Comparison of Web Service Architectures Based on Architecture Quality Properties

Chen Wu and Elizabeth Chang
School of Information Systems, Curtin University of Technology
Perth, Australia
{chen.wu, elizabeth.chang} @cbs.curtin.edu.au

Abstract - **Web service research has been focused on the issues of automatic binding, performance, scalability, and security, however, little research has been done in evaluation of web service architectures, namely Broker based. Examples of these are Matchmaker Broker, Layered Matchmaker, Facilitator, Layered facilitator, and Peer to peer (P2P) based, such as P2P Discovery, Match Maker and P2P, Split Code and P2P execution, Mobile Code with P2P etc. Another consideration is its impact on the adoption in distributed Internet environment. In this paper we introduce a methodology for measuring and evaluating web service architecture style, and we present our development of a set of architectural quality properties, and use these quality properties to carry out comparison and contract of current web services architectures. We provide a detailed analysis and critique of these, and these could be served as a guidelines for the next generation of web services development, which could adopted into the distributed environment.**

I. Introduction

While a large number of enterprises are now developing web services technologies (e.g. WSDL, SOAP, UDDI) to build information systems[1], the majority of the adoption of current web services only occur on Intranets rather than on the Internet, where formal 'Web Services' originated. One of the underlying reasons for this approach is that the true nature of web services adoption challenges has been relatively unstudied. It has been addressed using either the WS-* standards enhancement or the detailed work-around solution, rather than the rigorous software architectural approach established by formal software engineering discipline. Therefore, in this paper, we investigate and review state-of-the-art architectural styles, and evaluate them against those well-identified quality architectural properties. The major contributions of our research in this paper lie in four folds:

- We introduce a methodology of measuring and evaluating architecture styles for web services;
- We identify key architectural properties related to quality aspects of web services architecture;
- We summarize nine web services architectural styles in current literature; and
- The evaluation analysis result delivers useful decision support for web services architect.

This paper is organized as follows: Section 2 introduces the architectural methodology used in this research. In section 3, we summarize the architectural properties against which our evaluation and comparison work can be achieved. Sections 4 and 5 explore two different categories of common architectural styles used in academia and industry. Formal comparative evaluation studies of these architectural styles are discussed in Section 6. Conclusion and future prospects for future work are provided in Section 7.

II. Architectural Measurement and Evaluation

The early comprehensive research about software architecture can be found in Perry and Wolf [2]. They built the foundation for software architecture, and presented a model consisting of three components: elements (what?), form (how?), and rationale (why?). Based on Perry and Wolf [2] data-centric and property-based approach, Fielding [3] further described that 'software architecture is defined by a configuration of architectural elements constrained in their relationships in order to achieve a desired set of architectural properties'. Architectural properties define the minimum 'load-bearing wall' of the architecture design based on system requirements (functional or non-functional, e.g. quality), which in turn constrains the overall architectural design.

It is worth noting that non-functional properties are often ignored in architectural design [4]. Nevertheless, we believe it is such non-functional requirements, quality properties in particular, that make tremendous difference between a system that 'just works' and one that 'works well' [5]. In this paper we focus on identifying intrinsic quality-based architectural properties inherent in Internet-wide distributed web services systems. The use of patterns and styles is pervasive in software discipline to leverage past experience in order to produce better designs. Software architectural style encapsulates important decisions about the architectural elements, and it emphasizes important (not necessarily more) constraints on the elements and their relationships.

0-7803-9094-6/05/$20.00 ©2005 IEEE

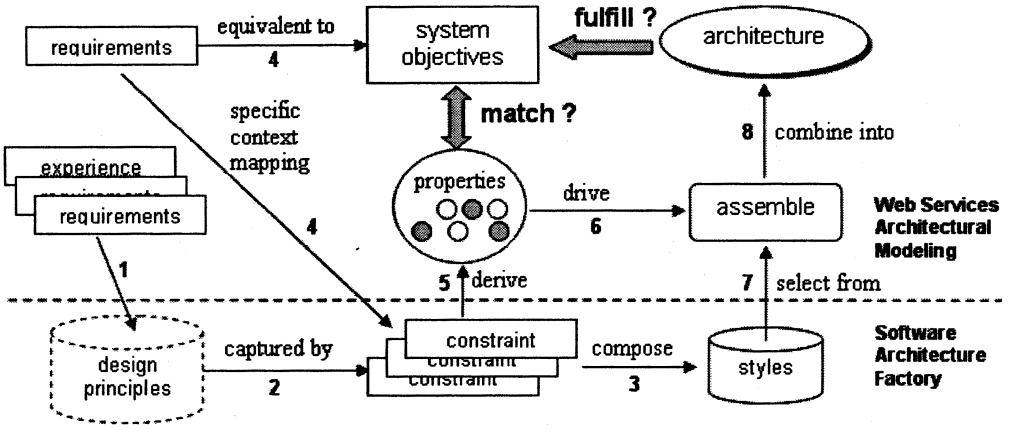

Figure 1. Architectural methodology

Having these basic understandings of software architecture, we now can propose our architectural methodology used in this paper to compare and evaluate web services architecture. Figure 1 depicts this methodology process flow and major artefacts with their relationships. The methodology is basically separated into two parts: architecture modelling and software architecture factory. Architectural factory collects user requirements from various applications, experienced advice from software engineers and architects led to the technology being organized into architectural design principles, which can be furthered captured by architectural constraints [2 and 3]. These constraints can be composed and combined into various architectural styles stored in the factory. During the architectural modelling time, the application requirement will form into a specific business 'context' mapping to one or more constraints, and eventually derive certain architectural properties from these mapped constraints.

Based on these derived properties, an architect is able to select relevant styles that induce such properties and combine them into the final architecture. From the figure 1, it is not difficult to find that in order to evaluate an architectural design, the most effective way is to check whether it fulfils all the system objectives reflecting equivalent system requirements. Thus, our goal is to examine the architectural design rationale behind the constraints it places on a system. In order to achieve this goal, we need to compare the quality properties derived from constraints which form into the architectural styles to the objectives of the target system – Internet-wide distributed web services.

