Research Journal of Applied Sciences, Engineering and Technology 5(22): 5260-5266, 2013
DOI:10.19026/rjaset.5.4274
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 22, 2012 Accepted: December 28, 2012 Published: May 25, 2013

Research Article

Optimized Architecture-centric Function Points' Clustering and Aggregating About Service Flow

Xiaona Xia, Baoxiang Cao and Jiguo Yu
School of Computer Science, Qufu Normal University, Rizhao, Shandong, 276826, China

Abstract: This study puts forward the "Cluster" concept about service function points. For the uniform of timely information's capturing, logical optimization and requirement implementation, service topology's relationship is adaptively formed. Tracking architecture goal and building distributed clustering logic timely, "Cluster head" and "Cluster node" are serialized and optimal architecture-centric "Cluster group" is formed, it is to achieve the balance service adjustment of service topology network driven by goal.

Keywords: Architecture-centric, optimized aggregating, service cluster, service clustering algorithm, service flow

INTRODUCTION

Control the complexity of software business and improves software services' quality, in order to raise the layer of software reuse (Mei and Shen, 2006). The specificity of software architecture in software life cycle and key role depend on the medium between users' requirement and goal's implementation (Garlan, 2000; Xiao-Feng *et al*., 2010), on architecture design, it is to achieve a flexible platform from a group of requirements to satisfy requirement or helping to meet for requirement by design (Bosch, 2000).

Architecture-structured and object-oriented technology provides a strong support for design, but it is difficult to adapt to the self-learning requirement of platform business logic and the high-layer special question of architecture. Architecture-structured stresses the core effect of function, while it does not achieve the uniform of question space and answer space and it is difficult to adapt to requirement changes, it expresses the "bloated" and "detailed" top-down requirement, but does not apply to high-layer design of architecture (Shao, 1998; Parnas and Clements, 1986). Requirement and business logic is planned by object, which unifies the modeling method between question and answer, but cannot be suitable for expressing the uncertain relationship among objects and granularities' self-organization, it exists serious "shortcomings ripple effect" for the more adaptive requirement, especially the non-functional features and integrated system properties are not be reflected in the architecture (Sangwan *et al*., 2008).

Architecture-centric service-oriented mechanism: In the architecture-centric platform, services is no longer the calculated granularities that is passively waiting goal requester's searching and adjustment, but can be real-time adjusted and take the initiative to search for architecture goal's implementation. In this process, it builds the corresponding relationship between goal requirement and service entities and then provides service function. If the goal is satisfied for service entity, they can collaborate on requirement. the goal's providing and solving is the topology of services' capturing and implementing sequence of architecture, then accepts the topology rules' logic and achieves the process and service mechanism about architecture goal. From the architecture perspective, it is the relationship of goal and services, from platform perspective; it is the aggregation of service topology and optimal building about goal, as shown in Fig. 1.

Figure 1 is divided into three parts, the middle is the intermediary platform (ACP: Architecture-centric Platform), it is to achieve the service process of Platform Requirement (PR), in the whole process, whole implementation logic is that architecture goal (AG) submits and feedbacks results, which exists parallel relationship between PR and AG. All business logic is the integration of service granularities (AG), that needs the aspects' supporting: first, AG of ACP achieves requirement logic's transferring of PR sets and completes logic's separation and aggregation, then forms the basic underlying topology relationship factors. Meanwhile, about service Topology Relationship sets (TR), Topology Relationship Analysis mechanism (TRA) analyzes these factors and selects the least and most simple relationships, cross reference, to complete the basic relationship factors' selection about PR implementation logic; Second, driven by basic relationship factors, helped by AG, these factors enter into Service Container (SC), Active Service (AS) captures the double deployment of goal's intention and

Corresponding Author: Xiaona Xia, School of Computer Science, Qufu Normal University, Rizhao, Shandong, 276826, China

