Towards Establishing Production Patterns to Manage Service Co-creation

Shigeru Hosono[1] and Yoshiki Shimomura[2]

[1] Knowledge Discovery Research Labs, NEC Corp., 1753 Shimonumabe, Kawasaki, Japan
[2] Department of System Design, Tokyo Metropolitan University, 6-6 Asahigaoka, Hino, Japan
s-hosono@bu.jp.nec.com, yoshiki-shimomura@center.tmu.ac.jp

Abstract. This paper proposes production patterns for web services to provide a standard procedure to assess and optimize the production processes. The patterns consist of design models in each of design, implementation, test and operation phase, and models of practitioners' tasks and activities through a service lifecycle. These two models are linked together and stored into a repository, and they form patterns of service production. The patterns can give a frame of co-creating services, and it will be used as a template for service production as well as a yardstick to keep track of production progress. A conceptual verification shows that the progress of practitioners' activities can be assessed and the total lifecycle cost of the service can also be estimated, enabling administrators to perform better production management and decision making.

Keywords: production patterns, agile development, critical chain for services, lifecycle cost, PSS for cloud-compliant web services.

1 Introduction

Service oriented architecture (SOA) [1] has introduced flexibility to web services, adapting to changes in business environment quickly. The SOA will be attained through a series of web service technologies [2] - standardized design interfaces and development methods for waterfall-based development. However, all the clients' requirements for web services cannot be determined at the early phase of development as the pace of change has accelerated these years. These changes require an adjustment for the integration of web services from traditional waterfall-based ways to a rather agile style, which develops prototypes of web service continuously while acquiring surfaced requirements. This new paradigm entails service integrators to change their development mindset from product-oriented to service-oriented one; the value of clients' services arises not from the scale or quality of systems' functions but from immediate improvements in their business. Therefore, the involvements of practitioners' activities have become more significant, and the integrators are required to co-create clients' services through a number of hypothetical prototyping continuously. To deal with such environments, activities and their deliverables in each development phase should be managed comprehensively.

2 Design and Operation Patterns

To assist continuous prototyping with hypothetical verifications, the authors have proposed service model chains [3], which is developed under the service design and operation framework [Appendix: *Service Lambda for Cloud Computing*]. A service model chain consists of a set of deliverables in a development process. It can be developed by the continuous design and operation tools of the framework: *service objectives, value chain, goal prioritization, service system design, quality-function insight, application use case, resource combination design, application prototyping,* and *application execution*. While a major concern for a service production, such as design-centered development, implementation-centered development, or lifecycle-centered management, have taken into definite forms, the *service domain* can be determined, and an appropriate tool set is specified from the tool chain. Then, a service model chain, which represents a pattern of design and operation for the domain, is established with the selected tool chain. The service design and operation patterns are stored into the shared service repository - *public service portfolio*, and service integrators are encouraged to reuse and customize these assets for another service production by importing them as templates to each project's repository - *private service portfolio*. With these mechanisms, the service model chains can bring a solution of *mass customization* to productions of cloud-compliant web services.

However, this service model chain is not sufficient to administrate service productions, as it lacks the interactions between stakeholders, i.e. practitioners and users, through a service lifecycle. The traditional development can be regarded as product-oriented, and service model chains are enough to improve service productions as the development is bound to software/hardware products' specifications and behavior. However, the development of cloud-compliant web service has characteristics of services, as negotiations between practitioners account for a higher proportion than composing software and hardware functions of cloud platforms. These practitioners' interactions would rise and fall, while the clients' requirements have developed. Then, tasks will become varied and it will be hard to administrate. To clarify management of such practitioners' tasks in product and service developments, the next section discusses project management approaches, which give a form of practitioners' activities.

3 Task Management in Production Process

Project management methods are approaches to explicitly handle practitioners' activities in product and/or service productions. This section contemplates two notable approaches - Critical Path and Critical Chain, and discusses how these methods can apply to production processes and how they should be extended to exploit characteristics in service productions.

