U.PORTO

iUSE – Visual Analytics & Web Framework for Desktop Runtime Intelligence Services

Luís Miguel Fonseca Rodrigues

Mestrado em Multimédia da Universidade do Porto

Orientador: Doutor António Manuel Lucas Soares (Professor Associado)

Junho de 2013

iUSE – Visual Analytics & Web Framework for Desktop Runtime Intelligence Services

Luís Miguel Fonseca Rodrigues

Mestrado em Multimédia da Universidade do Porto

Aprovado em provas públicas pelo Júri:

Presidente: Doutor Eurico Manuel Elias de Morais Carrapatoso (Professor Auxiliar)

Vogal Externo: Doutor José Benjamim Ribeiro da Fonseca (Professor Auxiliar)

Orientador: Doutor António Manuel Lucas Soares (Professor Associado)

Resumo

iUSE é uma plataforma web de *Runtime Intelligence Services* para aplicações *desktop* que disponibiliza os meios de recolha, transporte e análise visual do uso de uma aplicação informática. Coloca à disposição dos produtores de aplicações um conjunto de ferramentas cliente/servidor que, de forma não invasiva, recolhe informação sobre a utilização das suas aplicações em tempo real. A informação recolhida é condensada em visualizações que mostram padrões e tendências de uso que revelam o perfil dos utilizadores.

O *iUSE* inova no domínio de *Runtime Intelligence*, se comparado com produtos similares, na introdução de modelos visuais adequados à visualização de dados complexos e suas relações (modelos visuais baseados em grafos de informação ou modelos hierárquicos) e na sua capacidade de integração com a web semântica (*Linked-data*) através da exportação dos dados recolhidos para *RDF*, de acordo com o esquema de ontologia definido.

Abstract

iUSE is a Runtime Intelligence Service web framework for desktop applications that provides the means to collect, transport, analyze and visualize application usage. It offers a stack of client and server tools to software producers that unobtrusively capture real-time usage data directly from their applications. Collected intelligence is rendered as visualizations that show usage trends and patterns that unveil user profiles.

iUSE innovates, when compared to similar products, on how information visualization is applied and on its ability to provide integration with the semantic web and its linked-data world. *iUSE* introduces the concept of networks for Visual Analytics in the domain of software runtime intelligence and the representation of data using semantic models for the web (OWL and RDF).

Agradecimentos

Embora uma tese seja, pela sua finalidade académica, um trabalho individual, há contributos de natureza diversa que não podem nem devem deixar de ser realçados. Deixo, por isso, aqui algumas palavras de agradecimento e profundo reconhecimento.

Ao Professor Eng.º António Lucas Soares, pela disponibilidade manifestada em orientar este trabalho, pela exigência de método e rigor, pela orientação científica e pelos oportunos comentários, esclarecimentos e sugestões, e pela acessibilidade e cordialidade demonstradas, que se tornaram decisivos na elaboração desta tese.

Deixo também uma palavra de agradecimento aos professores da FEUP, que me acompanharam ao longo do primeiro ano da tese de mestrado. A sua orientação científica e os comentários críticos aos trabalhos realizados ampliaram significativamente a abrangência dos meus conhecimentos na área multimédia.

Não poderia também faltar uma palavra de apreço a todos os colegas de mestrado que tive oportunidade de conhecer, pelo companheirismo e disponibilidade demonstrados ao longo dos trabalhos que desenvolvemos em conjunto.

Por último, mas não de somenos importância, gostaria de agradecer à minha família. Aos meus pais, Rosária e Custódio, pelo apoio e compreensão inestimáveis e pelo investimento que fizeram em mim, ao longo de todo o meu percurso académico. Aos meus irmãos, Paula e Jorge, pela amizade e encorajamento. À minha namorada Ana pela revisão crítica do texto, pela compreensão e constante estímulo ao longo da elaboração deste trabalho. E, especialmente, ao meu filho Tiago, pela ternura com que sempre me presenteou, mesmo nos momentos em que terei estado menos presente. Espero que o entusiasmo, seriedade e empenho que dediquei a este trabalho possam servir-lhe de estímulo no futuro, para fazer sempre mais e melhor.

Luís Rodrigues

Index

Figures

Tables

Abbreviations

API	Application Programming Interface
CRM	Customer Relationship Management
CSS	Cascading Style Sheet
DOM	Document Object Model
GPU	Graphical Processing Unit
GUID	Globally Unique Identifier
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
KDD	Knowledge Discovery in Databases
LOB	Line-Of-Business
LOD	Linking Open Data
OOP	Object Oriented Programming
OS	Operating System
OWL	Ontology Web Language
R&D	Research and Development
RDF	Resource Description Framework
REST	Representational State Transfer
SDK	Software Development Kit
SOAP	Service Oriented Application Protocol
SVG	Scalable Vector Graphics
SWEO	Semantic Web Education and Outreach
TED	Technology Entertainment and Design
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
UX	User Experience
XML	Extensible Markup Language

Chapter 1

Introduction

1.1 Context and Motivation

iUSE was motivated by two fundamental principles: organizational and technological. Technologically, it was an opportunity to learn and practice emerging web technologies like HTML5 and SVG. However, the main motivation of this project was organizational.

