UNIVERSITY OF GOTHENBURG

Towards enhancing perceived performance through adoption of proposed benchmarking techniques

VJACESLAVS KLIMOVS
SERGEJS PISARENKO

Bachelor thesis in Software Engineering

Report No. 2009-029
ISSN: 1651-4769

University of Gothenburg
Department of Applied Information Technology
Göteborg, Sweden, May 2009

Abstract

> "What the eyes see and the ears hear, the mind believes."
>
> Harry Houdini, 1920

Despite the vast abundance of embedded/mobile devices being released every day only a minority of those gain enough momentum to secure mainstream adoption, let alone market domination. User experience and user perceived performance are taking an increasingly substantial stake in a successful product launch. In this paper, we present the results from a study of user perceived performance. Further, in this design oriented study we have applied benchmarking body of knowledge to develop a representative benchmark that assesses the end-user experience. We conclude that implementation of proposed techniques allows for repeatable, structured and systematic approach in user perceived performance testing, thus, potentially creating cost savings for the company and enhancing user perceived performance, which in turn leads to strengthening of the market position.

Keywords: perceived performance, benchmarking, embedded, user interface, platform sustainability.

1 Introduction

With a growing emergence of high speed wireless Internet, access technologies such as Bluetooth[1], Wi-Fi[2], 3G[3], Worldwide Interoperability for Microwave Access (WiMAX)[4], Long Term Evolution (LTE)[5] and a significant drop in prices for electronic components [Kleinrock, 2001], embedded/mobile devices are becoming essential common artifacts of the modern human society and an inseparable part of our everyday experience [Ling, 2004]. With radical improvements in reducing the sizes of form factors and drastic enhancements of microprocessor technology it is no longer unusual for a person to own and operate a multitude of embedded/mobile devices[6] [Lyytinen & Yoo, 2002]. Unseen by the end user, this ubiquitous computing is brought by a throng of companies fighting a fierce competition in hardware and software markets. While the competition itself is beneficial for the user, it puts pressure on vendors to come up with new concepts, new features, provide better performance and better user experience.

However, only a minority of all of the produced embedded/mobile devices gain enough momentum to secure mainstream adoption, let alone market domination. User experience and user perceived performance are taking an increasingly substantial stake in a successful product launch, yet many of the enterprises have underestimated its significance and consequentially lost their market share [Russinovich, 2008]. To complicate matters even further, existence of a variety of hardware and software platforms, for which the embedded/mobile device is built and deployed, inevitably creates the need to effectively measure how well an embedded/mobile device in development behaves [McKenna, 2000]. Effective measurement is paramount to gauge performance fluctuations over product evolution as well as to determine operational differences across platforms [Kevin, 2006].

Responding to the described challenges, this study has been set out to disentangle an existing quagmire of evaluating user perceived performance within the domain of embedded/mobile devices using applied benchmarking techniques while maintaining a strong focus on perceived performance. Opera Software ASA develops a variety of desktop and embedded/mobile solutions that enable content-rich web access on a wide spectrum of operating systems, platforms and consumer devices. As a highly competitive provider the company is determined to foster innovation on an ongoing basis to deliver the best

[1] Bluetooth™is an open wireless protocol for exchanging data over short distances from fixed and mobile devices.

[2] Also known as 802.11, is a set of standards carrying out wireless local area network (WLAN) computer communication in the 2.4, 3.6 and 5 GHz frequency bands.

[3] 3G refers to the third generation of telecommunication hardware standards for mobile networking.

[4] WiMAX is a telecommunications technology that provides wireless transmission of data using a variety of transmission modes.

[5] LTE is the next major revision of mobile radio communications superceding 3G.

[6] Further in the article any references to devices are interchangeable with the term '*embedded/mobile*'.

user experience for its products. Currently, developers of mobile versions of the Opera browser are required to carry out an enormous endeavor running their builds on a diverse collection of devices. To ease their development process and potentially lower associated costs, we have tightly collaborated with Opera to consider opportunities of simulating target devices performance-wise. In an effort to contribute to the much larger research scope of simulator development we have studied perceived performance measurement. To achieve the goal of collecting performance indicators of an underlying platform we have developed a representative device benchmark in a form of a compile-time plug-in for the Opera web browser.

In this paper we research the subject of user perceived performance, apply benchmarking body of theory, while acknowledging limitations and known approaches. A design science approach is taken [Kasanen & Lukka & Siitonen, 1993] to develop a representative solution for a specific domain with a particular emphasis on assessing the end-user experience. Furthermore, the study was set out to discover methods that would enable effective measurement collection and further representation of those in an unambiguous way. Particularly, an emphasis is put on the translation of user perceived performance into measurable performance indicators. As a twist to our solution, we discuss possible relation of sustainable platforms to the issue of perceived performance. This paper might appeal to researchers and practitioners working in the field of embedded/mobile software development as well as those interested in performance benchmarking and user interfaces (UI). In the future, our work could possibly be used to implement a set of enhancements to a device simulator or an emulator that could represent runtime characteristics of various devices.

The remainder of the paper is structured as follows: Section 2 introduces the concept of user perceived performance and elaborates on the current standards in computer performance measurement, their classification and also limitations of existing performance benchmarking solutions. In Section 3 we present the research method we have used to approach the problem. Our findings are uncovered in the Section 4, where we explain and present the developed artifact. Section 5 proceeds with discussing possible limitations of our solution and speculates on possible relations of long-term supported platforms (commonly referred to as 'sustainable platforms') to the issue of perceived performance. Finally, the paper concludes with our resulting thoughts and summary[7].

2 Theory

In this part of the paper literature review is presented, concepts and terms that central to our research are introduced.

2.1 Perceived performance

2.1.1 Definition and positioning

Merriam-Webster dictionary defines performance of the machine as the ability to perform or the manner in which a machine performs. When a computer system is thought of, two kinds of performance are distinguished. The objective machine performance or computational performance comes first, while the user perceived performance (UPP)[8] constitutes a completely distinguished dimension [TMurgent Technologies, 2003].

Machine performance is one of the ultimate engineering goals and is addressed by many, if not all, Computer Science branches. The field is characterized by an abundance of hardware and software engineers exercising a diverse universe of approaches and techniques to introduce improvements and to address existing shortcomings and bottlenecks. These include, but are not limited to [Dunlavey, 1993, TMurgent Technologies, 2003]:

- improving hardware to get less latency, more bandwidth, better response times
- improving software, so that it uses more efficient algorithms and data structures

[7] All errors in this paper are that of the authors.

[8] Terms "UPP", "user perceived performance" and "perceived performance" are used interchangeably in this paper.

It is the domain of machine performance, where most quantitative performance measurement techniques exist [Hockney, 1992]. Machine performance is the only type of performance where it is possible to do measurements in low-level units such as MHz, Mbit and relatively high-level units such as cost per transaction [Tognazzini, 2001]. Conversely, user perceived performance is concerned with the appearance of performance, that is, the feeling of how quickly a software feature seems to perform its task [Seow, 2008].

