# The Design of An Accountability Framework for Service Engineering *

Kwei-Jay Lin
*Dept. of Electrical Engineering and Computer Science*
*University of California, Irvine*
*Irvine, CA 92697, USA*

## Abstract

*The goal of our service science and engineering research is to advance IT-based services technology for service innovation, composition and delivery. In this project, we study the service accountability framework to detect, diagnose, and defuse the real problem in a service process. This is important when a service process has external service providers. We need an efficient mechanism to ensure that external services do provide an acceptable level of performance and stability, in order to meet the overall quality of service (QoS). Our design adopts Bayesian Networks to look for the most likely causes in a service process and inspects those services individually to identify the problem. We implement the LLAMA accountability service bus, a dynamic and efficient service infrastructure to support the monitoring, analysis, and reconfiguration of service processes. It includes accountability agents to continuously monitor services and an accountability authority to identify faulty ones.*

## 1 Introduction

The transition to services-centered economy is an important and inevitable step for many countries to continue their economic growth. The capability to innovate knowledge-intensive services, to create business and societal values from expertise in diverse domains, and to design, develop, and deliver new technological and business services is critical for building up successful service enterprises. In recent years, business globalization, increasing automation, and the componentization of business functions have lead to the remarkable growth of innovative web-supported services. Advances in the Internet technology and the popularity of web-centered life style have further created new business opportunities and service values at an unprecedent scale.

The goal of our service science and engineering research is to advance IT-based services technology by developing a service framework and companion toolset for service innovation, composition and delivery. Our research also aims at inventing and reinventing business models to target toward the 21st century societal and business environments. The study takes a systematic approach to service engineering. By building a service infrastructure using current and next generation IT technologies (including wired and wireless Internet, service-oriented architectures, embedded devices, sensor network, etc.), we hope to provide the scientific and technical foundation necessary for many enterprises to transition from product-oriented business operations to high-value service-centered businesses.

Many of today's enterprise services may include external service partners. For example, small hospitals rely on external specialists and laboratories to perform medical or laboratory tests; engineering consulting firms rely on law firms to conduct patent search or filing. However, when a service process has external service providers, it is imperative to have a mechanism to ensure that external services do provide an acceptable level of performance and stability, in order to meet the overall quality of service (QoS) requirements. In other words, the *accountability* of individual services should be effectively watched in order to attribute credit for success or responsibility for failure in a service process. Any ill-performed service should be replaced immediately to maintain the required QoS levels.

To reach such goals, service engineering should have easy-to-use mechanisms to conduct:

1. monitoring of services and identification of faults or likely problems,

2. inspection of the internal state of individual services, and

*This research was supported in part by a Visiting Fellow grant (96-2811-E-001-003) from the National Science Council of Taiwan, ROC, conducted at the Institute of Information Science, Academia Sinica, Taipei.

1530-1605/08 $25.00 © 2008 IEEE

3. the dynamic reconfiguration of services and service processes.

In this paper, we present the service accoutability framework to *detect*, *diagnose*, and *defuse* service deficiencies (such as logistic errors or service level agreement (SLA) violations). The accountability framework provides the mechanisms for: (1) QoS-based service selection; (2) service SLA monitoring; (3) Probabilistic reasoning to identify the likely cause of a problem; and (4) a trust and reputation mechanism for preventing future problems.

We have designed the LLAMA (inteLLigent Accountability Management Architecture) middleware software, which provides a dynamic and efficient e-service accountability IT infrastructure to support the monitoring, analysis, and reconfiguration of service processes. It includes the mechanisms to route and to monitor e-services within Service-Oriented Architecture (SOA), and dynamically adapt by rerouting around faulty services. We extend an open source MuleESB (Enterprise Service Bus) [13] to incorporate routing and interception mechanisms. Specifically, we create the LLAMA ASB (Accountability Service Bus) by adding additional features to support: 1) dynamic routing, via a routing table and a performance interceptor, 2) efficient and tunable profiling and logging components to report performance data regarding services and the host, and 3) a configuration gateway (CGW) to support the configurability of external service components.

This paper is organized as follows. Section 2 provides the background on accoutability and ESB. The system engineering process for accountability systems is presented in Section 3. The accountability model and reasoning algorithm are presented in Section 4. The LLAMA accountability-based infrastructure is presented in Section 5. Concluding remarks are given in Section 6.