Software development organizations are constantly searching for improved ways to line up their resources with business goals and to adjust development investments with business outcomes. To succeed, decision makers need tools whose analysis yield actionable intelligence. Working as a software developer for the past 15 years, I've realized that a common difficulty in development organizations that produce line-of-business applications is to accurately track the success and quality of implemented features.

All software implements a set of features believed to have a key role on improving user's processes in a specific area or domain. During application requirement analysis and design, features are introduced explicitly by customer's request, or implicitly by inferred data. The weakness of inferred features rests in the quality of collected data that supports it - an inferred feature may be assumed as a justified need but, for a varied number of reasons, users may not share the same opinion. And that happens to be more critical whenever there are mediators between the development organization and the customer (end-user), making it difficult to access truthful feedback from users. Another reason for inferred features to be, sometimes, unaligned with real end-user needs occurs when the universe of users is too large to efficiently collect opinions from a relevant sample. In short, after a software release, the lack of an efficient communication channel with the user, raises common questions to development organizations related to: when and if the product was installed; the context in what it is running on; the use that is being given to the various features; the existence or absence of patterns of use; the existence of trends; quality issues; and so on.

In what concerns quality, usually the developer lacks vital information to help him diagnose the issue. Commonly, the end-user's error reports are vague and therefore the possibilities are immense. From a bug in the software to a particularity of the user's running environment that the software is not handling, almost any scenario is possible. At this stage, the developer, along with the quality team, try to reproduce the problem in a similar context and following the exact steps as the end-user did. But often the results are different from the reported issue. What if software developers could get all the diagnostic information they needed, just by having end-users interacting with their software?

Real-time data streams seen in a perspective of individual usage and their underlying hardware and software technology stack, combined with "community" usage patterns, help in answering "development" questions.

1.2 Project

The Agile Manifesto states that "Working software is the primary measure of progress and development's highest priority is to satisfy the customer through early and continuous delivery of valuable software." In that context, development success can be measured where users and their applications meet. *iUSE* targets Development as “the customer” providing reliable analytics by taking advantage of the most immediate communication tool that development organizations have at their disposal and rarely use: their application.

iUSE offers a stack of client and server tools to software developers that unobtrusively capture usage data directly from their applications. Collected intelligence is rendered in a Visual Analytics Dashboard with visualizations that show usage trends and patterns that unveil how users use the software.

“Network thinking” (Lima, 2011) is of key influence in the project: The complex connectedness of modern times requires new tools of analysis and exploration; it demands a holistic system approach with macro/micro vision to the intricate mesh of connections among its smallest elements. It ultimately calls for network thinking.

Networks are omnipresent, so if we consider new methods of analysis or modeling, then we need to consider the network thinking. This notion is transversal to the project: *iUSE* ability to provide integration with the semantic web and its linked-data world (OWL and RDF); and the Visual Analytics Dashboard follows that consistency in thought by adopting visual models based on network topologies.

1.3 Problems, Hypothesis and Research Objectives

The project addressed several aspects of today’s desktop analytics software and runtime intelligence, some of which are the subject of this thesis. Because *iUSE* was projected to be a working product, various layers of the system had to be designed and implemented. The researched problematic concerned desktop software analytic requirements and strategies to manage runtime intelligence data and investigate new methods of information visualization. I will briefly comment on each one in the next paragraphs.

The value of analytics is well understood by Web stakeholders and accepted as a standard component of any Web project. Further, Web users have come to accept that Web sites collect runtime data. But, traditionally Web analytics has put sales and marketing roles, rather than development, as "the customer" for these technologies. This difference in focus leads to an immediate functional divergence and virtually ensures that the gap between development requirements and Web/mobile analytics functionality will continue to widen.

Understanding this difference in focus between marketing and development roles was the motivation to define *iUSE* analytics requirements. The first problem that this thesis addressed was a **definition of a basic set of “development” data suitable to feed a Runtime**

Intelligence system capable of providing actionable intelligence to the stakeholders of desktop software development organizations.

A definition to an elementary set of "development" runtime data that an analytic system should implement was created by analyzing data obtained through the methodologies described ahead in section 1.4 (Research Methodology).

In information technology, big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications. The challenges include capture, storage, search, sharing, analysis, and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found (e.g. spot business trends). In a smaller scale, *iUSE* had to cope with "big data" challenges when handling large quantities of information collected from thousands of users. Two of those challenges were: how to structure and aggregate information for web clients properly because communication latency can impact system performance and user experience; how to expose all the richness of collected raw intelligence to the outside of *iUSE* data silo. The later was subject of research.

Previously I have described the first project goal – define an elementary set of "development" runtime data – which is tightly connected with this second: **Proposal of a standard schema to expose runtime intelligence as raw data**. It was not just to provide the ability to export raw data, but doing so in a way that acknowledges that collected data is stored in the Web, a growing global "neurological" storage (Lima, 2011). Not only is data becoming more widely accessible but also it is becoming enriched with metadata, allowing new sets of comparison.

In a March 2009 talk at Technology Entertainment and Design (TED) conference, Berners-Lee made a vehement exaltation for linked data (Berners-Lee, 2009). One year later, in February 2010, he came back to support his vision with various practical examples, stating that "if people put data on the web – government data, scientific data, community data, runtime intelligence – whatever it is, it will be used by other people to do wonderful things in ways they could have never imagined." This project proposal to a standard schema was to **create an Ontology described with OWL that defines a common model of runtime intelligence data.** Raw data is exported as RDF.