3.1 Critical Path

Critical path [4] is a project management approach to identify one or more sequences of tasks, which will lead a project delay when a task in the sequence delays, and to manage the tasks to keep them on track. A critical path consists of a group of tasks, which have subordinate relationships - for example, task A should start after task B. All work in a production process is broken down to a collection of tasks - WBS (work breakdown structures). All tasks in a production process are depicted in a PERT (program evaluation and review technique) chart, and then the shortest route between the beginning and ending tasks can be identified in the figure. This route is the critical path in this production process. For instance, the shortest route in Figure 1 is the sequence of nodes (tasks) whose floating time are zero; A, B, D and F. With this chart, the total amount of production time and lead time are determined.

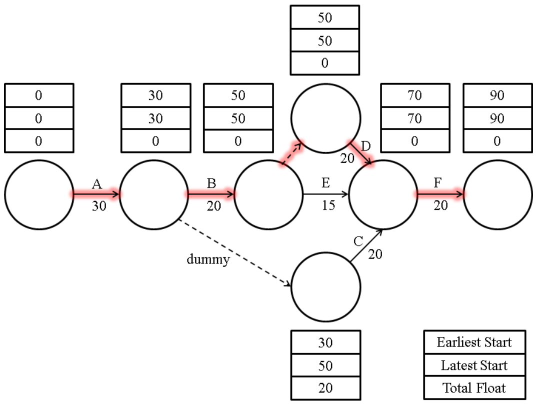


Fig. 1. Critical path

The critical path has been applied to a large scale project and it is aimed at optimizing complex schedules. A task delay in the critical path causes a delay of total schedule, and accelerating tasks to shorten the critical path reduces the total production time. Accordingly, managing the critical path will be pursued in product productions. However, this approach is unfitted to consider undetermined tasks, such as workers' psychology, behavior and social/organizational obstacles in a project.

3.2 Critical Chain

Projects with undetermined tasks can be managed through a critical chain [5] (Figure 2). A critical chain considers how the total development period should be kept or shortened. It contemplates workers' psychology, behavior, and social/organizational problems and enables to optimize project management comprehensively, such as scheduling, task execution and progress management.

The traditional project management based on the critical path assumes that each task should be completed as scheduled. Then, practitioners should try to secure buffering time not to extend the planned schedule with delays of their tasks. However, this redundant schedule may cause a *student syndrome* leading to further delay or phenomena of Parkinson's Law [7] preventing poorly synchronized integration.

To exclude such obstacles, the critical chain introduces *project buffers* and *joint buffers*. These buffers are controlled not by practitioners but by project managers.

Project buffers are defined at the position right after the end of critical path in a PERT chart, and joint buffers are placed at the position where a non-critical path joins the critical path, allowing latitude in project management. Critical chain also takes account of resource contentions, which are frequently observed in delayed projects. For instance, a task cannot be started while the required resources are used by another task. Whenever these tasks are not prioritized, it will cause further delays, requiring resource negotiation.

The critical chain approach considers resource contentions at the early stage of scheduling. When a resource contention occurs, it will become a bottleneck in a project. As this approach is based on the theory of constraints (TOC) [9] aimed at maximizing the ability of bottlenecked tasks, the tasks competing for same resource will be prioritized to avoid overlapping these tasks for the best scheduling.

In these ways, the critical chain approach differs from plans by the critical path management. The critical chain seems appropriate for managing service production. However, the undetermined tasks are not unique features to service production, and the next section discusses how critical chain for services differs from that for products.