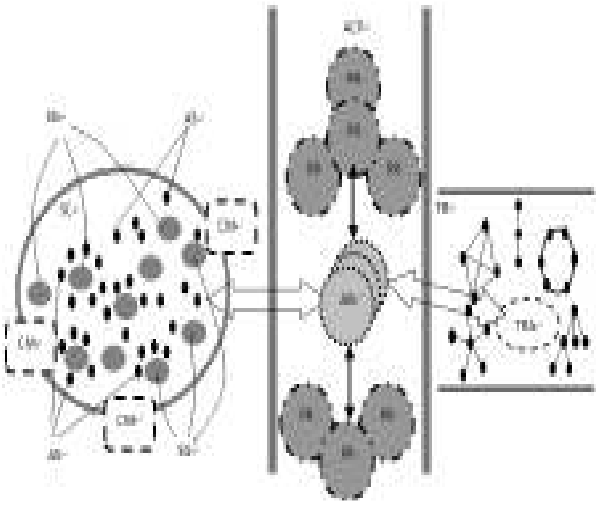

Fig. 1: Architecture-centric service serialization

accepts services' Classification Mechanism (CM) and AG, finishes the requirement's integration among services. The whole relationship is the transferring and analysis of AG, ultimately, it is the adaptive relationship's matching and implementing of services.

Service flow's definition: Architecture achieves the transferring process from tasks' requirement to implementation sequences that is the process mode about requirement of platform, here gives the definition of service flow:

Service is the basic granularity about implementation sequence and results' responses, controlled by architecture-centric main goal; this process is defined as service flow. We can see from Fig. 1, started from platform requirement, service flow accepts adjustment of architecture goal and finishes the corresponding and composition relationship about services and flow topology. In entire process, architecture goal plays an important role, which the topology corresponding and granularity's composition is an adaptive learning and completing transparency sequence. Service sets and architecture goal's achievement needs SC's integration and perception. Meanwhile, it is the basic supporting of service flow and the optimal implementation result. Every service flow's implementation is about goal's completing, this process is also responsible for adaptive services' selection and composition, the improvement and expansion of basic flow's topology.

This study puts forward the "Cluster" concept about service function points. For the uniform of timely information's capturing, logical optimization and requirement implementation, service topology's relationship is adaptively formed. Tracking architecture goal and building distributed clustering logic timely, "Cluster head" and "Cluster node" are serialized and optimal architecture-centric "Cluster group" is formed, it is to achieve the balance service adjustment of service topology network driven by goal.

SERVICE'S CLUSTERING LOGIC

The architecture-centric software platform based on services need to solve two question: function points' aggregation and service topology routing's building, its goal is to enhance the accuracy of service selection and the optimal degree of routing, at the same time, improve the ability that services participate in platform requirement's goal and achieve the entire adjustment of *AG* about whole and local requirement, which has the same similarities with WSNs about platform's design and implementation. Inspired by the clustering of WSNs, this study proposes the service cluster concept.

Cluster: Cluster structure is divided into two kinds of basic units: Cluster Head and Cluster Member. Cluster head is the adjustment center of cluster structure and manages cluster's forming; data's collection of cluster member and collects the gotten data and transfers to other cluster heads or directly sends to dada aggregation point.

During the process of forming clusters, through corresponding algorithm, the neighboring nodes' information of cluster head is exchanged and judges the intensity of nodes' distribution, based on this values, the covering area of nodes in cluster is calculated, by adjusting all cost by broadcasting information, it makes the dense area form clusters with small area, oppositely, form cluster with large area. Meanwhile, the selection of cluster head is based on the residual energy of nodes, when selecting cluster head, the direction of cluster head and the residual energy of cluster head is as the precondition and chooses the best cluster nodes to join.

This process to achieve WSNs topology based on clustering is beneficial to the optimization and nodes' selection for whole architecture and routing. In architecture-centric service platform, service is as the basic participation granularity, adjusted by *AG*, the adaptive transferring process is achieved that expresses "business goal-optimal topology-service node". Here presents the service cluster concept that *AG* is statute and optimal service topology is the clue. Based on *AG*, architecture goal is divided into local ones, then service cluster head, cluster node and cluster group is defined formally.

