Quality of Business Driven Service Composition and Utility Computing

Vijay Machiraju, Jerry Rolia, Aad van Moorsel
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-2002-66
March 15th , 2002*

E-mail: {vijaym, jar, aad} @hpl.hp.com

quality of
business,
management,
web services,
data center,
utility
computing

IT systems are more critical to the success of business than ever before. As a best practice, organizations are documenting their business processes and corresponding dependencies on supporting IT systems. As a result they are better able to conduct cost-benefit analysis regarding business objectives and IT investments. In this paper we consider two recent technological advances - service composition and utility computing, and describe how they impact best practice. We introduce the notion of quality of business (QoBiz) and present a methodology that uses QoBiz to continuously direct service composition and utility computing. To facilitate the execution of such continuous QoBiz-driven adaptation, we introduce a distributed architecture of control systems that manages service level agreements. Last, we consider requirements on modeling technologies in support of these advances in QoBiz-driven service-centric computing.

Quality of Business Driven Service Composition and Utility Computing

Vijay Machiraju, Jerry Rolia, Aad van Moorsel

Hewlett-Packard Laboratories
Palo Alto, CA, U.S.A.
{vijaym,jar,aad}@hpl.hp.com

Abstract. IT systems are more critical to the success of business than ever before. As a best practice, organizations are documenting their business processes and corresponding dependencies on supporting IT systems. As a result they are better able to conduct cost-benefit analysis regarding business objectives and IT investments. In this paper we consider two recent technological advances - service composition and utility computing, and describe how they impact best practice. We introduce the notion of quality of business (QoBiz) and present a methodology that uses QoBiz to continuously direct service composition and utility computing. To facilitate the execution of such continuous QoBiz-driven adaptation, we introduce a distributed architecture of control systems that manages service level agreements. Last, we consider requirements on modeling technologies in support of these advances in QoBiz-driven service-centric computing.

1 Introduction

Businesses rely increasingly on Information Technology (IT) to achieve business objectives. It is critical for businesses to have a process in place that relates their business objectives to the IT systems that support them. In this way a business is better able to make investment and technology decisions that support the business as a whole.

We define *agility* as the capability of a business to adapt and retire services quickly to better meet business objectives. Existing services may experience changes in load and/or significant changes in functionality. New services can place further stresses on IT systems. They can be expected to: support large numbers of users -- possibly mobile; be complex with many interdependencies; have bursty workloads that are difficult to predict; and may be such that rapid global deployment is necessary.

In this paper we look at a current best practice for relating business objectives to IT systems. We then consider the impact of new technologies, namely service composition and utility computing as enablers of agility. The paper approaches the problem from the perspective of those responsible for managing the relationships between business objectives and IT infrastructure.

We define a *business* as an organization that marks the boundary for administrative domain of control. A business has revenue and profit objectives and a budget for how much it wants to spend to meet those objectives. It typically relies on many *services* – some of which are revenue-generating customer services (or services for its customers), while others are internal services for its employees. Services themselves are realized by IT departments within a business through *application systems* deployed on *physical resources* (e.g., web servers running on a server farm or databases installed on storage devices).

Business processes define how various services will be put to use in meeting business objectives. A business process is defined as a sequence or workflow of *activities*, where the execution of each activity takes the process a step closer to completion. Each of the activities in a business process is realized by a service or by a portion of a service. Not all activities of a business process need to be implemented by in-house IT departments. Some can be outsourced as *remote services* offered by other businesses.
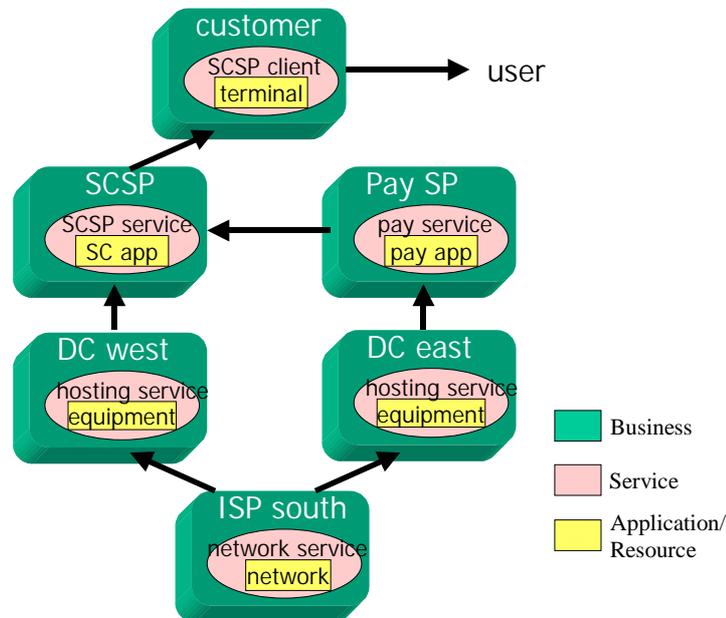
Fig. 1: An example showing businesses, services, and resources.