2.1.2 Key trends

According to Moore's law, the number of transistors that can be placed inexpensively on an integrated circuit has increased exponentially, doubling approximately every two years [Moore, 1965]. Almost every measure of the capabilities of digital electronic devices is strongly linked to Moore's law. Examples include but are not limited to processing speed, memory capacity and memory bandwidth [Myhrvold, 2006]. In other words, parameters we deem as a machine performance increase exponentially. User perceived performance grows as well, however, even a casual user of electronics will notice a slight discrepancy here. Due to increasing complexity of background functions that need to be executed after each user input, growth of user perceived performance is not that rapid [Tognazzini, 2001]. Interestingly, leaving the case when it actually grows alone[9], there are two distinct cases when user perceived performances degrades. First, while moving to a next generation product, a machine performance might have actually deteriorated, due to complication in software design or architecture, new demanding features or improper use of new hardware capabilities (or even abuse of those). This inevitably leads to decrease in user perceived performance. Similarly, if the perceived performance of the next generation product is unchanged or only marginally better, the user might perceive it as being worse, since it does not live up to the expectations of better performance. This is supported by Maister's First Law of Service, which states that user satisfaction is a function of disconfirmation [Maister, 1984]. In the second case, the newer version of the product improves the machine performance while actually degrading perceived performance. To casual users this appears as puzzling and inconclusive as the very least, causing major irritation and discomfort in most use cases.

2.1.3 Case in point

To illustrate the phenomena, let us consider file copy performance of two modern operating systems of the Windows[10] family, namely Microsoft Windows Vista and Windows XP. Users of Windows Vista complain that file copy performance of this operating system is worse than that of Windows XP, despite objective measurements, that show that actual file copy performance is better in most cases in Windows Vista than in Windows XP [Bott, 2008]. Windows XP and Windows Vista use different algorithms for copying files. Windows XP algorithm uses cached I/O, which lets Explorer[11] finish writing destination files to memory and dismiss the copy dialog long before the write-behind thread has actually committed the data to disk. With Vista's non-cached implementation, Explorer is forced to wait for each write operation to complete before issuing more, and ultimately for all copied data to be on disk before indicating a copy's completion [Russinovich, 2008]. Seemingly less advanced, that actually addresses some shortcomings of previous design, and in some cases, such as copying files over high-latency high-bandwidth links[12] brings tremendous improvements [Russinovich, 2008].

In addition to that, Vista's Explorer waits 12 seconds before making an estimate of the copy's duration as the estimation algorithm is sensitive to fluctuations in the copy speed [Russinovich, 2008]. Indeed, there is a theoretical background to that seemingly paradox situation. During the copy process the user is really concerned only about what is seen on the screen, the copy dialog being the only visible artifact during the process. Most of modern operating

[9] We are not that hypercritical.

[10] Windows, Windows Vista, Windows XP are registered trademarks of Microsoft Corporation.

[11] Explorer is the shipped file manager used in Microsoft Windows operating system.

[12] Most of the Wide Area Network (WAN) links adhere to that characteristics.

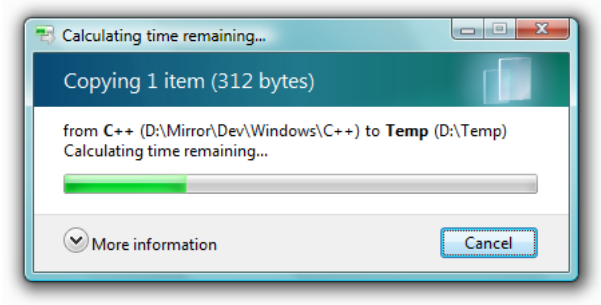

Figure 1: Windows Vista prolonged estimation process

systems provide a progress bar for process progress visibility. The file copy progress bar demonstrates the elapsed and remaining amounts of the operation. It is expected that the progress shown by a progress bar increases over time and it is natural for users to assume that a task requires time to reach a certain progress. [Harrison et al, 2007] compared user reactions to eight progress behavior functions. Although all of the progress bars took exactly the same absolute amount of time in the test, two characteristics made users think the process was faster:

- progress bars that moved smoothly towards completion
- progress bars that increased in speed towards the end

It is evident that Vista's copy dialog design violates these principles. Explorer waits 12 seconds before providing a copy duration estimate, which certainly provides no sense of smooth progress and the copy dialog does not disappear until the data is committed the data to disk, which means the copy is slowest at the end. Both of these behaviors exacerbate user frustration with slower copies [Russinovich, 2008, Maister, 1984]. All in all, despite all the algorithmic improvements, in spite of the superior file copy benchmark results, Vista's perceived file copy performance is worse than Windows XP.

2.1.4 Inherent contradictions

There is a natural conflict between obtaining ideal user perceived performance and inclination of engineers to optimize computational performance of the system [Tognazzini, 2001]. Consider a networked client desktop application which lets user operate on certain data. In the application database, there is abundance of domain data. In fact there is so much data, that it does not fit into one semantic screen[13] . In other words, the user might have a need to see different portions of data during the course of the task. To achieve ideal user perceived performance in that case, all data would need to be fetched from a remote server at once, so that when the user switches semantic screen, there is no delay in actually displaying that data. On the contrary, to achieve ideal machine performance, bandwidth and computational resources would rather be preserved, therefore no data would be loaded, except for when demanded.

2.1.5 Psychological perspective

The argument for the importance of perceived time is not new to areas outside benchmarking [Seow, 2008]. Within psychology, the highly researched concept of flow builds on this element. In extensive studies of different cultures, different professions, and over different ages, [Csikszentmihalyi, 1990] argues that experiencing so called "flow" involves a number of interconnected, but unique, elements, one of which involves losing track of time. The flow is a positive feeling of energized focus, full involvement, and success in the process of the activity. Therefore, design which strives for optimum user experience should not focus solely on designing widgets, for example, progress bars, in an optimum way. Instead, the design should look at the user experience as a whole, and place particular emphasis on overcoming areas where users may otherwise risk losing immersion [Seow, 2008]. This closely relates to the need for focus and loss of self-consciousness that [Csikszentmihalyi, 1990] also lists as important characteristics for flow experience.

[13] Not a physical screen, but rather screen as an simultaneously seen informative area. It has become common nowadays to have several physical screens.

2.2 Benchmarking

To approach the issue of perceived performance, investigation into existing benchmark research and the overall state of computer benchmarking is given. Below, the current body of knowledge on benchmarking is presented along with the core concepts and methods of measuring performance in desktop, workstation, server, mainframe computers and embedded/mobile devices. The presented analysis is then used to further support our findings.

Inevitably, benchmarking is of great relevance to this study given the quantitative measurement approach to the perceived performance conundrum. Benchmarking as a concept exists since the era of mainframes and, thus, is quite old if judged by software engineering standards. While the taxonomy of existing approaches for performance measurements has been widely expanded since then, the basic idea remains surprisingly simple. It all boils down to a concept of measuring either time over a predefined fixed task or measuring the amount of completed work over a fixed period of time [Gustafson et al, 1990]. Usually, the classification of benchmarks is done based on their internal focus, i.e., which component of the system the benchmark utilizes. It could be a set of integer, floating-point, I/O-intensive or network-centered operations. Simplified abstractions have demonstrated that synthetic benchmarks commonly fail to represent a realistic workload and a performance profile because of their narrow focus and, thus, are often effectively limited in its potential use as an accurate performance comparison tool. The issue is not universally solvable and relies on specifics of the target as well as expectations of the measured results. For example, a representative benchmark exercises tasks which are close, or even identical, to the ones used in real world situations [McKenna, 2000, Paul & Somers, 2008, Guthaus et al, 2001].