## 2 Background

Accountability has been a major concern in the financial industry, especially after ratification of the Sarbanes-Oxley Act of 2002 (also known as the Public Company Accounting Reform and Investor Protection Act of 2002), which establishes new enhanced accountability standards for all U.S. public company management and public accounting firms. The Act has made accountability a mandatory requirement for organizations. A new agency, called the Public Company Accounting Oversight Board, is given the responsibility of overseeing, regulating, inspecting, and disciplining accounting firms in their roles as auditors of public companies. The Act provides the motivation for our research on SOA accountability as services should be similarly regulated for effective QoS delivered by a service process.

In [9], a project on results-based accountability for public institutions has been reported. It identifies the following elements for systems with accountability:

1. Objective: Outcomes that articulate what programs are to achieve;

2. Quality: Indicators to measure whether or not outcomes have been achieved;

3. Benchmark: Performance standards to assess how programs are progressing;

4. Monitoring: Data collection instruments to regularly obtain indicator data;

5. Feedback: Periodic collection and analysis of data for decision making and reporting.

Among the five elements of a complete accountability measure, the first three are application-dependent and should be defined by application designers. Information technology may be used to implement the other two. We therefore focus our study on the mechanisms for performance monitoring and accountability analysis.

### 2.1 The Enterprise Service Bus

David Chappell defines [3] an ESB as "a standards based integration platform that combines messaging, web services, data transformation, and intelligent routing to reliably connect and coordinate the interaction of significant numbers of diverse applications across extended enterprises with transactional integrity". The goal of ESB is to provide a connectivity layer to facilitate the creation of Service-Oriented Architecture (SOA) by supporting the integration of existing service components. It does this by providing a reliable messaging bus and allows interoperability along several communication transport protocols. ESB provides a distributed approach to integration, that allows individual enterprise units to build up their integration projects in incremental steps, maintaining their own local control and autonomy, while still being able to connect together.

In addition to interoperability, some ESB provides facilities for message routing, filtering, and message transformations. Message routing allows a service request to be delivered to a service provider without the service client's detailed knowledge on the exact format of a target service. Service messages can be transformed and translated according to the platforms that clients and servers reside.

Other features that may be included in an ESB include service coordination between concurrently running services,

message distribution between a client and more than one services, QoS support such as security and reliability. Overall, a capable ESB may improve the simplicity of SOA deployment and the scalability of SOA systems. In this research, we extend an ESB to support service process accoutability.

## 3 Accountability System Engineering

Our research goal is to design an accountability framework that supports service workflow monitoring and reconfiguration. Through the framework and the supporting IT components, service development complexities may be offloaded from the service developer to the IT infrastructure. In order to do this, the framework must:

- easily identify and configure services and framework components, such as agents, loggers, and QoS brokers;
- *transparently* intercept service invocations to allow logging (to agents and local repository) for auditing and diagnosis;
- allow *dynamic and efficient reconfiguration* of service workflows to support continuous optimization according to the QoS requirements.

In our accountability framework (Figure 1), two major components are the Accountability Authority (AA) and Accountability Agents. They collaborate to perform service process monitoring, root-cause diagnosis, service network recovery, and service network optimization. As shown in Figure 1, multiple agents are deployed by AA to address the scalability requirement. Each agent is in charge of monitoring a subset of services during the execution of the process. Whenever the process performance is unacceptable, agents then forward the recorded service status information to AA so that AA may identify the actual cause of the problem.

### 3.1 A Motivating Example

Figure 2 shows a motivating example of the credit service workflow in the context of the lending application [1]. The business workflow provides the functionality that allows a loan officer to obtain the credit report of a customer in real time. In the figure, each rectangular node represents a simple service running either internally or by an external service provider.

The *lending life cycle service* initiates the credit checking as a response to a user request by sending a Process Credit message to the credit service. The *credit service* passes the request to the external vendor interactions service which in turn places a request with credit vendors A and B who provide credit query services. The credit reports obtained from vendors A and B are used in some local processing and then transferred and stored in *document service*. Finally, the credit report is sent to the customer. This business process is completely automated and is executed without human intervention.