Service Cluster Head (SCH): Service cluster head is the service individual about the Sub Goal (*SG*) for *AG* by cluster head's selection is to participate in the cluster nodes of *SG*, then *SG* is completed by service clusters. Service cluster head transfers the implementation results, or communicates requirement and results about *SG* with other service clusters. The service cluster head of architecture is formally defined as:

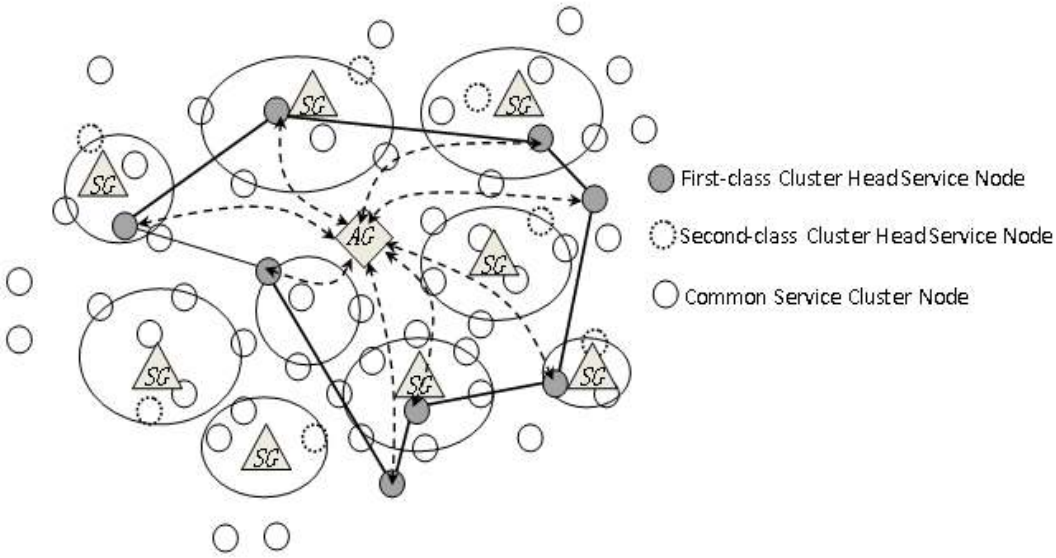

Fig. 2: AG-driven architecture service cluster's implementation topology

$SCH < i, j, k, ..., n >$, of which, i stands for the node number of first-class service cluster head, j stands for node number of second-class cluster head and so on, the final class n stands for the service cluster head number of *AG*. The achievement of *AG* is the topology's optimization and implementation of service cluster head set is defined as:

$$AG < SCH_i, SCH_j, ..., SCH_n >$$

Service Cluster Node (SCN): Service cluster node is the participation node of cluster based on *SG*, that is the individuals and selected and adjusted by *SCH*, then takes part in implementation and achievement of *SG*. It is formally defined as *SCH*< *i*, *j*>, of which, i stands for cluster head node's number corresponding to common service cluster nodes, j stands for number of common nodes. For entire *AG*, service cluster head and cluster node exist such a relationship: $AG \to set(SCH_x \times SCN_x)$.

Service Cluster Group (SCG): $FS = \{s_1, s_2, ..., s_i ..., s_n\} (1 \le i \le n)$ is the cluster concept about the function point set, cluster is the structure unit about service cluster head and node, its physical structure is the individual service, cluster is the aggregation of *SG*, *SCH* and *SCN*, it is formally defined as : *SCG*< *SG*, *SCH*, *SCH*>.

GOAL-DRIVEN CLUSTER PARTITIONING STRATEGY

The forming process driven by achieves hierarchical cluster in a recursive way, using QoS value of every service cluster node, is divided into set of, ensure the cluster heads of different clusters, according to the association coupling of cluster heads, the participation nodes and boundaries are formed.