Example: As an example of these abstractions, consider a business that is interested in providing a supply-chain management service (Fig. 1). We name this business SCSP (supply-chain service provider). Businesses that decide to outsource their supply chain are SCSP's customers. SCSP offers value in that it provides well-defined business processes for its customers to follow and IT systems that implement these processes. In order to implement its business processes, SCSP uses various services – some of which are built by assembling application components, while others are outsourced to other businesses. In our example, SCSP uses the payment service offered

by another business named Pay SP (payment service provider). Further, SCSP decides to deploy all its application components in a data center (DC West) to reduce the cost of maintaining the physical resources by itself. Similarly, Pay SP hosts its payment application on a different data center (DC East). The data centers in turn rely on an Internet service provider (ISP South) for connectivity via the Internet.

This example consists of many businesses, each with its own objectives and business processes for realizing those objectives. Sometimes, they rely on their own IT departments to implement and manage their processes, and sometimes they rely on each other through outsourcing.

A best practice for business process/IT system integration can be expressed as follows. It is based on systems re-engineering exercises [1].

1. Define business objectives including revenue and profit goals
2. Develop and verify documentation for existing business processes
3. Correlate IT systems with the business processes
4. Relate business objectives to business processes; identify the performance criteria that must be met, the monetary budgets for implementing processes, and the payment methods that shall be used
5. New IT initiatives come after the first 4 steps, and must better support business objectives

Business processes are grouped, with management teams for each group ranking ITinitiatives for the group so that the business as a whole can make most effective use of its resources.Though the practice is defined at a high level of abstraction it illustrates challenges that face business and IT decision makers. Often IT initiatives are proposed that benefit only one part of a business. At other times enhancements are suggested for business processes yet it isn't clear which IT systems are affected and how changes would affect other business processes. Few if any participants in such a practice have full knowledge of a business's processes and infrastructure. Though there are many challenges that face organizations wishing to implement such an approach the advantages are indisputable. It provides a systematic approach for deciding which IT investments and divestments offer maximum strategic and financial benefit for the organization as a whole. We refer to this as a *Quality of Business (QoBiz) driven practice*.

In this paper we consider two recent technologies – service composition and utility computing, how they enable agility, and their impact on the above best practice. The technologies offer the potential for adaptive control over services and IT systems used to achieve business objectives. They enable a dynamic environment that can help to realize a more agile business but present challenges regarding how budgets and other service level attributes should be managed.

The rest of this paper is organized as follows: Section 2 describes service composition technologies and their impact on businesses and best practice. Section 3 introduces the notion of utility computing and explains its impact. Section 4 defines the essential infrastructure components and an extended best practice to support the dynamic adaptation of IT systems to realize more agile businesses. Predictive modeling

techniques needed to achieve the goals are also discussed. A conclusion is given in Section 5.

2 Service Composition

We define *service composition* as one service relying on another service when serving its customers; in effect treating the other service as a component. In the SCSP example, SCSP's customers treat SCSP as a component. So their use of SCSP is an example of service composition. SCSP in turn composes services of other service providers - for e.g., payment service for authorizing customers' payments.

Service composition in its basic form is similar to outsourcing, which has always been part of businesses. Traditional examples of outsourcing include Internet network services and hosting services at remote data centers. But, with the emergence of *e-services* [2], service composition takes a whole new dimension. E-services are applications that communicate in loosely coupled ways. Requests to and replies from e-services are typically framed as XML (extended markup language) messages [3] that are transported using Internet protocols. XML and Internet are together enabling across enterprise boundaries what Enterprise Application Integration (EAI) platforms have enabled so far within a single enterprise [4].

In addition to basic technologies such as XML, a plethora of standards and other technologies are making e-service composition *dynamic* and *automated*. Dynamic service composition is the ability to discover, negotiate, and use remote services on the fly. This means that in the midst of its operation, a service could drop a component service and bind with another one, without even having prior knowledge of the newly bound service. Automated service composition means that such composition could be accomplished through software without requiring human intervention. The standards and technologies that facilitate dynamic and automated service composition include:

Service descriptions and repositories for storing these descriptions: One of the first requirements for dynamic service composition is that every service should be able to describe its capabilities in a format that is understood by everyone else. Further, these descriptions should be stored in a well-known repository making it possible for clients to discover them. Web services description language or WSDL [5] is an emerging standard for describing web services; Universal description, discovery, and integration or UDDI [6] is a repository for discovering e-services.

Domain ontologies and semantics: Once a service is discovered, the client and the service must speak the same language. Automated service composition requires that clients and services discover each other's language (or ontology) and conduct translations to bridge gaps. There are two possible ways to do this – the first is to define and standardize ontologies for each domain. All the services within a domain must communicate using its ontology. The second alternative is to enrich the exchanged messages with semantics [7]. RDF and other document formats are the subjects of research for this purpose.