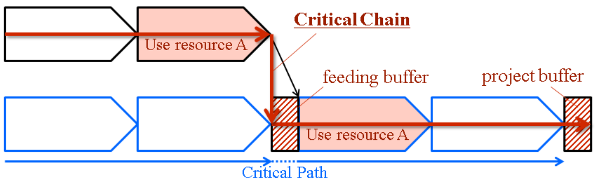


Fig. 2. Critical chain

3.3 Critical Chain for Products and Services

Critical chain for products (CCp) relies on physical resources and it depends on internal constraints. Meanwhile, users' involvement and interactions are mainly engaged in service development. Hence, the critical chain for services (CCs) has dependency on such external constraints. Furthermore, the characteristics of CCs are noteworthy from different perspectives. CCs allows more projects to be completed without increasing staff. Buffer penetration for CCs provides an ambiguous indicator of whether the project is on schedule. Eliminating bad multitasking in CCs makes individuals more productive and less prone to burnout [8].

Similarly to CCp, CCs also addresses resource contention. To put these principles of CCs into practice, it is required to quantify the amount of each task and the relationships between tasks. However, quantifying tasks in service productions includes a wider range of practitioners' activities, and each of activities has been managed by projects administrators and managers manually.

To establish a standardized method to quantify and formally address these practitioners' activities, the next section discusses how task chains can represent activities in conjunction with the service model chains. These two model chains will give a structure to production processes for services and become templates for another service production where agile development [9] has been adopted.

4 Production Patterns to Manage Service Co-creation

4.1 Task Chain

Service is defined as an activity between a service provider and a service receiver to change the state of the receiver [10][11]. Service is composed of 'content' and 'channel'. Service content directly changes the state of receivers. Service channel is the device which indirectly contributes to the change of state of receivers, e.g. communication, supply and amplification.

The service content is to be divided further into a set of functions, and the channel is to be denoted as a flow of the functions. The services, such as consulting, system planning, implementation, or monitoring IT platforms have functions. The channels are in the form of meetings, e-mail, networks, etc. From a lifecycle perspective, these functions and function flows exist not only in the operational phase of the service but also in the phases of ahead or behind: plan, design, implementation, and improvement. Development deliverables, such as requirement definition, system modeling and interface definition, will work as functions, and deliverables between stakeholders in production process will be flows of functions, i.e. 'channel'.

In this way, the whole production process can be regarded as a system in which stakeholders in development processes deliver functions to others. These relationships can reflect a sequence of tasks between practitioners, and a structure of task chain is established (Figure 3). Each task has task names as well as attributes, such as 'man-hour' to complete it.

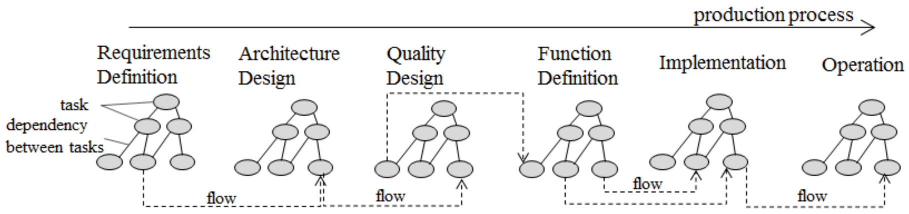


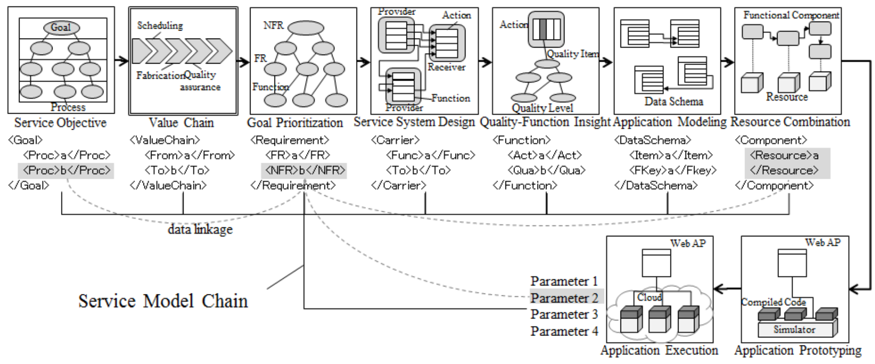
Fig. 3. Task chain

This task chain can be regarded as an abstraction form of CCs, as it is not only a set of each task. The CCs also abstracts feeding buffer, resource buffer, and project buffer. The data linkage represents a sequence of tasks with resource contention, as they reflect optimized sequences of real tasks.