As previously stated "big data" represents a challenge for visualization. One of the best ways to explore and understand a large data set is to place the numbers into a visual space and let the brain find the patterns. We are good at that. Runtime intelligence hides a story with its complex data, a large number of highly interconnected and interdependent variables, that might never be unveiled with just formal statistical methods or standard graph – bar charts, pie charts, scatter plots, line charts, and so on. To comprehend it, we need new methods of information visualization, a new kind of representation for information processing: a tool for understanding data – i.e., discovering patterns, connections, and structure. Such tool was the visualization

problem researched in *iUSE* - **Investigate Visual Analytics[1] models to enable human-information discourse of application usage levels, patterns and practices.**

1.4 Research Methodology

iUSE project emerged from several years of experience working as a software developer. The idea grew with the awareness of a common difficulty of desktop applications development organizations to accurately track the success and quality of implemented features (see 1.1). Also, market solutions (see Market Survey) presented weaknesses that could be turned into an opportunity to develop *iUSE*. Large data point collection costs, unstructured or limited access to collected data (an important asset for organizations) and failure to provide usage patterns and relationship analytics (or merely visually scattered information throughout multiple views) were just some of the weaknesses for the majority of the surveyed products, in spite of good support in analyzing trends in time and quantifying top usage.

The aspects presented in the previous paragraph identified an opportunity to create the project (*iUSE*) focused on strengthening some of the surveyed weaknesses, with special emphasis on providing high-density visualizations and Visual Analytics of usage relationships and patterns.

The definition of a basic set of Runtime Intelligence functional requirements for desktop applications (see 4.4) started by interviewing people from a business software company (Sage Portugal) in key management roles, in the development process (see Stakeholders Survey) and by creating Personas that described their profiles (see Personas).

For implementing client services and designing the API that integrators should use to publish information into *iUSE* cloud services (see 4.3), client APIs from *DeskMetrics* and *UserMetrix* (surveyed products) were installed and analyzed.

In what concerns the cloud services, models were created to optimize data storage and mining, and then enhanced in an iterative process with the visual models because of their tight connection (see 5.1). Technologies and standards were investigated to enable integration of *iUSE* Runtime Intelligence Services with the web (see 5.2).

OWL Ontology was modeled using Protégé[2] with the purpose to create a standard exporting model to accommodate the gathered Runtime Intelligence Data (see 5.3). A search for Ontologies in the domain of Runtime Intelligence was made throughout Ontology Stores to find a matching Ontology to be used or extended, but such Ontology was not found. Nevertheless, some related Ontologies served as inspiration for modeling some concepts (e.g., *Event* and *Activity* from the Enterprise Ontology).

[1] Visual analytics is "the science of analytical reasoning facilitated by visual interactive interfaces."(Cook, 2005)
[2] Protégé is a free, open source ontology editor and a knowledge acquisition system.

Visual Analytics was supported by a literature review on principles of information visualization (Tufte, 1990), (Tufte, 1997), (Tufte, 2001) and also a literature review on displaying complex information through the use of networks (Lima, 2011) – inspired by the surveyed practices on information visualization (http://www.visualcomplexity.com). From the state of the art review (see 2.3) and the surveyed contemporaneous information visualizations examples, three models were chosen to depict patterns of use and relationships between Runtime Intelligence Data: *Network*, *Hierarchical*, and *Radial Convergence* models (see 4.6).

In what concerns the implementation of the Visual Analytics Dashboard, a set of rich internet application technologies was examined (see 4.5) and culminated with the adoption of SVG (Scalable Vector Graphics) and Web Audio API (web synthesized sound), two specifications of the HTML5 standard proposal, and D3 a JavaScript visualization library. A high-fidelity prototype implemented interaction behaviors and visualization for the three selected models (see 0).

The prototype was evaluated using iterative cognitive walkthrough, performed by the author while developing the prototype, and by using the Talk-Aloud Protocol for usability testing (see Usability testing).

1.5 Thesis outline

Besides introduction, this thesis has four more chapters. Chapter 2 resumes the literature review regarding the state of the art in Visual Analytics, with focus on depicting complex information through the use of network topologies. Market survey examines how runtime intelligence is implemented by similar products and reported in Chapter 3. Requirement specifications for all components of *iUSE* framework are described in Chapter 4. Chapter 5 elaborates on the details of their implementation. The thesis concludes with Chapter 6 stating conclusions and future work.

Chapter 2

Literature Review

2.1 Introduction

This chapter focuses on visual representation. It starts with an overview on information visualization fields, specifically the path from Visual Data Mining towards Visual Analytics and concludes presenting a review of literature on methods of visualizing complex information, with special emphasis on portraying information using network topologies.

The literature review on methods of visualizing complex information through the use of networks was based on the work of Manuel Lima (Lima, 2011). Lima is a designer, lecturer and curator of one of the most influential online galleries that presents some of the best projects in information visualization: VisualComplexity.com is focused on visualizations of networks. In is work *Visual Complexity: Mapping Patterns of Information* (2011), Lima balances historical and theoretical discussions with the presentation of exemplary projects in network visualization; elaborates on detailed principles to handle network representation challenges; and discusses an embryonic and evolving taxonomy – a portrait of the current state of the practice that reveals the initial building blocks shaping a new visual language for depicting complexity through networks.

