

MANAGING THE QOS OF E-GOVERNMENT: METRICS FOR LARGE SCALE SOA

MAURICIO MONSALVE
DEPARTMENT OF COMPUTER SCIENCE,
UNIVERSIDAD DE CHILE

ABSTRACT. This paper present a set metrics for evaluating the operative aspects of the E-Government SOA systems, based on technical and economical criteria as they are intended to improve management of E-Government projects. We show analytically the desirable properties of these metrics and study them in the concrete case of E-Government in Chile.

Software Metrics, Government Information Systems, Service Oriented Architecture, E-Government

1. INTRODUCTION

Failure is a typical word heard among E-Government project managers. According to Heeks [1], the rate of failure of E-Government projects range from 60% to 85% among different estimations. This is essentially due to poor management, where the lack of standard metrics to evaluate the overall output of E-Government projects plays an important role. For example in Chile, the *PRYME*, the institution in charge of the modernization of the government, stated in a report [2] that *there is no general model to measure the advancement in E-Government; there are just models to measure isolated aspects of the E-Government projects*. These two facts, failure and lack of metrics, makes management of these projects a hard problem to deal with. In this paper, we will focus on the issue of metrics for E-Government.

The problem has no simple solutions. Using economic tools to analyze the advancement of an E-Government project may not be enough. Economic indicators are not intended to study low level technical work but are useful to study more general aspects of the projects. When the development reaches the technical design stage, economic tools cannot handle the project. But, somehow, it is still necessary to have those, especially when there is a need of economic measures.

On the other hand, software metrics also seem not precise enough for the context of E-Government. Kaner et al [3] say that software metrics lack precision and objectivity. Moreover, they say that the problem with software metrics is the formal model behind them. Sometimes it is missing, sometimes it is just a general idea. For example, using *Lines Of Code* (LOC) as a measure of work: really

do lines of code measure the amount of work? Efficient coders work less? This is a classic example of a vague metric. Another critic is made by Fenton et al [4], whom say that many software metrics lack the right causality. For example, estimating the *size* of the project in order to estimate the *effort* when the size is caused by the effort. This can also be seen as a problem in the formalization of the models behind the metrics. But the use of metrics is not avoidable, specially when working with the public administration.

In the specific case of E-Government context, Moore [5] states the problems of measuring the progress in E-Government projects. He criticizes many existing E-Government metrics as the *maturity* or *online sophistication*. He says that those metrics are not very meaningful and the *citizen* is not considered in these. This leads to another need: taking the citizen into account in the metrics. In order to incorporate the citizen in the metrics, it is necessary to use *social* measures.

In summary, management of E-Government projects need particular metrics about the project/system and many of the existing software metrics lack formalism, there is little link between social economic measures and software metrics, and the citizen is not taken into account in existing measures. Considering this, we propose a formal model and a set of metrics for dealing with large scale SOA systems, as SOA are of central importance to E-Government projects. Our objective is to lay a bridge between the languages and objectives from the public administration and those of the informatics area. Following these ideas, we define a formal model and the metrics are direct measures of this model, as an isomorphism of a real SOA.

Similar work, in terms of studying SOA systems for E-Government, can be found in [6], [7], [8] and [9]. The problem with these models and metrics is that require much more complexity and knowledge about web services and they do not take into account economical criteria, and are not intended to improve management. (There is little or no risk detection in these works.)

Contributions. In this work we present a model of measurement of some aspects of E-Government SOA systems based on Welfare Economics, that is, putting emphasis on the social aspects of the project.

- We present metrics that follow a formal model based on economic properties of SOA systems as network having welfare economics as an objective, as opposed to traditional ones based purely on software engineering or economics of the firm.
- We prove analytically that these metrics have desirable properties. Many metrics are statistical estimators, so these are natural attributes of the system, and other metrics facilitate the comparison of systems of different size.

- We apply these metrics to the case of Chilean E-Government development. In particular, we study thoroughly the case of the Foreign Trade Agency and show the usefulness of the proposed metrics.

Organization. This paper is organized as follows. In the section *Preliminaries* we present the formal model for the metrics. The link between economics and software, in the context of SOA projects, is made here. In the section *Common Problems* we summarize the problems we have found using the Chilean experience in E-Government development projects. In the section *Metrics* we introduce the metrics considering the formal model and the common problems. As metrics are intended to help management, their objective is to detect problems and keep a good level of service. In the section *Application* we analyze the Chilean foreign trade institution using metrics. Finally, in the section *Conclusions*, some final remarks about the research are presented.

2. PRELIMINARIES

2.1. Welfare Economics.¹

Social projects are intended to increase the *social welfare* of their countries. Therefore, E-Government projects have the same objective. Like any social project, E-Government projects should be valued using *socio-economic valuation*, not just typical NPV (Net Present Value) nor ROI (Return On Investment) measures. What matters is total amount of social wealth caused by the project, so a social NPV valuation shall be used.

