Economics of Nonfunctional "Quality of Service" Requirements

Steve Tockey
Principal Consultant, Construx Software
Bellevue, WA, USA
stevet@construx.com

JianYang Zhang
Independent Consultant
Beijing, China
applezhang0592@icloud.com

ABSTRACT

Nonfunctional "quality of service" requirements are that subset of software product requirements specifying how well the software is expected to perform. Examples include response time, throughput, reliability, and accuracy. Common problems are that this kind of requirement is poorly specified—if even specified at all. We propose that these problems could be reduced by considering this kind of requirement from an economic perspective. This paper starts by defining nonfunctional quality of service requirements and elaborating on the common problem. The remainder of this paper examines nonfunctional quality of service requirements from an economic perspective, both individually and collectively, showing how this perspective can reduce these common problems.

KEY WORDS

Nonfunctional requirement, quality of service, economic analysis.

NONFUNCTIONAL QUALITY OF SERVICE REQUIREMENTS, DEFINED

The Guide to the Software Engineering Body of Knowledge v4 [1] formally defines a software requirement as,

1) a condition or capability needed by a user to solve a problem or achieve an objective;
2) a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document;
3) a documented representation or capability as in 1) or 2) above.

The Guide then identifies the categories of software requirements shown in Figure 1. Quality of Service constraints are the subset of software requirements that:

- are relevant to the software product (product requirements), not to the project that is building or maintaining that software (project requirements);
- constrain automation technology (nonfunctional), instead of specifying necessary behaviors—policies to enforce and processes to be carried out (functional);
- declare acceptable levels of performance (quality of service, i.e., "how well"), instead of mandate—or prohibit—use of specific, named technologies (technology, i.e., "how").

Quality of service constraints would specify, for example, acceptable response time, throughput, accuracy, reliability, and scalability. ISO/IEC 25010: "System and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models" [2] contains a large list of the kinds of service qualities that are often relevant for software.

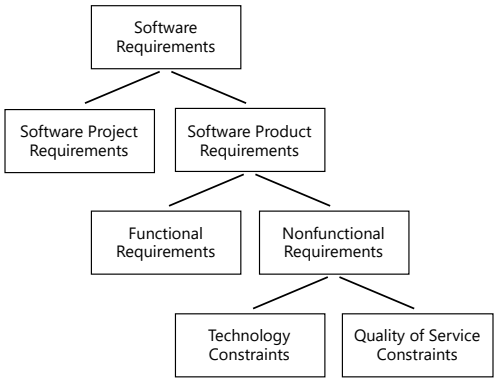

Figure 1. Categories of Software Requirements.

COMMON PROBLEMS WITH QUALITY OF SERVICE REQUIREMENTS

Several common problems surround quality of service requirements, not the least of which is not even considering them as part of the project's requirements activities. Development teams should use ISO/IEC 25010 (or a derivative) as a checklist in requirements elicitation to be sure that all relevant characteristics are exposed and specified.

Another common problem is that they are rarely quantified even when they are stated. Examples of stated but unquantified quality of service requirements include,

- "the system shall be fast;"
- "the system shall be user friendly;"
- "the system shall be maintainable."

In cases where quality of service requirements are quantified, for example,

- "the system shall support at least 500,000 customers;"
- "the system response time shall be 2.5 seconds or less;"

that quantification is almost certainly arbitrary, being based on essentially, "what do I, the stakeholder, feel justified in asking for?" But does that necessarily mean a system capable of supporting at best 499,999 customers fails because it does not "fully satisfy the requirement?"

We propose that reluctance to objectively quantify quality of service requirements is at least in part a result of not considering them from an economic perspective.

ECONOMICS OF SINGLE QUALITY OF SERVICE REQUIREMENTS

This section considers individual quality of service requirements from an economic perspective. The economics of multiple quality of service requirements will be discussed later.

To illustrate we use an e-commerce web site as a case study. Functionally, customers can browse the catalog, add items to their cart, place orders, and so on. We consider four quality of service requirements,

- the number of customers to be supported in the customer database;
- the number of products to be supported in the catalog database;
- response time;
- mean time to repair (MTTR) when recovering from system outages.