III. Architecture Quality Properties

We now explore related architectural properties in a context where distributed web services interacting with each other across the Internet. Based on previous related work [3, 6 and 7], we deduce architectural properties specifically for Internet-based web services in distributed e-Business environment.

- Loose-coupling – A flexible relationship between two or more computer systems that are communicating via data transmission. Loosely-coupled systems are bound to work well (e.g. without human intervention, at low cost) when either side of the computer systems are subject to frequent changes.
- Interoperability – The ability of two or more systems or components to exchange information and to use the information that has been exchanged. Current WS-Architecture utilizes XML and Internet protocols (such as HTTP) to achieve the interoperability as basic level interactions.
- Scalability – The capability of the architecture to support large numbers of web service consumers and providers, or large amount of interactions among these consumers and providers without making major changes to the existing systems or application software and hardware.
- Simplicity – The architecture does not impose high barriers to entry for its intended adopters: each individual component in this architecture should be substantially less complex as to the extent that they will be easier to understand and implement, otherwise functionality of that component needs to be reallocated (further decomposition or distribution).
- Extensibility – The ability of the architecture to dynamically accommodate changes without impacting the rest of the system. For instance, the architecture should allow the service consumer to dynamically add additional information to the message other than what is specified in the shared schema.
- Performance – This may include network performance, user-perceived performance, and network efficiency.
- Security – The security of Web services across distributed domains and platforms and privacy protection for the consumer of a Web service across multiple domains and services.
- Reliability – The degree to which architecture is susceptible to failure at the system level in the presence of partial failures within components, connectors, or data.
- Visibility – The ability of a component to monitor or mediate the interactions between two other components. Visibility can affect several other architectural properties such as performance, reliability and security. For instance, the header of the SOAP can help content routing and security inspection through the gateway or the company firewall.
- Composability – The ability of the architecture to enable the services be composed at lower cost (e.g. without human intervention) to execute certain business process smoothly and securely.

IV. BROKER-BASED ARCHITECTURE

Brokers (a.k.a Middle Agent in agent research literature) are widely used in distributed information systems such as multi-agent systems and distributed databases. Wong and Sycara [8] presented a comprehensive and systematic taxonomy for middle agent in the context of multi-agent systems. In their taxonomy there are generally two kinds of brokers: Matchmaker and Facilitator. The facilitator differs from matchmaker in that besides finding services, it also intermediates transactions between the consumer and provider. Their research is based on the observation of the multi-agent environments; however, the research result can apply to the web services literature very well since they share the same nature such as distributed environment, service consumer, and service provider.

A. *Matchmaker Broker Style (MB)*

Current web services architecture is based on the basic broker architectural styles – matchmaker style, within which a service provider registers with the central UDDI registry its capability information and later a service consumer contacts the registry to discover this service provider detail information so that it can bind and interact with it. All providers must make their services available by publishing their interface and thus advertising their service. This is a straightforward approach to distributed computing that provides the advantage that clients are coupled to the servers only via an interface rather than any service implementation details. An MB architectural style can be seen as the combination of client-server (CS), or client-stateless-server (CSS) and peer-to-peer style. Figure 2 presents such centralized services architecture and their used styles. From the figure, we can see that both 'Publish' and 'Find' will suffer the disadvantages of CS style. In addition 'Find' will obtain some advantages of CSS style if it is utilized. The binding and interaction (step 3 and 4) process will benefit from the advantages of Peer-to-Peer style, which could be varieties of message exchange patterns: synchronous request/response, event-based, or asynchronous message such as publish/subscribe. Matchmaker Broker (MB) style captures such loose-coupling property as the most classic web services architectural style.

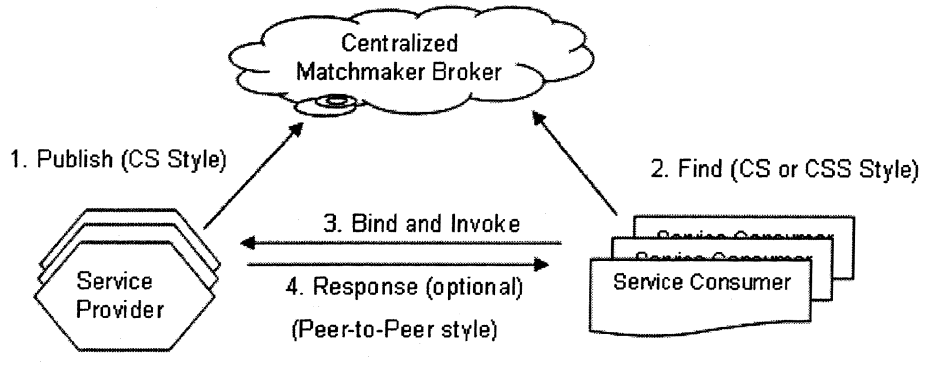

Figure 2. Matchmaker Broker Style

B. *Layered Matchmaker Broker Style (LMB)*

Some researchers enhance matchmaker broker style mainly deal with service selection based on the behaviour (e.g. Quality-of-Service, non-functional requirements) of, rather than simple function matching in the process of selecting service providers [9, 10 & 11]. Thus CCS style needs to be enhanced by being combined with other styles. Yu and Lin [10] proposed a Quality-of-Service capable web service architecture – QCWS by a QoS broker module collecting QoS information of providers, making selection decisions for clients and negotiating with providers to ensure QoS commitments. Wang, Yue, Huang and Zhou [9] maintained that because of the distributed nature of web services, it is very difficult to manage and predict the quality of web services. Thus a service broker – dynamic service selection engine, which provides QoS based service selection, is proposed in their broker architecture. The broker architecture given by Degwekar, Su and Lam [11] – Constraint-based Broker – allows the service requestor to specify their service requirements by extending the standard WSDL specification, thus a service broker is able to select the service that best satisfies the consumer's requirements. All these work, in an implicitly or explicitly manner, add a middle layer to the existing matchmaker broker style in order to augment the insufficient support of service behaviour (e.g. QoS) from existing registry – UDDI. We can use Layered Matchmaker Broker (LMB) style to describe such architectural design. As depicted in Fig. 2, LMB comprises of Layered-Client-(Stateless)-Server Style (LCS) and Peer-to-Peer style. The publisher and requestor brokers (or these two might come as a whole component) work as a 'gateway' to encapsulate the request by adding or removing some criteria such as QoS or user requirements, sending to the inner layer – the standard UDDI registry. Apparently, some extended data need to be stored in the intermediary layer to help augment the standard discovery and selection process.