Social welfare is measured as the total surplus of a given population. In perfect competition, consumer surplus is the total saving of consumers as they buy products at lower prices than their willingness to pay: $\int_{q=0}^{D^{-1}(P')} (D(Q) - P') dq$, where $P = D(Q)$ is the function of demand and P' is the current price of the products. An analogy can be done to the citizens, and say any saving increases their surplus.

How do E-Government Services increase the Social Welfare? An E-Government Service, as a project, consists on serving an existing service but by using Information Technologies. Therefore, the changes are related to *how* the service is provided. In a client-server paradigm, this means the client is serviced without interference like waiting, many trials, etc. For the client, that means *less costs*. What we are saying is that better services are cheaper to the client; they give *more surplus* to them. So, QoS (Quality of Service) is related to social welfare.

For example, let us recall the income declaration process. Every year, Chileans have to declare their income to the SII, *el Servicio de Impuestos Internos* (translated as *the Internal Revenue Service*),

¹More information can be found at [10].

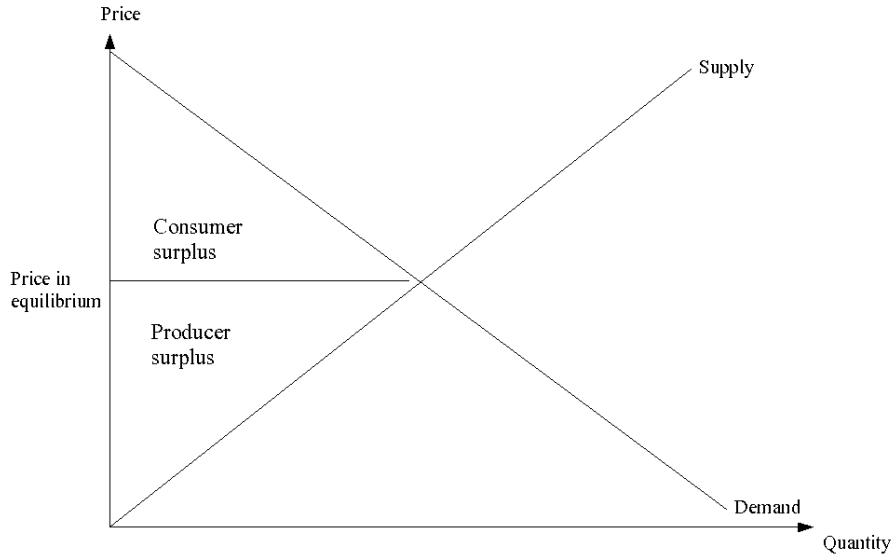


FIGURE 1. Surplus in perfect competition.

an institution in charge of everything related to tax collection. In the 90's, a person had to go to the closest SII office, fill some forms and wait in an usually long queue. Now, you only have to log in the SII's website and fill the forms on-line without waiting. Moreover, even the forms are self filled! The citizen has saved a lot of money with this. Therefore, his/her surplus has increased and so the social welfare.

An E-Government Service has two kinds of clients: citizens and public administration. Citizens benefits are related to time saving, mobilization avoided, etc. Public Administration benefits are related to process improvement and other management savings. But these savings may be affected as systems grow. We will focus in problem detection and the QoS measurements of E-Government SOAs, as systems made by coupling services, particularly to the case of web services (without much loss of generality). We we will not take into account management problems as lack of political involvement, budget restrictions and so on. We will just focus on the more operational and technical part of E-Government SOA projects.

2.2. **Network Economics.** SOA systems share three important economic properties:

- (1) *Scale economies*: as the system grows, it becomes easier (cheaper) to add new functionalities. On the contrary, the system has to be done from scratch in the beginning. SOA is commonly used because of this property of extensibility. (More formally we can say the cost of developing a SOA is a sub-additive function on the size of the system; if S_1, S_2 are

sizes, the cost of building a system of size $S_1 + S_2$ is $C(S_1 + S_2) \leq C(S_1) + C(S_2)$ because of the sub-additivity property.)

- (2) *Barriers to exit*: removing a service may be dangerous to the system. Other services may *depend* on the service being removed. If one service is removed, a great part of the network may be disabled. (This might be caused by the discard of the code on the service removed and some portion of the code of the services that are dependent on the removed one. In other words, the size of the system may greatly decrease by discarding a part of it.)
- (3) *Network externalities*: if a service slows down, the rest of the system may slow down because of this. If a service gets faster, also the rest of the SOA. If a service does not work properly, so does the SOA. Note that this effect is transmitted through the *dependent* services. (We are talking about the value of the system as a finished or functional product.)