Initial focus will be on the number of customers supported.

We also start by considering value only, ignoring the cost of delivering that value. This is to illustrate basic dynamics Cost will be considered later.

A "Level of Performance" (LoP) refers to a specific amount of that quality of service,

- the number of customers supported would be expressed in units of individual customers;
- the number of products supported would be expressed in units of unique catalog items;
- response time could be expressed in units of seconds;
- mean time to repair (MTTR) could be expressed in hours.

Value as a function of level of performance

Economically, there should be some specific business value associated with each relevant level of performance. For example, what is the business value of being able to support 400k customers? 450k customers? 500k customers? 550k customers? Generally, more supported customers should result in more profit. For many quality of service requirements we can expect business value to increase linearly with level of performance as shown in Figure 2.

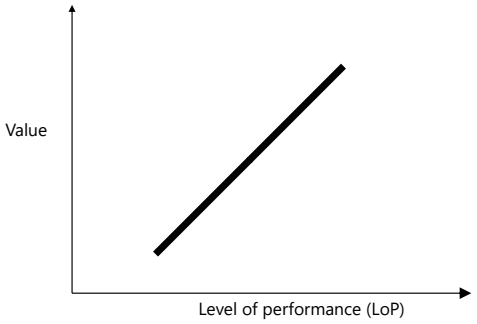

Figure 2. Value as a Function of Level of Performance.

There are non-software parallels, an electric car that can drive farther on a single charge is more useful (valuable) than one with shorter range.

Economic justification of a "Requirement Point"

Above, the stakeholders claimed, "the system shall support at least 500,000 customers" but is that necessarily the right quantification? In analyzing from an economic perspective, assume that the organization's annual expenses for the eCommerce system are $123,456k. If the organization's "Minimum Acceptable Rate of Return" (MARR[1]) on investments is 15% then the annual revenue target should be $123,456k * 1.15 = $141,974k. If the average annual revenue per customer is $375 then the required number of customers to support would be $141,974k / $375 = 379,216.

Although the stakeholders claimed they needed to support 500k customers, they only need 379k to achieve their organization's profitability goal. The requirement should instead be, "the system shall support at least 379k customers". Any economically justified requirement point would be on the value as a function of level of performance graph, as shown in Figure 3.

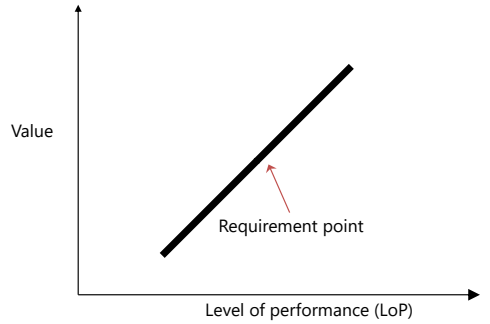

Figure 3. An economically justified Requirement Point.

This kind of economic analysis can be repeated for each relevant Quality of Service requirement: what Level of Performance yields the target annual revenue? The question now,

[1] See any reasonable Engineering Economy textbook, e.g., [3], [4], [5], [6], or [7] for a detailed description of the MARR

shown in Figure 4, is does value increase to infinity and does it decrease to zero?

Economic justification of a "Perfection Point"

Value can increase with level of performance, but only up to a maximum that is constrained either by the business environment or by available technology. The e-commerce system cannot have an infinite customer base—it cannot be greater than the world population. Hardware is certain to wear out at some point so the Mean Time Between Failures (MTBF) cannot be infinite. We introduce the concept of a "Perfection Point" which we define as,

"The most favorable level of performance, beyond which there is no additional benefit"

Either capacity to perform cannot exist, or it exists but there is some real-world reason why the business cannot take advantage of it. In the number of supported customers requirement, there is a finite target market and the organization in question may have some maximum share due to competition.

