Maintaining and Evolving Service Level Agreements: Motivation and Case Study

Mike Smit, Eleni Stroulia
Department of Computing Science
University of Alberta
Edmonton, Canada
Email: msmit,stroulia@cs.ualberta.ca

***Abstract*—Inter-organization service-oriented compositions are governed by Service Level Agreements (SLAs). While the software is maintained and evolved in response to changing business requirements or technology, the governing SLA and software configuration designed to meet this SLA do not always change in step. SLAs are negotiated and may have legal standing, which makes their maintenance expensive and time consuming. If an SLA is established that meets the requirements of the service consumer, changing the software without updating the configuration and/or the SLA may result in unmet requirements and reduced satisfaction. This paper begins by examining the business perspective on SLAs as a guide and motivation to maintaining SLAs. A simulation-driven approach to updating a configuration in step with ongoing maintenance efforts is presented. The approach is illustrated using a case study, where we examine how a configuration is updated in two maintenance scenarios: one where the application is migrated to a cloud computing platform, and another where it is migrated to use a distributed computing platform. Our approach suggests configurations that maintain existing performance levels.**

I. Introduction

Service-oriented applications are being deployed with increasing regularity. Once deployed, these applications must be maintained and must evolve to meet changing business goals. Deploying new services, adding new features, or creating new compositions are common maintenance and evolution tasks. However, major changes (or a series of minor changes) to the software can have unintended consequences on the non-functional qualities of the system behavior. Often neglected in the software-evolution process is the evolution of the software configuration and / or governance policies. This is particularly important when the software is distributed, heterogenous, and composed, as in SOA. Additional computing power may be required, or increased infrastructure, or storage capacity. New features might require revisiting security or privacy policies. New compositions or partnerships might require increased capacity.

Service-oriented applications are often governed by service level agreements (SLAs), particularly when more than one entity is involved. These agreements set expectations for various non-functional properties, such as performance, security, support, and reliability. As the functional properties of the service-oriented application evolve, and as the business goals evolve, the SLA can become less applicable, or become more difficult to meet, or its metrics can lose their value as indicators of quality. However, complete SLA re-negotiation can be time-consuming and expensive. When the changes affect the performance-related service level objectives, it may be possible to maintain the SLA, making small changes to some service levels or adjusting pricing. If the SLA cannot or need not be changed, the service level management policies – configuration policies designed to meet expected service levels – may need to be adjusted to maintain the same service levels.

One type of software evolution activity involves moving to different infrastructure, for instance cloud computing or distributed computing platforms. These platform shifts may impact the SLA substantially – reliability, latency, security, and privacy would all need to be revisited. The focus of this paper is on performance-related service levels, and in particular how to maintain the same (or better) service levels in a modified software system. Establishing performance levels that satisfy service consumers is a non-trivial task, as discussed in Section II, so maintaining the same cost-benefit ratio is desirable.

Academic literature has not yet addressed the problem of maintaining SLAs (to the best of our knowledge). This paper begins by discussing some of the challenges in establishing SLAs generally, then looks to business and marketing research to understand quality, value, and satisfaction. Collectively, these sections introduce some of the challenges in maintaining and evolving service level agreements while maintaing consumer satisfaction. We then describe our first approach to evolving Service Level Agreements (particularly performance-related agreements) and Service Level Management Policies in tandem with the software systems they govern. We measure the behavior of the software system both before and after platform shifts to identify changes in the performance of the software. The result is a before and after performance profile of the system. From this, we can indicate either how a configuration must change to meet the existing targets (and therefore identify an appropriate increase or decrease in cost), or how the service levels should change if the configuration must remain static. This analysis can also be done at the requirements elicitation / design time to identify how the SLA or cost structure would change if

the desired features are implemented, though with a higher expected error rate. Our simulation-driven approach to cost-benefit analysis [1] can be used to identify the cost-benefit trade-off in the existing satisfactory SLA, and look for the same trade-off based on the simulated results of a series of possible configurations.

We demonstrate this approach on a test bed application. This application has been evolved in two substantial ways: first, by migrating it to a cloud-computing infrastructure; second, by migrating to a distributed-computing infrastructure (Hadoop). We construct performance profiles for each scenario and compare them to the original. From these profiles we identify appropriate changes to service level management policies.

