Elaborating Scalability Requirements of Large Scale Systems through Goal-Obstacle Analysis

Muhammad Asjad

Computer Science Department, University of Wollongong

maak458@uowmail.edu.au

***Abstract*—** *This report provides a critical analysis and overview of current research on Scalability Requirement Engineering. We have focused on a goal-oriented approach for eliciting, modeling and reasoning about scalability requirements. Based on existing research on the topic, we will discusses a few shortcomings of the methodologies described in the literature on Scalability Requirement Engineering and will propose improvements.*

1. INTRODUCTION

In 1993, The London Ambulance service faced a critical failure. The cause was increased call traffic load, that prevented the system from maintaining location information about the ambulance units triggering a large number of exception messages. LAS's failure to handle extra load during peak usage is attributed to systems' missing scalability characteristics [7].

While discussing the concept of scalability, the questions we usually ask are; what does it mean for a system to be scalable or how do identify a system that lacks scalability? Even though scalability is considered as on of the most important attributes of Software-Intensive systems [8], it has received relatively less attention from RE research community thus far. Scalability concerns are usually ignored during the requirements elicitation procedure and there is lack of a proper definition and having systematic way of describing a system's scalability[7][9].Typically, Scalability is generally seen something of a concept that is related to system performance and a system is said to be scalable if it has the capability to increase resources to yield a linear (ideally) increase in service capacity[7][10]. In reali-ty Performance, however, is only one of many aspects of a scalable system, along with many other such as Availability, Reliability, Securability, Manageability etc. [10].

Scalability requirements(defined with respect to other quality goals of the system) should always be defined upfront during the requirements elicitation process. Defining scalability requirements of a complex system is not a trivial task. One major challenge that usually occurs during this step is that scaling of multiple characteristics in the application domain will affect the satisfaction of different, sometimes competing, quality goals[7]. Its imperative that our scalability requirements that are quantifiable, testable and justifiable in the context of higher-level business goals that the system must achieve [6][7].

Requirements Engineers should have methodology that precisely defines scalability and provides a systematic way of clearly eliciting scalability requirements upfront during the development process, thus preventing catastrophes such as the LAS from happening in the first place. Existing requirement engineering techniques like goal-oriented approach of KAOS, i* provide generic support for elaborating system requirements and don't specifically provide a method for dealing with scalability concerns of the system. (L. Duboc, D. Rosenblum, and T. Wicks, 2012) have proposed an extension of the conceptual framework of KAOS with precise characterizations of the concepts like scalability goals, scalability requirements that will allow requirements engineers to specify, analyze and refine explicit scalability goals[6]. Their approach deals with scalability concerns of a system, during the goal-obstacle analysis steps of the goal-oriented elaboration process [6][12].

2. BACKGROUND

2.1 Goal-Oriented Requirement Engineering

Goal-Orientation is a paradigm which uses goals for elicitation, specifying, elaboration, structuring, documentation and analysis of software requirements [1]. *Goals* have long been used in artificial intelligence and Yue was probably the first who showed us that goals modeled explicitly in requirements models provide a criterion for requirements completeness [64]. van Lamsweerde [3] defines a goal as an objective that the system should achieve through cooperation of agents in the software-to-be and in the environment. The system-to-be is in essence composite; it comprises both the software and its environment, and is. made of active components such as humans, devices and software called *agents* [1]. A *requirement* is a goal whose achievement is the responsibility of a single software agent, while an *assumption* is a goal whose achievement is delegated to a single agent in the environment. Requirements "implement" goals much the same way as programs implement design specifications [4]. Agent-based reasoning is central to requirements engineering since the assignment of responsibilities for goals and constraints among agents in the software-to-be and in the environment is the main outcome of the RE process [3]. Then we have descriptive statements, held true in the application domain, known as Domain Properties.

Goal models consist of AND/OR refinement structures. during the process of refinement, we decompose goals into subgoals so that each subgoal requires cooperation of fewer agents for its satisfaction. Refinement process continues until it reaches a point where each goal can be assigned as the responsibility of single agents. It should be noted that, In order to assignable to an agent, a goal must be realisable by the agent[6][21]. An *obstacle* to some goal is an exceptional condition that prevents the goal from being satisfied[1][6].