2.2.1 Traditional benchmarking

Performance benchmarks exist since the era of mainframes and are employed to calculate metrics, to compare and asses operational behavior of particular systems or their parts, including, but not limited to central processing unit (CPU), input/output (I/O), networking throughput. They also constitute a critical part in designing future products to overcome existing limitations and bottlenecks when performance-based design approach is used [Guthaus et al, 2001]. The first and the most important factor in designing a benchmark is its ability to characterize the specific domain it is covering, i.e., to predict the performance of an unknown system on a known, or at least well-defined, task or workload [Sill, 1996]. Thus, it is advantageous to study presently-used indicators and methodologies in their application area with a goal of developing a highly-relevant set of performance tests for the particular domain under research.

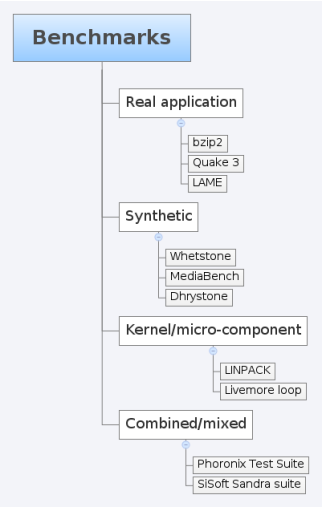

Figure 2: Benchmarking taxonomy

2.2.2 Classification by scope

Benchmarks could be classified by the nature of their internals and their target. Below we present our own list of benchmarks, which is based on concepts proposed by other researchers, and selected for their relevance to this research. We have categorized them for the sake of clarity. While the postulated types cover the concepts of the more granular category characterization, it would provide limited benefit to our

research as well as bring unnecessary complications and term conflicts to include each and every existing benchmarking category.

Real application — uses real existing applications on a predefined input data set to measure how quick the tested system completes typical usage scenarios. Usually, in such setups data compression tools, media file encoders/decoders, compilers and computers games are used. An associated advantage of such benchmarks is their accuracy and reflection of the objective realistic performance. Examples include compressors (e.g. bzip2, lzma, 7-zip), compilers (e.g. GNU Compiler Collection), media encoders/decoders (e.g. lame, ffmpeg) and a variety of recent gaming titles. Particular attention is given to open-source alternatives because those could be obtained for free for a multitude of platforms, though still compiled from the same sources, which adds credibility to the representativeness of the obtained results.

Kernel — in contrast to the real-world applications, these benchmarks focus on a very specific subsystem behavior. As such, these aim to strip away unnecessary program activity while still providing realism. Advantages of such approaches lies in the ability to compare efficiency of very specific operations across different architectures. For example, comparison of integer operations' performance between x86 and x86-64 bit CPUs would be possible. LINPACK (the kernel benchmark developed from a package of linear algebra routine "LINPACK") and Livermore loops (C loop code from Livermore labs) are examples of such benchmarks. Micro-benchmarks (also referred to as "Component benchmarks") are more extreme instances of kernel benchmarks which focus on specific instruction sets and operations [Guthaus et al, 2001, Lee & Mangione-Smith & Potkonjak, 1997].

Synthetic tests — developed and standardized to replicate typical workload, however, are subject to common abuse because of vendor optimizations. The problem lies in the fact that vendors tend to introduce changes which result in benchmarks giving significantly better scores to slightly better hardware. Most of such benchmarks target specific areas of computation, for instance: integer computations, floating-point intensive operations, media applications, input-output and other defined tasks [Curnow & Wichmann, 1976]. In addition, the majority of standard test suites (which include Whetstone, CPU2, SPEC and others) have heavily influenced design of microprocessors and microarchitectures [Guthaus et al, 2001].

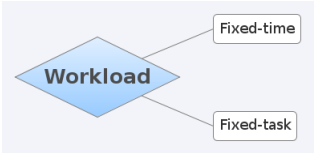

Figure 3: Classification by measurement

Mixed approach — combines several different tests to get the most realistic performance measure. Such tools typically constitute benchmarking suites and execute various tests in a batch, e.g. the suite runs synthetic tests first and then checks how quick the target devices handles encoding of a predefined source raw media data followed by a compression of fixed-size pseudo-random data. Optionally, some of the suites offer geometric representation of the results [Dujmovic, 1999].

2.2.3 Classification by measurement

Gustafson et al have proposed to characterize benchmarks in one of the two categories by the way measurement is approached:

Fixed problem-sized benchmark — the system under test is presented with a fixed defined task that needs to be solved. Execution time is then measured and used as a performance indicator. However, major increases in computational power over the last decade make it very difficult to measure performance without giving the scaling issue enough attention [Gustafson et al, 1990]. Quick evolution of processors and architectures obsoletes such tasks in terms of time and memory requirements and ultimately renders such benchmarks unrepresentative of realistic use.

Fixed-time benchmark — the system is given a predefined amount of time to solve a scalable task. It

is argued that such an approach offers a more realistic practical and meaningful estimation, includes I/O times and scales to vastly different architectures and programming environments [Gustafson et al, 1990].

2.2.4 Acknowledged challenges

The domain of benchmarking has a vast array of certain limitations that need to be taken care of, if a representative and a useful product is being developed:

One of the challenges a benchmark design should overcome is its ability to reflect characteristic system behavior, i.e., the test should be executing something that is very similar, if not identical, to what the system normally does under normal operational conditions [McKenna, 2000].

Benchmarks often fail to correlate gathered results with any realistic workloads. This is especially the case when a specific benchmark is used in domain for which it's not intended, for example, Drystone has been widely used for measuring performance of embedded systems while it has even been abandoned in its original general-purpose system target domain [Lee & Mangione-Smith & Potkonjak, 1997]. In such cases collected results typically have nothing but a value of a ballpark figure. By definition, benchmarks are tools for performance evaluation which associate the assessment rankings with user perceived operating attributes of the system being analyzed. Without this association, benchmarks essentially become meaningless and the benchmark methodology used can be judged by the strength of this correlation with user experience [McKenna, 2000].

It is inherently difficult, if not impossible, to abstract a single output number that is representative of any real usage profiles and interactions, especially in case a variety of multidimensional factors influences the result, e.g. energy efficiency, compatibility, performance [McKenna, 2000].

Most of the benchmarks target very narrow specific domains, e.g. particular architectures, subsystems, CPUs. The SPEC benchmarks, for example, were specifically developed to assist in commercial evaluation and marketing of desktop computing systems [Lee & Mangione-Smith & Potkonjak, 1997]. Thus, results gathered from such benchmarks present little value in assessing end-user's perceived performance.

Hardware and software vendors tend to optimize their products in order to achieve vastly better results in the standard de facto synthetic benchmarks without any visible or significant improvements in typical usage scenarios, which in itself confirms the limited usefulness of such testing suites. For example, optimizing options on today's compilers can drastically affect the results of benchmark tests so vendors take an advantage by publishing a range of test results using different optimization levels [Price, 1989].

Existing multipurpose benchmarks fall short on characterizing mobile computers because of their peak-performance orientation and are inadequate to address requirements of the new mobile paradigm which should be measured with attributes and usage patterns which are important for mobile users [McKenna, 2000]. In addition, modern-day usage of computers fundamentally differs from decade old specification styles and benchmarks which do not take into account the new paradigm of mixed content browsing and new computer usage patterns [Paul & Somers, 2008].