This paper is organized as follows. Section II describes Service Level Agreements in theory and the problems with these agreements in practice based on experience reports and surveys. Section III introduces concepts from marketing literature like value, quality, and sacrifice to help understand the mentality of a service consumer and how that impacts maintenance efforts. These sections together motivate future research into maintaining and evolving SLAs. Section IV describes our initial approach to maintaining SLAs in the face of changing software systems, illustrating the approach using a case study of two substantial software evolutions. Section **??** concludes the paper.

II. Service Level Agreements

Service Level Agreements (SLAs) can be used to govern interactions between service providers and service consumers. A service level agreement (SLA) is a contract between two parties promising a certain level of service. The level of service expected, whether technical (response time, CPU load) or non-technical (time to helpdesk problem resolution), is always measurable. There may be penalties for dropping below this level of service. The items in a service level agreement are called Service Level Objectives (SLOs).

One type of SLA is static and used for all customers, like those dictated by service providers (*e.g.* Amazon EC2). One can be a customer or not; this is the extent of "customization" available. Metrics are recorded by the consumer and must be reported to and validated by the provider in order for the penalty clause of a 10% refund to take effect. The other type are individual and specific based on the needs of the consumer and capabilities of the provider, in which case they are negotiated and include organization-specific guarantees. Typically these service providers have an SLA template which can be tailored to the needs of the customer. This tailoring can happen in a series of meetings prior to beginning the service experience; there are also methods for negotiating SLAs automatically at run-time (*e.g.* [2], [3], [4], [5]).

Services covered by SLAs typically have reporting requirements so both the provider and consumer can compare actual performance to target performance. These reports are typically high-level; for example, "green" means a service level objective was met, "red" means it was missed, and "yellow" means that there was trouble meeting the objective.

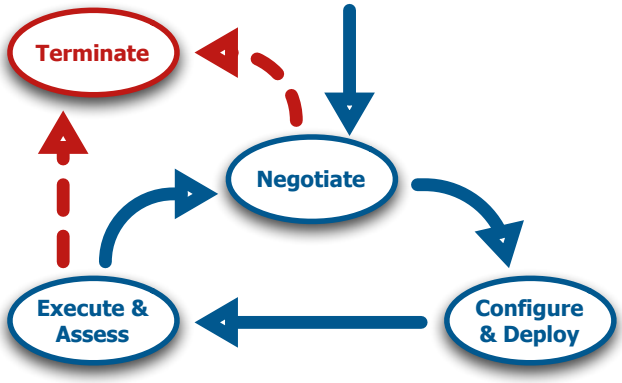

Figure 1. The lifecycle of an SLA: negotiated, implemented, assessed, then re-negotiated or terminated.

An SLA can be viewed as a three-way relationship between the producer, the consumer[1], and the service. The typical lifecycle of an SLA involves creation (negotiation), implementation and assessment, and eventually termination. This is the prevailing view on SLA lifecycles (*e.g.*, [6], [7]). Others recognize the need for the SLA to evolve and be re-negotiated over its lifetime (*e.g.*, [8], [9]). This latter view is adopted and elaborated in this paper: the SLA should evolve as business needs change, based on periodic review, and as the service implementation itself evolves (for example, to add new features).

This modified view of the SLA lifecycle is shown in Figure 1 by changing the linear process into a more realistic cycle. An initial SLA is negotiated to document the desired qualities of the service delivery. The service is configured and deployed to meet the SLA. Service interactions are monitored and evaluated to ensure that the terms of the SLA are met. The important part is the SLA is not static - it is periodically re-negotiated, or at least re-examined, to ensure it is still practical. Good points for re-examination are when the software is updated and when business needs change. The result of the evaluation or the re-negotiation may be to terminate the SLA.

An ideal SLA negotiation would involve two parties with equal knowledge and balanced power over the transaction (symmetric information and no duress), both with understanding of the technology and systems of the provider. The standards agreed upon would be based on a complete requirements elicitation from all of the stakeholders, trade-

[1]The consumer is not an individual; it is any entity that consumes a service. A service is not technically "consumed", but the term "user" is even more overloaded.

offs would be balanced optimally (assuming complete confidence of the consumer in his/her preferences regarding the relative qualities and costs), and the proposed service levels would be tested vigorously before entering production. Penalties would provide exactly the right incentive to ensure the targets are met if at all practical. The terms of the SLA would revisited regularly, and when change events occurred in either the provider or consumer. The reporting metrics would be well-understood by all parties, and unvarnished understandable detailed data would be available to both the provider and consumer.

This idyllic situation is not usually found in reality. The parties typically are not working with symmetric information, are not equal, and often the consumer has no complete understanding of the service system. A knowledge gap may be unavoidable, but the distance of this gap can be controlled by careful execution of the SLA lifecycle[2].