The primary problem with MB and LMB lies in their centralized indexing scheme provided by web services registry – UDDI. 'It does not scale well because the number and physical distribution of the UDDI clients can quickly overwhelm this centralized configuration and can lead to serious performance bottlenecks' [12]. Adding more servers

or implementing load-balancing strategies does not constitute a practical solution as it implies high cost for the operators of the UDDI [13]. Moreover, the current search facilities offered by the latest version of UDDI do not offer any special features for finding anyone of the distributed web services registries themselves [14]. Besides the performance bottleneck and single point of failure suffered from centralized systems, the possible storage of vast numbers of advertisements on centralized registries hinders the timely updates, thus it is questionable whether centralized registries will scale up to the needs of web services [15].

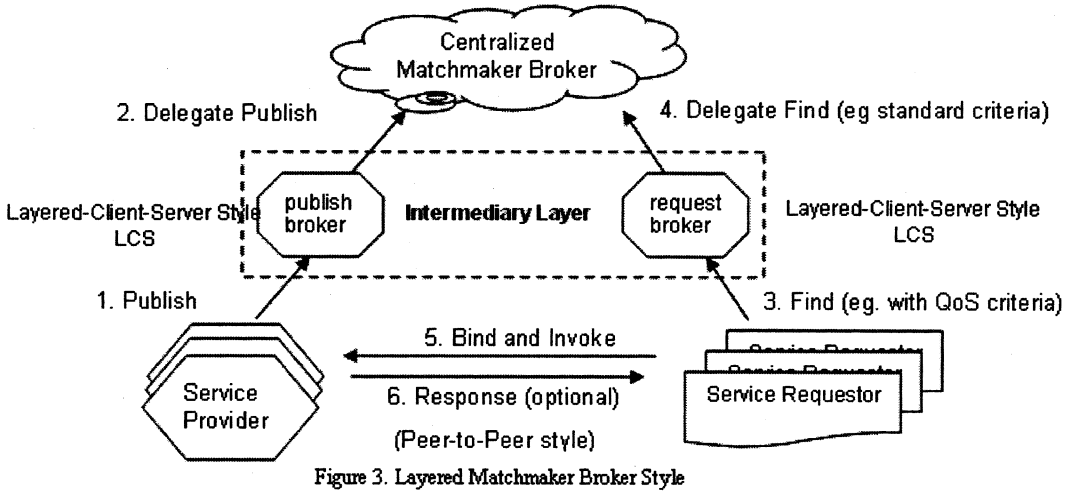

Figure 3. Layered Matchmaker Broker Style

C. *Facilitator Broker Style (FB)*

The facilitator brokers in services research literature basically carry the role of service interaction compared to discovery and selection provided by Matchmaker brokers. To further reduce the coupling between service provider and consumer, which in the case Matchmaker style is direct Peer-to-Peer style, facilitator broker delegates all the message request and response between them. Some practitioners and commentators regarded facilitator broker as 'SOA Fabric' – a central message environment that hides the complexity of message exchanges and other interaction issues from service providers and consumers [16]. Moreover, some researchers utilize such facilitator brokers to provide value-added mediation services such as homogenizing the heterogeneities among different web services [15 & 17] The essential issue that Fuchs's broker aims to solve is the heterogeneity: web services interfaces – WSDL – is 'too easy to be written to specific requirements without being required to support a variety of possibilities that can show up in a loosely-coupled heterogeneous environment where different parties are evolving at different rates'. Based on the existing SOA, the author proposed architecture to wrap the underlying services with an adaptation layer – the service facilitator broker – to deal with all the aspects of message handling in heterogeneous environment. The adaptation layer intermediates between the underlying, or base, operations and the outside world. Paolucci et al.[15] provided a detailed analysis about broker's requirement and architecture regarding a broker that performs both discovery and mediation (i.e. facilitator). Their result indicates broker should have powerful reasoning capability in accomplishing necessary tasks (e.g. interpretation, finding, invocation and returning results). The SOA Fabric forms the basic facilitator broker architectural style as shown in Figure 4.

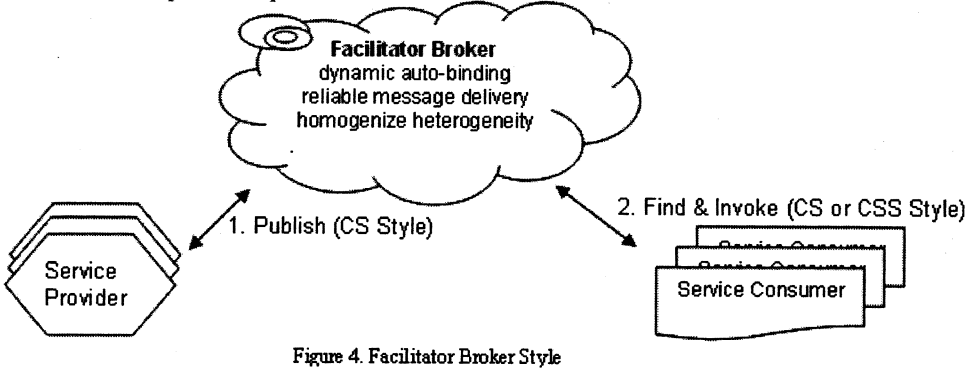

Figure 4. Facilitator Broker Style

D. *Layered Facilitator Broker Style (LFB)*

Basic facilitator style can be enhanced in several ways. Piers, Benevides and Mattoso's [18] broker layer addressed the issue of heterogeneity when composing distributed web services. They pointed out that existing mediator architectures cannot be directly applied to develop distributed web services composition since the web services mediation requires homogenization of different service interfaces – message formats. The authors explicitly articulated why and what are the heterogeneity problems (such as semantic dissimilarities etc.) in distributed web services composition. The multi-layered architecture of *WebTransact* illuminated some clues in

introducing the layered facilitator architecture style for web services.