2.2 Visual Data Mining/Analytics

> "The goal of visual data mining is to help a user to get a feeling for the data, to detect interesting knowledge, and to gain a deep visual understanding of the data set" (Ankerst, 2002)

Nieggman (2001) interprets visual data mining as visual representation close to the mental model. If humans understand information by forming a mental model, then a data visualization metaphor close to the mental model can reveal hidden information. In the domain of software usage, one such model could be the use of networks (graph-based-data).

Ankerst (2000), in addition to the role of the visual data representation, explored the relation between visualization and the data mining and knowledge discovery (KDD) process, and defined visual data mining as "a step in the KDD process that utilizes visualization as a communication channel between the computer and the user to produce novel and interpretable patterns."

Visual Analytics uses similar techniques for KDD but with different focus: Data Mining is computer-centred – computer performs data analysis and humans use the results by visual inspection and interactive tunning of association rules; Visual Analytics is human-centred – computer helps humans to solve a complex problem through visual perceptual and cognitive capabilities. Mining is performed by humans through perception of patterns, reasoning and intuition, insights otherwise not found by standard algorithmic means (Keim et al., 2010).

"Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces" (Cook, 2005)

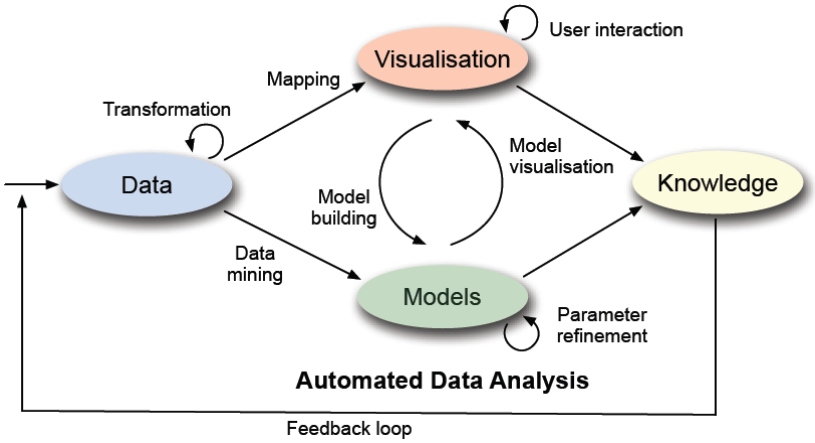


Figure 1 - The visual analytics process (Keim et al., 2010, p. 10).

The visual analytics process (see Figure 1) aims at tightly coupling automated analysis methods and interactive visual representations. In the context of visual analytics, the guide to visually exploring data "Analyze first, show the important, zoom/filter, analyze further, details on demand" (Keim et al., 2006) indicates that it is not sufficient to just retrieve and display the data using a visual metaphor (Shneiderman, 1996); rather, it is necessary to analyze the data according to its value of interest, showing the most relevant aspects of the data, and at the same time, providing interaction models, which allow the user to get details of the data on demand.

Visualization is at the heart of Visual Analytics (see Figure 2). Information visualization

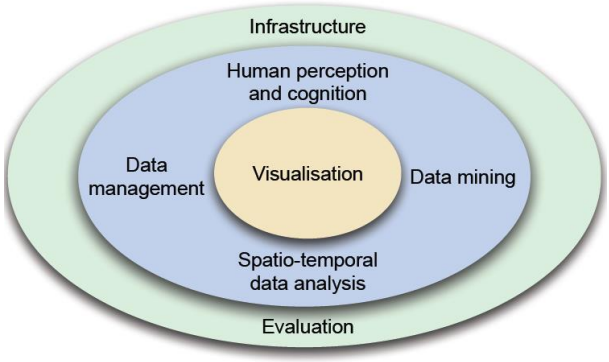


Figure 2 - The visual analytics disciplines (Keim et al., 2010, p. 12).

has developed methods for the visualization of abstract data where no explicit spatial references are given (Spenc, 2007). The data values cannot be naturally mapped to 2D or 3D display space, and standard charting techniques such as x-y plots, line graphs and bar-charts are ineffective with large multi-dimensional datasets. Moreover, as mentioned earlier, the capacity to interact with the data is extremely important. Novel visualizations have been developed such as treemaps, glyph and pixel-based visual data representations, to name just a few, together with a

variety of techniques to reduce display clutter (Dix, 2007). There are also special techniques for visualizing structured data, such as graph-based approaches for networks.

2.3 Networks and visualization

Networks and visualization are two techno-cultural phenomena of our time. While some scientists have already started to study networks in the middle of the twentieth century, globalization and the rise of the web in the nineties and the explosion of online social networks in the last decade have drawn attention to their importance. Furthermore, although scientists had already been making graphs and charts of their data since the early nineteenth century, the ubiquity of computers and the wealth of data unleashed by networks democratized information visualization, making it a rapidly growing new area of art and science.