Although SLAs are the current standard practice, they do not always ensure customer satisfaction. Blomberg [8] conducted a study of interactions between consumers and providers. She identified five problems with how SLAs were used in those interactions, three of which are particularly relevant. A 2008 Forrester Research study [10] and a paper identifying SLA principles and best practices by Fitsilis [11] support her conclusions and offer additional problems, as follows:

- Information is difficult to understand. A Forrester study [10] concluded most SLAs are defined in technical terms not accessible to business users. Fitsilis [11] emphasizes the difficulty in mapping service levels from low-level measurable information to higher-level meaningful information, reporting that only technical specialists understand SLAs. He also emphasizes the importance of changing this: the first two best practices identified are "service level definitions should be business-based, meaningful to the users..." and "service level definitions should be easily defined and measurable". The author also recognizes that "the ultimate measure of service-level performance is customer perception and satisfaction", but that this end-user experience is difficult to measure.
- Satisfied SLA metrics don't mean satisfied customers. The SLA reports are useful in the first few months as the provider adjusts service delivery to ensure the metrics are "green". However, once the service delivery is satisfied, they appear to become less important, and may not indicate customer satisfaction. A 2008 Forrester study [10] refers to "misalignment", the distance between consumer expectations and what the provider is trying to achieve. In pointing out that SLAs often aren't actually agreements, the report claims "... at the end of a budget period, business managers can say 'it was not good enough for me to do my business', even if the IT service levels were met on a mathematical basis from the IT point of view." They advocate for improved understanding of higher-level requirements. Fitsilis [11] reports that requirements are often poorly specified and difficult to enforce.
- Information is hoarded. Providers are reluctant to provide unfettered, un-nuanced access to performance data. This sometimes extends from an honest desire to provide meaningful interpreted performance data that has been analyzed and summarized for the client. A reasonable theory, though one as yet unsupported by evidence, is since SLAs are legally binding contracts, the provider may hesitate to give clients access to information that could be used to enforce penalty clauses if not properly "interpreted".
- SLAs are not proactive enough. Clients remarked that they wanted more from their service providers: a proactive approach to help the client identify their needs and proactively meet them. As Blomberg points out, this requires a broad sharing of information that may not always be possible or popular. It also requires technical staff to have the ability to elicit requirements from business users, and these two groups may not be able to communicate on the same level. Fitsili [11] remarks that by only penalizing performance below a minimum standard, providers are incentivized to strive for minimum levels and no better.
- SLA obligations are not met. Forrester [10] reports that SLAs are unmet 75% of the time, and suggests the problem is that IT has moved from managing servers and networks to managing the applications (or at least the middleware) that runs on them.

[2]The largest gap, and a prevalent case, is when the SLA is unilaterally dictated by a provider and there is no negotiation. In this case, this section applies to the provider as they author and re-examine this mandated SLA.

III. Value, Quality, and Cost

To maintain and evolve a Service Level Agreement while maintaining customer satisfaction, we have to consider the notions of value, quality and cost; to that end, this section turns to marketing and business literature, which has long been considering these concepts in the context of more traditional services. In particular, we review this literature in order to consider how *perceived* value, quality, and cost inform the decision-making process is described.

Value is not a constant definition with all consumers, and it varies from person to person and from product to product. Zeithaml [12] offers an overall definition: "*perceived value* is the consumer's overall assessment of the utility of a product based on perceptions of what is received and what is given." To us, that means "they do a cost-benefit analysis". There is a trade-off inherent to value: customers are willing to sacrifice some attributes in exchange for gains in other attributes. For example, they may give up convenient twist-

top lids in exchange for 100% pure juice, or they may pay more for a product they consider environmentally friendly. Decisions can be guided by single attributes (*e.g.*, monetary cost) or some combination. In simple terms, one can think of perceived value as "what I get for what I give", where each individual may consider various attributes to 'give' in exchange for various attributes they 'get'. The total of what is given up (or *sacrificed*) is the *perceived cost*.

There is a general agreement that value is a conclusion reached based on a variety of factors and attributes. Essentially, low level attributes imply quality, quality and cost imply value, and "value in general" affects "value to me personally". Sawyer and Dickson [13] defined value as "a ratio of attributes weighted by their evaluations divided by price weighted by its evaluation". They separate price from the other attributes. Though this is a formula that implies numeric analysis, it can be difficult to quantify the variables.