V. PEER-TO-PEER ARCHITECTURE

Peer-to-Peer computing is based on the principles that 'the world will be connected and widely distributed and that it will not be possible or desirable to leverage everything off of centralized, administratively managed infrastructure' [19].

A. *P2P Discovery Style (P2PD)*

The most common application of leveraging Peer-to-Peer technology in distributed web services environments is in the area of service discovery [20, 21, 22, 23, 24, 25 & 26]. Prasad and Lee [22] presented the PSI model to locate suitable services in peer-to-peer networks (a hybrid P2P infrastructure) using registry implemented by web services, though web services here just play trivial role in establishing the invocation interfaces between varieties of distributed nodes. Palucci et al. [23] proposed a structured P2P based web services discovery method that considers the process behaviour as well as functionality of web services. Emecki, Sahin, Agrawal and Abbadi [24] proposed decentralized discovery architecture based on a P2P connection between Web services, where services capability matching will be performed on the Gnutella P2P infrastructure network. Their core work is to combine the DAML-S matching with the Gnutella QUERY process and the use the basic Gnutella protocol for Web services discovery.

Schmidt and Parasher [25] described each web service by a set of keywords and then mapped the corresponding index to a DHT (Distributed Hash Table). One of the most valuable contributions of Ayyasamy, Patel and Lee [20] is the summarized justification of combining the Web Services and DHT based Peer-to-Peer networks in service discovery. Banaei-Kashani [26] introduced the WSPDS (Web Services Peer-to-Peer Discovery Service), a fully decentralized and interoperable discovery service with semantic-level matching capability.

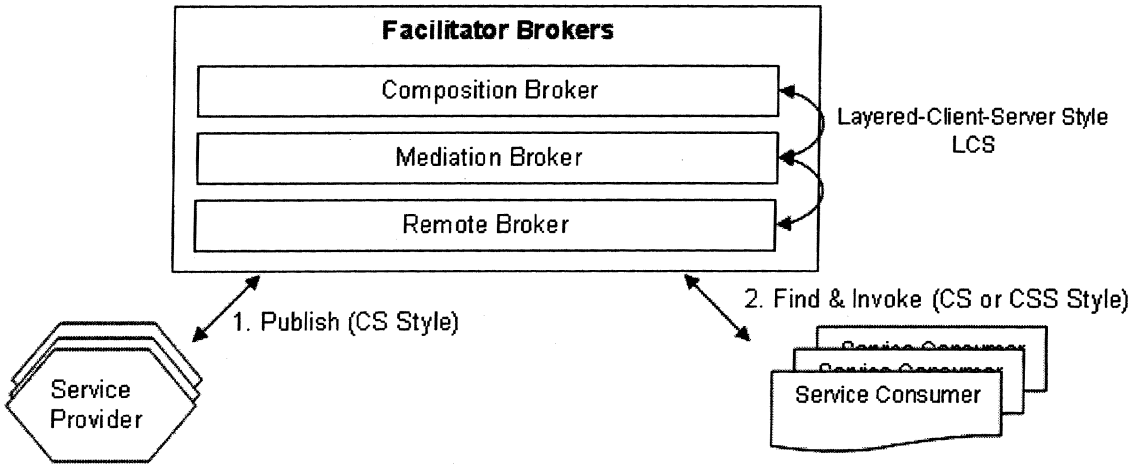

Figure 5. Layered Facilitator Broker Style

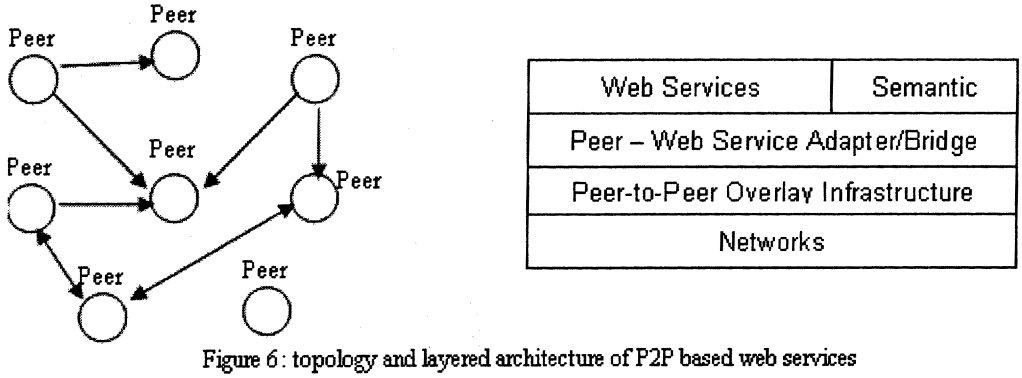

Figure 6: topology and layered architecture of P2P based web services

In figure 6, each peer represents a provider or a consumer or both. There is no a centralized registry to store the meta-data for all the participated peers, the service discovery relies solely on the capabilities of each peer by leveraging some P2P service discovery algorithms (such as flooded request, document routing, etc). Figure 6 also gives an example of integrating the web services and P2P using the layered structure. While the bottom two layers remain unchanged as are in common P2P model, the Peer-WS Adapter/Bridge layer plays the most important role here in that it bridges the gap between P2P communication protocol and existing web services protocols. In particular, this layer transforms the web services and services semantics description into the interfaces syntax and semantics that are understood by P2P overlay, thus wrapping the web services providers into common peers

which can consume the existing P2P services provided by the underlying overlay.

However, such pure P2PD style has one critical problem: no existing registries (e.g. UDDI) are utilized in their service discovery approach, hence the feasibility and compatibility of their research is questionable since they require the complete abolition of existing service discovery mechanisms, which are already well accepted as normative industry standards in web services practice. In addition, this style will introduce new security and trust threat inherent in P2P computing paradigm.

B. Matchmaker + P2P Discovery Style (CSS-P2PD)

To utilize existing registry (i.e. UDDI) in service discovery, some of the alternative P2P approaches have been developed to decentralize the conventional centralized UDDI architecture using P2P technology [27, 12 & 14]. Sivashanmugan, Verma and Sheth [14] focuses on leveraging registry federations supported by different domain ontologies, they mainly dealt with how web service discovery is carried out within a federation by presenting a scalable, high performance environment for federated web service publication and discovery among multiple registries [27] specifically addressed the problem of distributed registries (UDDIs) using peer-to-peer technology. They built the service registry peer-to-peer infrastructure based on Edutella P2P environment. Papazoglou, Kramer and Yang [12] also employed a federation of UDDI-enabled peer registries that operate in a decentralized fashion rather than requiring each peer to publish their own service descriptors locally or centrally (on the UDDI).