The more recent language of Information Visualization share a lot in common with standard graph – bar charts, pie charts, scatter plots, line charts, and so on – already in use for about one hundred years before computers. Both represent quantified data by systematically mapping it into visual images: points, lines, curves, simple shapes, and other primitive graphics. However, there are some unique characteristics of information visualization: contemporary designers, artists, and computer scientists are trying to represent considerably more data than ever before; they want to represent relations between more dimensions of data than is possible with older graph types such as bar charts (one dimension) or scatter plots (two dimensions), generating designs[3] that are visually denser, more complex, and more varied than the familiar charts; and information visualization as also an aesthetical and ideological dimension that lies in understanding the phenomena of complexity (e.g., chaos theory[4]), which is reflected in the kinds of visualization we find appealing.

A network is a structural and organizational model transversal to almost every subject, from genes to computer systems and social communities. This ubiquitous topology is the object of study in network science, a new discipline aiming to uncover and understand the inherent principles and behaviors that regulate a variety of natural and artificial systems, normally characterized by the complexity of a multitude of interconnecting elements.

Application usage runtime data represents interactions between “things” in the domain of use (i.e., people, software features, and organizations). It stores cause and effect evidences (e.g., user fires an event starting an activity that produces a software exception) whose interconnectedness *iUSE* tries to unveil. For this reason, networks are a natural choice for depicting *iUSE* complexities, because they share the goal to explain important aspects and clarify given areas of a system. By communicating in a simple, effective way, the network visualizations become powerful means for information processing and understanding.

[3] Examples at www.visualcomplexity.com, http://infosthetics.com/, http://visualizing.org
[4] **Chaos theory** is a scientific theory describing erratic behavior in certain nonlinear dynamical systems.

Deleuze and Guattari in their *Capitalism and Schizophrenia* (1972-80) introduced the concept of *rhizome*, aimed at acknowledging multiplicities and multilinearities:

> "*In contrast to centred systems with hierarchical modes of communication and pre-established paths, the rhizome is an acentered, non-hierarchical, nonsignifying system without a General and without an organizing memory or central automaton, defined solely by a circulation of states.*"

The rhizomatic model is a significant influence in postmodern thinking, particularly in areas like communication theory, cyberspace theory, complex systems, nonlinear narrative, and hypermedia. But perhaps one of the most famous demonstrations of the principle's applicability is hypertext – perhaps the largest rhizomatic system ever created by man.

A few decades before Deleuze and Guattari's conception of the rhizome, American scientist Warren Weaver was already aware of the inherent complexities of nature and the obstacles anticipated by the scientific community in deciphering them. In 1948 in an article entitled "*Science and Complexity,*" Weaver divided the history of modern science into three distinct stages: "*problems of simplicity*" – understanding the influence of one variable over another; "*problems of disorganized complexity*" – complex systems with many variables where interaction between many of these variables was thought to be random and sometimes chaotic; and "*problems of organized complexity*" – the last stage defined by Weaver, initiated in the second half of the twentieth century and continuing to these day (Weaver, 1948).

The complex connectedness of modern times requires new tools of analysis and exploration, but above all, it demands a new way of thinking. It demands a holistic system approach with macro/micro vision to the intricate mesh of connections among its smallest elements. It ultimately calls for network thinking.

There are various examples of how previous conceptions of organization (i.e., taxonomies) are giving way to new ideas capable to address the complexities of modern society (Lima, 2011, pp. 43-69). Complex systems, such as the Brain or the World Wide Web, are defined by a large number of interconnected elements, normally taking the shape of a network.

Networks are omnipresent – we act and live in networks, so if we consider new methods of analysis, modeling or simulation, then we need to consider the network thinking. This notion of network thinking is transversal to the *iUSE* project (e.g., *iUSE* Ontology) and visualization follows that consistency in though.

Network representation is commonly used by two main areas: graph drawing (under graph theory) and network visualization (under information visualization). In both disciplines the pictorial representation of a network throughout a set of vertices (nodes) connected by edges (links) is known as *graph*. Network visualization extends beyond the mere geometric drawing of graphs, using elementary design principles aimed at an efficient and comprehensible representation of the target system.

The network structure, based on nodes and links, can produce many insights: What are the nodes doing? How are they interacting? How many connections do they have? What are they sharing? This series of queries can lead to the identification of the topological significance. In this pursuit, network visualization can be a remarkable discovery tool, able to translate structural complexity into perceptible visual insights aimed at a clearer understanding. It is through its pictorial representation and interactive analysis that modern network visualization reveals many structures hidden from human perception, from eccentric visualizations of the World Wide Web to the representation of the brain's neural network.

As a visual decoder of complexity, the practice of network visualization is commonly driven by five key functions: *document*, *clarify*, *reveal*, *expand*, and *abstract* (Lima, 2011, p. 80). *Clarify* and *Reveal* are considered the most relevant for this project:

- *Clarify* – The central objective in this context is simplification – to explain important aspects and clarify given areas of the system. By communicating in a simple, effective way, the network visualizations become powerful means for information processing and understanding.

- *Reveal* – Find a hidden pattern or explicit new insight into the system. The goal of revealing should concentrate on causality by leading the disclosure of unidentified relationships and correlations while also checking initial assumptions and central questions.