Often, hardware testing is characterized by qualitative statements that are not quantifiable. Such disregards, done by benchmarking institutions, do not follow basic scientific method and, as a consequence, mislead the reader. This includes, but is not limited to: small sample size, lack of variable control, and the limited repeatability of results [Kevin, 2006]. Ultimately, absence of objective scientific verification reduces the confidence in benchmark results, yields too frequent and increasingly more difficult upgrades of benchmark suites, and causes unjustified global increase of the cost of benchmarking [Dujmovic, 1999].

The conclusions drawn from a benchmark study of computer performance depend not only on the basic timing results obtained, but also on the way they are interpreted and converted into performance figures. The choice of the performance metric, may itself influence the conclusions [Hockney, 1992]. In other words, "*producing and interpreting benchmark data crosses into the realm between art and science*" [Price, 1989].

Furthermore, issues identified in relation to the

user perceived performance pose additional challenges to be solved in the implementation of a representative benchmarking solution. Specifically, this implies extension of the above mentioned needs to carefully design the workload as well as to choose representative indicators. That in itself comprises a task to discover meaningful correlations between end-user tasks that constitute the subjective user experience (e.g. scrolling by utilizing touch interface) and actual software functions that implement the functionality (e.g. functions that re-draw the screen area).

Without careful assessment of the above mentioned issues and mitigation of associated risks a benchmark under design is threatened to provide very little visibility into the real operational characteristics that are paramount in definition of end-user experience.

3 Research method

This section introduces the proof of concept study and the reasoning behind a decision to chose it as a research method for the paper. Also, the section provides a detailed description about the empirical context of the paper. Finally, it postulates the applied data collection methodologies.

3.1 Research setting

The research has been carried out continuously over the course of 10 weeks at Opera Software ASA in Gothenburg. Opera started in 1994 as a research project inside Norway's largest telecom company, Telenor. Today, Opera Software develops the Opera Web browser which runs on a wide variety of operating systems, embedded Internet products and gaming consoles. Opera's vision is to deliver the best Internet experience on any device. Opera Software was a major stakeholder in this research, as the company was interested in potential applicability of the artifacts produced during the course of this work.

Opera's product line of browsers (including Opera Mini[14] and Opera Mobile) is proprietary and closed-source software. Coupled with the fact that some technologies used in the product line are unique and currently have no effective competition on the market, required the authors to sign a non-disclosure agreement (NDA) before being given access to the source code. Naturally, the ability to work on the existing source code, access to the intellectual property of the company and productive working conditions all were important contributions of Opera Software to this research. However, it is worth to note that nothing substantial relevant to the research question has been lost due to NDA.

[14]Opera, Opera Mini and Opera Mobile are registered trademarks of Opera Software ASA.

3.2 Research limitations

The scope of this paper was limited to exclusively deal with the mobile versions of the Opera browser, i.e., Opera Mini and Opera Mobile. Opera Mobile runs on a variety of platforms including Symbian and Windows Mobile. Opera Mini, however, employs Java Micro Edition (Java ME) as its virtual machine so it works on every platform Java ME is available for. It is worth to mention, that despite subtle differences in the way Opera Mini and Opera Mobile are built, performance benchmarking of underlying platforms is equally relevant to both of them.

3.3 Research positioning

The paper is positioned as a design oriented research work. In software engineering related fields, design oriented research approaches has been lately recognized as a valuable way of conducting scientific research [Hevner et al, 2004, Gasser & Majchrzak & Markus, 2002]. Besides the fact that some of the knowledge just might not be accessible without a preliminary designed artifact, there are other advantages to it [Kuechler & Vaishnavi, 2007]. Namely, the researcher who employs artificial constructs has an opportunity to influence the time and the conditions of the research rather than being limited by the research object itself [Simon, 1996]. Additionally, one may mention the fact that the designed artifact might bring additional unanticipated value [March & Smith, 1995].

On the other hand, the chosen research approach also exhibits a number of shortcomings. First, the research artifacts might be imperfect or inaccurate because of their artificial nature which, in turn, could lead to erroneous conclusions [March & Smith, 1995]. This problem is relevant to this research. However, the presence of a review body (i.e., Opera Software engineers) had somewhat mitigated the issue.

The necessity to build an artifact presents a problem in itself since that could turn out to be costly, dangerous, or disadvantageous altogether [Simon, 1996]. In the case of this paper, this does not apply at all, since the produced artifact (in the form of source code) and methodology it illustrates, constitutes the the main contribution of this research.

Finally, an attempt to answer a research question by means of an artifact one might be consciously or subconsciously tempted to develop the artifact biased towards an answer one would expect or like to find out [Kuechler & Vaishnavi, 2007]. Design oriented research imposes a danger of predisposition towards the research [March & Smith, 1995]. To mitigate this potential flaw in this study, data produced by the developed software, was empirically assessed to heuristically detect any inconsistencies in the benchmarking results.

The study was set out to create a solution for a known real-world problem. To some degree a contribution to the theory of the discipline has been made. According to [Kasanen & Lukka & Siitonen, 1993], the main attribute of constructive research methods is the concept of constructions. Since software or information system design fall under the definition of constructions it is evident that this research lies within the constructive research method. Important characteristic of constructions is that they are developed and not discovered, the later being a common constituent of other research methods. In addition, practical consequences as a meaning of truth were considered in this study, therefore, this research lies within the pragmatic school of thought.

3.4 Related research

It was of utter importance for this study to compile a list of suitable performance indicators. This task was systematically approached by performing a literature review and examining set of existing benchmarking tools. Appropriate keywords were identified to facilitate search and investigation of existing knowledge corpus. Those included, but were not limited to: embedded, mobile, performance, perceived, time, measurement, benchmarking. To structure work process, analysis and classification of found material was performed. Literature review aided in identifying key sources and big names in the field, and also revealed some major issues and debates about the topic. It also helped to define epistemological and ontological grounds of this study, in other words nature, scope and structure of the knowledge in the field. References used by many authors, including but not limited to [Seow, 2008], [Russinovich, 2008], [Tognazzini, 2001], were found to be tremendously useful for this study too. A set of existing tools for benchmarking embedded/mobile devices constituted an alternative input towards the more practical grounds of the paper. Literature review and the findings from examining existing benchmarking software are thoroughly covered in section 2.2 of this document.

3.5 Approach to implementation

Development of the compile time plug-in for a web browser constituted a substantial part of this work. Therefore, it has been vital to use an appropriate and efficient software process. An improper use of a software process, or lack thereof, could have resulted in the feature-incomplete product and possibly a failure of this study. For both of the authors the problem domain has been rather new. The definition of the main research question has been open-ended and initially not very well understood.

It was required to come up with a set of technical specifications and their later implementation. Prior to the commence of the research, necessary algorithms and data structures were not defined, though language of implementation has been introduced. The actual software, for which the plug-in has been developed, was easily extendable and conveniently explorable. Due to mentioned constraints, as well as after consulting existing body of knowl-

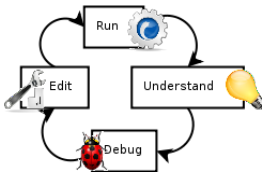

Figure 4: RUDE cycle illustration

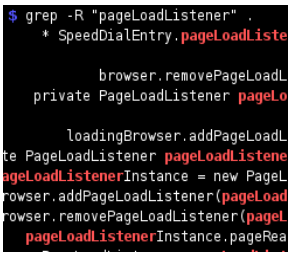

Figure 5: grep GNU regular expression matcher

edge, a exploratory software development model was deemed as a most appropriate one [Trenouth, 1991]. Use of extremely short Run-Understand-Debug-Edit (RUDE) cycles allowed for more flexibility, in terms of ability to quickly change the concepts to reflect newly gained knowledge [Partridge, 1991]. Additionally, the amount of changes introduced in each cycle was controlled, so that the software would remain continuously executable and meaningfully explainable. These undertakings made the software development relatively efficient.

