCost //get Communication Cost from stored information. Resource System‘s admin will fill it one time, per unit cost
3. ps = CALL ComputationSpeed // Computation Speed in MIPS (million instructions per second)
4. cost = CALL ComputationCost // get Computation Cost from stored information
5. nc = Find the no of cores of resource system
6. nds = get the count of deployed service
7. ns = get the count of registered sensors
8. na = get the count of registered actuators
9. return P = (cs , cc , ps , cost , nc , nds , ns , na)
Algorithm12 (Resource Systems Parameters) executes in resource system to find all the parameters of resource system.

Algorithm 13 : Service Scheduler(discoveredServiceList)
Input : discoveredServiceList
Output : TimeLayer[ ]
1. discoveredMAp= Map to store DiscoveredService object against taskID with the help of discoveredServiceList
2. Prepare weight matrix from given service list. // It finds dependency matrix (Initialize all diagonal elements as 0 means no dependency and if there is dependency then mark it as 1)
3. To prepare Time Layer array, check all predecessors. // Mark 1 for those who are having predecessors and default as 0)
4. Those who are not allocated Time Layer, take it in a separate queue and repeat for all unallocated services of queue
5. Repeat for all unallocated services of queue {
Find its predecessor
Check if predecessor is scheduled, then increment time layer by 1 }
If it has multiple predecessors then find Time Layer by same method and take maximum
6. return TimeLayer
Algorithm13 (Service Scheduler(discoveredServiceList) schedules discovered service list by resolving dependency by checking its predecessors.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5. EXPERIMENTAL SETUP DETAILS

The implementation is done in Java. For data model .XML is used. For experimental setup the hierarchical structure as described in CPS Model is followed. The hierarchy of the prototype consists of all computer with configuration as Intel i5, 2GHZ CPU, 8 GB RAM. On one computer middleware is running. There can be one or more computers representing CPS system and one or more computers representing Resource System. All the present CPS systems are registered in the middleware. Middleware holds a list of all CPS systems which are stored as per their domain information. All the Resource Systems are registered in the CPS system, so CPS systems maintain a list of all Resource Systems details. Finally all the information of Resources, may it be the sensors or actuators, they all are registered in Resource Systems.

Here the idea is to use sensors which are commonly found in any mobile handset. Every sensor is treated as an object, which makes it easy to deal with large number of sensors. To represent an actuator, a separate web applications depicting the function of actuator is developed. Eg. Light sensor is used from mobile handset as a sensor node which senses the input in lux then the actuator corresponding to that is a control which can depict ON and OFF operation through a web application. Each machine may it be a Middleware, CPS System or Resource System, they can communicate with each other by specifying URL of the computer to which you want to communicate. The data transfer will take place with the help of HTTP protocol. From this a runtime CPS Model is generated. The algorithms are validated in the above said setup. For results of services composition the algorithms are run in the above said setup which can be categorised as follows.

Category 1: CybReal PhyReal : Here Cyber system and physical system both are real.
Category 2: CybSim PhyReal : Here Cyber system is simulated and physical system is real. For working on a large number of services and resources, a simulator is written in java in which by giving a maximum and minimum value of all the positive and negative attribute, it will generate resource nodes and services.

6. RESULTS AND ANALYSIS

i) Category1 (CybReal PhyReal) : The results are achieved by varying number of CPS Systems, number of Resource Systems and number of Service instances. For various phases of Service Composition the total execution time is calculated as per equation (9). Table 2 shows the total service composition time with and without dependency consideration. Here number of CPS is 1, number of RN(Resource Node) is considered as 10 per resource system and number of RS(Resource System) is varied from 1,3,6,9 and 15. Service composition time is recorded for each case. Figure 4 shows results only for the first case when number of RS is 1, which indicates that Service Composition time with task dependency consideration is more than service composition time without consideration of task dependency and same is the observation when RS = 3,6,9,15. Table 3 shows the result when number of CPS is 1, number of RN considered per resource system is 50, and number of RS is varied as 1,3,6,9,15. Same observation is seen even when number of RN per RS is more. Table 4 and figure 4 shows the result of average service composition time without dependency consideration whereas table 5 and figure 5 shows the result of average service composition time with dependency consideration..