Negotiation and service level agreements: The third requirement for automated service composition is to build into services the ability to negotiate and reach an agreement on the level of service that will be offered. Automatic negotiation strategies [8] and modeling service level agreements (SLAs) [9, 10] are examples of research in this area.
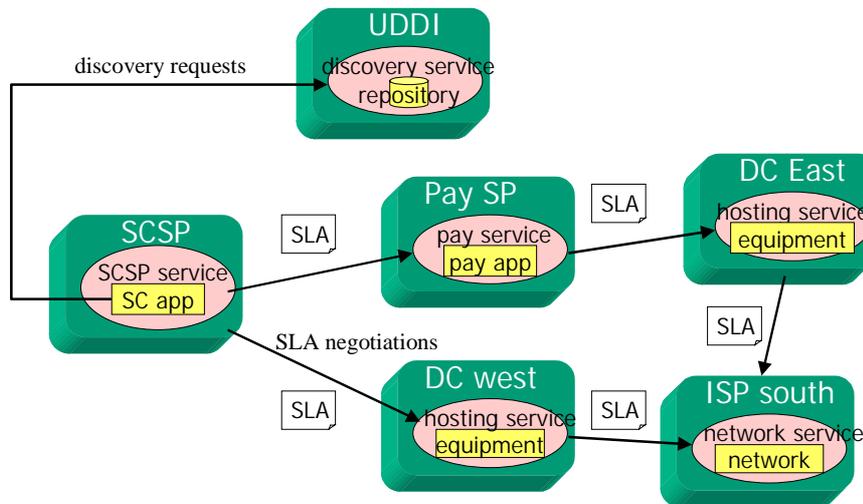
Fig. 2: Service composition technologies.

A conceptual architecture for how these technologies come together to enable service composition is shown in Fig. 2. With dynamic service composition, every service that relies on other component services must have both the ability to select or switch between component services, to select or switch between service level agreement ontology, and the ability to renegotiate and change service level agreement attributes with its component services on the fly. This can be quite powerful particularly, when the service has requirements that change over time.

To illustrate some of these benefits, the following scenario is an enhancement of the SCSP example described in section 1 with the addition of dynamic service composition.

Example – SCSP with service composition: SCSP requires a payment e-service. SCSP has a service level agreement (SLA) with Pay SP. The SLA distributes responsibilities and helps to identify causes of problems thereby allowing cleaner service composition. In the process of managing SLAs and their attributes, SCSP and Pay SP can exploit the power of service composition in the following ways:

1. On detecting a few SLA violations, warnings are issued by SCSP to Pay SP and penalties as stated in the SLA are collected. On repeated SLA violations, SCSP's business processes automatically trigger a lookup in UDDI for other potential providers offering the same type of service and that supports an acceptable SLA ontology. When an appropriate service provider willing to offer desired level of service is found, the processes automatically bind with the new service provider.
2. In another scenario, the SLAs with a component service are not violated. But, in order to keep up with the demand of its customers, SCSP requires a higher level of service from the payment service provider. In this case, SCSP's e-service negotiates a new SLA (i.e. different attributes, perhaps at an increased cost) with Pay SP.
3. Pay SP requires some short term overflow compute capacity for one of its applications. It makes use of UDDI mechanisms to locate a data center that has available resources. It then uses a resource negotiation ontology with the data center to negotiate configuration requirements, price, deployment, and execution of the application. SCSP is unaware of Pay SP's internal operations.

The level of automation as described in the above examples is far from reality. Today such scenarios still require significant human intervention. Examples of tasks that still need IT professionals are monitoring customer SLAs, monitoring component service SLAs, problem identification, renegotiation of SLAs and/or attributes when required, and the discovery of new services. To simplify the task of humans in the immediate future, and to enable further automation in the future, the following are suggested:

Evolve business processes to services: Business processes and activities should evolve towards e-services with interfaces, ontologies, and semantics that emerge within their particular domains. The e-services may be supported in-house or outsourced as is appropriate.

Rely on open e-service architectures: The integration between services should be done via open technologies and standards. This allows for potential composition of remote services in the future. E-service architectures rely on technologies such as XML, UDDI, and WSDL.

Services should be built to support service level management: Services should be built with service level management in mind. Monitoring SLAs, identifying problems, and paying penalties requires a rich management infrastructure that requires processes and services to be instrumented, metrics to be defined, measurements to be collected, and the analysis of that data.

3   Utility Computing

Section 2 described service composition as an outsourcing of application, networking, or hosting services. Today, hosting services typically offer resources via static agreements with resources dedicated to individual customers. A utility-based approach presents a paradigm where shared infrastructure can be provided on demand to multi-

ple customers. In this section we describe utility computing. Similar advances are being realized in the metro and wide area networking areas as network capacity on demand but are beyond the scope of this paper.