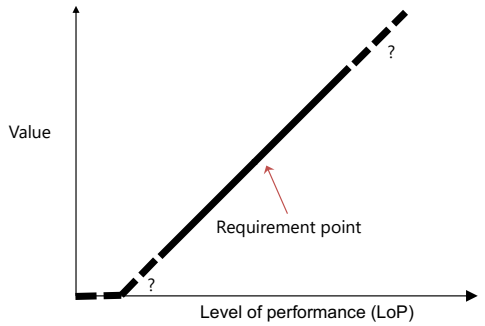

Figure 4. Does value increase to infinity? Does it decrease to zero?

Assume that the entire target market is 2500k customers and this organization has a maximum market share of 30%. The maximum number of customers to support would therefore be 2500k * 0.30 = 750k. The annual revenue at this level of performance would be $281,250k and the profit margin would be 128%.

Even if the system could support more than 750k customers, capacity beyond that is useless because those customers don't exist. The concept of the Perfection Point is shown in Figure 5.

This economic analysis can be repeated for each relevant Quality of Service requirement: what Level of Performance yields the maximum possible annual revenue in this dimension?

Economic justification of a "Fail Point"

Just as value increases up to some maximum, it almost certainly decreases to some minimum. We introduce the concept of a "Fail Point" which we define as,

"Least favorable level of performance, beyond which there is no reduction in benefit"

A social media application needs a minimum number of users to be viable. The Mean Time Between Failures (MTBF) must be long enough for the system to accomplish useful work. An electric car must have enough range to at least reach the next charging station.

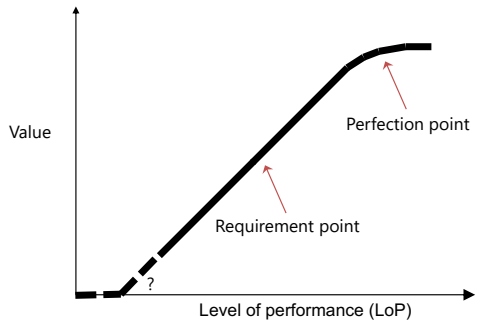

Figure 5. The Perfection Point on the Value as a Function of Level of Performance graph.

If the system is not able to achieve that minimal level of performance, it does not necessarily mean that system has no value at all. It just means value doesn't decrease any further. There may be value in other aspects, e.g., the electric car might be sold for its scrap value.

For the number of supported customers requirement, consider that annual expenses are $123,456k so the number of customers needed to break even is $123,456k / $350 = 329k. The Fail Point is shown in Figure 6.

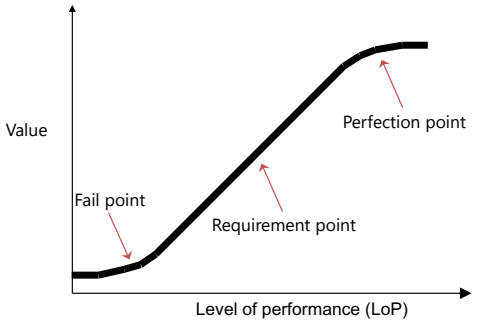

Figure 6. The Fail Point on the Value as a Function of Level of Performance graph.

This economic analysis can be repeated for each relevant Quality of Service requirement: what Level of Performance yields the minimum annual revenue in this dimension (e.g., zero profit)?

Figure 7 shows a concrete example using the number of supported customers requirement in the e-commerce system. Note the numeric values on both X- and Y-axes.

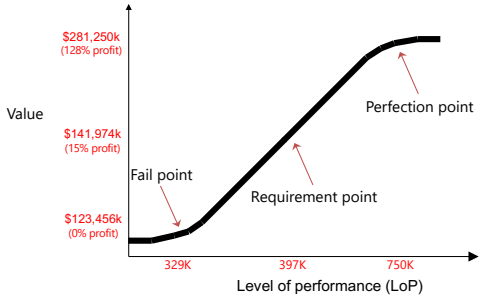

Figure 7. A concrete example using the number of supported customers for the e-commerce system.