- The service granularity uses nodes to spread around their QoS characterization and service features regularly, builds corresponding service function list. Service is adjusted by and provides its function and the collaboration information, so that, every service granularity will get candidate granularity group, through QoS and collaboration requirement information, the local has been calculated.
- About cluster structure with aggregation relationship logic, analyzes QoS characterization and service features of every node, ensures cluster head nodes and gives corresponding to cluster head.
- Based on the logical function of goal, analyzes every cluster head node, according to value, the association of local clusters and interconnection relationship among cluster heads are built.
- The first-class cluster about is built: the cluster head nodes achieved by the above three steps is broadcast clustering information in architecture and form their own clusters. After service nodes' receiving broadcasting cluster information, the matching about local cluster and is verified. If it has joined some cluster, the virtual interconnection about functions are built and the logical association is built among clusters. If it has not joined, this node directly join cluster and gets the participation identifier. Of which, expresses cluster head node's number, expresses the number in the platform.
- And so on, achieves the cluster and the association topology rules in or among clusters, when new requirement goals are provided, firstly search whether there're precedent instances in historical topology rules and improves the implementation

efficiency of platform. If existing parallel service requirement goals, executes the first four steps. The final implementation topology structure is shown in Fig. 2.

OPTIMAL AGGREGATION PROCESS OF SERVICE CLUSTER

Starting from service quality of user requirement, with meeting for *AG*, service granularities with less response time and cost and larger success rate, reliability and composite rate will be selected for clustering. Therefore, when processing new user requirement, the optimal aggregation and the first-class cluster's forming, that needs the five QoS performance indicators to be ordered and selected by right, their sorting and selection is adjusted by *AG*.

Service cluster is the set of basic granularities, that meets for production rules and interface semantic rules of service composition model, according to these rules, the optimal aggregation of cluster processes corresponding the next step action *a*, inputting *I* and outputting *O* to form definition C_a^1, furthermore, based on the interface semantic rules of service composition, the participation interface semantic rules of cluster is aggregated to C_a^2.

The optimal aggregation of service cluster is the balance and statute for second part of this study, here, suppose the QoS performance of service clusters is the definition above, respectively:

$Time\ (s_{sg})$, $Cost\ (s_{sg})$, $SuccessRat\ e(s_{sg})$, $\mathrm{Re}liability(s_{sg})$ and $CompositeR\ ate(s_{sg})$ and there are still the following relationships:

$Time(sg) \le Time(s_{sg})$, $\mathrm{Re}liability(sg) \ge Time(s_{sg})$ and $CompositeR\ ate(sg) \ge Time(s_{sg})$. The whole optimal aggregation process is expressed as Algorithm 1.

Algorithm1: Optimal Aggregation Algorithm of Service Cluster

PDL: Optimal Service Aggregation Process (S)

Input the composite function structure with *n* service granularities $C_s = \{s_1, s_2, s_3, ..., s_i, ..., s_n\}(1 \le i \le n)$ every service is respectively corresponding to five QoS performance values, that is:

$$Set_{s-QoS} = \{Time(s_i), Cost(s_i), SuccessRate(s_i), \mathrm{Re}liability(s_i), CompositeRate(s_i)\}$$

At the same time, every service cluster is as the service aggregation structure, it has this five QoS performance values, that is defined as:

$$Set_{sg-QoS} = \{Time(sg_i), Cost(sg_i), SuccessRat\,e(sg_i), \mathrm{Re}liability(sg_i), CompositeR\,ate(sg_i)\}$$

supposing the service cluster of user requirement is C_r.

Output the optimal aggregation of service clusters and the optimal association among service clusters, supposing the goal service composition structure of user requirement is C_g.

$user - \mathrm{Re}quest \xrightarrow{AG} C_s$; //*AG* controls user requirement and sends QoS performance values to C_s.