4.2 Production Pattern Development and Reuse

By defining task chains in line with model chains, a complete production patterns can be established with the following steps.

Step 1. Development of Model Chains.
Each service model is developed with corresponding design and operation tools [3]. The developed service model forms structured data in an XML (eXtensive Markup Language) form. The XML data are concatenated along the lifetime phases. Furthermore, each tag data in an XML is linked to a tag in the next phase on the basis of reference relationships. As a result, a service model chain is formulated (Figure 4).



```
<Goal>
  <Proc>a</Proc>
  <Proc>b</Proc>
</Goal>

<ValueChain>
  <From>a</From>
  <To>b</To>
</ValueChain>

<Requirement>
  <FR>a</FR>
  <NFR>b</NFR>
</Requirement>

<Carrier>
  <Func>a</Func>
  <To>b</To>
</Carrier>

<Function>
  <Act>a</Act>
  <Qua>b</Qua>
</Function>

<DataSchema>
  <Item>a</Item>
  <FKey>a</Fkey>
</DataSchema>

<Component>
  <Resource>a
  </Resource>
</Component>
```

Fig. 4. Service model chain

Step 2. Development of Task Chains.
While developing service model chains, corresponding tasks to produce each deliverable, such as reviewing documents and implementing application software, are identified. These tasks are structured and linked to corresponding parts in the service model chain. Task sets in each development phase are also linked along with the production process (Figure 5). The man-hour attributes in the task chain will be translated to costs of practitioners' resources in the next step.

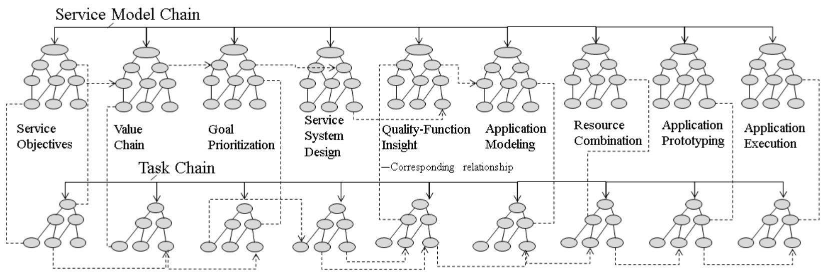


Fig. 5. Association between service model chain and task chain

Besides, this man-hour value can be brought from project management tools, which issue task tickets with planned value and actual performance records of human resources. These practitioners' resource data are integrated to the task chains. Then, a pair of model chain and task chain is stored into repositories as a production pattern for later use (Figure 6).

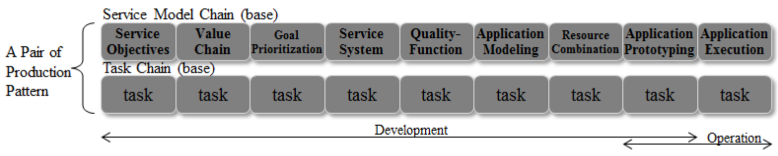


Fig. 6. Production patterns

Step 3. Reuse of Production Patterns.
Service portfolio will be the key to manage these production patterns efficiently. The service portfolio should consist of repositories for managing requirements, design, implementation, operational data and tasks for deliverables. The portfolio should also manage the common and service-specific data separately. Then, two extended portfolios are introduced; the *public service portfolio* is shared by any development project and the *private service portfolio* is owned by each application provider (Figure 7).