For a similar reason as the software process, appropriate software design has been equally important for this work. However, achieving consistency in that area has been less of an issue as the design of the software for which plug-in was constructed was to be adhered. A simplistic multilayer approach had been chosen due to the inherent flexibility provided by such design and by the fact that developed software is a compile time plug-in, which is required to integrate seamlessly with the rest of the system. In that design, the software would pose as a thin layer between existing UI code and existing backend code, effectively intercepting calls to lower level to allow measurement of time where appropriate.

3.6 Tools

To systematically approach the problem of constructing the software as well as to get better understanding of the inner-workings of the domain application several tools were employed. Below, a list of applications that have assisted us in this endeavor is presented:

3.6.1 Development tools

As an aid in the implementation of the benchmarking plug-in a set of development instruments has been exercised. The tools contributed to the research process by narrowing original research scope and by concentrating research efforts at fundamental areas first.

Profiler — to collect runtime statistics and discover performance bottlenecks the in-house profiler, developed specifically for the target application, was employed. Through the use of the profiler distinct critical parts of the browser were analyzed, while typical tasks have been initiated by the user (e.g. opening a web page, scrolling in different directions, zooming, etc).

grep — once key methods and functions have been identified a common UNIX tool *grep* has been executed to localize and specify the area where necessary modifications are required to be introduced. Then, identified source code files were targeted for further analysis.

Eclipse — the Eclipse Integrated Development Environment (IDE) has been mandated by the project for the reason that the development team provides an addition to the IDE which make the code base substantially easier to work with.

3.6.2 Productivity tools

In order to organize ideas, concepts and knowledge several collaboration tools have been used, specifically:

Xmind — an open-source mindmapping graphical tool which allows to build diagrams to structure identified concepts, ideas, words through visualizing

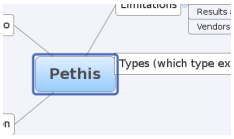

Figure 6: Xmind mindmapping tool

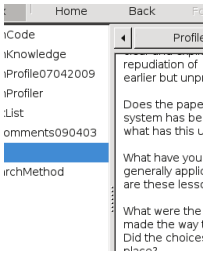

Figure 7: Zim knowledge management tool

those around central topics.

Zim — an open-source desktop knowledge management software for maintaining interrelated information in a wiki format. The tool has been used to store structured texts, processed gathered data, research logs, writing aids, check lists and other related information.

3.7 Collected results

While the focus of other research methods is set on collecting empirical data, it is a primary property of constructive research to think of a practical applicability of the construction as an empirical data, thus implying its validity. Therefore, it is important to define a fail-success continuum for a project. It is considered that the perfect result for a project exercising constructive research method is that a newly developed construction solves a real-world problem, which, in turn, forms a challenge of defining what a solution is. It is considered that a result that is satisfactory from the viewpoint of all stakeholders of the project indeed is a success [Kasanen & Lukka & Siitonen, 1993]. To add more validity to the data in the form of construction, the study was set out to verify it's design principles. This has been done by heuristically evaluating results obtained from running the software on a variety of hardware. When doing so, it is expected from inherently slow hardware to demonstrate lower results than a more powerful next generation device.

4 Software construction

In this part of the paper a clarification to the carried out research is presented, additional concepts and terms are introduced, identified findings are postulated, compared and combined to support the discussion in the upcoming section.

Opera Software develops web-browsing products for wide device ranges that feature numerous manufacturers and models. Such a span creates a burden on the software engineers who need to ensure that the application behaves exactly the same and that the performance does not degrade on any of those devices. In contrast to certain companies which are in total control of both aspects of their products, i.e. software and hardware (for example, Apple Computer Inc., that is often identified as an industry leader in user experience, with its iPhone smartphone), Opera is constrained by the software boundaries. In other words, the company is expected to adapt their software to guarantee that it shall work with the many devices and basically circumvent any existing problems or limitations in the hardware. Nevertheless, user perceived performance still constitutes a major part of the company's goal to deliver the best user experience which brings the necessity to assess how well a particular product or its version behaves on the target embedded/mobile device. In section 2.1 we have looked at the importance of UPP for the success of a consumer product, in this part of the paper we shall introduce and describe a solution that contributes to solving the described challenge.

To maintain insight into operational characteristics of the software, such as user experience, it is necessary to at least execute software on the device in question. However, running the software on each and every device that exists on the market would be a tedious, time and resource consuming process. Given that there is a possibility to run the application in a simulated environment it would be much

simpler and efficient to enhance an existing simulator to mimic the behavior of various devices. That way developers would have a possibility to assess how well a particular build of the software is handled on a particular device. Before we proceed, it is beneficial to review existing device simulators. As we have limited ourselves to two versions of the Opera Browser — Opera Mobile and Opera Mini, only the platforms which they run on are of interest to the research. Below, a list of the more common platforms for which the specified products are built is presented:

- Windows Mobile
- Java ME
- Symbian (also known as S60)

Each of the listed platforms have been provided with simulators by the original platform providers. Additionally, third-party simulators are available for some of the platforms, e.g. Microemulator[15] for the Java platform. However, as of 2009 none of the listed solutions provide any means to simulate performance aspects. To compensate for the lacking feature an implementation of a mechanism that would limit the computational power for the running application is needed. To be able to simulate existing devices it is crucial to have data that accurately represents capabilities of those. As part of solving a more complex issue, our approach is to build a benchmark which is capable of producing a so-called "performance profile" that could be used later in the simulator to imitate behavior of the target device. Although deep internals or construction specifics of enhancing a simulator is out of scope for this research, a rudimentary analysis is still essential for this undertaking to be useful. Specifically, a fundamental understanding of its future design is required for the sake of functionality of produced profiles.

One approach to solve the simulation problem is to introduce intentional delays in the execution of crucial resource-intensive operations, e.g. certain system calls or program functions that wrap around those system calls. Relating back to the problem of UPP, part of the problem is to discover correlations between end-user tasks that constitute the subjective user experience (e.g. scrolling by utilizing the touch interface) and actual software functions that implement the functionality (e.g. functions that re-draw the screen area) that would be subjected to mentioned delays. It is important to notice that such an analysis is bound to specifics of a particular application under question and is possibly a limitation of the approach, however, it is exactly the case that the original research problem is tied to such specifics.

[15]MicroEmulator is a pure Java implementation of Java ME in Java Standard Edition (Java SE), http://www.microemu.org/.

4.1 PETHIS benchmark

To produce the previously mentioned "performance profiles" a compile-time plug-in PETHIS for Opera's browsers has been developed. At a high level of abstraction the aggregation of the plug-in with the browser itself constitutes a benchmark of "real application" type, according to the classification in section 2.2.2. Basically, an existing application, i.e., Opera Browser (with certain modifications), is used to measure the performance of a certain device.