$AG \xrightarrow{AG} (SG_1, SG_2, ..., SG_i, ..., SG_n)$ //According to user requirement *AG* completes division of requirement goal and forms *SG* set.

Empty (C_g); //Empty () achieves goal service composition to be null.

$x = 1$; // *SG*'s Loop count variable。

While (all $(SG_1, SG_2, ..., SG_i, ..., SG_n)$)

Start

While (all s_i)//All service granularities is grouped by service size based on *SG*.

$Cluster(s_i) \xrightarrow{SG_x} C_r^1$// controlled by *SG*, achieve the service composition about *SG* and achieve that different *SG* is corresponding to service function composition.

While, ($all\ s_i \in C_r^1$)

$ClusterSem\,entics(C_r^1) \xrightarrow{AG} C_r^2$ //*AG* and *Cluster Sementics* () records topology relationship of service function composition for *SG*.

End

$QoSsequenceR1 \xrightarrow{SG} sort\{Set_{s-QoS}(Time(s_i), Cost(s_i))\}$ //Service granularities corresponding to *SG* are sorted based on response time and cost asc.

$QoSsequenceR2 \xrightarrow{SG} sort\{Set_{s-QoS}(SuccessRate(s_i), \mathrm{Re}liabilty(s_i), CompositeRate(s_i))\}$ /Service granularities corresponding to *SG* be sorted based on success rate, reliability and composite rate desc.

$QoSsequenceRC1 \xrightarrow{AG} QoSsequenceR1 \Rightarrow sort\{Set_{sg-QoS}(Time(sg_i), Cost(sg_i))\}$// *AG* sorts service clusters corresponding to *SG* according to response time and cost asc.

$QoSsequenceRC2 \xrightarrow{SG} QoSsequenceR2 \Rightarrow sort\{Set_{sg-QoS}(SuccessRate(sg_i), \mathrm{Re}liabilty(sg_i), CompositeRate(sg_i))\}$ // *AG* sorts service clusters corresponding to *SG* according to success rate, reliability, composite rate desc.

$C_{sg-r}^2 = Cluster\ (C_r^1, C_r^2)$

$\times QoSsequenceRC1 \times QoSsequenceRC2$ //Achieve the topology routing relationship of service clusters.

While ($C_r^2 \neq \phi$ and C_g)//achieve the topology of service clusters about user requirement, its precondition is the topology relationship is not null about different service clusters.

Start

If ($Time(sg) \leq Time(s_{sg})$ and $Time(sg) \leq Time(s_{sg})$ and $SuccessRate(sg) \geq SuccessRate(s_{sg})$ and $\text{Re}\,liability\,(sg) \geq \text{Re}\,liability\,(s_{sg})$ $CompositeRate(sg) \geq CompositeRate(s_{sg})$

While ($all\ sg_i \in C_{sg-r}^2$)

Start

$C_g \xleftarrow{AG} \{\min(Time(sg_i) \&\& Cost(sg_i))\}$ and $max(\ SuccessRate(sg_i) \&\& Reliability(sg_i) / \&\& CompositeRate(sg_i))$

Select the service clusters topology with maximum response time, cost and maximum success rate, reliability and composite rate.

End

$Evolution\ (QoSsequenceRC1, QoSsequenceRC2, C_g)$; // Optimal process among service clusters.

Return (C_g)//C_g is the optimal aggregation result about service clusters.

End

TOPOLOGY'S BUILDING OF SERVICE NETWORK

The optimal aggregation result of service clusters is to meet for the user requirement's implementation topology of *AG*, the cluster head nodes need some transmission power to transfer goal to all service clusters and nodes. After service nodes receive corresponding goal and requirement, *AG* calculates implementation routing distance based on QoS of selected cluster and nodes and forms service topology network of *AG*.