Table 2: SC time(ms) of Category 1(CyberReal-PhyReal) : Case 1

No. of Services	Service Composition time(ms) when Number of CPS =1, Number of RN =10 per RS									
	No. of RS=1		No. of RS=3		No. of RS=6		No.of RS=9		No.of RS=15	
	W/O DEP.	WITH DEP	W/O DEP.	WITH DEP	W/O DEP.	WITH DEP	W/O DEP.	WITH DEP	W/O DEP.	WITH DEP
5	2	16	11	12	3	61	1	9	13	52
10	2	22	4	34	2	304	2	12	100	190
20	5	15	3	69	2	102	3	350	24	108
30	3	6	2	81	8	280	4	234	58	479
40	3	8	4	100	2	302	4	193	325	452

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

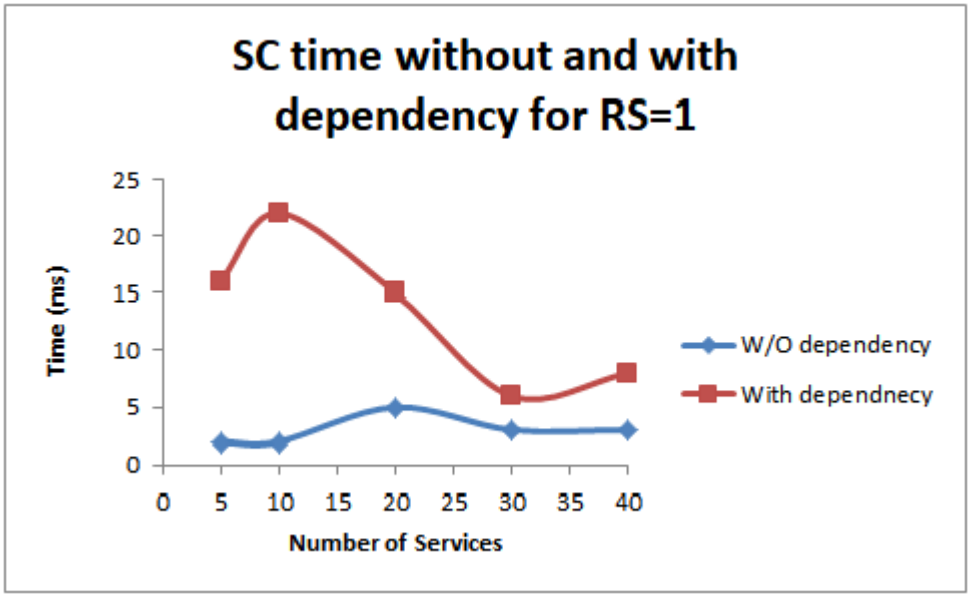

Figure 3: SC time of category1 of Category1 (CybReal PhyReal) : Case1

Table 3: Service Composition Time of Result of Category(CyberReal-PhyReal) : Case 2

No. of Services	Service Composition Time in ms when Number of CPS =1, Number of RN =50 per resource system									
	No. of RS=1		No. of RS=3		No. of RS=6		No.of RS=9		No.of RS=15	
	Without Dep.	With Dep	Without Dep.	With Dep	Without Dep.	With Dep	Without Dep.	With Dep	Without Dep.	With Dep
5	37	77	47	60	47	60	5	45	20	61
10	7	82	16	170	16	170	10	60	55	2105
20	19	71	15	270	15	270	13	1202	54	581
30	11	26	10	303	10	303	16	980	26	325
40	11	31	16	394	16	394	16	1103	182	2121

Table 4: Average SC time Without dependency of Category(CyberReal-PhyReal)

No of Resources Systems	No. of CPS=1, No of Services =5, 10, 20, 30, 40 No. of RN=50 per resource system	
	Average Service Composition time(ms) when RN=10	Average Service Composition time(ms) when RS=50
1	4.4	17
3	4.6	20.8
6	3.4	16.4
9	2.8	12
15	3.4	67.4