The last two properties are important because we can identify a *dependency relationship* among the services. This relationship explains why there are barriers to exit and externalities. The *dependency relationship* is the main engine behind the *value transmission* across the system.

2.3. The Graph Approach. Having stated the relevancy of the dependency relationship, it is natural to start modeling the system by the use of this relationship.

Let R be the dependency relationship, a binary relationship such that $(a, b) \in R$ means “ a depends on b ”, and $a, b \in W$ are services (represented by the set W). So, R has the following properties:

- *A call between two services is a dependency*: if a calls b then $(a, b) \in R$. Let us call this a *direct dependency*.
- *Transitivity*: if $(a, b), (b, c) \in R \Rightarrow (a, c) \in R$. Let us call this an *indirect dependency* as it is explained by the transitivity property.

If we are to use this kind of relationship, we can model it by means of a graph. We will call this graph $G(W, D)$ where W is the set of services and D is the set of *direct dependencies* or *calls*. It is clear that $D \subseteq R \subseteq W \times W$.

As we are modeling stochastic issues (a *call* may or may not be realized during the execution of a service), we will have to rely on some statistical values. In particular, we will need the probability a service does a call and the expected number of calling made by a service.

Definition 1. Let $(v, w) \in D$ be a call and T a time interval. We define the metric $Pc(v, w)$ as the probability that the service v calls w during an execution, and the metric $Nc(v, w)$ the expected number of times the service v calls w during an execution.

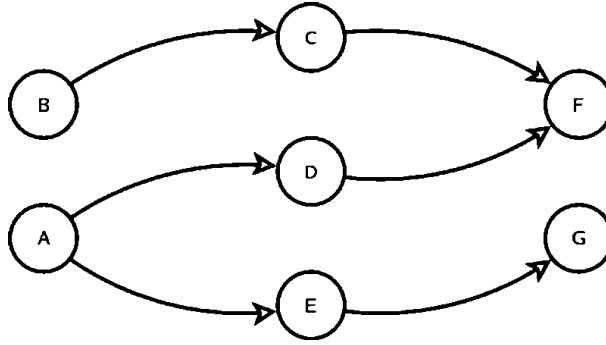


FIGURE 2. Graph approach to modeling SOAs.

In this graph $G(W, D)$, the services are $W = \{A, B, C, D, E, F, G\}$ and the direct dependencies are $D = \{(A, D), (A, E), (B, C), (C, F), (D, F), (E, G)\}$.

Note that P_c shall be measured as the frequency of times that the service does the specific call. N_c shall be measured as the average number of times that the service does the certain call during a service. Both metrics are natural attributes of the system.

Using the graph approach is convenient because it is possible to visualize the graph in a comfortable way and it is pretty natural to understand what the different parts of the graph are. But the best part is the straightforward formalism for the development of metrics, as we show below.

3. COMMON PROBLEMS

We have stated how a SOA is useful in the E-Government context (by explaining the social value of the system) and how to model it by means of a graph. But our model is not important if it does not help the development and management of SOA systems.

During our research, we have found the following set of potential problems:

- A bad implementation of one or more web services.
 - A suboptimal algorithm was used.
 - Bad programming practices. For example, a web service that sends a big response when the message can be simplified (hundreds of Kilobytes or even Megabytes per message!).
- Server overload due to a high demand.
 - Some services are too demanded.
 - There are redundant dependencies.
- There are critical services; if their QoS goes down, also goes down the overall QoS of the system.

- Network latency. For example, the extreme south of Chile has connection problems.
- A server stopped working, so its services.
 - External failure (lack of energy, earthquakes, etc.).
 - Cracking (black hats). Fortunately, this isn't a great issue.
- A web service is slow due to its dependencies.
- The technology got obsolete.
 - A server has old hardware. This is a rather common problem in small public institutions.
 - A web service was built on top of obsolete technology (legacy systems). A consequence of this is the current variety of technologies used by the different institutions.

Some problems are hard to discover with a metric, like cracking. But other problems are related to response times, demand and the SOA structure. Therefore, we will build metrics using that data so we can detect some of these problems.

How to deal with potential problems (risks) or active problems? The following set of strategies may be useful in dealing with the related problems:

- (1) Server level
 - (a) Invest in better/newer hardware.
 - (b) Update the server software (operating system, server management, etc.)
- (2) Service level
 - (a) Choose a better algorithm.
 - (b) Use better programming practices.
- (3) SOA level
 - (a) Distribute the load on different servers.
 - (b) Redesign the dependencies.

These strategies are general guidelines but give a lot of options for dealing with problems in SOA projects.