Papazoglou et al. [12] envisioned that 'a P2P network architecture that promotes a logically decentralized arrangement of registered service descriptions and that also provides web-service descriptions much in the same way that UDDI does'. In this research, registry is responsible for discovering the immediate services registered locally. If the requested services cannot be found locally, the registry will form the global query delegated to other registries via the P2P network. The major difference between Papazoglou, et al. [12], Thaden et al. [27] and Sivashanmugan, et al. [14] is that Sivashanmugan et al. [14] used DAML-S as service description language instead of standard UDDI tModel. In summary, their architectures can be illustrated in Figure 7. Service providers register themselves into the local registry, which in turn form the P2P network registry federation in a decentralized way. Local registry will normally be responsible for discovering the immediate services registered locally. If the requested services cannot be found locally, the registry will form the global query delegated to other registries via the P2P network. Such architecture combines both P2P style and CS (CSS) style

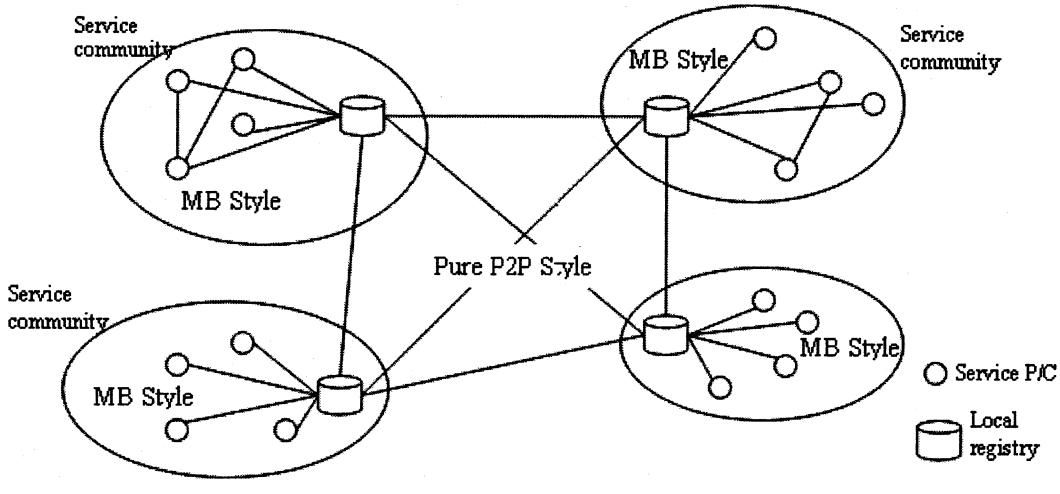

Figure 7. Federated UDDI architecture using P2P model

C. P2P styles in service composition

Existing web services processes (e.g. BPEL4WS - Business Process Execution Language for Web Services) are normally executed by a single centralized coordinator node (e.g. the BPEL engine in the case of BPEL4WS). The coordinator is responsible for controlling and driving the process execution, all the data should be transferred via this central node to the involved web services providers. Apparently, this leads to overload traffic on the network, poor scalability and performance degradation [28]. In industry practices, existing IBM's MQSeries Workflow or Microsoft's BizTalk offering well-engineered web services execution guarantees, however, are not able to optimally distribute the overall load among all service providers at run-time. Since 'these systems follow a centralized architecture consisting of dedicated workflow engine(s), their scalability is limited' [29]. For instance, both Lakhal, Kobayashi and Yokota [30] and Chafle, Chandra and Mann [28] in their respective research postulated multiple distributed engines architecture to coordinate and communicate with each other in service composition and execution; nevertheless, they in fact utilized two different methods respectively.

D. Split Code + P2P Execution Style (SC-P2PE)

This style assumes that distributing the execution of processes necessitate the partition of process specification at design-time, and during the run-time each local engine

only obtains the partial copy of the whole process, and finally executes it at local site where the invoked service resides. The early rigorous studies on partitioning process (workflow) specification can be found in Muth, Wodtke, Weissenfels and Kotz [31]. They provided an algorithm for transforming a centralized state and activity chart into a provably equivalent partitioned one, suitable for distributed execution using multiple process engines. However, their work is not suitable for web services process composition and execution. Project SELF-SERV [32] is the early work that addressed the problem of decentralization of web services processes composition and execution using such partition method. There are, to our best knowledge, two researches so far working on the partition of BPEL4WS. Nanda, Chandra, and Sarka [33] presented a new code algorithm to partition a composite web services written as a single BPEL program into an equivalent set of decentralized processes, with the goal of minimizing communication costs and maximizing the throughput of multiple concurrent instances of the input program. Based on Nanda et al.'s [33] research result, Chafle et al. [28] proposed a decentralized BPEL composite scheme which contains multiple engines, each executing composite web service specification at distributed locations. One of the problems is that the process might have to halt in the case of unavailability of certain involved service providers. Another problem is how to deploy these partitioned process pieces to those involved peer process engines at run-time.

E. Mobil Code + P2P Execution Style (MC-P2PE)

Unlike design-time partition where involved services providers are already explicitly specified during the partition, in process-mobilization method, the process specification does not indicate the concrete service providers (i.e. the peer engine). Moreover, both the whole process specification and its related instances, which contain their execution states, have to be dynamically brought to the hosts on which the services reside during the run-time of the process. Based on these issues, Haller and Schuldt [34] presented the AMOR system that utilizes the mobile agent to encapsulate the process specification brought to each host where the desired services are invoked by such mobile agent. Lakhal, Kobayashi and Yokota [30] propose the architecture – THROWS – for a highly available distributed execution of web services compositions. In THROWS, the execution control is hierarchically delegated to peer-to-peer collaborated service process engines discovered dynamically using the CEL (Candidate Engine List). In other words, the engine which executes that service is also decided in an ad-hoc manner. The mobility of process is implemented using the message communication between peer engines.