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

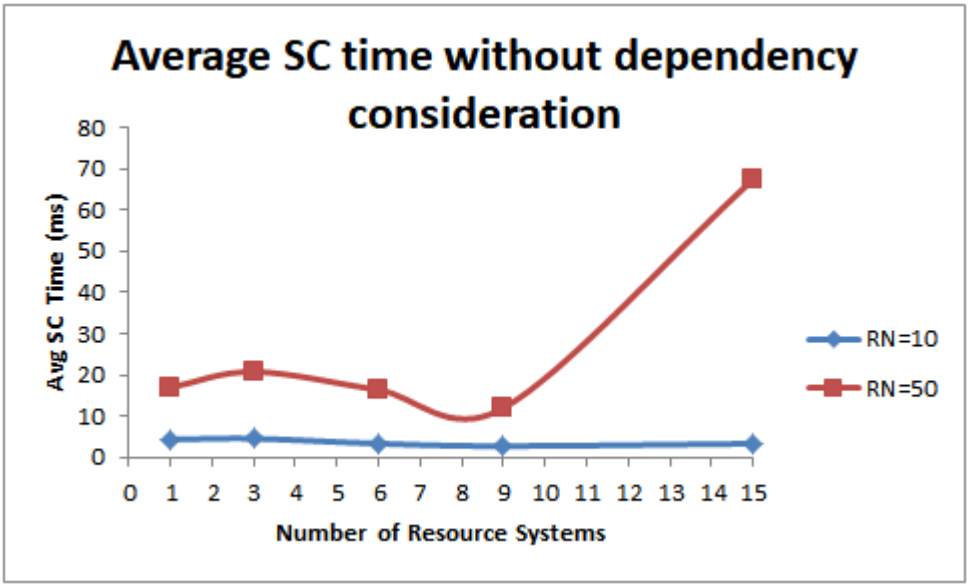

Figure 4: Average Service Composition Time Without dependency consideration

Table 5: Average Service Composition Time with dependency consideration of Category(CybReal PhyReal)

No of Resources Systems	No. of CPS=1, No of Services =5, 10, 20, 30, 40	
	Average Service Composition time(ms) when RN=10	Average Service Composition time(ms) when RN=50
1	13.4	57.4
3	59.2	293.4
6	209.8	876.6
9	159.6	678.0
15	504.8	1038.6

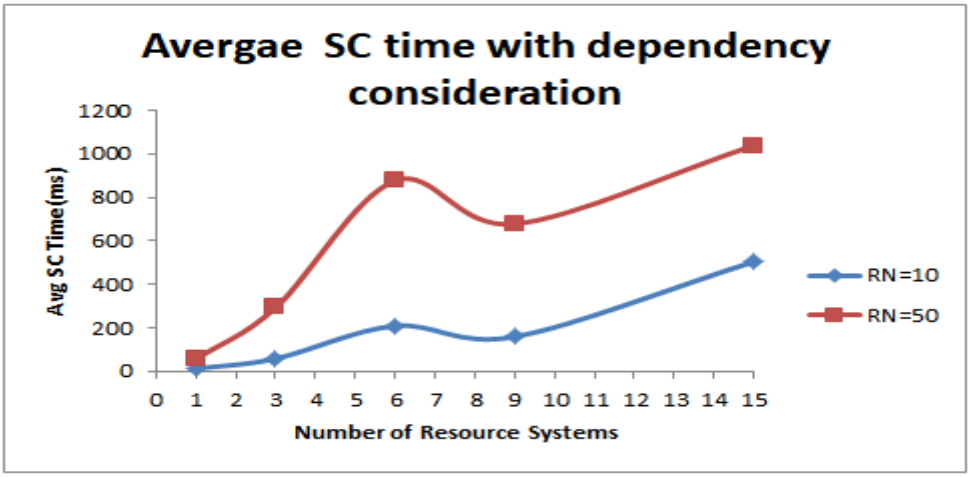

Figure 5: Average Service Composition Time with dependency consideration of Category(CybReal PhyReal)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ii) Category 2 (CybeSim PhyReal) : The results of service ranking is calculated using three algorithms a) Multi Attribute Comparator(MAC) b) Preference Selection Index(PSI) and Technique for Order Preference by Similarity to Deal Solution(TOPSIS) [17]. There are many MADM methods present in the literature but PSI and TOPSIS are chosen because they do not have any requirement of weight assignment to the attributes. Table 6 shows the result of Category 2 and figure 6 shows graph of comparative results of MAC with PSI and TOPSIS.