4. METRICS

This section consists in the definition and discussion of the metrics. The metrics are presented in three groups: *Time Related Metrics*, *Overload Metrics* and *Criticality Metrics*. Time Related Metrics are intended to aid in the vigilance of the QoS of the system and in the identification of time consuming services, Overload Metrics are intended to identify where the load is placed, and Criticality

Metrics are intended to show the sensibility of the system to the dependency of services. Also there is an Extensions section in where we discuss other applications of our model and metrics.

4.1. Time Related Metrics. Our first concern is time because it has a directly related cost: the opportunity cost of time. The amount of time the client waits for a service (the service time) has to remain small enough to guarantee the QoS.

Let us define *Level of Service* of a service –or simply *LoS*– as the probability that a client is serviced under a specific time defined by the management (τ). We propose the following two metrics as measurements of LoS:

Definition 2. Let w be a service, T be the set of execution times of w and τ the service time parameter. We define $LService_w(T) = \frac{|t \in T: t < \tau|}{|T|}$, a metric for the LoS given a set of times.

$LService$ is a metric that works given a set of execution times over a period of time. The source of the data should be the logs of the server.

Definition 3. Let w be a service, t be the time of execution of w , τ the service time parameter, λ the relevancy of the past data parameter and $u(x)$ the step function. We define $LServiceS_w(t) = \lambda LServiceS_w(t) + (1 - \lambda)u(\tau - t)$, a recursive metric for the LoS for automatic update.

$LServiceS$ is a metric that should be updated each time a service is executed. It deals with the problem of defining a time interval for the measurement. $\lambda \in [0, 1]$, the *relevancy of the past data*, is a parameter that should be close to 1 (above 0,9). If this is not the case, the last execution time will have too much effect in the estimator.

Theorem 1. Both $LService$ and $LServiceS$ are estimators of $P(t < \tau)$, where t is the time of execution of a given service.

Proof. It is easy to see that $LService$ is an estimator by the frequency definition of probability. In the case of $LServiceS$, let us see $E(u(\tau - t))$. The step function is equal to 1 when $\tau > t$ and 0 otherwise. Then, $E(u(\tau - t)) = 1 \times P(t < \tau)$. Therefore, $E(LServiceS) = \lambda E(LServiceS) + (1 - \lambda)P(t < \tau) \Rightarrow E(LServiceS) = P(t < \tau)$. \square

Now we will define rather common metrics for the execution time of a service: average time and variance.

Definition 4. Let w be a service and T the set of execution times of w . We define $ATime_w(T) = \frac{1}{|T|} \sum_{t \in T} t$ and $VTime_w(T) = \frac{1}{|T|-1} \sum_{t \in T} (t - ATime(T))^2$ as the metrics for the average execution time and the variance of the execution time, respectively.

A $Time$ and V $Time$ are metrics that show how the LoS is achieved. If the execution times follow a normal distribution, A $Time$ and V $Time$ can be used to estimate the LoS of a given service. And it is possible to *estimate the social cost of a service per client* by using the A $Time$ metric: just compute $A\mathit{Time}_w \times \mathit{SocialCostOfTime}$. The *social cost of time* may be estimated using the salary information –valuated under social assumptions– and the cost of accessing the service via web.

We need the following two definitions to proceed:

Definition 5. Let $G(W, D)$ be a graph of services and dependencies, $w, v \in W$ be services such that $(w, v) \in D$ holds. We define $Pc(w, v)$ as the probability that w calls v during an execution and $Nc(w, v)$ as the expected number of times that w calls v during an execution.

Slower services are bottlenecks in a SOA. The previous metrics are not enough to discover which services are the bottlenecks of the system so we will define a new metric for that purpose: the *self execution time*.

Definition 6. Let $G(W, D)$ be a graph of services and dependencies, $w \in W$ be a service and T the set of execution times of w . We define $S\mathit{Time}_w(T) = A\mathit{Time}_w(T) - \sum_{(w,v) \in D} A\mathit{Time}_v(T) \times Nc(w, v)$, a metric that measures the self execution time of a service.

S $Time$ measures the *self execution time* by discounting the time used by the dependencies. If there are time problems in the SOA (for example, caused by a low LoS), looking for services with bigger S $Time$ could be an intuitive strategy.

4.2. Overload Metrics. Excessive demand may cause the overall performance of the system to go down. Metrics that try to detect where the overload may be useful in such situations.

Redundancy of calls may represent a great problem to highly demanded systems. If a service is called more times than necessary, there is an undesirable overload. But, how to detect redundancy of calls? Perhaps the review of code is the only exact method. But we can try to detect just a certain kind of redundancy of calls: when a service calls many times another service.

Definition 7. Let $G(W, D)$ be a graph of services and dependencies, $w, v \in W$ be services such that $Pc(w, v) > 0$ holds. We define $RNc(w, v) = \frac{Nc(w, v)}{Pc(w, v)} - 1$, the relative number of calls.