The forming of service implementation is the multi-layer distributed cluster routing with balancing QoS value, in order not to increase the difficulty about topology's forming and goal's implementing and improve the efficiency of routing's building, service cluster's forming is based on the optimal aggregation strategy of the fifth part of this study . Because the cluster head nodes directly receive the adjustment of *AG* or *SG*, it plays a key role on cluster's selection and routing's matching, during keeping to optimal aggregation strategy, this study will think about QoS performance of cluster head nodes. Therefore, routing topology needs the clustering of WSNs and routing topology algorithm, based on clustering and clustering's selection thought, the first-class service cluster head nodes are as the carrier about main goal and topology and build requirement-driven platform's implementation sequence. To reflect the adaptive capturing and timely feedback, multi-layer second-class service cluster head nodes are built to implement local goal *SG*.

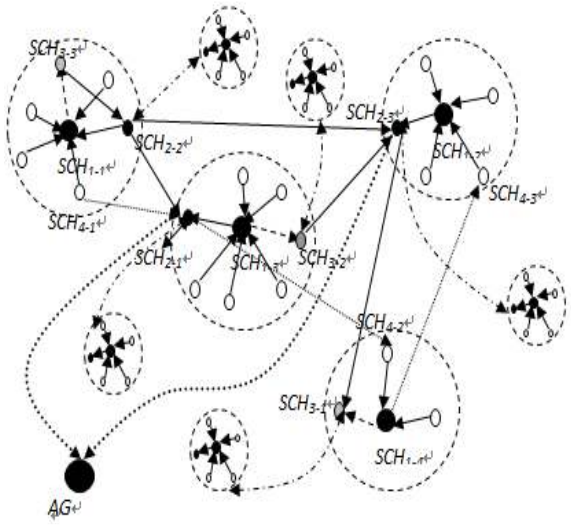

Fig. 3: Multi-layer tolology of service routing

In multi-layer topology of Fig. 3, service nodes in the same ring adjacent to each other, every ring is first-class service cluster structure, the black nodes in middle part $SCH_{1-1}, SCH_{1-2}, SCH_{1-3}, SCH_{1-4}$ is the first-layer service cluster nodes, that are the first-class service cluster heads of AG; $SCH_{2-1}, SCH_{2-2}, SCH_{2-3}$ are the second-layer service cluster node set, $SCH_{3-1}, SCH_{3-2}, SCH_{3-3}$ are the third-layer service cluster node set, $SCH_{4-1}, SCH_{4-2}, SCH_{4-3}$ is the forth-layer service cluster node set, these cluster heads is the sub-class cluster head services about different *SG*, sub-class cluster heads can be divided into one sub service cluster structure by goal, the small clusters in Fig. 3 is division of sub-class cluster head. During the forming of routing, new cluster heads will be produced from current non-cluster head nodes of implementation topology. That is, the next layer's cluster heads are not in the first-class service cluster head set, the service cluster head in this current layer only processes the logical aggregation and integration of this layer's *SG*.

Through different levels' service cluster heads, participation service nodes assign and select its own neighbor service node list. During the forming of the hierarchical topology, with the help of *AG*, from the whole to the local or from top to bottom, the cluster head nodes are selected. That selects first-level cluster heads and forms the first-level cluster, through traction, until forms k-level and this level is as the basic single service nodes, this completes multi-layer routing topology. By Algorithm 2 that can achieves the multi-layer topology, in the selected and aggregating process

about different service node and different clusters, implementation topology completes service routing logic by broadcasting message rules.

Algorithm 2: Multi-layer Topology's Forming Algorithm

Service Topology Form

Begin

Service Topology = Null;//Service routing topology is empty, it's a two-dimensional structure, one-dimension stands for layer, two-dimension stands for the corresponding cluster head set.

Computing and Broadcasting Request Goal (*AG*, *SCH*)

Set (*SG*) = Split (*AG*)//*AG* is divided into sub-goal set *SG*.

ServiceClusterHead⊂Set (*SCN*)//Cluster head node of service cluster is produced by service nodes.

ServiceClusterHeadSet = null//

ServiceClusterHeadSet is two-dimensional data, one dimension stands for topology routing layer, two-dimension stands for cluster head node set of some layer.