When business value decreases with level of performance

Figures 2 through 7 show value increasing with level of performance. For many quality of service requirements value decreases instead. Examples would be response time and Mean Time To Repair (MTTR), less is better. In this case the graph is mirrored as shown in Figure 8. Figure 8 also shows response time in the e-commerce case study as a concrete example with numeric values on both axes derived using similar economic analysis.

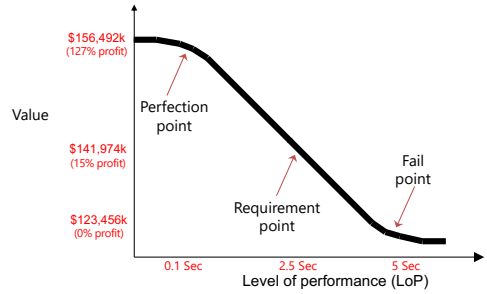

Figure 8. Value Can Decrease with Level of Performance: Response Time Example.

The Perfection Point, 0.1 seconds, is where the user becomes unable to perceive any delay so even shorter response times are no more valuable. The Fail Point, 5 seconds, is when the user becomes frustrated and loses attention, so they look for other ways to satisfy their needs than using this system.

The relationship between the Fail, Requirement, and Perfection Points can affect project decisions

It should be reasonable to assume that the Requirement point will be between the Perfection Point and the Fail Point. We now consider how the relationship between these three points can influence project behavior.

Figure 9 shows a Requirement Point very close to a Fail Point. What if, for some reason, the system cannot even meet that required level of performance? Not only do the stakeholders have every right to be upset because their expectation is not being met, but they also have every right to be very upset because value is close to its minimum. On the other hand, if it is possible to substantially exceed the required level of performance (assuming minimal cost to do so) then it makes sense to take advantage of it.

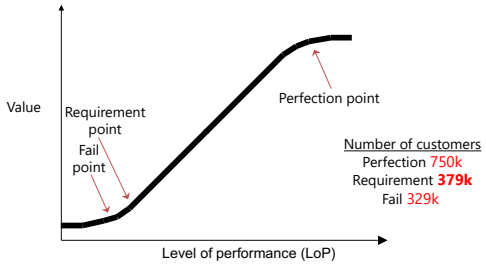

Figure 9. Requirement Point Close to Fail Point.

Figure 10 shows a Requirement Point very close to a Perfection Point. What if, for some reason, the system cannot meet that required level of performance? The stakeholders have a right to be upset because their expectation is not being met, however they should not be very upset because the system is delivering most of the value it could ever deliver in this dimension. On the other hand, if it is possible to substantially exceed the required level—even assuming minimal cost—it may not make sense to act.

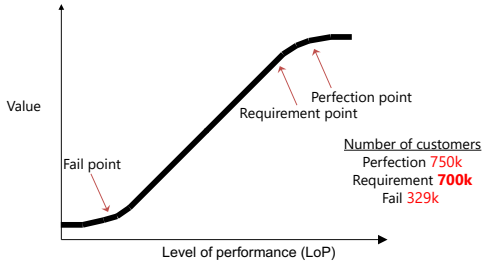

Figure 10. Requirement Point Close to Perfection Point.

Behavior should not be driven only by whether a quality of service requirement is being met. Instead, behavior should be driven by the level of performance achieved compared to the Fail Point and Perfection Point, which then begs the question,

"How will the team know how to behave unless they know what the Fail, Requirement, and Perfection Points are?"

Recommended specification of a quality of service requirement

Specification of a quality of service requirement should explicitly define Fail, Requirement, and Perfection Points[2]. For example, "the system shall have an acceptable response time" requirement can be minimally specified as shown in Table 1.

	Level of Performance
Fail	5 seconds
Requirement	2.5 seconds
Perfection	0.1 seconds

Table 1. Specifying a response time requirement.

As another example, "the system shall have acceptable reliability (Mean Time Between Failures, MTBF)" requirement can be specified as shown in Table 2.

	Level of Performance
Fail	1 day
Requirement	2 weeks
Perfection	1 month

Table 2. Specifying a reliability requirement.

It would be even better to include the value at each level of performance, as shown in Tables 3 and 4 for the response time and reliability requirements.