The response time for a service flow is critical to customer satisfaction. If a *Lending Lifecycle Service* is initiated but has not received the credit report back within a reasonable amount of time, the service is considerd as a failure. However, the problem could originate from any of the services involved in the process. Therefore, an accountability mechanism can be used to identify the problem. For example, in Figure 2, a delay from the *Vender A Order-Process Service* could render the *Local Processing Service* to become slow as it has to wait for the credit report from A, even though the *Local Processing Service* actually functions well by itself. Therefore, the most critical service that has violated the SLA should be identified and repaired instead of those intermediate services affected by the root cause.

### 3.2 Accountable Service Engineering

The engineering flow of an accountable service process deployment is shown in Figure 3. The accountability support is provided during service process execution (in the circled area of Figure 3) so that run-time exceptions can be easily observable. The service process engineering flow includes the following steps.

- *Service Network Planning*: This is the first step in service process deployment. Given a service request from a client, all applicable service plans are collected to build a function graph. The mapping from a client request to applicable process plans should satisfy the functional requirements of the client without considering its QoS or accountability issues. Parametric consistency checking between individual services is performed in order to integrate them together. This problem has been studied by earlier research projects such as [16, 18].
- *QoS-based Service Selection*: Given a function graph, this step performs the service flow composition by selecting individual services based on their QoS parameters and a user's QoS requirements, such as response time, cost, and reputation of a service. In [19], QoS based service selection is achieved by efficient combinatorial and graph algorithms. If any component service fails or becomes overloaded during the execution of a service process, a back-up service path excluding

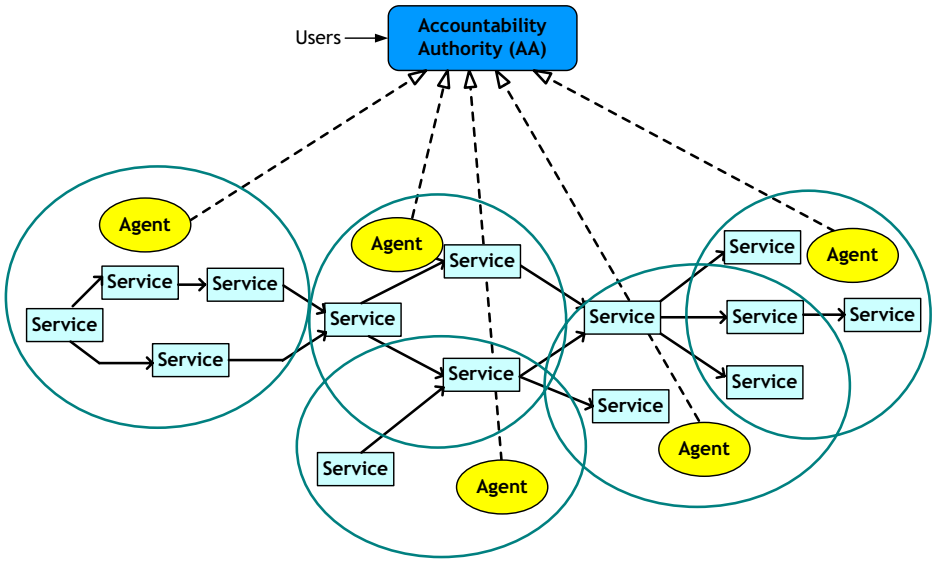


**Figure 1.** System architecture of accountability framework

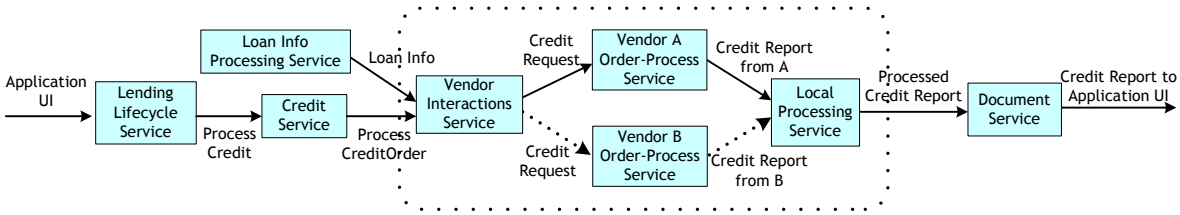


**Figure 2.** The credit service workflow in the context of a lending application

the failed service is created, so that ongoing executions will not be disrupted. Algorithms to compose back-up service paths have been studied in [17].