Utility computing treats a data center's compute, storage, and networking components as shared resources. The data center may belong to the business or may act as an infrastructure provider to otherwise independent businesses. Two models for utility computing are emerging. One is a *shared utility model* where many services, possibly executing on behalf of different businesses, execute on the same compute servers at the same time [19][20][21]. The other is a partitioned *utility model* where compute servers are allocated to only one service at a time. The shared model provides the greatest opportunity for resource sharing but has more severe implications for security and Quality of Service.
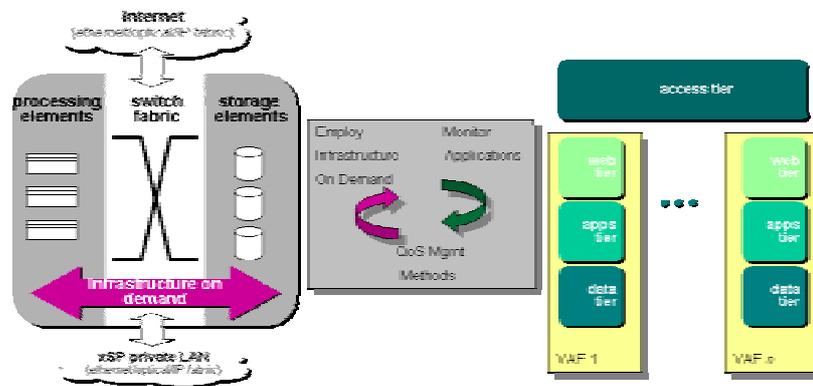

Fig. 3: A programmable partitionable data center that can provision infrastructure on demand.

In the partitioned utility model resources are physically wired once and programmatically partitioned across hosted applications [11,23,22]. In general, any resource can be allocated to any application at any given time but to only one application at a time. Resource allocation is managed programmatically by the data center to ensure correct secure atomic allocation. Goals, policies, and application control systems can cause events that initiate the addition or removal of resources from applications. In this way hosted applications receive *infrastructure on demand*.

In non-utility data center environments, resources are often informally cited as being busy 15-30% of the time. Sharing offers the possibility of increasing resource utilization, the number of hosted applications per cubic meter, reducing power requirements per application [19], and minimizing the potential impact of incorrectly estimating an application's workload intensity or mix [18].

Fig. 3 illustrates a programmable partitionable data center that realizes utility computing. To manage scalability within large data centers, we partition the physical infrastructure of the data center as a set of service cores. Each service core has on the order of 1000 compute nodes, layer 2 networking, and fiber channel based virtual storage subsystems. Service cores within and between data centers are interconnected

using layer 3 networking and virtual private network services. Service cores can be aggregated to achieve vast computing and storage capabilities.

Within each service core, data center management services programmatically create virtual application environments (VAE) on demand [11,23]. A VAE is the infrastructure for one hosted service (i.e. application). It consists of computation, networking, and storage resources. Resource types and connectivity are consistent with the application's configuration requirements. It may have explicit layers of servers (shown as tiers in Fig. 3) and many local area networks. Network and storage fabrics are configured to make a portion of the data center's resources appear to the application as a dedicated environment. The application's tiers may support clusters of servers, for example Web or application servers and appliances such as firewalls and load balancers. Such servers can be programmatically added and removed from networks as required by changes in workload intensity and mix.

Now we describe several technologies and design issues that help to realize the programmable data center. Data center management services create VAEs in a manner that isolates them from one another. This is done, for example, by exploiting technologies such as virtual local area networks (VLANs) [12], storage area networks (SANs) [13] and disk arrays [14]. From a security perspective applications are fully isolated from one another. This enables support for applications of competing businesses within a single utility computing environment.

Communication network resources within the data center are over-provisioned using cost-effective level 2 switching technologies to reduce the performance interactions between these environments. Similarly, special attention is given to storage networks to ensure that virtual disks can be accessed without significant performance interaction. Intelligent provisioning mechanisms [15] have been proposed that allocate resources to minimize performance interactions between otherwise independent VAEs.

In general, utility computing offers a way of outsourcing support for resource infrastructure. This has numerous advantages that include: reducing the burden of systems support on local IT staff, reducing risks associated with capacity planning (either underestimating or overestimating), reducing overall costs associated with IT systems, and enabling the quick deployment of systems on a global scale. Disadvantages include: an increased reliance on networking and a lesser ability to tailor the configuration of servers to specific applications.

Example – SCSP with utility computing: We now consider several examples of how infrastructure on demand can benefit a business such as the SCSP.

1. The SCSP rolls out a new promotion, it expects thousands of new customers to make use of its service. Without a utility computing environment, the SCSP would have to accept substantial risk and overprovision its infrastructure in anticipation of the new load. A utility data center can more cost-effectively over-provision for the support of many hosted applications that do not all have peak resource requirements at the same time.
2. SCSP customers cause an unexpected surge in load. The SCSP can request and acquire additional resources from the data center within minutes. Similarly, supply

chain activity may be very low on the weekends. The SCSP may release some of its resources for use by others. In doing so it reduces its own IT costs.
3. The importance of assorted SCSP applications may vary based on time of day or month or even in response to an unexpected event. Instead of provisioning each application for peak loads, the amount of infrastructure associated with each application can change with business needs.

We note that the data center itself is also a business with a goal to maximize profits while satisfying its customers' service level agreements. It must ensure that adequate resources are available to meet the above needs yet will also aim to maximize resource utilization. It can be expected to charge for numbers of servers and storage actually used as well as for bounds on peak quantities of resources that may be used. The ability of data centers to satisfy the needs of customers while minimizing costs will determine the success of this approach to computing.

We now consider the impact of utility computing on the QoBiz best practice.