Graphs are, as of today, the most suitable method for the depiction of networks due to their intrinsic organization based on nodes and links, but they are far from perfect. Many of the current limitations – such as resolution and screen size – can quickly lead to cluttered displays. The adoption of interactive techniques solves some but not all of the problems on the challenging state of affairs in network visualization. In order for the general usability of network visualization to improve, we need to embrace the existing body of knowledge from graphic design, cartography, and visual perception, including notions of color theory, composition, typography, layout, and spatial arrangement.

Lima (2011) proposes a list of eight principles to support the creation of network visualizations. The first four are general principles of graphical representations; the subsequent are detailed principles to handle network representation challenges:

1. *Start with a question* – The definition of a question is vital and ties back to the need for a clear purpose and goal in every execution. The initial question is what evaluates the effectiveness of the project as a measure to filter the essential from the superfluous.

2. *Look for Relevancy* – Human cognition is relevance oriented (Sperber & Wilson, 1995): we pay attention to information that seems relevant to us. The measure of relevance is

therefore primarily based on the intent of the project and the validation of the initial question that set it forward. The selection of the most suitable visualization method for the project is largely determined by the central question. However, this particular quest is equally dependent on the end users, their immediate context and expressed needs. Acknowledging the different contexts of use – when, where, and how the final execution will be used – is crucial in the pursuit of relevancy.

3. *Enable Multivariate Analysis* – The ties among elements in a network are immensely rich and detailed, and the inclusion of additional information – able to provide additional information on the nature of nodes and respective ties – can be fundamental in expose causality in patterns and relationships, contributing decisively to the holistic understanding of the depicted topology.

4. *Embrace Time* – Time is one of the hardest variables to map in any complex system. It is also one of the richest. If we consider a social network, a snapshot in time can only tell us a bit of information about that community. Alternatively, if time were to be properly measured and mapped, it would provide us with a comprehensive understanding of the social group's changing dynamics. Time analysis not only identifies historical evolution, but also highlights the inherent dynamics of real-time oscillations (Lima, 2011, p. 85).

5. *Enrich you vocabulary* – Whenever considering the representation of a network, there are two vital elements to consider: nodes (vertices) and links (edges). The expressive capabilities of these elements are often neglected. A consideration of a full spectrum of visual properties – color, shape, size, orientation, texture, value and position, as outlined in Jacques Bertin's list of seven graphical attributes from his seminal work *Semiology of Graphics (1984)* – can and should be used comprehensively, always reinforced by a specific semantics able to tie the different data attributes to corresponding visual elements.

    - *Richer Nodes* – Nodes can be more intelligible with an appropriate use of color and graphical features. They can also become responsive and provide contextual information through the use of interactive features. Nodes can expand or shrink, show or hide relevant information, and ultimately morph according to the user's criterion and input.

    - *Expressive Edges* – Edges can express much more than a single connection between entities. The following factors should be considered in visualizing edges: *length* to suggest a gradation of values; *color* to differentiate or highlight particular

groups, categories, and clusters, or alternatively, singular connections; *shape* to communicate the type of relationship.

- *Clear Visual Language* – One of the caveats behind the implementation of diverse graphical attributes is to beware of creating a visual language that might not be immediately recognized by everyone. Embrace the cartographic technique: the legend. The map legend is vital, allowing for a quick interpretation of the various graphic components and facilitating an immediate understanding of topology.

6. *Expose Grouping* – Spatial relationships are as important as explicit visual ties and are a critical element in exposing contrast and similarity. The idea of grouping is simply to combine several units of information into related chunks in order to reinforce relationships, reduce complexity, and improve cognition. In most cases, elements can be grouped in five distinct ways: alphabetically, by time, by location, by a particular continuum (or scale), and by a specified category (e.g., images, videos, text). This procedure, first proposed by Richard Saul Wurman in *Information Anxiety* (2000), is known as the five hat racks, and it delivers an effective way to organize most types of information. Another remarkable source of knowledge on the notion of grouping comes from Gestalt psychology (Köhler, 1947). Of particular relevance are the devised rules of perceptual organization, also known as Gestalt laws of grouping. Three of the Gestalt laws – *similarity*, *proximity* and *common fate* – are particularly important rules in exposing groups in network visualization.

   - *Law of similarity (graphical treatment)* – The law of similarity asserts that elements that are similar – either in color, shape, or size – are perceived to be more related than elements that are dissimilar. This Gestalt principle highlights the need for a differentiated graphical vocabulary in the depiction of nodes, as a critical measure for spotting similarities and differences and in order to apprehend the overall distribution within the system.

   - *Law of proximity (spatial arrangement)* – The law of proximity states that elements that are close together are perceived as being more related than elements that are farther apart. This organizing principle proves that relatedness is not only expressed by graphical properties but also by spatial proximity. The mere placement of homologous nodes closer to each other suggests inherent relationships not solely manifested by edges (links).

   - *Law of common fate (motion)* – The law of common fate proclaims that elements that move simultaneously in the same direction and at the same speed are perceived as being more related than elements that are stationary or that move in different

directions. This notion is particularly pertinent when trying to highlight contrast through animation (e.g. depicting the changing dynamics of a network over time).