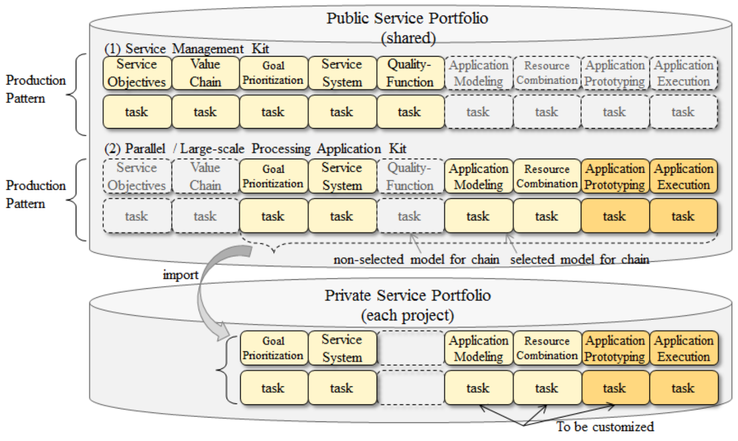


Fig. 7. Public and private service portfolios

When a production pattern is developed, it is exported to the *public service portfolio* after pruning off confidential data, minimizing anticipation that the core asset may have clients' specific data. Then, production pattern can be used for another service development. With this mechanism, a new project of service development can be started by importing it. Service developers can start the web service production from the beginning, and they just customize the interfaces and attain the target web services. In the same way, the second or after cycle in iterative/agile development can be started by importing a partial chain, which corresponds to the spectrum of the original pattern. Therefore, the system integrators can achieve clients' requirements easily and quickly. Furthermore, when the development cycle ends and it returns to the earlier phase for the next development cycle, the man-hour attribute in the partial chain can indicate the lifetime cost of the new development cycle.

4.3 Implementation

To evaluate the feasibility of the steps in the previous section, a production process framework is implemented by enhancing the *Service Lambda for Cloud Computing* framework [Appendix]. This framework consists of service design editors, task management tools, repositories, and analytic tools, which identify similar production patterns from the repositories and identify the influenced part in the patterns (Figure 8).

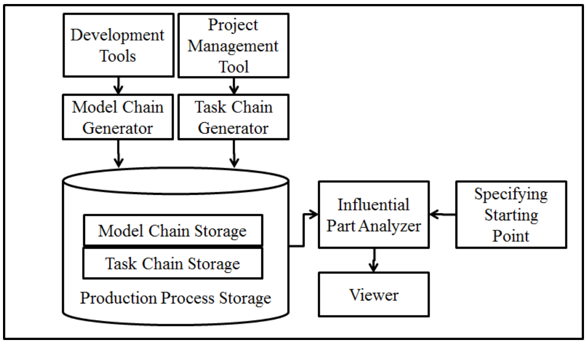


Fig. 8. Overview of production process framework

This framework is applied to a cloud service infrastructure and environment - PaaS (platform as a service), and the productivity and manageability are evaluated by developing a typical cloud-compliant web service.

5 Case Study

The feasibility of our approach is examined through developing a web service, which shows trends of newly-released web contents, such as new product information, from RSS feeds on websites. The crawling module of the web service collects RSS feeds from websites. The service runs alternatively, multi-threads or multi-nodes, based on its configuration rules. While developing this web service, the production pattern is stored into the *public service portfolio*. Then, a similar web service for other clients is developed in an agile development way as follows.

Reuse of Production Patterns.
The model chain and task chain are used as a standard process for web service development. When developers specify a target service to develop, the framework (Figure 8) finds a model chain and the corresponding task chain, which is similar to the target function from the *public service portfolio*, and imports them into a *private service portfolio* and uses them as a base of web service development.

After developing a prototype of web service, the development cycle circulates and returns to the earlier development phase, as the requirements for non-functional requirements are updated. For example, the requirement of its performance is updated to a higher level. To follow this requirement change, the similar nodes/trees, which correspond to the change is identified from the *public service portfolio*. The identified whole/partial chains with task chains are used as a reference for a baseline of the next round development. Using the man-hour attributes in the identified chains, the project owners can estimate the production cost of this development round and also evaluate the progress of development (Figure 9).