Quality is, at an abstract level, a measure of excellence or superiority [12]. It is not an attribute; rather, it is a high-level construct formed in the minds of individuals based on their assessment of a product or service. Research distinguishes between *actual quality* and *perceived quality*. *Actual quality* is some quantifiable and verifiable measure of excellence, compared to some norm or ideal. We measure water quality in parts-per-million of substances that should not be in pure (ideal) water. We measure manufacturing quality in terms of the number of defects per item produced, with an ideal of "0". *Perceived quality* is a more complex measure that factors in customer expectations, customer experience, comparator sets, and other factors to form a global assessment of quality. It is a customer's judgment about the superiority or excellence of a product. It is this definition that most influences perceived value. Perceived quality, in its reliance on experience and knowledge, is assessed based on what the consumer knows about the product and all competing products. Which products are 'competing' is up to the consumer (Ford might think their competition is Toyota, but to some consumers it may be Vespa).

There are varying schools of thought on how to assess quality. Perhaps the best known is SERVQUAL [14], which assesses quality based on the gap between what the consumer expects and what they actually experience. It measures the gap between expectation and perception in five primary categories: Reliability, Assurance, Tangibles, Empathy, and Responsiveness (RATER). It is driven fundamentally by satisfaction, where the smaller the gaps, the greater the satisfaction. Over the 25 years since was proposed, it has been adopted and refined several times (*e.g.*, [15][16]), has attracted criticism and has motivated the proposal of alternatives. One of the best known contenders is SERVPERF [17], which is performance-based or attitude-based: Cronin and Taylor assert that perceived service quality can be best measured by a customer's "perceived" attitude about the service. The debate (*e.g.*, [18][19]) has continued since and will not be re-produced here[3]. Lutz [20] proposes that we consider two other forms of quality: "affective quality" and "cognitive quality". A cognitive judgment is one based primarily on attributes that can be determined without actually consuming a product (brand, reputation, packaging), while affective quality is determined by actual experience. He suggests that service quality is an affective judgment: that is, global assessment of service quality is based on personal experience.

Zeithaml [12] presented propositions about value and quality; in particular, how low-level attributes map to higher-order constructs like quality. Her work focuses on regular consumers making personal decisions about products (groceries, travel plans, and so forth), and not individuals making decisions about electronic services on behalf of an enterprise. The assumption in this work is that the concepts are largely transferable.

The points most relevant to this paper are as follows (from [12]):

- **Consumers rely on lower level attribute cues to infer quality**; for example, to some consumers the amount of suds indicates high-quality detergents. Though more complex metrics are available, one or two easily-compared attributes are used to serve as reliable signals of quality. In the context of our work, we interpret this point as follows: clients of service-oriented systems should be given the opportunity to understand and formulate SLAs based on (at least) a few low-level widely available quality attributes.
- **Consumers rely on measurable, observable attributes when evaluating their experience**, particularly when these attributes are easy to understand and are accurate predictors of their satisfaction. Among the implications of this point is that clients of service-oriented systems will benefit from a visual interface that makes quality attributes observable in formulating SLAs.
- **Monetary price is not the only sacrifice perceived by consumers.** When you make a decision, you weigh many trade-offs, only one of which is price. In software service systems, one may for example consider availability of the service, the response time guarantee, the legal jurisdiction which governs the service provider, or any number of other factors in addition to the total cost.
- **The benefit components of value include low-level attributes and relevant high level abstractions.** Low-level attributes influence perceived value via other high-level abstractions. So while "juice in a flavour my

[3]Clearly a one-paragraph summary cannot adequately represent almost three decades of ongoing debate, refinement, and extension. Google Scholar reports 6776 citations for the original SERVQUAL paper and 3557 for SERVPERF, and in the thousands for the key debate papers.

child likes" is an intrinsic measurable attribute, its influence on value perception was through a high-level abstraction "I feel appreciated by my child when I buy this juice".