	Level of Performance	Value
Fail	5 seconds	$123,456k
Requirement	2.5 seconds	$141,974k
Perfection	0.1 seconds	$148,956k

[2] Gilb's Planguage [8] includes Survival, Goal, and Wish as similar concepts but without the economic justification described in this paper.

Table 3. Fully specifying a response time requirement.

	Level of Performance	Value
Fail	1 day	$123,456k
Requirement	2 weeks	$141,974k
Perfection	1 month	$164,700k

Table 4. Fully specifying a reliability requirement.

Considering cost to deliver

It was stated earlier that the cost of delivering each level of performance was being ignored to simplify discussion of basic dynamics, namely economically justifiable Fail, Requirement, and Perfection Points. We now consider cost to deliver at each level of performance to complete the analysis of individual quality of service requirements.

Typically cost to deliver will be a step function as shown in Figure 11. Steps in the cost function are driven by one or a combination of,

- additional effort (and thus, cost) to develop and maintain more optimized, and necessarily more complex, software;
- additional costs to acquire, operate, and maintain more capable hardware.

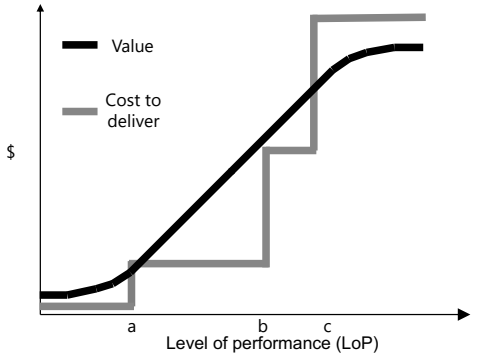

Figure 11. Cost to Deliver each Level of Performance is usually a step function.

The cost to deliver the levels of performance in Figure 11 could be as shown in Table 5. Cost to deliver should not be documented as part of the requirements, it is relevant to design not to requirements.

Step	Cost to deliver
Up to LoP_a	$122,500k
LoP_a to LoP_b	123,456
LoP_b to LoP_c	137,250
LoP_c to maximum	155,750

Table 5. Example costs to deliver.

The most cost-effective level of performance

The most cost-effective level of performance can be defined as that level of performance having the largest positive difference between business value at that level and cost of delivering it. This is shown in Figure 12 at LoP_b.

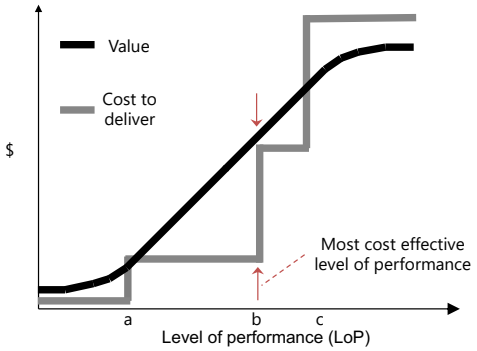

Figure 12. The Most Cost-Effective Level of Performance.

Figure 13 shows the same analysis in terms of net business value, i.e., subtracting cost to deliver from business value at each level of performance. The most cost-effective level of performance has the highest net value.

ECONOMICS OF MULTIPLE QUALITY OF SERVICE REQUIREMENTS

All previous discussion was in the context of an individual quality of service requirement. Each was analyzed independent of the others. We now consider multiple quality of service requirements at the same time. Two forms of analysis will be presented, Sensitivity Analysis and Overall System Optimization.

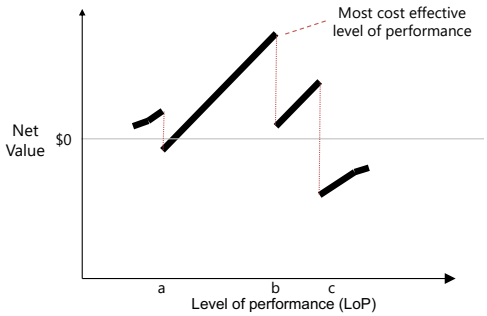

Figure 13. Net Value at each Level of Performance.