Theorem 2. If $Pc(w, v) > 0$, $RNc(w, v) \geq 0$. Moreover, $RNc(w, v) = 0$ implies that there is at most one call per execution.

Proof. Let be $f_n(w, v)$ the probability that exactly n calls (w, v) are realized during an execution of w . Clearly, $\sum_{n \geq 0} f_n = 1$, so $Nc(w, v) = \sum_{n \geq 0} n f_n \geq \sum_{n \geq 1} f_n = Pc(w, v)$. Then, $Nc(w, v) \geq$

$Pc(w, v) \Rightarrow RNc(w, v) = \frac{Nc(w, v)}{Pc(w, v)} - 1 \geq 0$. Now, $RNc(w, v) = 0 \Rightarrow Nc(w, v) = Pc(w, v)$, so $\sum_{n \geq 0} n f_n = \sum_{n \geq 1} f_n \Rightarrow f_1 + \sum_{n \geq 2} n f_n = f_1 + \sum_{n \geq 2} f_n$. As $n f_n > f_n$ if $n \geq 2$ and $f_n > 0$, then $\forall n \geq 2, f_n = 0$. Therefore, $RNc(w, v) = 0$ implies that at most v is called once per execution of w . \square

The previous theorem says if RNc is zero, the number of calls. If RNc is greater than 0, there *might be* redundancy of calls.

Another important measurement is the amount of overload a given service causes by its execution. This is a recursive property, as the overload caused by the called services is also caused by the caller.

Definition 8. Let $G(W, D)$ be a graph of services and dependencies and $w \in W$ a service. We define $Load(w) = \sum_{(w, v) \in D} Nc(w, v) \times (1 + Load(v))$, the metric of load caused by the execution of the service w .

Related to the load caused by the execution of a service, we define the concept of *internal demand* of a service as the demand generated by the system and *external demand* as the demand that comes from outside the system. The demand of a service is the sum of both quantities. Anyway, the available data is related to the total demand per service.

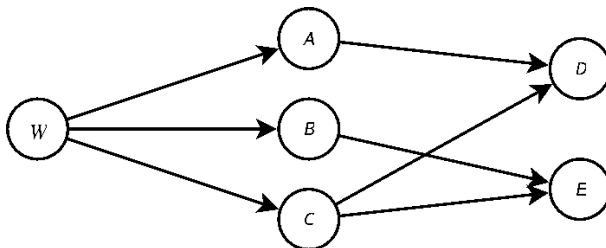


FIGURE 3. Load in a graph. If $Nc = 1$ for all arcs, $Load(W) = 7$.

Definition 9. Let w be a service, T a time interval and Q the number of executions of w during T . We define $Demand(w) = \frac{Q}{|T|}$, the total demand of a service.

Note that a demand is a *frequency*. It has to be measured in terms of [executions/day], [executions/week], [executions/month], etc.

Definition 10. Let $G(W, D)$ be a graph of services and dependencies and $w \in W$ a service. We define $IntDemand(w) = \sum_{(v, w) \in D} Demand(v) \times Nc(v, w)$, the internal demand of a service, and $ExtDemand(w) = Demand(w) - IntDemand(w)$, the external demand of a service.

The internal demand is the demand caused by the other services in the SOA. Therefore, it is possible to control the internal demand by redesigning the dependencies in the system. A good goal may be minimizing the *Load* of certain services. Other goal may be distributing the load through many servers.

The external demand is also a measure of the impact of a failure in a service. Critical services are those with higher external demand as their reliability is essential.

If the external demand are human clients or other institutions, it is possible to measure the *social cost* of the system; just compute $SocialCost = \sum_{w \in W} ExtDemand(w) \times ATime(w) \times SocialCostOfTime$. As the total demand is an important factor in the LoS (the bigger the demand, the greater the overload, thus the lower the LoS), the internal demand should be distributed through many servers. The external demand is a sunk cost because is independent of the SOA structure (as it comes from outside the system) and the removal of a service may cause third parties to lose their possibility to be serviced.

4.3. Criticality Metrics. This set of metrics is intended to measure the sensibility of the SOA to the removal of one or more services.

Definition 11. Let $G(W, D)$ be a graph of services and dependencies and $w \in W$ a service. We define $Incomers(w) = |(v, w) \in D|$, the number of services with a dependency on w , and $Outgoers(w) = |(w, v) \in D|$, the number of dependencies of w .

$Incomers(w)$ counts the number of direct services that may fail if w fails. On the other side, $Outgoers(w)$ measures the sensibility of w to fail due to the failure of a direct dependency; if any dependency fails, the dependent service will fail.

Definition 12. Let $G(W, D)$ be a graph of services. We define $Density(G) = \frac{|D|}{|W|(|W|-1)}$, the density of the network G .