- **Some attributes serve as "value signals" and can substitute for active weighing of benefits and costs.** Consumers are generally "mindless" [21]; there is not careful consideration so much as there is overall impression. Feelings like "it's slower than it was last week" are used in place of hard facts. There are some consumers who spend time and effort to give purchases careful consideration; Zeithaml theorizes that more rational evaluation is used when there is lots of information available, lots of time, lots of processing ability, and high involvement in the purchase. A similar theory is that as the cost of the product goes up (or, as the cost of choosing the wrong product goes up), more careful consideration is more common. In the context of our work, we interpret this point as follows: clients of service-oriented systems should be given the opportunity to freely examine, at their own pace, their service-level options, likely in a manner that allows them to explore and consider alternatives.
- **The perception of value depends on the frame of reference in which the consumer is making an evaluation.** Value determination is situational and depends on context. A consumer may evaluate quality differently if a substantial shift or migration occurs. If they are involved in the maintenance, without experiencing the new SLA they may make suggestions at negotiation time that differ from how they experienced the SLA at run time. In the context of our work, we interpret this point as follows: clients of service-oriented systems should be given a consistent interface through which to formulate the SLA for their systems, before and after migration.
- **Perceived value affects the relationship between quality and purchase.** Customers do not always purchase the highest quality option; they make a decision that balances their perceived quality and perceived sacrifice, namely their perceived value. The trade-off is the key factor and should be incorporated into decision support tools. In the context of our work, we interpret this point as follows: clients of service-oriented systems should be supported in exploring service-level quality trade-offs.

IV. Initial Approach: Case Study

As a first approach to bringing performance-related service-level objectives and service-level management policies, we employ a simulation-driven approach. Simulation allows a type of experience, which may help identify better value for the consumer. It gives simulated experience with low-level attributes to build understanding and assist consumers in reaching conclusions about the quality of the SLA before deployment time.

We measure the behavior of the software system both before and after platform shifts to identify changes to the performance of the software. The result is a before and after performance profile of the system. From this, we can indicate either how a configuration must change to meet the existing targets (and therefore identify an appropriate increase or decrease in cost), or how the service levels should change if the configuration must remain static. The details of this approach are best explained by example.

To test this approach, we examined two migration scenarios based on real-life scenarios. The first involves moving a service from a bare-metal platform to a cloud computing environment; the application is not modified but the service delivery platform is. The second involves a substantial rewrite of the application, with the same output produced using a pre-computed index. Both require re-configuration to meet the existing service levels. This section describes the test bed and original deployment, then describes both evolution scenarios.

A. Original TAPoR Test Bed

The Text Analysis Portal for Research (TAPoR) is a web-based application that provides a suite of text-analysis tools to scholars and researchers in the Digital-Humanities [22]. It includes a front-end portal and a back-end web service called TAPoRware. It has been simulated in previous work [1] and so is ideal for our simulation-supported approach.

TAPoRware is a single web service with 44 operations. Each operation runs in time linear with respect to the size of the input. The subset of operations covered here includes listing the words with their counts, generating word clouds, finding the use of a word in context (concordance), and finding two words located near each other in text (co-occurrence).

A typical usage scenario begins with the end user identifying a piece of text to analyze with a given tool. Then via a web-services client or the web front-end, the user selects the relevant parameters for the analysis. Common options include word stemming, excluding stop words, and the number of words/sentences/paragraphs to display results in-context. In addition, each operation has its own configuration options. The entire text to be analyzed and the configuration options are encoded in a SOAP request and transmitted to the web service.

The processing time for any given request depends on the nature of the operation being performed and on the size of the input text. All operations are CPU-intensive, with lower memory and IO requirements.

The performance-related service level objective for TAPoR used in this paper specifies an expected distribution of request sizes and operation types, and mandates an

ListWords	Word Cloud	Concordance
29,750	35,610	7,270

Table I
RESPONSE TIME FOR THREE OPERATIONS OF THE ORIGINAL TAPOR APPLICATION (MS).

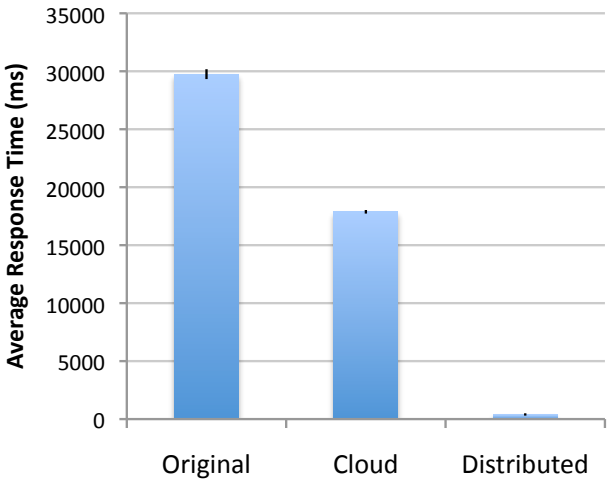

Figure 2. The average response time for the List Words operation running on three different platforms.

average response time of less than 15 seconds measured server-side.