- *Service Process Execution*: Shwon as the circled area in Figure 3, this is where the accountability mechanism is supported. It includes the following functional steps.

    1. *Agent Deployment*: Given a composed service network, agents are deployed to monitor all atomic services selected as well as the services in backup paths. Agents are responsible for reporting performance exceptions and the states of service outputs to the Accountability Authority (AA). The issue of how to effectively deploy agents is modeled as a set covering problem: given (1) each agent's capability on the set of services the agent can cover, and (2) set of services, the goal is to select the minimum number of agents that are capable of covering all selected services. The set covering problem is NP-hard. However, approximation algorithms have been shown to provide feasible solutions [6].
    2. *Evidence Channel Selection*: The amount of collected information and the number of locations from where it is collected have a big impact on the monitoring performance. Although it is possible for agents to check the output of all services, it is not efficient to do so since status collection imposes extra system overheads and delays. Agents may want to monitor only a subset of *evidence channels*, just enough to support problem diagnosis. The evidence selection problem can be modeled as *k-median facility location problem* [2, 4]: given a service graph, identify the best locations to place shared facilities so that the total travel cost for all clients are minimized. The k-median problem is NP-Hard. Approximation algorithms have

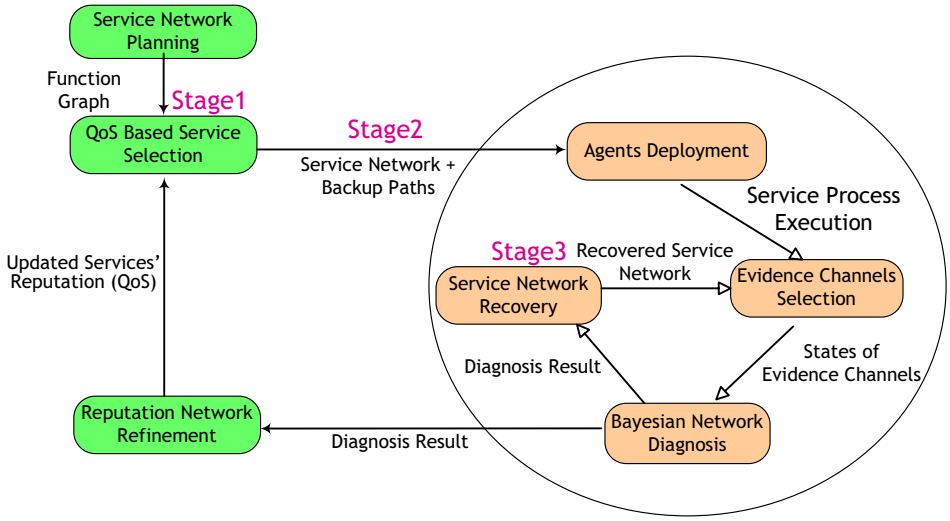


**Figure 3.** Engineering of service deployment with accountability

also been designed to provide solutions. For example, [2] provides local search heuristics for the *metric k-median problem.*

3. *Network Diagnosis*: AA leverages the Bayesian network reasoning mechanism to identify the root cause of a problem when violations of SLAs occur in a system. Based on the information monitored by agents, AA performs an effective diagnosis to determine the most likely service that causes the service process problem. The Bayesian network reasoning process will be discussed further in Section 4.
4. *Service Network Recovery*: When problematic services are identified based on the diagnosis result, back-up paths can be used to replace problematic services quickly [17]. This simple fix can resume the process execution flow immediately. Since agents are deployed to cover all services including backup paths, there is no need to re-deploy the agents. This service recovery step can save significant time and cost. Once the service process is repaired, a new set of evidence channels will be selected.

- *Reputation Network Refinement*: The reputation of a service is an important QoS parameter that affects the service network recomposition in the long term. Services that are less likely to violate its SLA are more likely to be selected next time when a new service process is composed. Distributed reputation management framework, such as that in [12, 20], can be used to evaluate, aggregate, and manage the reputation information scattered in a distributed computing environment.