Goals are specified using standard temporal logic operators, which indicate the conditions (e.g. Achieve, Maintain, Avoid). The following standard operators are available to us for automated reasoning on goal models:

$\Box P$	(P is true in all future states)
$\Diamond P$	(P is true in some future state)
$\Diamond_{\leq d} P$	(P holds in some future state that is less than d time units from the current state)

3. SCALABILTY REQUIREMENTS ENGINEERING

(L. Duboc et al, 2012) have defined scalability as *"Ability of a system to maintain the satisfaction of its quality goals to levels that are acceptable to its stakeholders when characteristics of the application domain and the system design vary over expected operational ranges" [11].* As previously discussed, old definitions of scalability only included characteristics like capacity and resources and ignored things like domain characteristics (and their expected operational ranges). Thus the above definition is definitely an improvement. However we believe that the scope of this definition is still somewhat limited, that is when applied to some of the large scale web systems (as discussed later in the paper), of whose requirement elicitation are we primarily interested in.

3.1 IDENTIFYING SCALABILITY OBSTACLES

Inspired from the goal-obstacle analysis steps of the goal-oriented method. (L. Duboc et al., 2012) describe the following steps for identifying obstacles:
1) Identify *scalability obstacles* that may obstruct the satisfaction of the goals. 2) *resolve* the obstacles by modifying existing goals, requirements and expectations, or generating new ones so as to prevent, reduce or mitigate the obstacles.
3) The resolution of scalability obstacles will lead to the identification of scaling assumptions and scalability goals.

Once the scalability obstacles have been identified, scaling assumptions and scalability goals can be added to resolve them. (L. Duboc et al., 2012) have define a *scalability goal* as *"A goal whose definition and required levels of goal satisfaction (as specified by its objective functions) make explicit reference to one or more scaling assumptions."* Author further illustrate the concept by giving an example:*"The goal Achieve [Batch Processed Overnight Under Expected Batch Size Evolution] is a scalability goal because its definition refers to the scaling assumption Expected Batch Size Evolution".*

3.2 ELICITING SCALABILITY ASSUMPTIONS:

According to the authors the idea of adding scalability assumptions while describing scalability goals is of vital importance. This is due to the fact that agents that have been assigned to fulfill the goals of the systems have finite processing capacities.
Further explaining the previous example, A goal Achieve [Batch Processed Overnight] after adding a scalability assumption will now be described as:

Goal Achieve [Batch Processed Overnight Under Expected Batch Size Evolution]
Where the scalability assumption is *Under Expected Batch Size Evolution*. Which is described as:
Assumption Expected Batch Size Evolution
Category Scaling assumption
Definition Over the next five years, daily batches of transactions for all banks are expected to vary between 10,000 and 95 million transactions.

According to authors(L. Duboc et al., 2012) the process of specification of scaling assumptions should therefore be defined in terms of the following items:
1) One or more *domain quantities* whose expected variations are defined in the assumption; 2) the *periods of time* and *classes of system instances* over which the assumption is defined; and 3) the *range of values* each quantity is expected to assume for each system class over each period of time.

This approach of specifying bounds on time and domain characteristics might work for medium sized projects such as banking systems, but lets discuss a couple of reasons for why we believe the same approach can't be extended and applied to Large Scale Systems. From our experience, such medium sized systems are developed once and have life span of 4-5 years, after which they are usually scrapped. Such systems are developed once and after deployment they usually don't see major upgrades unless a new product replaces them. Hence, scalability here would usually mean that developers of the system have to pre-optimize the system to handle the upper bounds of the expected workload that the system is expected to handle during its life period. Premature-optimization(apart from being the root of all evils according to Knuth) adds to the time/development costs of the system. There can be huge cost savings, if instead of scrapping such products at the end of their lifetimes we could add modules that can help such systems meet their updated requirements efficiently.

3.2 Specifying Growth Rate: Large Scale systems such as Youtube and Google search, and many other startups can't afford to scrap their whole product that was initially developed and start over. For systems that see that see exponential growth, the only thing that can be predicted with some level of accuracy is the potential growth rate. i.e quantitative increase of certain domain parameters per unit time(like increase in number of users per month, or number of uploaded photos per week etc). These systems will run forever and don't have a limited life span of a few years. In such a scenario, developers don't have the luxury of redesigning a system, that is when system's design/architecture can't keep up with the scalability requirements. Hence it is imperative that the growth rate of one or more domain more parameters is included in the scalability requirements of the system. In most businesses, a monthly growth percent is too volatile to be meaningful. However the TSM Average smooths out the monthly change[20]. It is calculated as: (ending_value/starting_value)^(1/(num_periods-1))-1.