The version of TAPoRware we define as "original" in this paper is an installation created for research purposes. It runs on a single core of a partitioned Core 2 Duo (1.86 GHz). To establish a performance profile for the original installation, we sent a series of test SOAP messages to various operations. These messages include the text for processing, a text corpus almost 1 MB in size (Mary Shelley's *The Last Man*). This is larger than the average message expected.

The sending machine was on the same network as the TAPoR machine. The results are shown in Table I. Only the List Words results are used to create the performance profile; previous experiments have established that it is an accurate predictor of performance.

B. Moving to Cloud Computing

A previous project moved TAPoRware to the cloud to enable scalability. The unmodified application was deployed to an Amazon Web Services instance running Linux (Ubuntu Server 10). As the application is CPU-intensive but does not make extensive use of the file system, the default instance was not modified in any way to improve IO performance. A machine image was created to enable fast future deployments.

For this paper, we created a new instance (single processor, two compute units) based on this machine image. A second instance in the same Amazon zone was created

ListWords	Word Cloud	Concordance
17,880	23,133	4,277

Table II
RESPONSE TIME FOR THREE OPERATIONS ONCE MOVED TO THE CLOUD (MS).

Operation	Distribution	Avg size	Weighted Distribution
WordCloud	49.1%	56459	19.2%
ListWords	22.6%	109507	17.2%
DateFinder	0.6%	242663	1.0%
AcronymFinder	0.2%	82111	0.1%
Concordance	16.3%	247854	28.0%
Collocation	5.1%	790077	28.1%

Table III
METADATA ABOUT THE DISTRIBUTION OF THE TOP 6 OPERATIONS AND THEIR MESSAGES.

to send test SOAP messages and generate the performance profile. The results are shown in Table II; in particular, List Words requests completed in 17,880 ms ($\pm$154.5), a 40% decrease.

This adjusted hardware performance profile is used as a parameter to the simulation that is also configured with the expected load on the service (Table III)[4]. The simulation is run with varied configurations to produce expected response time values based on the operations invoked and the expected size of the messages. The result is an expected response time of 7800 ms across all operations, unsurprisingly well within SLA tolerances. The customer could be offered higher performance (perhaps at a price premium). This simulation can also be systematically reconfigured to explore alternative configurations (for instance a less expensive Amazon instance), hypothetical scenarios where the expected load changes, robustness, or reliability.

A benefit of simulation is the consumer can use the simulation framework's visualization tools to observe the simulation as it runs, comparing the metrics generated to their experience with the previous version of the service. This will assist in producing an SLA that matches the consumer's expectations.

C. Moving to Distributed Computing

A previous project re-implemented TAPoRware operations to use distributed data processing in the form of Hadoop (MapReduce) [23]. The MapReduce framework [24] offers a simple and elegant paradigm for processing large data sets, exploiting the increased availability of a large number of computing nodes, possibly virtualized (on a cloud). In this paradigm, a master node decomposes the input data set into smaller data sets and distributes them to worker nodes (map phase). The worker nodes process

[4]These values are imperfect; the size follows a double-Gaussian distribution for most operations.

(or possibly further decompose and distribute) their data sets and return the answer back to their master node which is responsible for composing the answers to produce an answer corresponding to the overall input problem (reduce phase). Hadoop is an open-source implementation of the MapReduce framework in Java.

We used an index-and-query strategy that would make best use of MapReduce. Counting every word in a document, for example, necessarily involves reading every word in the document. To achieve the kind of speed-up we needed, we had to begin with an index. We also identified new functional requirements that could not be met by the old implementation but that we could provide during the transition. Our requirements included generating indices of large text corpora that would facilitate querying the collection, or defined sub-collections, in an offline fashion; querying indices on request; and maintaining backward compatibility.

We designed our indexes to allow operations to re-use the same index, to allow us to divide collections into sub-collections easily (*e.g.*, to analyze the collected works of William Shakespeare one day, and only Hamlet the next), and to be quickly searchable and sortable. An index has, for each word, a count of its occurrences in the collection, a list of the files that word appears in, and the byte locations for each of those files. For us, a collection consists of a set of files; sub-collections are subsets of this file set. The indexes are stored in a distributed file system (HDFS) and accessed by the querying algorithms.

The new web service runs in the Apache Axis SOAP stack[5] on Apache Tomcat 6[6], and is implemented in Java. The majority of the service was generated using the Axis `wsdl2java` tool. The WSDL for the new service is nearly identical to the old. The main technical change is a new URI for the service; the main semantic change is the "inputText" parameter is now the name of the text collection to query, instead of being the actual text to query.