F. Split Code + Mobile Code + P2P Execution Style (SC-MC-P2PE)

Schuler, Weber, Schuldt and Schek [29] employed the process mobilization method to propose a true peer-to-peer service process execution runtime. The process mobility is achieved by deploying the Two-Phase-Commit protocol to ship the process instance to the target node against the meta-information replicated from the global repositories to each local HDB layer. However, in order to achieve better performance and reduce the amount of data (process specification and process instances) to be replicated, Schuler, Weber, Schuldt and Schek [29] also utilized the process partition method to decompose a process into a set of distributed execution units, which only contains the information to execute the locally corresponding service and to navigate the process depending on the result of the service invocation.

VI. Compare and Contract of Web Service Architectures

A. Method

Here we use a table of style versus architectural properties as the primary tool for evaluating and comparison studies. Firstly, for each quality property that we investigated, we assign a weight value to indicate the significance that property contributes to the total objectives of the system under certain business context. The metrics for this weight is an integer ranging from 1 to 5, showing the increasing trend of significance. The rationale of assigning weight is that architectural property may present different momentum in contributing to the system objectives under different business context. Secondly, for each style we will also assign a number against certain property it will induce.

The value of this number indicates the degree to which such style is able to exhibit the characteristics of this particular architectural property under certain business context. The metrics for this number is an integer ranging from -2 to 2 inclusive. It is worth noting that if we adjust the weight for each property; the total score will change accordingly, impacting our architectural style decisions. This reiterates nothing but the principle that different requirements and context will generate different architectural styles.

Style	Derivation	Loosely Coupling	Interoperability	Scalability	Simplicity	Extensibility	Performance	Security	Reliability	Viability	Composability
Weight in the architectural context of "Internet-wide distributed web services"		5	5	5	3	4	5	3	4	2	2
Broker	Matchmaker Broker	1	2	-1	1	0	-1	0	-1	-1	-2
	Layered Matchmaker Broker	1	0	0	2	1	0	1	1	1	-1
	Facilitator Broker	2	2	-2	-2	2	-2	-1	-2	2	1
	Layered Facilitator Broker	2	2	-1	-1	2	-1	-1	-1	2	2
P2P	Pure Peer-to-Peer Discovery	0	-2	2	0	0	1	-2	2	-2	-2
	Matchmaker + P2P Discovery	1	1	1	0	0	1	-1	2	-1	-1
	Split Code + P2P Execution	2	2	2	-1	-1	2	0	1	-2	2
	Mobile Code + P2P Execution	0	1	2	-1	1	1	-2	0	-2	1
	Split Code + Mobile Code + P2PE	1	1	2	-2	-1	2	-2	0	-2	1

Table 1. Architectural style evaluation table

B. Analysis and Critiques

As depicted in Table 1 in matchmaker broker style, service consumer and provider are loosely-coupled, they are not aware of each other until binding occurs between them. The WSDL obtained from a UDDI registry makes such syntax-level automatic binding and further interaction possible though on the semantic level, service matching and selection is still far from desirable, thus needs much human intervention. Although interfaces and protocols in such component technologies are described in proprietary binary encoding rather than 'standardized' format which results in implementation interoperability problems preventing traditional component technology from becoming dominant in the Internet era. However, we should distinguish such 'implementation' interoperability from architectural interoperability, which only considers the component and their interaction relationships at abstract level. Matchmaker broker represents the current most popular architecture of web services, hence the interoperability is relatively guaranteed if all the components use well-accepted standards such as (SOAP, WSDL, and UDDI) to communicate. Scalability is one of the main concerns [12, 13 & 15] of this architectural style since the centralized broker will soon become the bottleneck and single-point-failure as the number of consumer and provider increases across the Internet.

Meanwhile in all the broker styles, the complexity of the architecture concentrates on one centralized broker. This situation can be alleviated by adding intermediate layer into the broker styles (LMB and LFB) where complexity is split and separated into each layer of the broker; however, this will impact other properties such as performance, security, etc since it introduces extra communication and confidential data transfer between layers of the central broker. For the similar Internet network environment, the performance of matchmaker broker is generally better than facilitator in that P2P interactions greatly reduce the communication and computation time burdened by the central broker. The rationale of introducing facilitator in the architecture is to ensure the functional requirements – the service consumer sometimes is unable to automatically bind to the identified provider without any intermediation from brokers due to high heterogeneities across the distributed Internet. Hence, we found that the architectural design decision between these trade-offs should not be determined or as late as possible until detailed business contexts including hardware configurations are clearly identified. Facilitator broker further deteriorate the reliability in that even basic connections among any services cannot proceed regardless of whether they are aware of each other. Replication of broker data is one of the methods to improve the reliability. The latest UDDI specification (UDDI v.3) has enhanced the replication model among distributed UDDI registry servers. However, some researchers argue that such replication solution is not feasible and empirical: it needs not only a replication contract between both registry providers but also manual system administration for each (new private) registry. Therefore, practical replication between UDDI registries does not occur [27]. Wang et al also pointed out that such replication approach causes other problems such as expensive data replication, unnecessary global service querying when local providers can satisfy the demands [21]. We should notice that broker architectures especially the facilitator broker greatly reduce the coupling between consumer and provider. For instance, facilitator brokers can easily achieve loose-coupling by adopting the asynchronous messaging exchange pattern, thus working like an asynchronous message middleware broker. Owing to this, extensibility property induced by this architectural style is quite prominent. Ideally, the providers can even change their interfaces without impacting the involved service consumer since facilitator will be responsible to interact with the updated provider

interfaces during the run-time. Matchmaker style does not monitor too much on the interactions among services. Layered matchmaker stores some data in the intermediate layer so that some historic data can be captured for future usage and this will result in the visibility of the interactions and security threats as well. This visibility in matchmaker still rely on the active involvement of stakeholder providers and consumers, who need to explicitly notify the broker intermediate layer attributes about the past P2P interactions, which brings about extra burden for those peers. The visibility in facilitator is rather implicit as long as detailed monitor mechanism is enabled, stakeholders can only focus on their own business, which is visible to the facilitator broker for further usage. Privacy and trust concerns of brokers are some other potential problems accompanied with visibility. One of the advantages of visibility is the centralized and consistent cache control from the brokers.