ServiceClusterLevel = 0// Initialize the service level for the initial zero-layer clusters.

ServiceClusterRule = Null; //Service topology routing rules are set empty. //beClusterHead = False//Judge Service nodes whether or not is the cluster head.

TopologyDegree = OptimalDegreeClassified(*AG*, Set (*SG*))//*AG* adjusts the optimal implementation levels of *SG*, Topology Degree is a two-dimensional data, one dimension stands for topology layers, two dimension stands for the *SG* set about different topology layers.

//the above pseudo-code achieves the generated cluster's initialization.

While (not empty (Set(*SG*)) and not empty(Topology Degree))

begin

For any SCN ∈ Topology Degree

If Satisfy (SCN, C_g) and be Cluster Head (*SCN*) = False //Judge SCN whether or not meets for the optimal aggregation process of C_g and it is not the cluster head node.

Then

begin

beClusterHead = true

SCN→SCH

(SCH, OptimalServiceAggregationProcess(SCH)) → ServiceTopology;

SCH→ServiceClusterRule

(SCH, ServiceClusterLeve) → ServiceClusterHeadSet

End If

End For

//When SCN meets for cluster head nodes ,then is defined as cluster head and completes the optimal aggregation process of this service node and achieves the corresponding C_g, then it enters into ServiceTopology and this service node optimizes routing.

ServiceClusterLevel+=1;//Topology layer adds based on aggregation and implementation's PRI.

Output (Service Topology, ServiceClusterHeadSet, ServiceClusterRule)

End While

end

Based on algorithm, each service node accepts direct reduction of goal and cluster heads and nodes can be selected and optimized among layers, by function "*ServiceTopology, ServiceClusterHeadSet, Service ClusterRule*" completes implementation topology of requirement goal, cluster heads' selection and routing rules' planning adjusted by *AG*, finally, cluster head nodes complete the participation about *AG* by message, form whole topology and this topology is the smaller and more optimal participation set about goal's implementation. During the sequence process, the feedback about clusters' goal in different layers use directly transferring method, cluster heads are the direct local adjustment mechanism, due to the cluster head nodes in different layers may be in the same cluster, thus, the communication data of cluster will be further reduced, but also effectively reduce the communication and adjustment's burden of cluster head services.

The sub-goal execution sequence in local cluster unit's implements optimal aggregation process of participation service clusters that make the QoS performance values of every service nodes with most optimal participation method build topology.

ACKNOWLEDGMENT

This study is supported by National Natural Science Foundation of China (No.60072014), Natural Science Foundation of Shandong Province of China (ZR2012FQ011, ZR2009GM009), the Key Science-Technology Development Project of Shandong Province of China(No. 2009GG10001014) and Promotional Foundation (2005BS01016) for Middle-aged or Young Scientists of Shandong Province, SRI of SPED(J12LN06, J07WH05), DRF and UF(3XJ200903, XJ0609)of QFNU.

REFERENCES

Bosch, J., 2000. Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach. Addison-Wesley, Harlow.

Garlan, D., 2000. Software architecture: A roadmap. Proceeding of the Future of Software Engineering (FoSE 2000). ACM Press, Limerick, pp: 93-101.

Mei, H. and J.R. Shen, 2006. Progress of research on software architecture. J. Software, 17(6): 1257-1275.

Parnas, D.L. and P.C. Clements, 1986. A rational design process: How and why to fake it. IEEE T. Software Eng., 16(2): 251-257.

Sangwan, R., C. Neill and M. Bass, 2008. Houda ZE. Integrating a software architecture-centric method into object-oriented analysis and design. J. Syst. Software, 81(5): 727-746.

Shao, W.Z., 1998. Object Oriented System Analysis. Tsinghua University Press, Beijing.

Xiao-Feng, C., S. Yan-Chun and M. Hong, 2010. Decision-centric software architecture design method. J. Software, 21(6): 1196-1207.