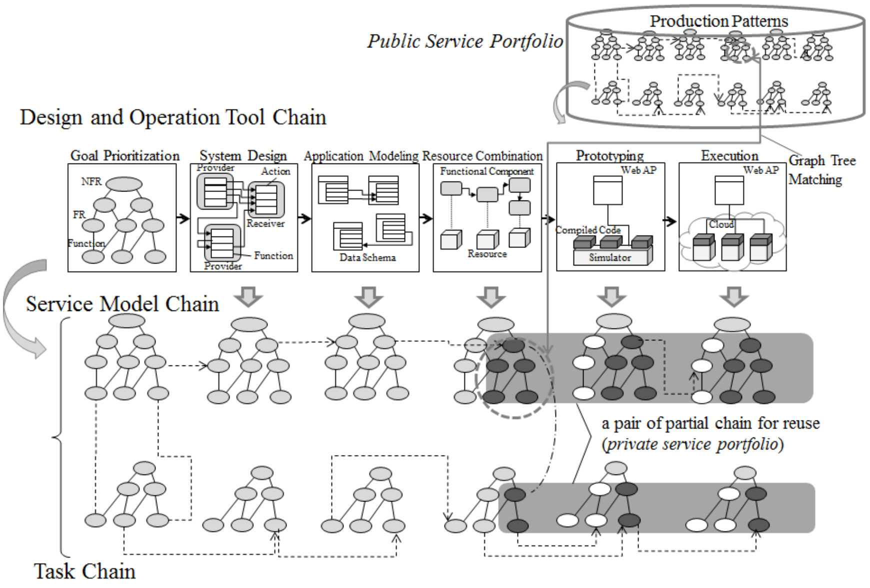


Fig. 9. Reuse of partial production pattern

Assessment of Production Progress.

The reuse of production patterns can be extended for the entire development phases. While developing web services in this agile way, the frequent interactions between practitioners make assessing progress obscure. Earned-value analysis [12] is a good way to assess production progress at the time of evaluation. Through the earned-value analysis, planned value (PV) is defined as the budget given at the evaluated date. Actual cost (AC) is given as the cost that they actually used. Earned value (EV) is determined by the date of measurement. The value of PV, AC and EV can be deemed with the following equations:

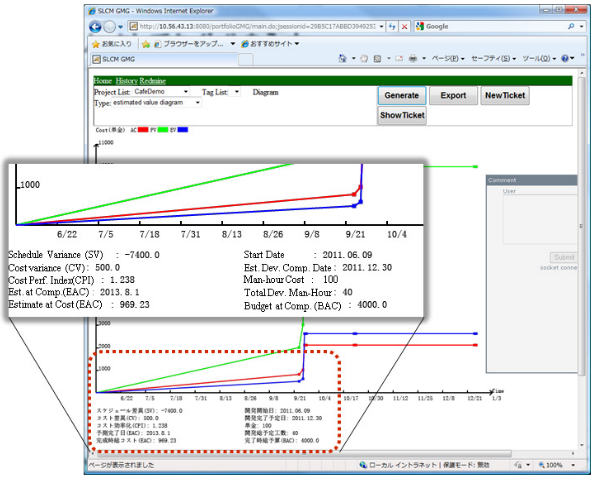
$$\text{PV (planned value)} = \Sigma\ (\text{planned man-hours} \times \text{a unit price}) \tag{1}$$

$$\text{AC (actual cost)} = \Sigma\ (\text{actual man-hours} \times \text{a unit price}) \tag{2}$$

$$\text{EV (earned value)} = \Sigma\ (\text{planned man-hours} \times \text{a unit price} \times \text{progress rate}) \tag{3}$$