Formal and precise cache control will dramatically improve the performance and scalability of the systems. Matchmaker brokers only concern the functional requirements (interface signature) during service discovery hence it helps little in the later composition phases of web services. Layered matchmaker broker improve this by fostering the non-functional criteria in assembling related services providers, thus paving the way for feasible composition though itself does not involve the composition process. Facilitator broker is able to compose the services by mediating the invocation among services, while layered facilitator enhance such functionality by masking the local and remote services and providing the uniform environment layer for composition.

A paradox occurs when introducing broker-based architectural style into the web services architecture: it is imperative to have central broker facilitating service discovery, binding and interaction due to the heterogeneity, trust, and security reasons among highly autonomous web services from different organizations across the Internet; on the other hand, broker style raises scalability and performance issues since it originated from traditional distributed object systems where components providing services are confined in the controlled domain with limited boundaries.

Existing P2P web services architectural styles centre on one principle: to decentralize resources control – discovery, synchronization, coordination, execution, etc. The pure P2P discovery style presents the most common usage of P2P computing model leveraged in web services architecture. Most of the cases complied with such style use existing P2P overlay as the default service discovery mechanism. Since such overlay often supports ad-hoc connectivity which in turn achieve the loose-coupling among peer web services. Loose-coupling based on the assumption that all the peer-service consumer or provider are all employ the same P2P communication protocol such as P2P overlay discovery mechanism. The downside of such default P2P overlay discovery mechanism comes from its scarce acceptance by the majority of web services so far. It is questionable whether such service network can interoperate with other web services outside of such overlay across the Internet, which is apparently not ubiquitous and uniform as is current WWW protocol and web services standard (UDDI, SOAP). This is exactly where style comes into play.

To respect the existing service discovery standard, this style makes each UDDI registry as peer rather than turning each service provider or consumer into peer. By doing so, it achieves the decentralization of data while keeping the current centralized service discovery mechanism within each UDDI community. Such hybrid style is making a compromise between two extremes: total centralization and total decentralization. However, the interoperability problem in this style is not thoroughly clear: the communication protocol between UDDI registries is undefined by any standards so far and will certainly introduce complexity when any attempts of augmenting UDDI specifications occur. Scalability is greatly improved in pure P2P discovery due to the decentralized query and search mechanism. While for the Matchmaker + P2P discovery style, load balancing among federated UDDI registries becomes significant in determining the overall scalability of the architecture. Appropriate learning algorithms are desirable to ensure that query load is indeed 'decentralized' across each one of these registries and dynamically switch to those with relatively more free resources. All this work requires inevitable enhancement of UDDI registry server, thus raising the interoperability issues again. Thanks to the ad-hoc connectivity nature of P2P network, the reliability issue appears obtain sheer resolved by avoiding utilizing any single-point-failure and bottleneck-prone brokers during the service discovery process. Nevertheless, the performance is not guaranteed to improved, and in some occasion might become even deteriorated due to the unstructured nature of P2P network and frequent communications among registries which can cause much network latency in an unexpected manner. Security is probably the biggest concern when introducing P2P style into web services. Recent research of security and trust in P2P computing [35] proposed some constructive solutions to address this issue; however, from an architectural perspective, such anarchic management model inherent in P2P architectural style is regarded the most threatening. Visibility in such decentralized autonomous network is rather opaque, which undoubtedly incurs a series of monitoring and management issues. Generally, we believe P2P styles of service execution are 'making things more complex rather than simple'. Shifting all the tasks used to burden the broker to each peer will make them too complex to be deployed and evolved at large volume; in addition, distributed data management becomes extremely difficult.

Different styles have their own advantages and disadvantages, improving one property might lead to the reduction of another. Architectural design is indeed a trade-off among a number of architecture styles constrained by the particular circumstances.

VII. Conclusion and Future Work

We approached the challenges of web services from the architectural perspective. We introduced an architectural methodology used to measure and evaluate architecture styles of Internet-wide distributed web services. We then identified key architectural properties related to quality aspects of web services architecture. Next we summarized nine web services architectural styles in current literature. Finally, we evaluated these against the architectural properties identified. Detailed methods to evaluate these

styles are also given. For future work, we believe there are three issues that need to be addressed:

1. Based on this architectural evaluation, propose an improved architecture for distributed web services architecture with software prototype.
2. Evaluation metrics should be further formulated based on certain factors determined by business context.
3. The proposed architectural methodology need to be further studied to cater for Internet-wide distributed web services environment.

VIII. REFERENCE

[1] Ciganek, A.P., Haines, M.N. & Haseman, W., 2005, 'Challenges of adopting web services: Experiences from the financial industry', *Proceedings of the 38th Hawaii International Conference on System Sciences* – 2005.

[2] Perry, D.E. & Wolf, A.L., 1992, 'Foundations for the study of software architecture', *ACM SIGSOFT Software Engineering Notes*, vol. 17, no. 4, Oct. 1992, pp. 40–52.

[3] Fielding, R.T., 2000, 'Architectural styles and the design of network-based software architectures', *PhD Dissertations*, University of California, Irvine CA, USA.

[4] Bosch, J., 2000, *'Design and use of software architectures – Adopting and evolving a product-line approach'*, Addison-Wesley, ISBN: 0-201-67494-7.

[5] Birman, K., Renesse, R. & Vogels, W., 2004, 'Adding high availability and autonomic behaviour to web services', *Proceedings of the 26th International Conference on Software Engineering* (ICSE'04).

[6] Austin, D., Barbir, A., Ferris, C. & Garg, S,. 2004, 'Web services architecture requirements', *W3C Working Group* Note 11 February 2004, Retrieved: 5 May, 2005, from http://www.w3.org/TR/wsa-reqs.

[7] MacKenzie, M. & Amand, S., 2004, 'Electronic business service oriented architecture', *OASIA Working Draft 047*, 20 August 2004, retrieved: 30 May 2005, from http://www.oasis-open.org/committees/ebsoa.

[8] Wong, H.C. & Sycara, K., 2000, 'A taxonomy of middle-agents for the internet', *Proceedings of the Fourth International Conference on MultiAgent Systems*, July, 2000, pp. 465 - 466.