## 4 Accountability Diagnosis for Service Processes

### 4.1 Assumptions

Our system model describes service applications with multiple services as a *flow* $G = (V, E)$, such as business services networks and service supply chains. Each vertex in $V$ represents an atomic service and each edge in $E$ represents an interaction between two services.

In the current study, we make the following assumptions on service systems.

1. The service flow $G = (V, E)$ is a directed acyclic graph (DAG). For any service $s$, there is no directed loop edge starting and ending on $s$ in the flow.
2. The behavior of a service is independent of other services.
3. The network connection between services is error-free, but individual services may be problematic.

### 4.2 Diagnosis with Bayesian Networks

Bayesian networks [5] are directed acyclic graphs for reasoning under uncertainty, where the nodes represent random variables (discrete or continuous) [11]. The arcs connecting the nodes can be used to represent direct causal relationships [14]. An arc from A to B indicates that A causes B. A Bayesian network stipulates that each node, $V_i$, in the graph is conditionally independent of any subset of the nodes that are not descendants of $V_i$.

Bayesian networks is adopted in our accountability mechanism as the diagnosis engine for the following reasons. (1) It is based on the directed acyclic graph (DAG) model, a match for the service flows we are studying; (2) It is capable of reasoning with the causal relationship in the service network, another important feaure for business processes; (3) It can handle the uncertainty in service networks. In a service-oriented computing environment, service nodes can be considered as random variables whose actual performances are probability values. Furthermore, the causal relationships among service nodes also may not be deterministic. Bayesian networks thus match major diagnosis requirements in our design.

#### 4.2.1 Configuring Bayesian Networks Parameters

In order to make Bayesian networks reasoning effective, we need to make sure that the conditional probability functions of each node are conditioned only on its parents. For nodes without parents, the probabilities are not conditioned on other nodes; these are called the *prior probabilities* of these variables. For nodes with parents, a *conditional probability table (CPT)* needs to be defined for each node variable to make the Bayesian network inferencing feasible [14].

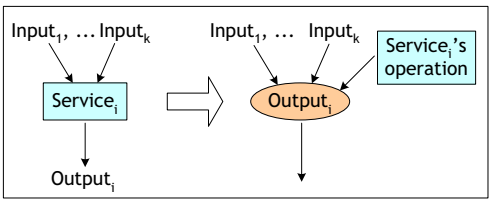


**Figure 4. Transforming a service node to random variable nodes in Bayesian network**

In business networks, all arcs represent the execution flow among nodes. While in Bayesian networks, all arcs connecting the nodes must represent direct causal relationships among nodes. Therefore, some steps are necessary to transform the business network topology into a Bayesian network topology.

Therefore, the following parameters need to be specified.

- Prior Probabilities of the root nodes (rectangle nodes in Figure 4). These root nodes represent the web services functionality according to the network transforming rules. Therefore, their prior probabilities can be considered as the reputation of services, which is a continuous value in [0,1]. The reputation of services are dynamic and accumulated from the historical feedbacks of the service behaviors. The AA in our accountability model is in charge of the management of those reputation values for each service.

- CPTs of the non-root nodes (elliptical nodes in figure 4). These non-root nodes represent the outputs of web services. These CPTs are elicited from the service providers. Those numerical probabilities may be frequency data extracted from databases (frequentist interpretation), be based on expert judgment (subjectivist interpretation), or be a combination. They give a complete list ($2^{n+1}$) of probabilities in the format of $P(output|input_1, input_2, ...input_n, service)$, which specifies the probability that the $output$ is correct in a combination of the correctness of $input_1, input_2, ..., input_n$, and the $service functionality$.

As shown in the left part of figure 4, in the business network each service node has several inputs and outputs. The result of output is impacted by correctness of all inputs and the service node's function. In other words, all of the inputs, as well as the service node's functionality, are the causes of the output. Therefore, as represented in the right part of figure 4, to represent the causal relationship correctly in the Bayesian network, the service node functionality should be extracted out as a root node (the rectangle shaped node) and the output should exist as an individual node (the elliptical shaped node) in the Bayesian network. At the same time, the output node of the services in the upstream become the input node for services in the downstream.