7. *Maximizing Scaling* – One of the biggest misconceptions in network visualization is the notion that a representation that works at one scale will also work at a larger or smaller scale. Not only do networks showcase different patterns and behaviors at different scales, but also the user's needs vary depending on his or her particular position with respect to the network. When representing a network, it is important to consider three fundamental views in line with a specific method of analysis: macro view, relationship view, and micro view.

    - *Macro View (pattern)* – A macro view should provide a bird's-eye view into the network and highlight certain clusters, as well as isolated groups, within its structure. In most cases, the use of color (within nodes or edges) and relevant positioning (grouping) is enough to provide meaningful insight into the network's broad organization.

    - *Relationship View (connectivity)* – The relationship view is concerned with an effective analysis of the types of relationship among the mapped entities (nodes). It not only indicates the existence of connections but also offers further revelation, such as proximity between the nodes, and type and intensity of association. This is a fundamental view of network visualization and normally requires analysis from different perspectives or points of view in order to obtain a solid grasp of the different topologies.

    - *Micro view (individual nodes)* – A micro view into the network should be comprehensive and explicit, providing detailed information, facts, and characteristics on a single-node entity. This qualitative exposure helps to clarify the reasons behind the overall connectivity pattern, from an isolated node to one highly connected to a large number of other nodes.

8. *Manage Intricacy* – Even though the three main views for network visualization appear to be autonomous, it is imperative that users are able to navigate between them in a seamless way. Progressive disclosure is an interaction-design technique aiming at simplification that allows additional content and options to be revealed gradually, as needed, to the user. This technique is particularly relevant if we consider Hick's Law, put forth by psychologist William Edmund Hick, which states that the time required to make a decision increases as the number of variables increases. Alluding to the risk of displaying a full, convoluted network at once in a single view, Hick's Law is an important point of awareness of the perceptual limits of network visualization. Even

though other methods can and should be devised, there are three important concepts that can help minimize intricacy and unify the three views of network visualization:

- *Adaptive Zooming* – This widely used modern cartographic technique – strongly tied with the notion of progressive disclosure – enables the system to render a different set of visual elements depending of the present zooming view. A similar method – semantic zoom – could be employed in the depiction of networks, by focusing on a gradation from macro to micro view, showing the most prominent nodes first, and then slowly disclosing additional graphical and textural elements: major hubs and primary links, labels, secondary nodes and links, and so on.

- *Overview and detail* – A common interaction-design technique, overview and detail usually comprises a primary viewing area (detail) that allows for different levels of zoom, accompanied by a smaller macro view (overview), which permits users to see where they are in the general context. This is particularly relevant in reassuring users they are free to navigate the system without getting lost.

- *Focus and context* – This widely used information-visualization concept is one of the field's strongest contributions and its most studied technique. It simultaneously provides a detailed view (focus) and a macro view (context) within a single configuration. Popularized by the widespread fish-eye view, this method merges both views in the same space without the need to segregate them.

The network depictions produced in the last decade, enriched by the diversity of subjects, portray a variety of visual techniques (Lima, 2011, pp. 97-158). Frequently generated by computer algorithms and enhanced by iterative features, most projects showcase a broad palette of visual elements and variations that consider color, text, imagery, shape, contrast, transparency, position, layout and configuration. Despite this rich graphical diversity, many projects tend to follow noticeable trends and common principles, which in turn result in a type of emergent taxonomy. See Figure 3 (Lima, 2011, p. 158) .

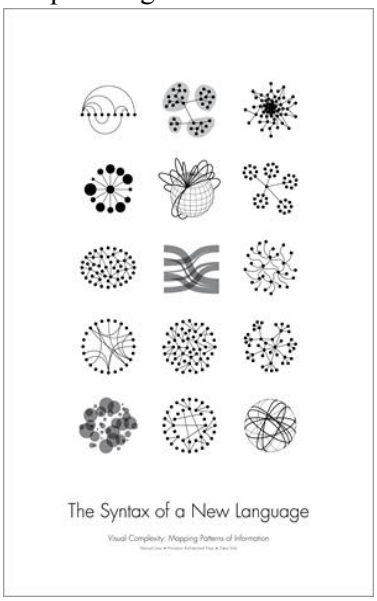


Figure 3 - New Visual Language.

This embryonic and evolving taxonomy provides a portrait of the current state of the practice and reveals the initial building blocks shaping a new visual language: Arc Diagram, Area Group, Centralized Burst, Centralized Ring, Circled Globe, Circular Ties, Elliptical Implosion, Flow Chart, Organic Rhizome, Radial Convergence, Radial Implosion, Ramifications, Scaling Circles, Segmented Radial Convergence, and Sphere.

Complex networks are intriguing and stimulating – emotional values that these visual methods intent to depict through the use of aesthetics. When it comes to express particular intentions, the mere appliance of individual elements – dot, line, color, shape, direction, texture, scale, dimension, or motion – is not enough. This is why Dondis, in *A Primer of Visual Literacy (1974)*, provided a complementary inventory of visual methods of how to combine these ingredients (i.e., balance and instability, symmetry and asymmetry, transparency and opacity). Dondis's study provides a set of communication-design patterns for building the most suitable visual composition for any given intent. It is important to understand such communication strategies by analyzing the different methods for reaching a particular goal.