In order to achieve a realistic workload which is representative of a real end-user task a concept of scenarios has been introduced. In this context, scenario refers to a typical action or a collection of actions executed by the user which is autonomously performed by PETHIS. For example, user types in a Uniform Resource Locator (URL), loads a page and then scrolls it in different directions. While a scenario is executed PETHIS collects run-time performance data, i.e., how long it takes to perform a particular method, from several so-called probes. Probes are implemented as intercepts of underlying functions and are implemented in a special programming language construct. The construct allows us to transparently insert code along the existing system functions and measure execution time without significant overall changes to the application. The results are then passed to a suitable data-structure and then sent through Hypertext Transfer Protocol (HTTP) to a results server.

PETHIS is built as a collection of classes, structured into 4 major packages: *ui*, *core*, *probes* and

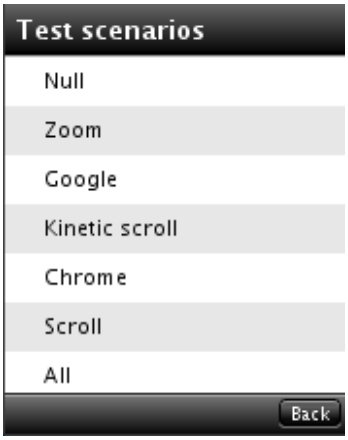

Figure 8: PETHIS scenario selection menu

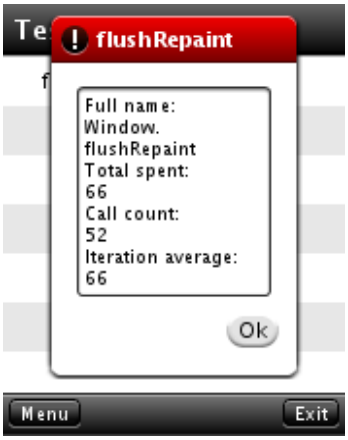

Figure 9: PETHIS showing intermediate result

scenarios. The design follows a layered architecture approach and restricts interactions between non-adjacent layers. The UI provides means to execute different scenarios and to upload the gathered results which would be used later for performance simulation. Currently, PETHIS provides following scenarios:

- Load page
- Scrolling
- Kinetic scrolling
- Zooming
- Demanding menu invocation

Additional scenarios and probes are required to be developed to further extend the usefulness and accuracy of the obtained results, however, the current solution already demonstrates a proof-of-concept implementation of the ideas described in this work. To prove that the solution is representative, results for few embedded/mobile of different generations are compared. Still, in case the comparison demonstrates controversy it does not necessarily imply existence of major faults but rather a lacking set of implemented probes and/or scenarios.

5 Discussion

Previously, in section 2.2, an overview has been given to the current state of benchmarking in general and its limitations in particular. To address the issues identified in the current body of knowledge on performance assessment we discuss how the existing limitations apply to our solution and what is necessary to avoid those intricate situations. A significant factor that needs to be taken into account is that a real application is used as a benchmark which means that several of the acknowledged limitations do not apply.

Additionally, in this section, after proceeding to the discussion of our solution - PETHIS, we look at the topic of sustainable mobile platforms with a

particular emphasis on perceived performance, to examine whether efforts to maintain long-term platforms enable enterprises to successfully deliver products that are accepted by consumers in high demand. Furthermore, a speculation is presented on possibility of combining techniques used by PETHIS, with the development of such long-term platforms.

5.1 Leveraging PETHIS

In the section we discuss our proof-of-concept implementation as well as various issues related to it. In our research and development process each of the previously identified benchmarking limitations has been addressed:

Synthetic nature of the test and inadequate workloads — the original problem of synthetic tests which do not represent any realistic load and, as a consequence, provide results that can hardly be used to compare different devices is addressed by the fact that a real application with very specific and common tasks is used. PETHIS scenarios represent typical operations a user performs on a browser, such as loading a page, scrolling in different directions and using animation-intensive menus.

Vendor optimizations — the problem does not apply in our case since Opera is dealing with devices that are already released on the market. Moreover, the issue generally does not apply to any benchmarks except those used for comparative extensive hardware tests (i.e., kernel benchmarks), such as CPU benchmarks. Furthermore, for marketing reasons only well-known de facto solutions are targets for such abuse so very specific niche products, such as PETHIS, have a little chance to be employed for generic benchmarking purposes.

Lack of reliable data and result interpretation — PETHIS collects runtime low-level performance data, such as functions' execution times, which is quantitative, and therefor objective indicator. However, additional analysis might be required to study which particular functions present the biggest correlation to the user perceived performance because little or no use could be extracted from measuring random method performance timings. Also, since the obtained results shall be consumed by the simulator there is no subjective interpretation involved. On the other hand, there is more than one way of implementing the mechanism that shall simulate the behavior of devices based upon the collected results so it is crucial to carefully design the future simulator.

The developed solution PETHIS that we propose raises certain concerns and calls for future investigation to improve the methodology of accurately measuring perceived performance. While similar approaches are known to be taken in the past by other researchers (e.g. [TMurgent Technologies, 2003]) there have been major obstacles in implementing a working prototype due to inherent complexity of intercepting correct function calls to extract meaningful data for analysis.

Unlike the previous research attempts we have dealt with a particular specific application developed by Opera Software so we had an opportunity to closely study its structure and discover exact places where the performance bottlenecks occur. And although the developed artifact does not yet provide performance statistics for every application aspect a convenient framework for extending its usefulness with enough modularity is in place.

A better understanding of correlation between the perceived performance and parts of the software that represent actual features is necessary to produce performance data which resembles typical realistic behavior. That in itself is not a straightforward process and calls for a separate research undertaking. One might argue that it is possible to analyze the execution of low-level procedures (for example, system calls), however, there is a trade-off between:

- providing meaningful representative statistics which can be used to compare and judge whether particular devices have better user experience and possibly be used in a simulator to mimic the behavior
- presenting meaningless, though very detailed and specific, performance indicators that have very little or no correlation with the perceived performance

Thus, it is crucial not to end up with a bench-

mark that has a vast array of limitations that are typically associated with synthetic benchmarks. Another possible limitation of this research is its narrow scope since it might not be feasible to execute a very detailed software analysis in order to discover interception points where measurements are to be taken. Nor would it be feasible to place such a framework in a generic environment such as the Linux kernel for the same complexity reason.

5.2 Sustainable mobile platforms

Building a solution like PETHIS requires a significant development involvement and ultimately translates into high development costs. The reason for high development costs comes from the nature of PETHIS as it is closely coupled to the underlying application or platform it is measuring. In other words, solution like PETHIS would require an on-going maintenance unless it is changed in some crucial way. Therefore, we argue for introducing PETHIS into platforms or products that are supported in the long-term perspective.

5.2.1 Relation to UPP

One of the most successful consumer devices that has been attributed to the high-end user experience and is believed to be a revolution on the mobile market is the Apple iPhone smartphone. While iPhone is generally considered not the best from its features or technical specifications (based on comparisons to rival products), it delivers a unique user interface and deep integration with the web. Specifically, iPhone's web browser Safari has been regarded by consumers as being fast and responsive. In reality its Extensible Hypertext Markup Language (XHTML) rendering engine has been assessed to perform on par with the rivals' products [Michaluk, 2008, Haselton, 2008, Lee, 2008]. The reason why such positive perception has been awarded to Apple's product is the contribution of perceived performance and company's efforts to engineer a highly consumer-friendly device.