Based on above, the original travel planner business network is transformed to the Bayesian network with parameters configured as illustrated in Figure 5. The parameters are randomly made up in the figure to give a sense of the parameters configuration. In practice, the prior probabilities should come from the services' history behaviors and the CPTs are estimated by the service providers.

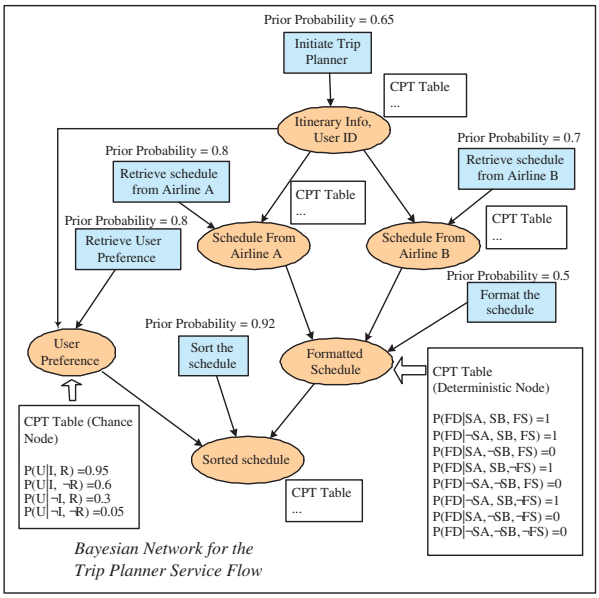


**Figure 5. The Bayesian Network with CPT Parameters**

#### 4.2.2 Reasoning with Bayesian Networks

In Bayesian network transformed from the service network, all of the nodes representing services' outputs become the evidence nodes. Their status may be observable by the monitoring agents. And all of the nodes representing the services' functionality become query nodes. Their status is not observable by the agents. The accurate knowledge on their correctness is done by checking the service log, which is costly and time-consuming.

The goal of the diagnostic network is to find the root cause of the problem node. Firstly, we produce the posterior probability of every query node Q, notated as Bel(Q), upon the observation of a set of evidence nodes $\{E_1, E_2, ..., E_n\}$. The inference results provide the recommendations on where is the most likely problem in term of probability. After that, we can further inspect the logs of service nodes in the order of their faulty probabilities to finally diagnose the root cause service node.

Compared to randomly select the service nodes to check the service log, the Bayesian network inference process can save much service log checking time and cost. The extraction of service functionality as the query node during the network transforming is helpful to reason the root cause of problem in the Bayesian network. With the causal relationship set and all probabilities parameters specified, the diagnosis process always recommends to the service node which is most likely to break the SLA in a faulty path.

Suppose there are $m$ service nodes in a service network, the diagnosis procedure is as follows:

***Diagnosis Algorithm***
***Input***: *The status of an evidence nodes set* = $\{E_1, E_2, ..., E_n\}$
*1.* ***for*** *every query nodes Q*
*2. Infer the posterior probability Bel(Q)*
*3. Sort the inference results: Bel($Q_1$), Bel($Q_2$), ..., Bel($Q_m$) in the order of decreasing probability*
*4.* ***for*** *each query node $Q'$ in the sorted order*
*5.* ***If****(the service log is found to be problematic)*
*6. return $Q'$ as the node in error;*
7. **End If**
8.**End for**

**Table 1. Netica's output for travel planner (*UserPreference* and *FormattedSchedule* are set to false)**

| Query Node | Bel(Q=false)(%) |
|---|---|
| $InitiateTripPlanner$ | 85.8 |
| $RetrieveUserPreference$ | 31.2 |
| $RetrieveScheduleFromAirlineA$ | 25.9 |
| $RetrieveScheduleFromAirlineB$ | 34.7 |
| $FormatSchedule$ | 47.8 |
| $SortSchedule$ | 8 |

In the trip planner example, assume we observe that both *UserPreference* and *FormattedSchedule* are slow, we want to figure out which node causes this problem. Table 1 shows the inference results using Netica [7], one of the world's most widely used Bayesian network development software.