Table 6: Comparative Result of MAC, PSI and TOPSIS of Category 2: CyberSim-PhyReal

Number of Services	Multi Attribute Selector(MAS) Execution Time(ms)	PSI Execution Time(ms)	TOPSIS Execution Time(ms)
100	0.334	1.534	1.515
200	0.788	2.194	3.29
300	0.974	4.52	5.187
400	1.553	5.107	5.074
500	2.112	6.682	10.626
600	2.021	7.275	10.592
700	2.839	1.312	1.989
800	2.776	1.239	2.335
900	4.03	1.726	2.779
1000	4.682	2.038	2.36

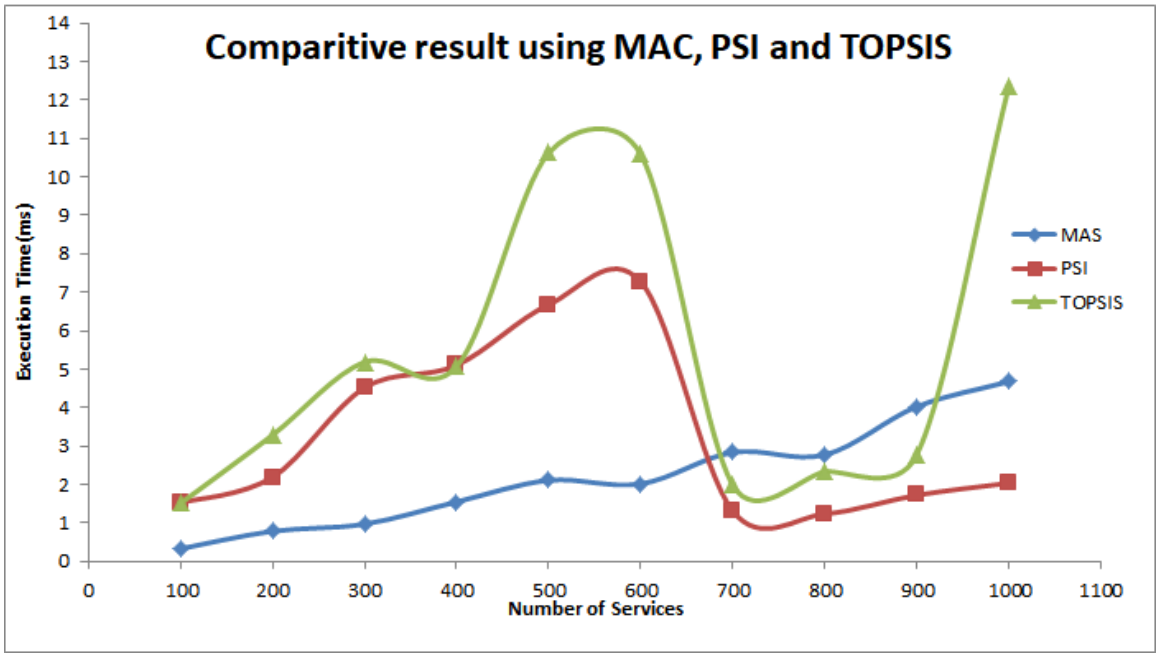

Figure 6: Performance of MAC, PSI, TOPSIS .

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

7. COMPARISION WITH RELATED WORK

As mentioned in section 2 many of the previous researchers have not defined CPS service attributes whereas they have considered the attributes of web service only. But considering attributes of web service may not be sufficient as there is a difference between Web service and a CPS service. Our work clearly defines CPS service along with its attributes. Secondly the service composition algorithms were not defined phase wise and also all the phases were not elaborated, whereas our work consists of phase wise definition and description of service composition algorithms along with results. Thirdly previous researchers have worked either on framework or given theoretical discussion on ontology but very few have discussed the implementation details of service composition method whereas in our work the complete description of middleware and implementation details are given. And lastly the methods of MADM were not yet explored for optimal CPS service election problem, but our work includes comparison of all the potential methods of MADM[19] and then the best methods are applied for final service selection phase. The results are achieved in simulated environment where number of CPS systems, Number of resource systems and number of sensors and actuators are limited so remaining combinations of CybReal, CybSim, PhyReal and PhySim are not considered in this paper for discussion.

8. CONCLUSION

The service composition problem is divided into different phases and algorithms are designed for phases. Optimal service selection is seen as a sub problem of service composition. Multi attribute decision making methods are used in CPS service ranking and selection for service composition. For solving optimal service selection problem Multi Attribute Comparator algorithm is written. Existing methods of Multi attribute decision making methods mainly PSI algorithm and TOPSIS algorithm are used for comparison of service selection phase. The impact of task dependency on service composition is shown with various deployment scenarios which indicates that task dependency resolution time impacts overall service composition time.