The density is a typical graph metric. In this case, the density reflects the level of dependency of the system. It is easy to see that the density equals to 1 in the acyclic clique of G . This metric is also useful because it is *dimensionless*, as the result does not depend on $|W|$ nor $|D|$.

Another concept that might be important is the concept of *centrality*. In our context, *centrality* refers to the situation where many services depend on a single one (the central). We already defined the *Incomers* metric as the numbers of services that depend on a service but that value is not independent of the size of the graph. If we want to compare two graphs, we should try to use a dimensionless metric. In this spirit we will define the *CentralDegree* metric.

Definition 13. Let $G(W, D)$ be a graph of services. We define $CentralDegree(G) = \sqrt{\frac{\sum_{v \in W} Incomers^2(v)}{(2|W|^3 - 3|W|^2 + |W|)/6}}$, the degree of node concentration of G .

This metric, as it sums $Incomers^2(w)$, it is greater when the dependencies are gathered around specific services. (This is explained by the following property: $\forall a, b \geq 0, a^2 + b^2 \leq (a + b)^2$.) Thus, $CentralDegree$ measures if the network has many services which gather dependencies from other services.

Theorem 3. Let $G(W, D)$ be a graph of services. Then $CentralDegree(G)$ grows as the $|D|$ grows. In other words, the $CentralDegree$ increases as the number of dependencies does so.

Proof. Given a graph $G(W, D)$, add a new direct dependency (a, b) to D . Then $Incomers(b)$ will grow (by 1), so do $CentralDegree(G)$ as it sums all $Incomers$. \square

Theorem 4. If $G(W, D)$ is an acyclic graph, then $CentralDegree(G) \in [0, 1]$.

Proof. It is easy to see that if $G(W, D)$ is such that $D = \emptyset$, $CentralDegree(G) = 0$ as all $Incomers$ are 0. To prove that $CentralDegree(G) \leq 1$, we will find the maximum and show that is 1, for $|W| > 1$ (so $|D| > 0$ is feasible). First, let us state that $G(W, D)$ is a DAG. And as a DAG, it is possible to find a topological sort of its nodes (W). So we have $w_0, \dots, w_{n-1} \in W$, the sorted nodes such that $(w_i, w_j) \in D \Rightarrow i < j$. Adding each possible dependency it is easy to see that $(w_0, w_k), (w_1, w_k), \dots, (w_{k-1}, w_k)$ can be added to D without violating the topological sort. After that we will have the acyclic clique, with the property $Incomers(w_k) = k, k = \{0, \dots, |W| - 1\}$. Therefore, $\sum_{v \in W} Incomers^2(v) = \sum_{k=0}^{|W|-1} k^2 = \frac{2|W|^3 - 3|W|^2 + |W|}{6}$. It is easy to see that this leads to a $CentralDegree(G) = 1$. \square

Theorem 5. Let be $G_n(W, D)$ a graph such that $n = |W|$ and $\frac{Incomers(w)}{|W|} \rightsquigarrow f, \forall w \in W$ ($Incomers$ follow a distribution, regardless the size of the graph). Then $\exists \lim_{n \rightarrow \infty} CentralDegree(G_n)$.

Proof. Let us define $X_i \rightsquigarrow f$. Now we can write $Incomers(w_i) = nX_i, \forall i = \{0, \dots, n - 1\}$. Then, $\sum_{v \in W} Incomers^2(v) = \sum_{i=0}^{n-1} (nX_i)^2$. Taking expected value, $E(\sum_{i=0}^{n-1} (nX_i)^2) = n^2 \sum_{i=0}^{n-1} E(X_i^2) = n^3 E(X^2)$, because all X_i share the same distribution of probability. Then, $\lim_{n \rightarrow \infty} \frac{n^3 E(X^2)}{(2n^3 - 3n^2 + n)/6} = \frac{E(X^2)}{3}$ (exists). It is trivial to see that $\exists \lim_{n \rightarrow \infty} CentralDegree(G)$ (and it is smaller than $\frac{E(X^2)}{3}$ due to the concavity of the square root). \square

Despite its strange form, $CentralDegree(G)$ is a metric that is not *too* dependent on the size of the graph. In small graphs, it is natural that $CentralDegree$ cannot assume many different values.

In the worst cases, if $G(W = \{w\}, D = \emptyset)$ then $CentralDegree(G) = 0$ and if $G(W = \emptyset, D = \emptyset)$, $CentralDegree(G)$ has no sense at all. But with bigger graphs, it becomes easy to compare two graphs with this dimensionless metric, as stated in the previous two theorems.

Other important concept is the *descentrality*. Descentrality happens when a service depends on many other services. This is originally measured by the Outgoers metric, but this metric is not dimensionless and it is useless when comparing graphs of different sizes. So, we will define the metric *DescentralDegree* by analogy to *CentralDegree* but using Outgoers instead of Incomers.