**The Complexity of the Diagnosis Algorithm** The main complexity of the diagnosis algorithm is from the Bayesian network reasoning. Bayesian network inference algorithms are computationally complex, i.e. they are NP-hard problems. However, there exist several efficient algorithms that allow Bayesian network inferencing for tens or hundreds of variables tractable. Those algorithms include the message-passing scheme [15] and an efficient algorithm that first transforms a Bayesian network into a tree [10]. In most practical networks with tens or hundreds of nodes, Bayesian updating is fast and takes only between a fraction of a second and a few seconds [8]. For most business processes, the size of the service networks should be within a few hundreds

nodes. Therefore, we believe the diagnosis algorithm is computationally tractable for real-world applications.

## 5 Accountability Support using LLAMA

Figure 6 shows the components involved in the accountability architecture. In addition to the service requester and service providers, LLAMA includes:

- **QoS broker**. A broker provides process planning and service selection, in order to assist the service requester in meeting the end-to-end QoS requirements for a service process. A broker's jobs include:
  - **Tracking**, which is responsible for storing both functional and QoS data about various service candidates.
  - **Planning**, which helps the user to compose a business process at the functional level based on various requirements.
  - **Selection**, which helps the user choose specific services to fulfill each specific functional need based on the user's quality of service requirements.
- An **accountability authority** (AA). AA is responsible for the diagnosis of services with problematic behaviors in order to ensure that the process produces desired outcome. Moreover, it is responsible for initializing a runtime process reconfiguration when the original process fails to produce desired outcome.
- **Accountability agents**. They are used to monitor the behavior of the web services to which they are assigned and report monitoring information to the accountability authority.
- **Trust brokers**. Trust brokers are responsible for evaluating, aggregating, and managing the reputation of services. Service's reputation is a QoS parameter that affects the service network composition: services with better reputation are more likely to be chosen.
- A **reputation authority**. This is a third-party component that is queried by the trust broker for information regrading reputation information for various services.
- A **negotiation broker** for service negotiation. The broker can be used by the service requester to cutomize or semi-customize a service process to meet the user's needs.
- A **service process engine** for service process orchestration. Such an engine can be used by the service requester to coordinate a service process.
- The **LLAMA ASB**. This component of the framework provides the infrastructure for easy and scalable integration of system components and services in the process. Moreover, it provides fundamental supports for automatic and dynamic profiling and reconfiguration of service processes.

Many of the LLAMA components are being implemented in our research. The components have well-defined capabilities to support the accountability management. They are able to accept policy specifications defined by users so as to provide customized accountability measures.

## 6 Conclusion

Service industry is qualitatively different from merchandize exchange as business activities. Service activities may be used to deliver help, utility, experience, information, or other intellectual content to its clients. They are much more dynamic and unpredictable in both demand and supply. They are also very quality-sensitive and trust-dependent. Questions such as what the best ways are to maintain desired service levels while increasing efficiency and productivity, and how a service can guarantee end-to-end manageability and visibility throughout the entire services cycle should be addressed in the study of service science and engineering. In this paper, we present the accountability framework for service engineering. By deploying services on the LLAMA ASB, we make services and service processes easier to monitor, to inspect and to manage. In the future, we will study how to facilitate the innovation of new services on LLAMA by integrating existing and new service capabilities.

## References

[1] M. Acharya, A. Kulkarni, R. Kuppili, R. Mani, N. More, S. Narayanan, P. Patel, K. W. Schuelke, and S. N. Subramanian. SOA in the real world - experiences. In *Proceedings of the 3rd International Conference on Service-Oriented Computing*, pages 437–449, 2005.

[2] V. Arya, N. Garg, R. Khandekar, K. Munagala, and V. Pandit. Local search heuristic for k-median and facility location problems. In *ACM Symposium on Theory of Computing*, pages 21–29, 2001.

[3] D. Chappell. *Enterprise Service Bus*. O'Reilly Media, 2004.

[4] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k -median problem (extended abstract). In *ACM Symposium on Theory of Computing*, pages 1–10, 1999.