3.3 Specifying Operational Costs: Traditionally, for the development of medium sized projects often times server/running costs are ignored. Since programmer time is considered expensive, it makes sense to tradeoff some runtime performance for programmer productivity [18]. However this not always the case, as observed by quote by Bill Venners in an interview with Twitter's Alex Payne: ***" If you have enough traffic, at some point the cost of servers outweighs the cost of programmers"***.

While eliciting scalability requirements, its important to note that increase in system workload has a direct co-relation with the operational costs and resources required to run the system. As the system scales, operational costs must remain within the bounds of what is acceptable to the stakeholders. Increased capacity of a system was previously thought to be in proportion to the cost[13], but its important to note that for large scale systems costs must grow more slowly than system capacity [14]. The growth pattern should look like the one described in figure 1.

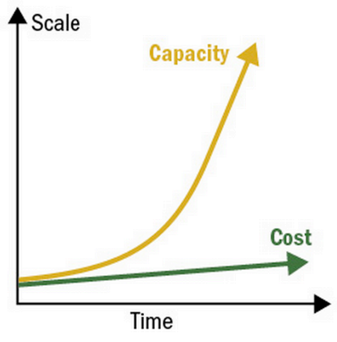

Figure 1. Costs grow slowly as capacity grows quickly

Application scalability requires a balanced partnership between two distinct domains, software and hardware [10]. If operational costs are explicitly mentioned for a given workload then system developers can make much informed decisions about algorithmic and architectural choices which can effect system performance and compute costs down the road.
So for example replacing an O(n2) sort algorithm with an O(n log n) sort algorithm increases the efficiency of a computation in the sense that a larger workload can be processed in the same amount of time [19]. Operational costs can also significantly decrease in such a situation.

Lets discuss an example to see how algorithmic choices effect operational costs. Consider a bubble sort algorithm being used in a photo-sharing that when executed 1,000 times for an average case, costs about $1.7 in computing costs. This cost is acceptable to the stakeholders when the system has 0.5 million users but might become unacceptable when it reaches say 50 million users. Replace the bubble sort algorithm with quick sort O(nlog(n)) and the cost drops will significantly to what might be an acceptable value for the stakeholders. While developing a business model the company comes up with a cost per user (for a service to sustain overtime). Thus we can say that predicting operational costs and ensuring that they remain acceptable to stakeholders overtime is vital for scalable systems.

3.3.2 Estimating Operational Costs:

Factors that influence the operational costs of a system are electricity costs, compute costs that are calculated by using various factors such as time needed for computation, storage costs, Load Balancing, and data transfer out costs[15][16]. Currently most companies rely on cloud service providers(like Amazon, Rackspace etc) to deploy their applications. Such providers usually provide tools like cost-calculator (having a detailed list of parameters), that can be used to estimate operational costs[17]. Going through this process and defining expected operational costs in the scalability requirements will minimize risk, and will help in avoiding surprises during the later stages of the project. Awareness on operational costs right from the start will also help Stakeholders choose between essential and non-essential requirements.

3.4 Scalability - Revised Definition:

This above discussion can lead us to propose a revised definition of scalable systems: *"Given a growth rate X(of one or more domain properties), if a system(with varying processing/load handling capacities) maintains over time, the operational costs and user experience(of the system) to a level that is acceptable to its stakeholders then the system is said to be scalable."*

If we were to use the concept of scalability assumption as described by the author then an example of system requirements (in a somewhat simplified manner) would look like this:

"develop a software system which can handle 10-15million(1st year of deployments) to 95million(5th year of deployment) transactions in 10hours."

Using our new definition if we define the growth rate of the system then developers can do a long term evaluation of the system to see if it will remain scalable without scrapping or a major overhaul. This can be done for example by doing back of the envelope calculations for the running times of the algorithms used and costs of computations. Evaluating the architecture/design to see if it have enough provision for modules that can be(like for doing loading balancing) added to ensure that the system remains scalable according to the criteria defined in the scalability requirements. Furthermore they can come up with a cost- effective strategy for extending a system's capacity.