Though the topic of sustainable platforms and their impact is out of scope of this paper and require additional research treatment, we nevertheless consider it useful to hypothesize about its relation to concepts of perceived performance. Continuing with the Apple's product success an observation can be made that the iPhone not only revolutionized the handset market but the adjacent technical areas too. For example, the browser included with iPhone, albeit not being the first one on the market with full XHTML support, was first one to make it seem that browser is a natural part of the mobile device. There have been many mobile browsers, and even highly-specific network protocols (e.g. Wireless Application Protocol), before Apple joined the competition but UPP and platform consistency greatly modified the market landscape so it is now unlikely that any mobile manufacturer will release a smartphone without including a XHTML-compliant browser. This should positively affect companies such as Opera business-wide. As CEO of Opera Software, von Tetzchner, puts it: "What the iPhone did was make people want a full browser for their phones, and we have that." [von Tetzchner, 2009].

5.2.2 Historical perspective

Apple iPhone stands out for a much more significant reason than just an example of a well thought-of UI — in essence, it is a perfect example of a sustainable mobile platform. While competitors are known to release devices with substantially better technical characteristics (examples include but are not limited to LG Prada, Motorola Q series, high-end Nokia phones, HTC smartphones) they are still considered as an "afterthought", or even as the secondary part of the market in contrast to the iPhone, which stands aside. Though the mentioned manufacturers have been moving in the same direction as Apple, at the time the iPhone went out they had poorly underdeveloped concepts of a long term platform without an emphasis on the business model behind it. Oppositely, Apple has some experience in building sustainable platforms under its belt, having the world's longest running graphical desktop platform with the Macintosh and being one of the first companies to successfully release a personal computing platform Apple II back in the 70s. As a matter of fact, more and more manufacturers are picking up Apple's strat-

egy, and are either developing their own software platform, or reusing an existing one, like the Open Handset Alliance Android.

What Apple has managed to comprehend is the importance of user experience for the success of a consumer product. Company's engineers have translated that into a consistent, reliable, attractive and innovative platform which has been maintained throughout the development of the line of its products. There is a price for everything and maintaining mobile platforms is not an exception to that rule. In the early 90s Apple's efforts to introduce one of the first personal digital assistant (PDA), Newton Message Pad, have been bogged down by a collection of technological problems (such as short battery life, thermal constraints, limited storage and processing capabilities) inherent to all mobile device of the time. Originally, Apple has cooperated with a company called Acorn Computers Ltd to develop a new processor called ARM6[16], built an entirely new operating system and amply invested in marketing[17] to overcome these difficulties. Despite these efforts, Newton has still failed. Primary reason for its reason is accounted to the overmarketing of hand-writing recognition capabilities, which, although were ahead of its time, did not fulfill the expectations of users [Tesler, 2001].

There have been several other attempts to fill the market with a long term sustainable platform, the well-known examples include Microsoft Windows CE (it was later renamed to Windows Mobile), Symbian and Palm OS. However, none of these have managed to attract satisfactory levels of audience because of the problem that lies in the opposite side of the platform design continuum. While a lot of engineering talent has been involved in the development (not without major technological breakthroughs) little thought has been given to assessing user experience as a whole and perceived performance in particular. To a certain extent, a product could be compared to an artwork. Engineering efforts, just like artistic efforts, can extremely valuable when packaged appropriately, yet without it, the talent itself has no intrinsic value.

Having learned on their own and others' mistakes, three years later after discontinuing Newton, Apple created a new device, the iPod. It had some rudimentary PDA features, including a calendar and contacts, though it has been assigned a primary task — music playback. Pushing iPod into existing, wildly successful, market of digital playback devices turned out to be an extremely smart strategy. The device stood out with its revolutionary comprehensive interface and simple synchronization capabilities which reliably worked with both, PC and Macintosh computers, something that has been a huge problem for the other big vendors of the time. Once it mastered audio playback, Apple has licensed audiobook through a service called Audible, introduced podcasting support, video playback and some capabilities for extending functionality (e.g. ability to install games). The company succeeded in building an iPod empire, licensing hardware accessory makers and working with selected software vendors to offer games. That, in turn, has put company in an excellent position to launch a smartphone based on a new, at the time, software technology which is based on their desktop operating system Mac OS X.

In 2007, Apple launched the iPhone as a high-end iPod, phone, and web browser device to a large, enthusiastic user base of Mac and iPod users. The iPhone effectively attached all of the iPod's goodwill and content expertise with the technical superiority of Mac OS X to a new market even larger than that of music players, a market dominated by products that were largely based on simple devices trying to operate well out of their league. Users, not mistakenly though, assumed that the product had some desirable quality because it carried the same brand name as their other favorite products, a condition otherwise known as halo effect [Seow, 2008]. However, it was not the PDA features that made it stand out, but rather effective packaging of a Wi-Fi enabled phone, iPod media player, software features, albeit quite limited, rich Internet-connected applications and effective marketing. Not less important how Apple handled that transition, and most importantly the user

[16] This ARM architecture branch became start to what ARM is today. The ARM architecture is the most widely used 32-bit CPU architecture in the world.

[17] The device still remains familiar within a lot of users as of 2009, unlike all other devices sold at the time or even years later.

base. All generations of iPod and iPhone share most parts of the UI, whilst there is a huge difference in functionality between, say, iPod 1.0 and iPhone 3.0, which is more than a smartphone — it is now a central interface for controlling all sorts of hardware.

5.2.3 Implications of sustainable platforms

Despite the need of further research in the area of mobile platforms, it appears that developing a long-term sustainable platform contributes to a company's efforts to deliver devices that are capable of demonstrating excellent perceived performance. Moreover, having a software platform that is supported and maintained throughout technology generations, allows incorporating an infrastructure similar to PETHIS to obtain measurable UPP indicators.

6 Conclusion

While machine performance is a subject of interest to many, if not all, Computer Science branches, user perceived performance is a relatively uncharted area. User perceived performance of embedded/mobile devices and its benchmarking are subject of this paper.

Machine performance or user perceived performance alone may not win user acceptance and subsequent marketability. If the end-users feel that a product is not easy to learn, not easy to use, or too cumbersome, an otherwise excellent product could fail. Furthermore, the way users form their mental model of the system and their feeling of responsiveness and performance is subjective and complex. It is not unusual for a user perceived performance to deteriorate on the contrary to the objective measurement and common sense. However responsive UI can make a product easy to understand and use, which results in greater user acceptance.

At the same time, competition on the hardware and software market forces vendors to come up with new products in shorter development time, thus, among all, reducing time available for quality assurance procedures. This, together with complexity of modern UIs inevitably creates pressure on respective teams, forcing them to adopt automated testing techniques.

During the course of our work, carried at Opera Software ASA, we have created a proof of concept piece of software, which has a form of compile time plug-in, for a yet to be released product. On its basis we have successfully demonstrated techniques for effective perceived performance measurement. Implementing these techniques allows for repeatable, structured and systematic approach in user perceived performance testing, thus, potentially creating cost savings for the company and enhancing user perceived performance, which in turn leads to strengthening of the market position.