Decide which business processes and activities would best benefit from utility computing: In general those processes and activities with particularly bursty workloads, or those that must be rapidly deployed on a global scale are good candidates. For others businesses must characterize the costs and strategic benefits of outsourcing resource infrastructure for each of its processes and activities. The characterization of IT costs is always a challenge. It must include human and technology costs. Human costs include investments in skill-sets, ongoing training, and actual day-to-day system's support. Technology costs must include a financial characterization of risk associated with under or over-provisioning. Similarly potential losses in revenue due to inadequate service from a utility data center need to be factored into the comparison. A decision must be made whether shared or partitionable utility computing is most appropriate for the problem at hand. Benefits arise from outsourcing when a business is able to focus more on its core competencies and/or when the outsourcing enables greater agility.

Establish objectives for utility computing: A business must specify QoS objectives, goals and priorities, and monetary budget requirements for its utility computing needs. A business's goals may change based on time of day or month or even due to unexpected events so specifications must be robust. Utility providers must provide requested resources with some agreed probability.

Business	An organization that executes business processes. The business marks the boundaries of an administrator's *domain of responsibility* over the execution of the business processes. If the business *owns* the resources and application system, then these are within the IT manager's *domain of control*.
Service provider	A specific business that provides services for use by its customers.
Customer	A specific business, which uses the services of a service provider. In a chain of service providers, the *end customer* exclusively uses services (that is, it is a business that is not a service provider itself). We also use *end user* if the end customer is an individual.
Business process	A sequence of one or more workflow *activities* that achieve some intended purpose on behalf of the business.
Activity	Logical representation of one or more application services within the workflow. Is realized by one or more application services.
Resources	Physical or logical components that must be provisioned to implement e-services.
Application service or e-service	Implements an activity for a customer. Exposes a business process for a service provider.
Infrastructure service	Provides resources to other services. These may be statically allocated resources or infrastructure on demand.
SLA	An agreement that includes a specification of time, budget (prices, penalties), qos metrics, workload, and priority. See, e.g., [10].
Control system	A system that continuously receives monitoring feedback on SLAs and dynamically adapts SLA attributes to achieve SLA objectives.
Workload and customer fluctuations	For a service provider, customer fluctuations refer to the fluctuating number of customers over time. Workload fluctuations refer to the changing demands over time of one particular customer.
Quality of Business	In general terms, a QoBiz metric is any metric expressed in monetary units. We also use a more specific interpretation, namely the monetary value corresponding to the quality of the delivered service, expressed through linking the QoS metrics in the SLA with monetary value. See also [16,17].

Table 1. Terminology

4 QoBiz Driven Service Composition and Utility Computing: Practice and Models

In previous sections we discussed service composition and utility computing. In general a business should exploit opportunities to treat its business processes as services. When advantageous, these should be outsourced as e-services. If that is not desirable or possible, they should be implemented in house but exploit internally or externally hosted utility computing.

In this section we consider the impact of service composition and utility computing on the best practice presented in the Introduction and detail the various entities and attributes to be considered in QoBiz-driven use of service providers.

QoBiz Entities and Attributes of Service Composition and Utility Computing

We already introduced some terminology in the Introduction; these and other terms are defined more precisely in Table 1. Also, Fig. 1 in the Introduction illustrated parties that interact with the SCSP business. In this section, we drill down and focus on the core system components that enable QoBiz-driven usage of outsourcing. To this end, we present a UML class diagram in Fig. 4 that represents the various entities and attributes that play a role in QoBiz-driven service-centric computing.

The UML diagram in Fig. 4 deals with two main aspects: (1) core entities that constitute outsourced computing, and (2) control systems. The core entities are in the center of Fig. 4, while the control systems in Fig. 4 are placed around these core elements. In addition, (QoBiz-oriented) SLA attributes play a role in various classes, since SLAs determine the 'rules' of partnerships and motivate choices on how to control.

As we mentioned in the Introduction, *businesses* run one or more *business processes*, which contain various *activities*. These activities are implemented as *application services*. If an application service is outsourced, then from its service provider's point of view the application service exposes a business process as an e-service.

Application services exploit and offer application functionality via the *uses* and *usedBy* roles, respectively. For example, an HTTPS request may be issued by SCSP to Pay SP to cause a payment on behalf of one of SCSP's customers. The *implementedBy* relationships capture service level requirements that govern these interactions. For example 95% of such requests must complete within 4 seconds. As discussed below, service level negotiations are the responsibility of control systems.

An application service can be deployed on its own *resources* or may rely on an *infrastructure service* provider for resources. An infrastructure service provider may offer static infrastructure as described in Section 2 or utility infrastructure as described in Section 3. Note that a shared utility's application service is its ability to offer an execution environment to its customers. A shared utility infrastructure provider may rely on a partitionable utility infrastructure provider for resources, i.e. for infrastructure on demand. *Service level requirements* for resources are captured by the *provisionedBy* relationship.

Control systems take as input objectives and monitoring feedback for *SLAs*. As output, a control system divides its aggregate budget and SLA attributes across those entities affecting its domain of control. This division must be adaptive in response to changes in objectives and feedback including defined events. A business's control systems operate together to enable the business to make best use of infrastructure at over time.