The indexes were created on a small Hadoop cluster, with the actual processing happening on a pair of similar machines (Xeon Quad Core, 16-32GB RAM). The same machines hosted the distributed file system on which indexes were queried.

The performance of this re-implemented system was measured by indexing the same text used in the previous two tests, then sending a series of requests to three representative operations and measuring response time. The results are shown in Table IV.

This adjusted hardware performance profile is again used as a parameter to a simulation of TAPoR (see *e.g.*, [1]). For this scenario, however, the workflow has change substantially: there is now a pre-processing step where the submitted text is indexed. In addition to the load described in Table III), we also need an estimate on the number of requests with repeated text corpora. We turned to historic access logs and estimated that 61% of requests were repeated versions of previous requests (based on text input size for input longer than 10KB). The simulation was configured to include indexing costs for 39% of all incoming requests.

Indexing	ListWords	Word Cloud	Concordance
60,000	400	1200	600

Table IV
TIME TO BUILD THE INDEX, AND RESPONSE TIME FOR THREE OPERATIONS AFTER THE INDEX IS BUILT (MS).

The result is an expected response time of 6,881 ms across all operations, again within SLA tolerances. However, the change in workflow may change customer perceptions; in particular, the faster response time once a text corpus is indexed may result in additional requests (which will take 1 second instead of 10-15). Our intuition is that the ability to run additional requests quickly will increase the perceived value even at the cost of increased index time, but it is also possible that users will be unhappy with the initial indexing time. The simulation visualization features can be used to test this before deploying the new version of the software.

The hardware costs of this configuration are also substantially higher – this hardware is only needed for indexing, the actual queries of the indices require very little computation power, but to index on-demand it must be available constantly. It is unlikely that the provider could offer the same service levels at the same price. It should be noted that the benefits of this shift in platform are intended for larger text corpora than the load expected in the past. The smaller sized requests that make up our sample request set will not benefit as much. For instance, a data set 100 times larger can be indexed in 847 seconds, only 14 times longer. The original version is not capable of processing text input of that size. If the expected load changes to include larger text corpora, the SLA will need to be revisited.

Again, simulation can be used to explore alternatives - for instance, while indexing a text corpus, using the older version of the software to produce results quickly, then shifting to the index once it has been built.

V. CONCLUSION

We introduced and motivated SLA maintenance and evolution, an activity we assert should occur in step with the maintenance and evolution of service-oriented software. We described the problems with SLAs in practice, which include poorly understood, poorly specified, and poorly adhered to. Once these hurdles are overcome and a satisfying SLA is arrived at, minor maintenance efforts in step with the changing software are preferable to re-negotiation. We then discussed value, cost, and quality in the context of marketing literature in an attempt to understand how service consumers

[5]http://ws.apache.org/axis2/
[6]http://tomcat.apache.org/

will evaluate SLAs and how this affects the maintenance of SLAs. The importance of experience, accurate evaluation of low-level metrics, and the importance of context to value are key to SLA maintenance strategies. In particular, there are likely to be certain key metrics that will impact evaluation of the SLA on paper, and perhaps another set of metrics that impact evaluation of satisfaction in practice.

We then described an initial simulation-supported approach to identify how the maintenance and evolution activities on software impact the performance of the software system, and to modify the performance-related aspects of the SLA in step. We used TAPoR as a testbed, in particular two major evolution activities (moving to the cloud and moving to a distributed computing platform). We showed how we created a hardware profile from the original and two evolved deployments, then provided this profile to a simulation of TAPoR. Using the expected load, we were able to compute the projected impact on the performance of the software system. This information can be used to adjust the elements of the SLA relevant to performance.

ACKNOWLEDGMENTS

The authors receive funding from IBM, NSERC, and iCore. The move to distributed computing was done with Himanshu Vashishtha.

REFERENCES

[1] M. Smit, A. Nisbet, E. Stroulia, G. Iszlai, and A. Edgar, "Toward a simulation-generated knowledge base of service performance," in *MWSOC '09: Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing*, Nov 2009.

[2] P. Rubach and M. Sobolewski, "Dynamic sla negotiation in autonomic federated environments," in *Proceedings of the Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: ADI, CAMS, EI2N, ISDE, IWSSA, MONET, OnToContent, ODIS, ORM, OTM Academy, SWWS, SEMELS, Beyond SAWSDL, and COMBEK 2009*, ser. OTM '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 248–258.