References

[Csikszentmihalyi, 1990] M.Csikszentmihalyi, 1990, *Flow: The Psychology of Optimal Experience*, Harper Collins, New York, USA;

[Curnow & Wichmann, 1976] H.J.Curnow, B.A.Wichmann, 1976, 'A synthetic benchmark', Central Computer Agency, Riverwalk House, London, UK;

[Bott, 2008] E.Bott, 2008, *Another take on Vista vs. XP benchmarks*, http://blogs.zdnet.com/Bott/?p=369, retrieved 04.05.09;

[Dunlavey, 1993] M.R.Dunlavey, 1993, 'Performance Tuning: Slugging It Out!', *Dr. Dobb's Journal*, vol. 18, no. 12, pp. 18-26;

[Dujmovic, 1999] J.J.Dujmovic, 1999, 'Universal Benchmark Suites', Dept. of Comput. Sci., San Francisco State Univ., CA, USA;

[Gasser & Majchrzak & Markus, 2002] L.Gasser, A.Majchrzak, M.L.Markus, 2002, 'A Design Theory for Systems that Support Emergent Knowledge Processes', *MIS Quarterly*, vol. 26, no. 3, pp. 179-212;

[Gustafson et al, 1990] M.Carter, S.Elbert, J.Gustafson, D.Rover, 1990, 'The design of a

scalable, fixed-time computer benchmark', Ames Laboratory, Iowa State University, Ames, Iowa, USA;

[Guthaus et al, 2001] T.M.Austin, R.B.Brown, D.Ernst, M.R.Guthaus, T.Mudge, J.S.Ringenberg, 2001, 'MiBench: A free, commercially representative embedded benchmark suite', Michigan Univ., Ann Arbor, MI, USA;

[Harrison et al, 2007] B.Amento, R.Bell, C.Harrison, S.Kuznetsov, 2007, 'Rethinking the Progress Bar', *Symposium on User Interface Software and Technology*, Proceedings of the 20th annual ACM symposium on User interface software and technology, pp. 115-118;

[Haselton, 2008] T.Haselton, 2008, *Mobile Browser Showdown: iPhone 3G vs Opera Mobile and SkyFire*, http://blog.laptopmag.com/mobile-browser-showdown-iphone-3g-vs-opera-mobile-and-skyfire, retrieved 19.05.2009;

[Hevner et al, 2004] A.R.Hevner, S.T.March, J.Park, S.Ram, 2004, 'Design Science in Information Systems Research', *MIS Quarterly*, vol. 28, no. 1, pp 75-105;

[Hockney, 1992] R.Hockney, 1992, 'A Framework for Benchmark Performance Analysis', Southampton University, Southampton, UK;

[Kasanen & Lukka & Siitonen, 1993] E.Kasanen, K.Lukka, A.Siitonen, 1993, 'The constructive approach in management accounting research', *Journal of Management Accounting Research*, vol. 5, no. 1 pp. 241;

[Kevin, 2006] C.Kevin, 2006, *Hardware Testing and Benchmarking Methodology*, http://donutey.com/hardwaretesting.php, retrieved 06.04.2009;

[Kleinrock, 2001] L.Kleinrock, 2001, 'Breaking loose', *Communications of the ACM*, vol. 44, no. 9, pp. 41-46;

[Kuechler & Vaishnavi, 2007] W.Kuechler, V.Vaishnavi, 2007, *Design Research in Information Systems*, http://tinyurl.com/chgadd, retrieved 06.04.2009;

[Lee, 2008] J.K.Lee, 2008, *Website load times compared: Archos 5 vs. iPhone 3G vs. Nokia N810*, http://www.pocketables.net/2008/10/website-load-ti.html, retrieved 19.05.2009;

[Lee & Mangione-Smith & Potkonjak, 1997] C.Lee, W.H.Mangione-Smith, M.Potkonjak, 1997, 'MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems', *International Symposium on Microarchitecture*, Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture, pp. 330-335;

[Ling, 2004] R.S.Ling, 2004, *The mobile connection*, Morgan Kaufmann, San Francisco, CA, USA;

[Lyytinen & Yoo, 2002] K.Lyytinen, Y.Yoo, 2002, 'Issues and Challenges in Ubiquitous Computing', *Communications of the ACM*, vol. 45, no. 12, pp. 62-65;

[Maister, 1984] D.H.Maister, 1984, 'The Psychology of Waiting Lines', Harvard Business School, Harvard, CA, USA;

[March & Smith, 1995] S.T.March, S.T.Smith, 1995, 'Design and natural science research on information technology', Information and Decision Sciences Department, Carlson School of Management, University of Minnesota, Minneapolis, MN, USA;

[McKenna, 2000] D.McKenna, 2000, 'Mobile Platform Benchmarks, A Methodology for Evaluating Mobile Computing Devices', Transmeta Corporation;

[Michaluk, 2008] K.Michaluk, 2008, *Real World iPhone 3G vs. Storm Browser Speed Test*, http://crackberry.com/iphone-3g-vs-blackberry-storm-real-world-browser-test, retrieved 19.05.2009;

[Moore, 1965] G.E.Moore, 1965, 'Cramming more components onto integrated circuits', *Electronics*, vol. 38, no. 8, pp. 114-117;

[Myhrvold, 2006] N.Myhrvold, 2006, 'Moore's Law Corollary: Pixel Power', *The New York Times*, 7 June;

[Paul & Somers, 2008] J.M.Paul, M.Somers, 2008, 'Webpage-Based Benchmarks for Mobile Device Design', Virginia Tech Electr. & Comput. Eng., Blacksburg, VA, USA;

[Partridge, 1991] D.Partridge, 1991, *A New Guide to Artificial Intelligence*, Norwood, N.J. Ablex Pub. Corp., New York, USA;

[Price, 1989] W.J.Price, 1989, A Benchmark Tutorial, *IEEE Micro*, vol. 9, no. 5, pp. 28-43;

[Russinovich, 2008] M.Russinovich, 2008, *Inside Vista SP1 File Copy Improvements*, http://blogs.technet.com/markrussinovich/archive/2008/02/04/2826167.aspx, retrieved 04.05.09;

[Seow, 2008] S.C.Seow, 2008, *Designing and Engineering Time: The Psychology of Time Perception in Software*, Addison-Wesley Professional, San-Francisco, CA, USA;

[Sill, 1996] D.Sill, 1996, *comp.benchmarks Frequently Asked Questions, With Answers*, http://tinyurl.com/ct6aqu, retrieved 06.04.2009;

[Simon, 1996] H. Simon, 1996, 'The Sciences of the Artificial', *MIT Press*, Cambridge, MA, USA;

[Tesler, 2001] L.Tesler, 2001, *Why the Apple Newton Failed*, http://g4tv.com/techtvvault/features/25271/Why-the-Apple-Newton-Failed_pg2.html, retrieved 07.05.09;

[von Tetzchner, 2009] J.S.von Tetzchner, 2009, *Browser Wars: Opera Says It's Not Down or Out*, http://gigaom.com/2009/03/02/browser-wars-opera-says-its-not-down-or-out/, retrieved 07.05.09;

[Tognazzini, 2001] B.Tognazzini, 2001, *Maximizing Human Performance*, http://www.asktog.com/basics/03Performance.html, retrieved 14.05.09;

[TMurgent Technologies, 2003] TMurgent Technologies, 2003, 'White Paper: Perceived Performance';

[Trenouth, 1991] J.Trenouth, 1991, 'A Survey of Exploratory Software Development', Department of Computer Science, University of Exeter, Devon, UK;