QoBiz controllers operate over long time scales, deciding SLAs (including budgets) for business processes as required to best satisfy the objectives of the business as a whole. As an example a QoBiz controller may have as its goal the long term maximization of stable profit for the business. Based on current knowledge of customers, their agreed upon workloads, QoS expectations, revenue and resulting profits the control system may decide on certain SLA attributes for its business processes.
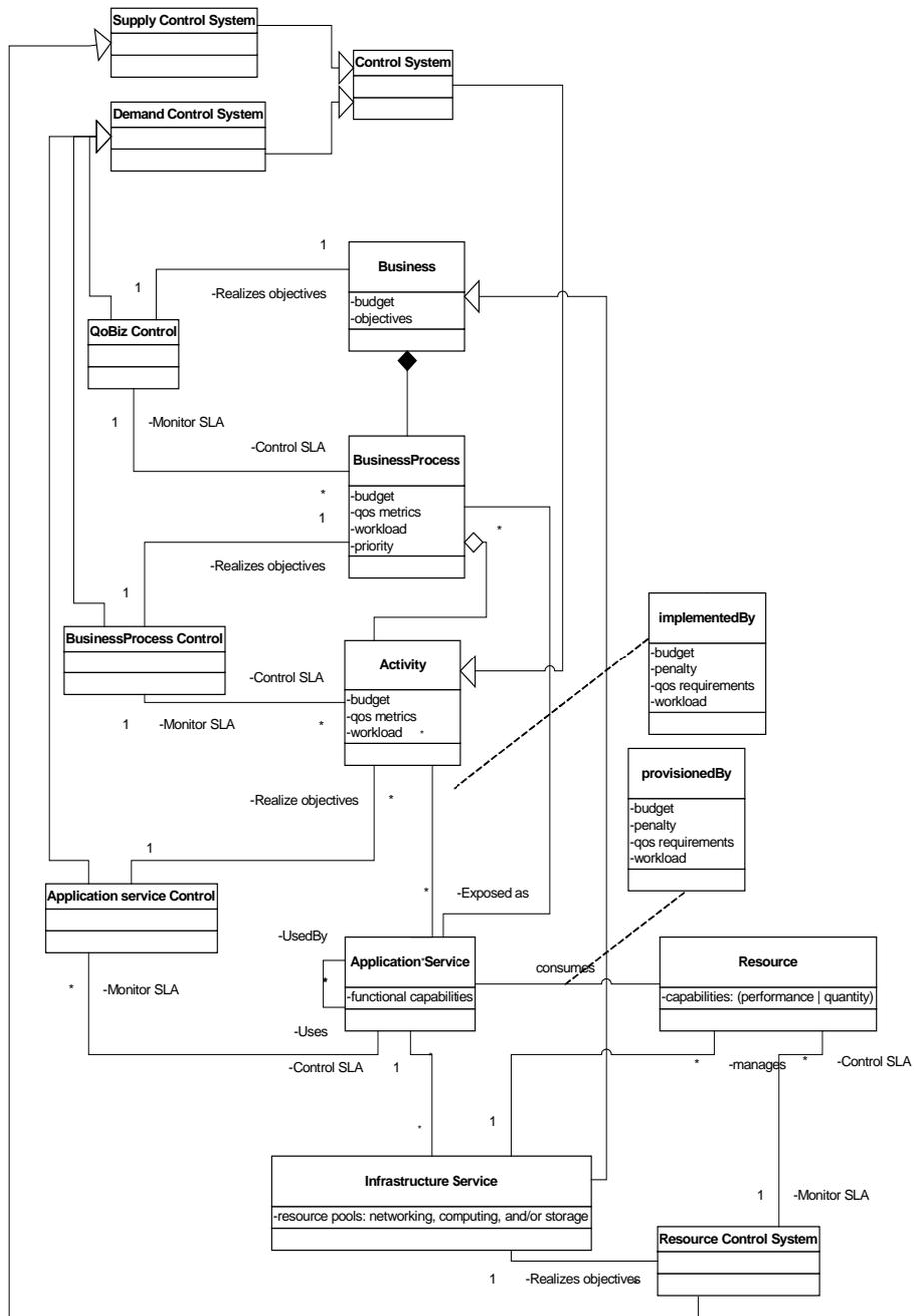
Fig. 4: Class diagram for QoBiz elements in service composition and utility computing

Business process controllers operate over medium time scales to control the attributes within the implementedBy relationships to best satisfy the needs of each process. These relationships may be re-negotiated due to changes in workload conditions or service level violations without affecting the QoBiz control system. If for example it is not possible to achieve required service levels within budget an event must be issued to the QoBiz controller to reevaluate the budget or adjust the service level expectation.

Application service controllers operate over shorter time scales, for example on the order of minutes [18] or 10's of minutes. They help to ensure each application service uses only the resources it needs to satisfy its service level requirements. They govern the provisionedBy relationships.