Sensitivity Analysis

"Sensitivity" refers to the relationship between a relative change in the level of performance and the corresponding effect on value. Does value change a little or a lot if the level of performance is changed by some amount? The more sensitive a quality of service requirement is, the more its value changes for a given percent change in level of performance. Quality of service requirements with higher sensitivity have greater economic leverage, and thus deserve paying closer attention to in development and maintenance.

Sensitivity analysis starts by holding all other requirements at their specified requirement point and varying only one. For that requirement, vary its level of performance between its Fail Point and Perfection Point recording each one. Tables 6 through 9 show how value changes as only one of the four quality of service requirements are varied while the others are held constant.

Supported customers	% change in supported customers	PW(i)	% change in PW(i)
337k	-10%	$126,194k	-11.1%
379k	0	141,974k	0
421k	10	157,753k	11.1
463k	20	173,533k	22.2
505k	30	189,312k	33.3
547k	40	205,091k	44.4
589k	50	220,871k	55.6
631k	60	236,650k	66.7
673k	70	252,430k	77.8
715k	80	268,209k	88.9

Table 6. Considering change in number of supported customers alone.

Number of products	% change in number of products	PW(i)	% change in PW(i)
875	-50%	$125,308k	-11.7%
1100	-40	128,641k	-9.4
1325	-30	131,974k	-7.0
1550	-20	135,307k	-4.7
1775	-10	138,641k	-2.4
2000	0	141,974k	0
2225	10	145,307k	2.4
2450	20	148,640k	4.7
2675	30	151,973k	7.0
2900	40	155,307k	9.4

Table 7. Considering change in number of supported products alone.

Response time	% change in response time	PW(i)	% change in PW(i)
0.5	-40%	$156,492k	10.2%
1.0	-30	152,862k	7.7
1.5	-20	149,233k	5.1
2.0	-10	145,603k	2.6
2.5	0	141,974k	0
3.0	10	138,344k	-2.6
3.5	20	134,715k	-5.1
4.0	30	131,085k	-7.7
4.5	40	127,456k	-10.2
5.0	50	123,826k	-12.7

Table 8. Considering change in response time alone.

Mean Time to Repair	% change in MTTR	PW(i)	% change in PW(i)
0.12	-80%	$227,304k	60.1%
0.48	-70	216,638k	52.6
0.77	-60	205,972k	45.1
1.06	-50	195,305k	37.6
1.34	-40	184,639k	30.1
1.64	-30	173,973k	22.5
1.92	-20	163,306k	15.0
2.21	-10	152,640k	7.5
2.50	0	141,974k	0
3.00	10	123,456k	-7.5

Table 9. Considering change in Mean Time To Repair (MTTR) alone.

When all quality of service requirements have been considered, plot the results on a graph. This is shown in Figure 14 for the for four quality of service requirements.

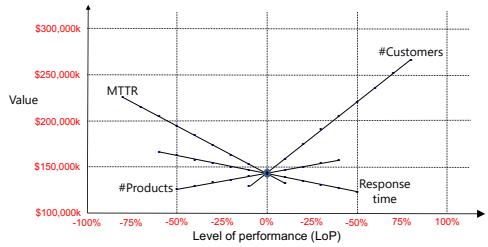

Figure 14. Example Sensitivity Analysis for the e-commerce case study.

Visually, the number of supported customers is far more sensitive than the number of supported products because it has a much higher slope. Computing the slope of each line can help when visual inspection is insufficient to distinguish. The slope of each sensitivity line is shown in Table 11 sorted in order of decreasing absolute value (i.e., from most sensitive to least sensitive),

Requirement	Sensitivity[3]
Supported customers	$1578
Mean Time To Repair	-1067
Response time	-363
Supported products	333

Table 10. Numeric sensitivity for each requirement in decreasing order of absolute value.

The level of performance for the number of customers requirement has more leverage on business value so the system should be designed in a way that favors this performance. The same can be said for Mean Time To Repair (MTTR), the system should be designed in a way to favor minimizing it.