The values of PV, AC and EV are obtained through the initially determined data in the framework (Figure 8), such as project start date, estimated end date, the number of iteration time, and the estimated date to complete the cycle. The value of EV at the time of assessment can be generated at any time during the service development as the data stored in the *public service portfolio* and the *private service portfolio* is always available. With PV, AC, and EV, the production progress at the time of assessment and the expected completion date of each iteration development are visualized (Figure 10). In this use case, the production patterns give standard data for 'estimated ending date' and 'required man-hour for the next development cycle', as the partial chain corresponding to the next cycle will provide past actual records of the production period and man-hours.

As illustrated above, the model chain and task chain can be used for estimating lifetime and for decision making of investing to the service development. The models retain a complete production process from the design to operation phases. Therefore, the assets of web service development can be reused and development on customization will become easier.



Schedule Variance (SV) : -7400. 0
Cost variance (CV): 500. 0
Cost Perf. Index(CPI) : 1. 238
Est. at Comp.(EAC) : 2013. 8. 1
Estimate at Cost (EAC) : 969. 23

Start Date : 2011. 06. 09
Est. Dev. Comp. Date : 2011. 12. 30
Man-hour Cost : 100
Total Dev. Man-Hour : 40
Budget at Comp. (BAC) : 4000. 0

Fig. 10. Assessment of Production Progress

6 Discussion

The case study shows that the model chains have increased productivity and manageability of service production. As exemplified in the agile development, a reusable process is easily specified as the beginning of the spectrum is the start point of reusable models, and the spectrum can be easily imported to a *private service portfolio* for reuse. Similar structures are identified through a graph tree matching algorithm, and appropriate chains stored in the repository are selected when a new service project starts. In addition, the selected parts of chains are used to determine the influenced activities and estimate lifetime costs.

On the other hand, the case study of earned value analysis shows that the progress of service production as of time is visualized. The task model structures practitioners' activities and they are used for a standard for assessing the progress of cloud-compliant web services. Hence, the presented approach gives another solutions to lifecycle cost modeling and estimation in the researches on product-service systems (PSS) [13][14][15][16].

7 Conclusion and Future Work

This paper proposes production patterns providing development principles for web services. The patterns consist of design models and task models through a service lifecycle. The task chains give a structure to sequences of tasks and resource coordination between the tasks. This data structure can be stored in a repository. Then, they are reused as a template for service production as well as a yardstick to keep track of production progress. A conceptual verification shows that the progress of practitioners' activities can be assessed and the total lifecycle cost of the service can also be estimated enabling administrators to perform better production management and decision making.

However, the adequacy of each task model chain has been left for future work. As production procedures depend on each organizational culture, the future study extends consideration of such cultural background and identifies an appropriate buffer size for such environment. The customization interfaces for task chains can be identified by introducing roles representing organization to task chains. This mechanism will increase the accuracy of the CCs, and better co-creating services will be depicted clearly. In addition, task chains in this paper focused on practitioners' tasks mainly as hardware and software resources are affluent from the cloud, and they are excluded from the attributes of task chains. To apply this approach to general service developments, the presented approach should be extended to handle resources explicitly.