Control systems are also activities that are realized as application services. They interact with other control systems including those of other businesses via the uses and usedBy roles. Business process controllers interact to negotiate pricing and service level attributes. Application control systems interact with resource control systems to acquire and release resources. Control systems propagate monitoring information to their parent controllers to inform on the current state of the system.

In Fig. 4, the demand side control systems act to prioritize and limit demand on resources. The supply side control systems act to intelligently provision resources to best satisfy infrastructure provider goals. ImplementedBy and provisionedBy SLA attributes are ultimately determined by business objectives, current and anticipated workloads, and the partitioning of SLA attributes by control systems.

To restrict Fig. 4 to the essential abstraction we do not depict events that trigger control mechanisms. Events that warrant such action can happen at both customer and service provider side and are of various kinds. For instance, a price increase or service degradation may motivate a change for a business' service composition. This could entail changes to some relationships between activities and application services. Similarly an increased workload may cause a re-negotiation of SLA attributes with a utility computing provider so that more resources can be made available. At the same time, application and infrastructure service providers must consider issues that maximize their own profits according to their own QoBiz models.

Example: QoBiz Control for SCSP. SCSP considers an IT solution that deploys its service at a partitioned utility data center. SCSP has prepared an instantiation of the model in Fig. 4, relating its customers with SCSP business processes and IT infrastructure hosted at the UDC. The model reflects the SLA agreements with customers along with the expected revenue from each customer. It also reflects the priorities of business processes for situations of diminished resource availability. To fully exploit the potential of utility computing, SCSP analysts specify a long-term expected demand profile for UDC resources. The profile is based on the SCSP's expected number of business customers and their expected number of employees, and on traces that provide historical usage patterns. These projections are continuously updated based on actual behavior. Revenue per customer determines budget constraints for SCSP. Numbers of customers and usage profiles help bound the expected demand for the UDC. The SCSP has a control system for its application services that decides when additional resources are required from the UDC or when un-necessary resources can

be released [18]. The application control system always seeks approval from its business process control system prior to going beyond its IT budget. Similarly the business process control system always seeks approval form the SCSP QoBiz control system prior to going beyond its IT budget. If inadequate numbers of resources are available, the control systems guide the provisioning of resources to those business processes and activities (and hence their IT systems) with highest priority. The likelihood of receiving a resource when requested must be expressed as an attribute in the SCSP's SLA with the data center provider. The required SLA must be consistent with SCSP's SLAs with its own customers. Last, to negotiate a favorable SLA with its service provider, SCSP calculates the monetary consequences of various pricing and penalty options to choose the best one.

Practice for QoBiz Driven Service Composition and Utility Computing

Whereas the practice in the Introduction deals with 'what' (what business processes within your business to give priority and support), we now have to deal with issues of 'how' to implement and 'when' to adapt. 'How' must consider the strategic advantage of the new technologies. 'When' must consider dynamic adaptation of service provider relationships in response to changing events. Hence, to account for the dynamics of service composition and utility computing, the practice in the Introduction must be augmented with the following steps:

- o For each business process and activity, consider which IT solution best meets the strategic goals of the business; the options must include service composition and utility computing.
- o If service providers are used, define SLA attributes. These include bounds on budgets, measures of throughput, responsiveness and availability. Determine what events should trigger changes in attributes or providers.
- o If utility computing is used, define SLA attributes. These include bounds on budgets and numbers of resources to be requested when adding/releasing resources in response to fluctuations in workload and also the probability of acquiring resources when they are requested. Determine what events should trigger changes in attributes or providers.
- o If the business offers a service, establish criteria that should guide the selection of SLAs and values for SLA attributes it should support. Establish criteria that can be used to prioritize business processes and customers.

The results of these steps must then guide the development of the QoBiz model. The control systems themselves can then be "achieved" by operations staff with the support of predictive modeling tools or by autonomous control systems that also rely on such tools. Full automation is an ultimate goal for such control mechanisms. As a last step for the augmented best practice:

- o Periodically revisit the model of Fig. 4 to ensure it continues to achieve its goal of best satisfying the objectives for the business as a whole.

Modeling Techniques

Model-based assessment and optimization are critical for control systems that implement and benefit from service composition and utility computing. Understanding demand characteristics, linking service quality with business value, and determining the optimal setting of SLA prices and penalties are all essential. The following lists some of the key modeling technologies we anticipate will be used:

Statistical methods for workload demand characterization, based on the analysis of traces for trends and other regularities

Optimization methods, to help to find an operating point (within and across control systems) that offers best value for the business at any time. In particular, to minimize costs while dividing budgets and guaranteeing SLA, under fluctuating demands.

Queuing analysis may also be appropriate for relating the performance aspects of service level requirements to numbers of resources; this also provides feedback on expected costs

Markov and other analytical models may be appropriate for relating the availability aspects of service level requirements to resource requirements, also providing feedback on expected costs.

Economic models, for anticipating the impact of an open market on E-service and infrastructure services costs over time, and for negotiating prices and penalties for SLA violations.