Using our definition an example of a requirement described informally would look something like this:
*Develop a search engine where (year*0.1million)^2 users are expected to join our service per year. The running or operational costs of the system(once deployed) should be such that cost per user doesn't exceed $3 per month and the response time for search queries under the defined growth rate should remain than 5ms.*

CONCLUSION

In this report we discussed a goal-oriented approach for eliciting, modeling and reasoning about scalability requirements. We discussed a few shortcomings of the methodologies described in the current literature on scalability and recommend a few improvements that are better suited to developing modern day web applications that provide services to millions of users on daily basis. We also proposed a revised definition of scalability that provides us a criteria for judging the scalability characteristic of any system.

REFERENCES

[1] A van Lamsweerde, Goal-Oriented Requirements Engineering: A Guided Tour.

[2] K. Yue. What Does It Mean to Say that a Specification is Complete? Proc. Fourth International Workshop on Software Specification and Design (IWSSD-4), Monterey, USA, 1987.

[3] A. van Lamsweerde. Requirements Engineering in the Year 00: A Research Perspective. 22nd International Conference on Software Engineering (ICSE'2000), Limerick, Ireland, June 2000.

[4] A. van Lamsweerde, E. Letier. From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering. Proc. Radical Innovations of Software and Systems Engineering, LNCS, 2003.

[5] E. Letier and A. van Lamsweerde, "Agent-based tactics for goal-oriented requirements elaboration," in *Proc. 24th International Conference on Software Engineering*. New York, NY, USA: ACM, 2002, pp. 83–93.

[6] Duboc, E. Leiter, D. Rosenblum,Systematic elaboration of scalability requirements through goal-obstacle analysis, IEEE Transactions on Software Engineering,VOL. X (2012)

[7] L. Duboc, E. Letier, and D. S. Rosenblum,"Death, Taxes, & Scalability," IEEE Software, vol. 27, no. 4, pp. 20–21, Jul. 2010.

[8] R. P. Gabriel, L. Northrop, D. C. Schmidt, and K. Sullivan, "Ultra- large-scale systems," in *Proc. 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages and Applications*. New York, NY, USA: ACM, 2006, pp. 632–634.

[9] L. Duboc, E. Letier, D. Rosenblum, and T. Wicks, "A case study in eliciting scalability requirements," in *Proc. 16th IEEE International Symposium on Requirements Engineering*, Barcelona, Spain, 2008.

[10] Scalability, **MSDN**-the microsoft developer network
<http://msdn.microsoft.com/en-us/library/aa292172(v=vs.71).aspx>

[11] L. Duboc, D. Rosenblum, and T. Wicks, "A framework for characterization and analysis of software system scalability," in *Proc. 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. New York, NY, USA: ACM, 2007, pp. 375–384.

[12] A. van Lamsweerde and E. Letier, "Handling obstacles in goal-oriented requirements engineering," *IEEE Transactions on Software Engineering*, vol. 26, no. 10, pp. 978–1005, 2000.

[13] Luke, E.A. Eng. Res. Center, Mississippi State Univ., MS, USA "Defining and measuring scalability"

[14] Four Principles of Engineering Scalable, Big Data Software Systems
<http://blog.sei.cmu.edu/post.cfm/four-principles-engineering-big-data-systems-195>

[15] How AWS Pricing Works
< https://media.amazonwebservices.com/AWS_Pricing_Overview.pdf>

[16] IEEE Spectrum, New Algorithms Reduce the Carbon Cost of Cloud Computing
<http://spectrum.ieee.org/computing/networks/new-algorithms-reduce-the-carbon-cost-of-cloud-computing>

[17] Amazon, Cost Calculator
<http://calculator.s3.amazonaws.com/index.html>

[18] High Scalability, At Some Point The Cost Of Servers Outweighs The Cost Of Programmers
<http://highscalability.com/blog/2009/4/5/at-some-point-the-cost-of-servers-outweighs-the-cost-of-prog.html

[19] Charles B. Weinstock, John B. Goodenough , On System Scalability,Performance-Critical Systems ,Technical Note CMU/SEI-2006-TN-012

[20] Your Startup's 10 Most Important Metrics
< http://tomtunguz.com/your-startups-10-most-important-metrics/>