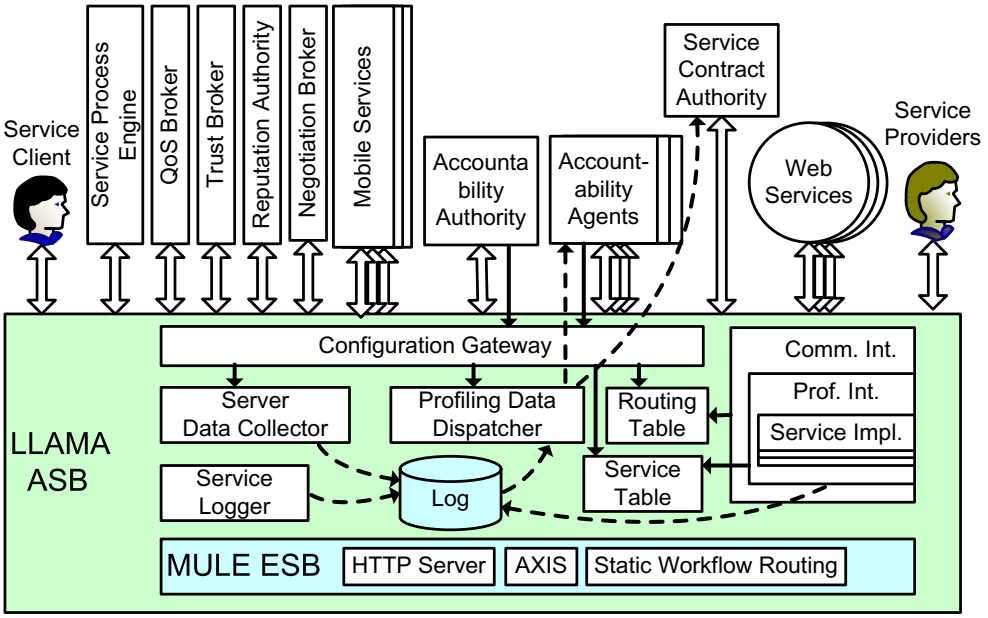


**Figure 6.** LLAMA Components


[5] E. Charniak. Bayesian networks without tears. *The American Association for Artificial Intelligence*, 1991.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. The set-covering problem. In *Introduction to Algorithms, Second Edition*, pages 1033–1038. MIT Press and McGraw-Hill, 2001.

[7] N. S. Corp. Netica: Bayesian network development software. *http://www.norsys.com/*, 1995-2006.

[8] DSL. SMILE (structural modeling, inference, and learning engine). *Decision Systems Laboratory(DSL), School of Information Sciences, University of Pittsburgh, http://genie.sis.pitt.edu/*, 2006.

[9] K. Horsch. Results-based accountability systems: Opportunities and challenges. *The Evaluation Exchange*, II(1):http://www.gse.harvard.edu/hfrp/eval/issue3/theory1.html, 1996.

[10] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2001.

[11] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. Chapman & Hall/CRC, London, UK, 2004.

[12] K.-J. Lin, J. Y. Hsu, Y. Zhang, and T. Yu. A distributed reputation broker framework for web service applications. *Journal of E-Commerce Research*, 7(3), 2006.

[13] MuleSource. The mule project. *http://mule.codehaus.org/display/MULE/Home*, 2007.

[14] N. J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann Publishers, Inc, San Francisco, California, 1998.

[15] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

[16] S. R. Ponnekanti and A. Fox. SWORD: A developer toolkit for web service composition. In *11th World Wide Web Conference, Honolulu, Hawaii*, May 2002.

[17] T. Yu and K.-J. Lin. Adaptive algorithms for finding replacement services in autonomic distributed business processes. In *The 7th International Symposium on Autonomous Decentralized Systems, Chengdu, Jiuzhaigou, China*, April 2005.

[18] T. Yu and K.-J. Lin. A broker-based framework for QoS-aware web service composition. In *IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-05), Hong Kong, China*, March 2005.

[19] T. Yu and K.-J. Lin. Service selection algorithms for composing complex services with end-to-end QoS constraints. *In Proc. 3rd International Conference on Service Oriented Computing (ICSOC2005), Amsterdam, The Netherlands*, December 2005.

[20] Y. Zhang, K.-J. Lin, and R. Klefstad. DIRECT: A robust distributed broker framework for trust and reputation management. *IEEE Conference on e-Technology, e-Commerce and e-Service (EEE'06)*, June 2006.