References

1. Erl, T.: Service-Oriented Architecture: Concepts, Technology and Design. Prentice Hall (2005)
2. Zhang, L.J., Zhang, J., Cai, H.: Service Computing. Springer (2007)
3. Hosono, S., Shimomura, Y.: Towards Establishing Mass Customization Methods for Cloud-compliant Services. In: Proc. of the 4th CIRP Int'l Conf. on IPS^2, pp. 447–452 (2012)
4. Fuller, R.B.: Critical Path. St. Martin's Griffin (1982)
5. Goldratt, E.: Critical Chain. North River Press (1997)
6. Goldratt, E.: The Goal: Beating the Competition. Gower Pub. Co. (1996)
7. Parkinson, C.N.: Parkinson's Law. Buccaneer Books (1996)
8. Ricketts, J.A.: Reaching The Goal: How Managers Improve a Services Business Using Goldratt's Theory of Constraints. IBM Press (2007)
9. Anderson, D.: Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results. Prentice Hall (2003)
10. Arai, T., Shimomura, Y.: Proposal of Service CAD system - A tool for Service/Product Engineering. Annals of the CIRP, 397–400 (2004)
11. Arai, T., Shimomura, Y.: Service CAD system - Evaluation and Quantification. Annals of the CIRP, 463–466 (2005)
12. Fleming, Q., Koppelman, J.: Earned Value Project Management, Project Management Institute (2000)
13. Abramovici, M., Jin, F.: A New Approach to Executive Information Management as Part of IPS^2 Lifecycle Management. In: Proc. of the 2nd CIRP Int'l Conf. on IPS^2, pp. 543–547 (2010)

14. Roy, R., Erkoyuncu, J.A.: Service Cost Estimation Challenges in Product-Service Systems. In: Proc. of the 3rd CIRP Int'l Conf. on IPS^2, pp. 1–10 (2011)
15. Fernandes, P., Roy, R., Mehnen, J., Harrison, A.: An Overview on Degradation Modeling for Service Cost Estimation. In: Proc. of the 3rd CIRP Int'l Conf. on IPS^2, pp. 309–314 (2011)
16. Komoto, H., Tomiyama, T.: Life Cycle Cost Estimation using a Modeling Tool for the Design of Control Systems. In: Proc. of the 18th CIRP Int'l Conf. on Life Cycle Engineering, pp. 663–668 (2011)

Appendix: Service Lambda for Cloud Computing

The authors have proposed the design and operation framework, *Service Lambda for Cloud Computing* to provide comprehensive platforms for developing and operating cloud-compliant web services [3] (Figure 11, 12). This platform is a client-server system. The client-side tools cover system design and implementation phases, providing development tools as well as collaborative development environments for practitioners. The server/cloud-side tools manage the applications' lifecycle and also provide an execution environment for them.

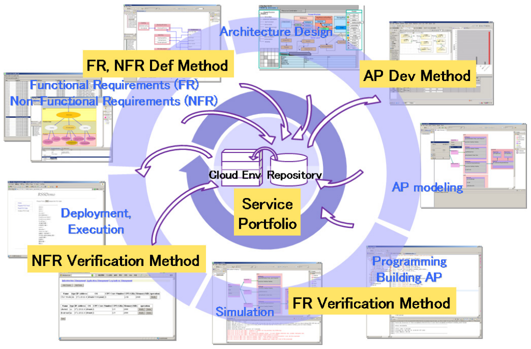


Fig. 11. Overview of Service Lambda for Cloud Computing [3]

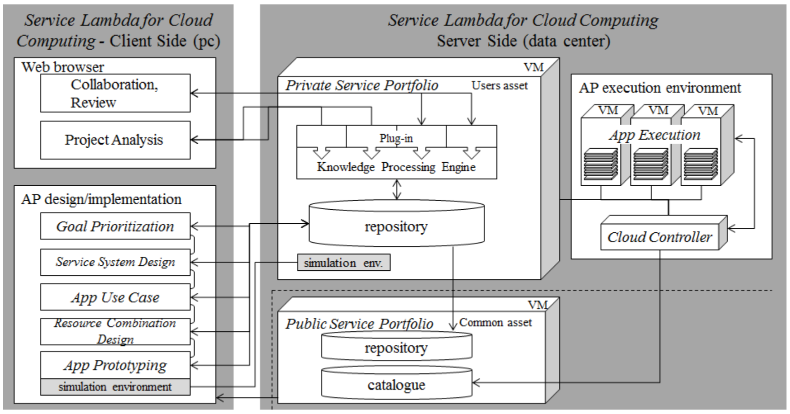


Fig. 12. Architecture of Service Lambda for Cloud Computing [3]