REFERENCES

[1]. Cyber Physical Systems Summit Report, NSF, iccps2012.cse.wustl.edu.

[2]. Edward Lee, "Cyber Physical Systems : Design, Challenges", *Object IEEE International Symposium on oriented real time distributed computing (isorc),* pp. 363-369, 2008.

[3]. Jiafu Wan, Hehua Yan, Hui Suo and Fang Li, "Advances in Cyber-Physical Systems Research", *KSII Transactions on Internet and Information Systems*, Vol. 5, Issue 11, pp. 1891-1908, 2011.

[4]. Swati Nikam, Rajesh Ingle, "Survey of Research Challenges in Cyber Physical Systems", *International Journal of Computer Science and Information Security,* Vol. 15, Issue 11, pp. 192-199, 2017.

[5]. Jototh Chandrashekhar, G.R.Gangadharan, Ugo Fiore, Rajkumar Buyya, "QoS Aware Big Service Composition using MapReduce based Evolutionary Algorithm with guided mutation", *Future Generation Computer Systems,* 2017.

[6]. Kalasapur, Swaroop, Mohan Kumar, and Behrooz A. Shirazi, "Dynamic service composition in pervasive computing", *IEEE Transactions on Parallel and Distributed Systems,* Vol. 18, Issue 7, pp. 907-918, 2007.

[7]. Tamhane, Sagar A, Mohan Kumar, Andrea Passarella, and Marco Conti, "Service composition in opportunistic networks", *IEEE International Conference on Computing and Communications (GreenCom),* pp. 285-292, 2012.

[8]. Wu, Taotao, Wanchun Dou, Chunhua Hu, and Jinjun Chen, "Service Mining for Trusted Service Composition in Cross-Cloud Environment", *IEEE Systems Journal,* Vol. 11, Issue 1, pp. 283-294, 2017.

[9]. Swati Nikam, Rajesh Ingle, "Comparative Study of Service Composition in CPS and IoT", *International Conference on Advances in Cloud Computing,* pp 1-7, 2014.

[10]. Zhao Yajing, "Abstract Cyber Physical Systems Service Composition", *Chapter 14, Service Life Cycle Tools and Technologies, Methods, Trends and Advances , IGI Global,* pp. 2012-2014, 2012.

[11]. Hell Bruck Horst et.al, "Name Centric Service Architecture for Cyber Physical Systems", *Proceedings of 6th IEEE International Conference on Service Oriented Computing and Applications,* pp. 77-82, 2013.

[12]. Tao Wang et. al., "Automatic and Effective Service Provision with Context-aware Service Composition Mechanism in Cyber-

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Physical Systems", *International Journal on Advances in Information Sciences and Service Sciences,* Vol. 4, Issue 11, pp. 151-160, 2012.

[13]. Wang Tao, "A Two-phase Context Sensitive Service Composition Method With a Workflow Model in Cyber Physical Systems, *Proceedings 17th IEEE International Conference on Computational Science and Engineering,* pp. 1475-1482, 2014.

[14]. J. Huang et. al., "Extending Service Model to Build an Effective Service Composition Framework for Cyber-Physical Systems", *IEEE International Conference on Service-Oriented Computing and Applications,* pp. 130-137, 2009.

[15]. Jian Huan et.al., "Towards a Smart Cyber Physical Space- Context Sensitive Resource Explicit Service Model", *33rd International IEEE Conference on Computer Software and Application,* pp. 122-127, 2009.

[16]. Wang, Shangguang, Ao Zhou, Mingzhe Yang, Lei Sun, and Ching-Hsien Hsu, "Service Composition in Cyber-Physical-Social Systems, *IEEE Transactions on Emerging Topics in Computing , 2017.*

[17]. Quan Shen et.al., "Web Service Composition- A Decades Overview", *Informatics Sciences,* Vol 250, pp 218-238.

[18]. Swati Nikam, Rajesh Ingle, "Middleware for Service Composition in Cyber Physical systems", *Journal of Computational and Theoretical Nano science, 2018, (Accepted and Under Publication)(Link- rbingle.in)*

[19]. Swati Nikam, Rajesh Ingle, "Autonomics of Self Management for Service Composition in Cyber Physical Systems", *Second Springer International Conference on Smart Innovations in Communications and Computational Sciences (ICSICCS), 2018, (Accepted and Under Publication)(Link-rbingle.in).*