[9] Wang, X., Yue, K., Huang, J.Z. & Zhou, A., 2004, 'Service selection in dynamic demand-driven web services', *Proceedings of the IEEE International Conference on Web Services* (ICWS04).

[10] Yu, T. & Lin, K., 2004, 'The design of QoS broker algorithms for QoS-capable web services', *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service* (EEE'04).

[11] Degwekar, S., Su, S.Y.W. & Lam, H., 2004, 'Constraint specification and processing in web services publication and discovery', *Proceedings of the IEEE International Conference on Web Services* (ICWS' 04).

[12] Papazoglou, M.P., Kr¨amer, B.J. & Yang, J., 2003, *'Leveraging web-services and peer-to-peer networks*, Springer-Verlag Berlin Heidelberg 2003.

[13] Pilioura, T., Kapos, G. & Tsalgatidou, A., 2004, 'PYRAMID-S: A scalable infrastructure for semantic web service publication and discovery', *Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications* (RIDE'04).

[14] Sivashanmugam, K., Verma, K. and Sheth, A., 2004, 'Discovery of web services in a federated registry environment', *Proceedings of the 16th International Conference on Software Engineering & Knowledge Engineering* (SEKE2004): Workshop on Ontology in Action, Banff, Canada, June 21-24, 2004, pp. 490-493.

[15] Paolucci, M., Soudry J., Srinivasan, N. & Sycara, K., 2004, 'A broker for OWL-S web services', *Proceedings of First International Web Services Symposium*, 22- 24 March, 2004.

[16] Malek, H.B., 2005, 'Service-orientation: A brief introduction', *OASIS SOA Reference Model TC Public Documents*, Retrieved: May 29th 2005, from http://www.oasis-open.org/committees/download.php/12834/Service-Orientation.pdf

[17] Fuchs, M., 2004, 'Adapting web services in a heterogeneous environment', *Proceedings of the IEEE International Conference on Web Services* (ICWS'04).

[18] Piers, P., Benevides, M. & Mattoso, M., 2003, 'Mediating heterogeneous web services', *Proceedings of the 2003 Symposium on Applications and the Internet* (SAINT' 03).

[19] Milojicic, D.S., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S. & Xu, Z., 2003, *'Peer-to-peer computing'*, Hewlett-Packard Company Technology Report.

[20] Ayyasamy, S., Patel, C. & Lee, Y., 2003, *'Semantic web services and DHT-based peer-to-peer networks: A new symbiotic relationship'*, Position Paper, School of Interdisciplinary Computing and Engineering University of Missouri – Kansas City.

[21] Wang, Q., Yuan, Y., Zhou, J. & Zhou, A., 2003, *'Peer-Serv: A framework of web services in peer-to-peer environment'*, WAIM 2003, LNCS 2762, pp. 298 – 305, 2003, Springer-Verlag Berlin Heidelberg 2003.

[22] Prasad, V. & Lee, Y., 2003, 'A scalable infrastructure for peer-to-peer networks using web service registries and intelligent peer locators', *Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, p. 216.

[23] Paolucci, M., Sycara, K., Nishimura, T. & Srinivasan, N., 2003, 'Using DAML-S for P2P discovery', *Proceedings of International Conference on Web Services*, ISWS, 2003

[24] Emekci, F., Sahin, O., Agrawal, D. & Abbadi, A., 2004, 'A peer-to-peer framework for web service discovery with ranking', *Proceedings of the IEEE International Conference on Web Services* (ICWS'04), 0-7695-2167-3/04 IEEE.

[25] Schmidt, C. & Parashar, M., 2004, 'A peer-to-peer approach to web service discovery', *World Wide Web*, vol. 7, no. 2, pp. 211-229.

[26] Banaei-Kashani, F., Chen, C-C. & Shahabi, C., 2004, 'WSPDS: Web services peer-to-peer discovery service', *Proceedings of International Symposium on Web Services and Applications* (ISWS'04), Las Vegas, Nevada, USA, June 2004, pp. 733-743.

[27] Thaden, U., Siberski, W. & Nejdl, W., 2003, *'A semantic web based peer-to-peer service registry network'*, Technical Report, Learning Lab Lower Saxony, University of Hanover, Germany.

[28] Chafle, G., Chandra, S. & Mann, V., 2004, 'Decentralized orchestration of composite web services', *Proceedings of World Wide Web*, 17-22 May, 2004, New York, USA.

[29] Schuler, C., Weber, R., Schuldt, H. & Schek, H., 2004, 'Scalable peer–to–peer process management — The OSIRIS approach', *Proceedings of the IEEE International Conference on Web Services* (ICWS'04).

[30] Lakhal, N.B., Kobayashi, T. & Yokota, H., 2004, 'THROWS: an architecture for highly available distributed execution of web services compositions', *Proceedings of the 14th International Workshop on Research Issues on Data Engineering: Web Services for e-Commerce and e-Government Applications.*

[31] Muth, P., Wodtke, D., Weissenfels, J. & Kotz, D.A., 1998, 'From centralized workflow specification to distributed workflow execution', *Journal of Intelligent Information Systems* (JIIS), vol. 10, no. 2.

[32] Benatallah, B., Dumas, M., Sheng, Q. & Ngu, A., 2002, 'Declarative composition and peer-to-peer provisioning of dynamic web services', *Proceedings of the 18th International Conference on Data Engineering* (ICDE'02).

[33] Nanda, M.G., Chandra, S. & Sarkar, V., 2004, 'Decentralizing execution of composite web services', *Proceedings of the 19th annual ACM SIGPLAN Conference on Object-oriented programming, systems, languages, and applications*, vol. 39, iss. 10, October 2004, Vancouver, British Columbia, Canada.

[34] Haller, K. & Schuldt, H., 2003, 'Consistent process execution in peer-to-peer information systems', *Proceedings of the 15th Conference on Advanced Information Systems Engineering* (CAiSE), Klagenfurt/Velden, Austria, 2003.

[35] Hussain, F.K., Chang, E. & Dillon, T., 2004, 'Trustworthiness and CCCI metrics in P2P communication', *International Journal of Computer Systems Science & Engineering*, vol. 19, no. 3/4.