Definition 14. Let $G(W, D)$ be a graph of services. We define $DescentralDegree(G) = \sqrt{\frac{\sum_{v \in W} Outgoers^2(v)}{(2|W|^3 - 3|W|^2 + |W|)/6}}$, the degree of node concentration of G .

It is easy to see that *DescentralDegree* share the same properties of *CentralDegree*, making it an useful metric for comparing two systems with different sizes.

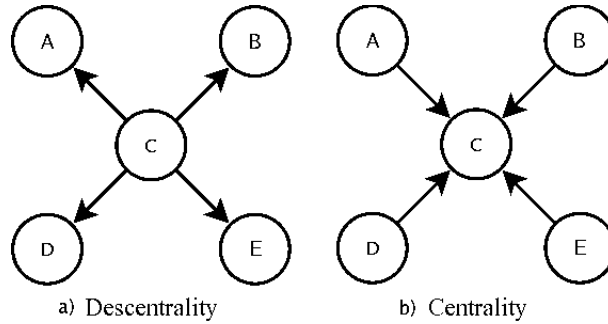


FIGURE 4. Centrality and Descentrality in a graph.

In the central network, C may fail if A, B, D or E fail. In the descentral network, if C fails, A, B, D and E may fail.

Both *CentralDegree* and *DescentralDegree* measure risks and both should remain in a controlled level. If a risk materializes, its impact will be bigger as *CentralDegree* or *DescentralDegree* are greater. So, their level should remain pretty low. If this is not the case, the system should be redesigned at SOA level.

4.4. Extensions. As we are modelling large scale SOA systems, it is possible to apply the same metrics to any other SOA. There is no real restriction to web services or the private sector. (In the latter, new economical costs shall be estimated. For example, the social cost of time shall be replaced by the opportunity cost of time of the firm.) This set of metrics is intended to analyze a network (graph) in order to provide a certain level of QoS. Of course, the relation between the metrics and the social welfare is meaningless outside de public administration context.

The previous model and metrics can be reused in other contexts similar to the SOA. For example, by modelling public institutions as nodes and information exchange as arcs, it is possible to use the previous metrics as an analysis tool of the interoperability between institutions but without the possibility of redesigning the graph, at least not directly. Also, if the nodes of the graph are SOA systems themselves, it is also possible to use the metrics to study the interoperation between networks. Note that the metrics lose a lot of meaning in this context.

5. APPLICATION

Now we will present a few experiences from *El Servicio Nacional de Aduanas*. We will contrast the use of the model and the metrics with their decisions.

El Servicio Nacional de Aduanas or simply *Aduana*, is an institution in charge of foreign trade and border crossing. Technically, *Aduana* faces great operative requirements due to the nature of its work. For example, more than ten thousands of vehicles cross the border each weekend to and from Argentina. Each vehicle shall be identified, thus demanding a lot of resources to the services developed by *Aduana*.

Historically, *Aduana* lacked efficiency and had many other issues. But after it started its modernization process, the institution became pretty innovative and efficient. *Aduanas* has many years of experience with E-Government systems. Moreover, one of its objectives is to improve the management by the use of Information Technology. Actually, it has successfully worked with SOAs since the beginning of its modernization process and is connected to many other institutions.

Aduana has a few years working in the interconnection with other institutions. (Currently, there is a need of interoperability between institutions at system level.) Successful connection has been made with many big Chilean institutions, a few small institutions and the border with Argentina. We will review some of these experiences.

First, we will review the connection with a small institution: *Sernapesca*. *Sernapesca*, *the National Service of Fishing*, is in charge of the negotiation with local fishers, large companies, water rights and fishing rights. This service had no experience in software development, so the connection with *Aduanas* was difficult. After the service in *Sernapesca* was set up, the problem was the time used by services (in terms of metrics, $ATime(W_{Aduana})$ was too large). From the point of view of *Aduana*, the time used by the service in *Sernapesca* was too much ($STime(W_{Aduana}) \ll STime(W_{Sernapesca})$). In order to improve the service time used by the service of *Sernapesca*, the

server update strategy was used (both hardware and software were replaced). But also the service were redesign using better programming practices. This solved the problem.

Other interesting situation happened in the interconnection with Argentina. This required co-operation across the border. As Aduana had more experience in the development of web services, the Argentinean border web service was built by Aduana and the result was successful. However, problems arise when the *Registro Civil*, institution in charge of the identification, updates or alters its systems (typically on weekends). When this happens, the services in the border are rendered useless, as any other service dependent on personal identification. The impact of disabling this service can be seen as the external demand of those services (*ExtDemand*) as it does not belong to the same system as the services of the border. As we said previously, $ExtDemand > 1000[\frac{1}{day}]$ and on weekends can reach $10000[\frac{1}{day}]$ in the border. Clearly, the impact of disabling a service is pretty high and it will become higher as systems grow. A solution for this may be the use of mirror services. (As Aduana has no power over Registro Civil, there was no solution for this issue.)