5   Conclusions

Service composition and utility computing are two core technologies that can provide businesses with the agility to realize a world of more advanced services, where workloads are difficult to predict, and global deployment is desirable. The flexibility results in new IT solution paradigms, thus creating new challenges for businesses and their IT departments. Together they must decide if/when to use service providers, what service level agreements to accept, and how to exploit infrastructure-on-demand solutions in response to workload fluctuations.

In this paper, we propose a practice for IT managers to address these issues. It is based on a practical best practice process, augmented to deal with service composition and utility computing. The process is driven by business concerns ('QoBiz-driven') and assumes a key role for SLAs and associated control systems to manage future agile service infrastructures. Expressing QoBiz inherently relies on human judgment and experience, but will increasingly rely on model-based assessment and optimization. These models merge advancements in the field of economy (to set SLA prices and penalties), operations research (to minimize resource consumption), statistics (to characterize service demand fluctuations) and business (to compute cost of ownership and do business risk analysis).

It is our desire that the proposed practice forms a template that can be fine-tuned and specialized in the field, to help IT managers deal with decisions about the use of

outsourcing and utility computing options. In addition, tools and techniques for model-based assessment and optimization must continue to be developed to improve the quality of IT decisions.

References

1. Wrenn, J.: The IT Train That Could, Jan 15, 2002 Issue of CIO Magazine, Also at http://www.cio.com/archive/011502/peer.html.
2. Wooyoung, K., Graupner, S., Sahai, A., Lenkov, D., Chudasama, C., Whedbee, S., Luo, Y., Desai, B., Mullings, H., Wong, P., Web e-speak: Facilitating web-based e-services, IEEE Multimedia, Vol. 9, No. 1, Jan-Mar 2002, pp. 43-55.
3. Extensible Markup Language, http://www.w3.org/xml.
4. Sahai, A., Machiraju, V.: Enabling of the ubiquitous e-service vision on the Internet, e-Service Journal, Indiana University Press, Vol. 1, No. 1, 2001-02, pp. 5-19.
5. Web Services Description Language, http://www.w3.org/TR/wsdl.
6. Universal Description, Discovery, and Integration, http://www.uddi.org.
7. Berners-Lee, T., Hendler, J., Lassila, O., The semantic web, Scientific American, May 2001.
8. Rodrigez, J.M., Sallantin, J, A system for document telenegotiation (negotiation agents), COOP 98: 3rd International conference on design of cooperative systems, Cannes, France, May 26-29, pp. 61-66.
9. Long, T.P., Jong, W.B., Woon, H.J., Management of service level agreements for multi-media Internet service using a utility model, IEEE communications interactive magazine, Vol. 39, No.5, May 2001.
10. Forbath, T., Why and how of SLAs [service level agreements], Business Communications Review, Vol. 28, No. 2, Feb 1998.
11. Rolia, J., Singhal, S., Friedrich, R., Adaptive Internet Data Centers, SSGRR 2000, Computer and eBusiness Conference, L'Aquila, Italy, Jul 31-Aug 6, 2000.
12. IEEE 802.1q standard. Available at http://standards.ieee.org/.
13. Storage Networking Industry Association, http://www.snia.org.
14. Patterson, D.A., Gibson, G., and Katz, R.H., A Case for Redundant Arrays for Inexpensive Disks (RAID), In Proceedings of ACM SIGMOD Conference, Jun 1988, pp. 109-116.
15. Zhu, X., Singhal, S., Optimal Resource Assignment in Internet Data Centers, In Proceedings of the Ninth International Symposium on Modeling, Simulation, and Analysis of Computer and Telecommunication Systems (MASCOTs), Cincinnati Ohio, pp. 61-69, Aug 2001.
16. Van Moorsel, A., Metrics for the Internet Age: Quality of Experience and Quality of Business, HP Labs Technical Report HPL-2001-179 http://www.hpl.hp.com/techreports, July 2001, Also in Fifth International Workshop on Performability Modeling of Computer and Communication Systems, Arbeitsberichte des Instituts für Informatik, Universität Erlangen-Nürnberg, Germany, Band 34, Nummer 13, pp 26—31, September 2001.
17. Wolter, K and Van Moorsel, A., The Relationship between Quality of Service and Business Metrics: Monitoring, Notification and Optimization, HP Labs Technical Report, HPL-2001-96, http://www.hpl.hp.com/techreports, April 2001, Also in OpenView Forum 2001, New Orleans, Louisiana, USA, June 11-14, 2001.
18. Ranjan S., Rolia J., H. Fu, and E. Knightly, QoS-Driven Server Migration for Internet Data Centers, Submitted to IWQoS 2002 on Feb 15, 2002.

19. Chase J, et. al. Managing energy and server resources in hosting centers. In *Proceedings of the Eighteenth ACM symposium on Operating Systems Principles (SOSP)*, October 2001.
20. Ejasent. Utility computing white paper, November 2001, http://www.ejasent.com.
21. Sychron. Sychron Enterprise Manager, 2001. http://www.sychron.com.
22. K. Appleby et al. Oceano – SLA based management of a computing utility. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*, May 2001.
23. Hewlett-Packard. HP utility data center architecture, http://www.hp.com/solutions1/infrastructure/solutions/utilitydata/architecture.