Overall system optimization

One very important element of Systems Engineering is to find an optimum configuration that yields the highest value of overall system performance for the lowest overall system cost.

On the one hand it would seem simplest to just drive every quality of service requirement to its most cost-effective level of performance. Unfortunately, this is not guaranteed to yield overall optimum system performance because the optimum configuration for one quality of service requirement may negatively affect performance in another quality of service requirement. There are almost certainly competing forces, for example:

- actions that make the code more efficient necessarily make it less maintainable;
- actions that make the code more maintainable may make it less portable.

Sensitivity analysis can guide intelligent questioning about potential trade-offs. Stakeholders are more likely to be willing to trade lower than as-specified required performance in a less sensitive quality of service requirement if it leads to higher performance in a more sensitive one. In the eCommerce system, stakeholders would probably be willing to trade:

- lower performance in number of products if it leads to higher performance in number of customers;
- lower performance in response time if it leads to higher performance in Mean Time To Repair (MTTR).

Stakeholders should not be interested in trade-offs between response time and number of supported products because neither is particularly sensitive.

SUMMARY

Nonfunctional "quality of service" requirements are that subset of software product requirements specifying how well the software is expected to perform, such as response time, throughput, reliability, and accuracy. This kind of requirement is commonly poorly specified—if even specified at all. We proposed that this problem could be reduced by considering this kind of requirement from an economic perspective. This paper started by defining nonfunctional quality of service requirements and elaborating on that common problem. The remainder of this paper examined nonfunctional quality of service requirements from an economic perspective, both individually and collectively, showing how this perspective can reduce this common problem.

[3] Each 1% increase in LoP causes this much change in value.

REFERENCES

[1] *Guide to the Software Engineering Body of Knowledge v4.0*. IEEE, 2023. Available at www.swebok.org.

[2] ____, ISO/IEC 25010: "System and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," *International Standards Organization*, 2011.

[3] S. Tockey, *Return on Software: Maximizing the Return on Your Software Investment*, Boston, MA: Addison-Wesley, 2005.

[4] E. DeGarmo, W. Sullivan, and J. Bontadelli. *Engineering Economy, Seventeenth Edition*. Upper Saddle River, NJ: Prentice Hall, 2018.

[5] Ted G. Eschenbach, *Engineering Economy: Applying Theory to Practice, Third Edition*. New York, NY: Oxford University Press, 2010.

[6] Grant; Eugene L., Ireson, Grant W., and Leavenworth, Richard S. *Principles of Engineering Economy, Eighth Edition*. New York, NY: Wiley, 1990.

[7] G. J. Thuesen and W. J. Fabrycky. *Engineering Economy, Ninth Edition*. Englewood Cliffs, NJ: Prentice Hall, 2000.

[8] T. Gilb, *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage*, Oxford, UK: Elsevier Butterworth-Heinemann, 2005.

Steve Tockey is the Principal Consultant at Construx Software. During more than four decades in the software industry, he has worked as a programmer, analyst, designer and researcher for organizations that include Lawrence Livermore National Laboratory, The Boeing Company and Rockwell Collins, Inc. Steve has a Master's of Software Engineering from Seattle University and a B.A. in Computer Science from the University of California, Berkeley. Steve is the author of two books. His first book, Return on Software, is a guide for companies that want to maximize the return on their software investment. His second book, How to Engineer Software, shows how software can be developed and maintained under a true engineering discipline.

JianYang Zhang is an independent consultant specializing in software quality management. She graduated from Beijing Union University with a bachelor's degree in physics. She has 18 years of project delivery management experience in connected car, insurance, tax administration, and oil services. She has worked at Peking University, OOCL, Schlumberger, Atos@Bull, Alibaba, IBM, and Valtech Mobility in a variety of quality, test management, and build management roles.

ACKNOWLEGEMENTS

The authors would like to acknowledge the following individuals for their insightful review comments: Earl Beede (Construx Software),

Tom Gilb (Gilb International), Mike McKee (Seattle University), Meilir Page-Jones (Wayland Systems), Maria-Isabel Sanchez-Segura (Universidad Carlos III de Madrid).