Now let us study the *Proyecto Ventanilla nica de Comercio Exterior* (a programme to simplify bureaucratic procedures in foreign trade), where Aduana tries to connect to many other institutions using web services. This means the interoperability of many particular systems from different institutions. This is a large system, and the upper management should use the metrics to evaluate the progress. Forecasting the value of the metrics, *Load* and demand metrics may grow a lot due to the requeriments of many new clients and new functionalities. *CentralDegree* and *DescentralDegree* may go down as the network becomes sparse. And time metrics will increase as the level of service will go down because of this. Redesigning parts of the network may be a necessity as the system grows in size.

6. CONCLUSIONS

The metrics used can help management in the development of large SOA systems. The metrics presented have many useful properties that can help in the comparisson between different graphs and in the detection of problems. By using the presented strategies, solving the problems should be a simpler issue.

In order to make the metrics more useful, more experience is required. As comparissons between systems are made, such knowledge will appear and should be shared so the metrics can become more practical. Also, upper level management should use watch the overall system built. If it is solely built by parts, its quality of service will go out of control.

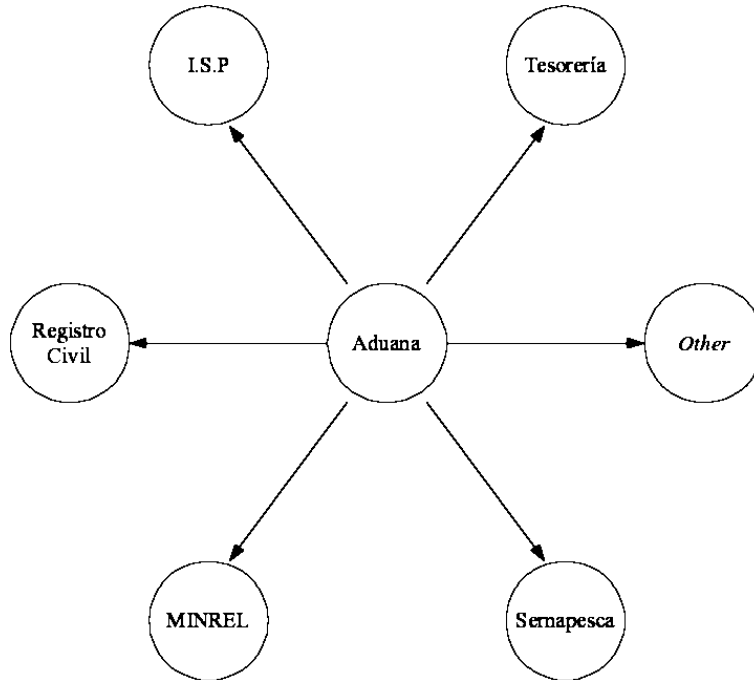


FIGURE 5. Proyecto Ventanilla Única de Comercio Exterior. Simple model.

REFERENCES

- R. Heeks. *Implementing and Managing eGovernment: An International Text*. SAGE Publications, Athenaem Press, United Kingdom. 1st edition, 2006.
- Gobierno Electrónico en Chile 2000-2006: Estado del Arte 2*. PRYME, Gobierno de Chile, Jan 2006.
- C. Kaner, W. Bond. *Software Engineering Metrics: What Do They Measure and How Do We Know?. 10th International Software Metrics Symposium METRICS 2004*, Chicago, IL, Sep 2004.
- N. Fenton, M. Neil. *Software Metrics: Roadmap. Procs ICSE*, May 2000.
- D. Moore. The problems of measuring eGovernment progress (online). *IQ Content website*, Aug 2005. http://www.iqcontent.com/publications/features/article_58/
- C. Kubicek, M. Fisher, P. McKee, R. Smith. *An Architecture for QoS Enabled Dynamic Web Service Deployment. 4th All Hands Meeting*, United Kingdom, Sep 2005.
- F. Corradoni, Ch. Ercoli, A. Polzonetti, O. Riganelli. *An Automata based approach to e-Government cooperation. eGov-INTEROP' 06 Conference, Cit Mondiale, 18 Parvis des Chartrons, Bordeaux (France), Mar 2006*.
- J. Cardoso, J. Miller, A. Sheth, J. Arnold. *Modeling Quality of Service for Workflows and Web Service Processes. Technical report UGACS -TR-02-002, LSDIS Lab, Computer Science Department, University of Georgia, May 2002*.
- Research report on interoperability and co-ordination of Web Services. QUALEG*, 2005.
- P. Samuelson. *Foundations of Economic Analysis*. Harvard University Press, Cambridge, MA, 1983.