[3] K. Mahbub and G. Spanoudakis, "Proactive sla negotiation for service based systems," in *Proceedings of the 2010 6th World Congress on Services*, ser. SERVICES '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 519–526.

[4] F. Zulkernine, P. Martin, C. Craddock, and K. Wilson, "A policy-based middleware for web services sla negotiation," in *Proceedings of the 2009 IEEE International Conference on Web Services*, ser. ICWS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1043–1050.

[5] G. Koumoutsos, S. Denazis, and K. Thramboulidis, "Sla e-negotiations, enforcement and management in an autonomic environment," in *Proceedings of the 3rd IEEE international workshop on Modelling Autonomic Communications Environments*, ser. MACE '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 120–125.

[6] N. Hargrove, I. Heritage, P. Imandi, M. Keen, W. Neave, L. Olson, B. Perepa, and A. White, *Service Lifecycle Governance with IBM WebSphere Service Registry and Repository*, IBM, December 2009.

[7] P. Hasselmeyer, B. Koller, I. Kotsiopoulos, D. Kuo, and M. Parkin, "Negotiating SLAs with Dynamic Pricing Policies," *Proceedings of the SOC@ Inside'07*, 2007.

[8] J. Blomberg, "Negotiating meaning of shared information in service system encounters," *European Management Journal*, vol. 26, no. 4, pp. 213 – 222, 2008.

[9] Q. He, J. Yan, R. Kowalczyk, H. Jin, and Y. Yang, "Lifetime service level agreement management with autonomous agents for services provision," *Information Sciences*, vol. 179, no. 15, pp. 2591 – 2605, 2009, including Special Issue on Computer-Supported Cooperative Work - Techniques and Applications, The 11th Edition of the International Conference on CSCW in Design.

[10] Forrester Research, "Managing IT services from the outside in," http://www.compuware.com/dl/ManagingITServicesFromTheOutsideIn.pdf.

[11] P. Fitsilis, "Practices and problems in managing electronic services using SLAs," *Information Management & Computer Security*, vol. 14, no. 2, pp. 185–195, 2006.

[12] V. Zeithaml, "Consumer perceptions of price, quality, and value: a means-end model and synthesis of evidence," *The Journal of Marketing*, vol. 52, no. 3, pp. 2–22, 1988.

[13] A. Sawyer and P. Dickson, "Psychological perspectives on consumer response to sales promotion," *Research on Sales Promotion: Collected Papers, Marketing Science Institute, Cambridge, MA*, pp. 1–21, 1984.

[14] A. Parasuraman, V. Zeithaml, and L. Berry, "SERVQUAL: A Multiple-Item Scale For Measuring Consumer Perception," *Journal of Retailing*, vol. 64, no. 1, pp. 12–40, 1988.

[15] A. Parasuraman, L. Berry, and V. Zeithaml, "Refinement and reassessment of the SERVQUAL scale." *Journal of Retailing*, 1991.

[16] V. Zeithaml, A. Parasuraman, and L. Berry, *Delivering quality service: Balancing customer perceptions and expectations*. Free Pr, 1990.

[17] J. J. Cronin and S. A. Taylor, "Measuring service quality: A reexamination and extension," *The Journal of Marketing*, vol. 56, no. 3, pp. pp. 55–68, 1992. [Online]. Available: http://www.jstor.org/stable/1252296

[18] ——, "SERVPERF versus SERVQUAL: Reconciling performance-based and perceptions-minus-expectations measurement of service quality," *The Journal of Marketing*, vol. 58, no. 1, pp. pp. 125–131, 1994.

[19] A. Parasuraman, V. Zeithaml, and L. Berry, "Reassessment of expectations as a comparison standard in measuring service quality: implications for further research," *The Journal of Marketing*, vol. 58, no. 1, pp. 111–124, 1994.

[20] R. Lutz, "Quality is as quality does: An attitudinal perspective on consumer quality judgments," 1986, presentation to the Marketing Science Institute Trustees' Meeting, Cambridge, MA.

[21] E. Langer, "Rethinking the role of thought in social interaction," *New directions in attribution research*, vol. 2, pp. 35–58, 1978.

[22] G. Rockwell, "Tapor: Building a portal for text analysis," in *Mind Technologies; Humanities Computing and the Canadian Academic Community*, R. Siemens and D. Moorman, Eds. Calgary, AB: University of Calgary Press, 2006, pp. 285–299.

[23] H. Vashishtha, M. Smit, and E. Stroulia, "Moving text analysis tools to the cloud," in *IEEE Congress on Services*. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 107–114.

[24] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.