Aesthetic judgment has always been seen as an unempirical domain, but many researchers are striving to quantify and understand it. Information visualization is traditionally viewed as a tool for data exploration and hypothesis formation. In recent years, however, both the mainstreaming of computer graphics and the democratization of data sources on the Internet has had important repercussions in the field of information visualization. With the ability to create visual representations of data on home computers, artists and designers have taken matters into their own hands and expanded the conceptual horizon of information visualization.

2.4 Conclusions

Literature review provided a deeper understanding about the Visual Analytics role, requirements and challenges. The theoretical discussions in the context of exemplary projects and the presentation of an evolving taxonomy of visualizing complex information, using network topologies, enabled the author to identify three appropriate visualization models for *iUSE* (visualizations that depict usage, patterns and relationships)*: Organic Rhizome (graph based), Radial Convergence,* and *Scaling Circles*. Detailed principles to handle network representation challenges helped to define visual and interactive requirements for the proposed models.

Chapter 3

Market Survey

3.1 Introduction

In the past few years the offer of desktop analytics software has increased, driven by the widespread of internet connectivity that boosted the universe of desktop software with access to cloud services. The growing number of connected customers has created the opportunity to track usage of desktop software which provided useful data to software developers and product managers that helped them shape business strategies, based on real facts about their software usage.

This chapter analyses five products in the area of desktop software analytics. Products targeting specific web applications (e.g. Google Analytics) or specific software frameworks were not considered (e.g. PreEmptive is a powerful tool but limited to .NET and Java client applications). The five products were chosen from an extensive search in the internet and internet forums and selected according to the following criteria: supporting some analytics relevant to desktop web based software solutions; supporting multi-platform clients (e.g. .Net, C, Java, etc.); and support providing (e.g. Chat, FAQ, documentation, SDK, Wiki/Blog/ Forum).

A comparative table (see Table 1) was created to relate relevant analytic features between the five chosen providers: Millimetrics, DeskMetrics, TrackerBird, UserMetrix, and EQATEC. Some of these products also implement web metrics, as an example: DeskMetrics implements Loyalty, New vs. Returning and Funnel Analysis. But because *iUSE* centers on desktop applications, marketing metrics aiming web applications were not considered.

A special paragraph about Mixpanel (a web/mobile analytics product) was also included because web analytics influence on desktop software analytics can't be ignored, since there are some overlapping analytic features that must be considered.

Below, in this survey, there's a brief report for each product. Each report is not intended to be an exhaustive analysis of every single feature, usability or design, but a record about first impressions in contrast to analytic requirements, as well as a picture of the information visualization methods, used by current providers.

The chapter ends with a brief conclusion about inspected analytics and presentation methods.

Table 1 - Runtime Intelligence Services comparative analytics.

Logging	Millimetrics	DeskMetrics	TrackerBird	UserMetrix	EQATEC
OS	platform, architecture, version	platform, architecture, version	platform, architecture, version	platform, version	platform, architecture, version
Culture settings	language	language	language	-	language
Peripherals	cd devices, monitors, printers, scanner, camera	monitors, display resolution, display DPI	monitors, display resolution	-	monitors, display resolution, display DPI
Hardware	GPU, CPU, memory, experience index	Memory	CPU, memory, pc type	-	CPU, memory
Component information	Flash, Java, .NET, Silverlight	.NET, Java	-	-	.NET
User activity	-	event	event	event	event, activity
User info	☑	☑	-	-	
Geolocation	☑	☑	☑	-	☑
Exceptions	☑	-	☑	☑	☑
Custom Data	☑	☑	-	-	☑
Installation/ Uninstallation	-	☑	-	-	☑
Executions/ Unique Executions	-	☑	-	☑	☑
Licenses	-	-	☑	-	-
Messages from Users	-	-	-	☑	-
Direct2desktop messaging	-	-	☑	-	-
Log messages	-	☑	-	-	☑
Export	jpg, csv, excel	csv, json	csv, xml	-	excel

3.2 Millimetrics

Collected information is divided into three separate categories: General, Custom Reports and Error Reports. The first one includes data about the executing environment; the second one is used to track different types of information that the software producer decided to trace; the last one tracks unhandled application exceptions with type, source, message and stack trace. All information can be filtered by application version and, depending on the context, by date and OS.

Data reports are displayed as charts and tables. Pie charts and tables display categorized summaries of information. Trends in time are shown as bar charts and devoted to quantify the total of session reports (general, custom or error reports). There isn't a possibility to view trends in a particular property, such as variations in OS or device. One interesting feature is the mix between the bar chart that shows trend in total reports and the line graph that shows the relative increase of new reports (Figure 4). This overlay increases data density, meaning a user can consume more information from the visualization.

In error report, a table of all exceptions lists detailed information about errors that have occurred in the application. There is no visual or other tool to quickly assess relations between errors and environment factors like OS or Architecture. Analytics has to be done by manipulating the filtering options.

Millimetrics is strong in quantifying environment information it collects, and although graphically limited, environment data can be easily assessed and proportions understood, at least while the number of different categories is limited. From this angle, the presentation using pie charts and tables is efficient. Its analytic weakness is in the failure to show trends other than the total reports collected that show the trend of usage numbers. There is also no support to unveil relationships between collected data, especially important when inspecting error reports. Therefore it was impossible to find the mechanism that could answer the question "which of my customers is having a specific problem?"

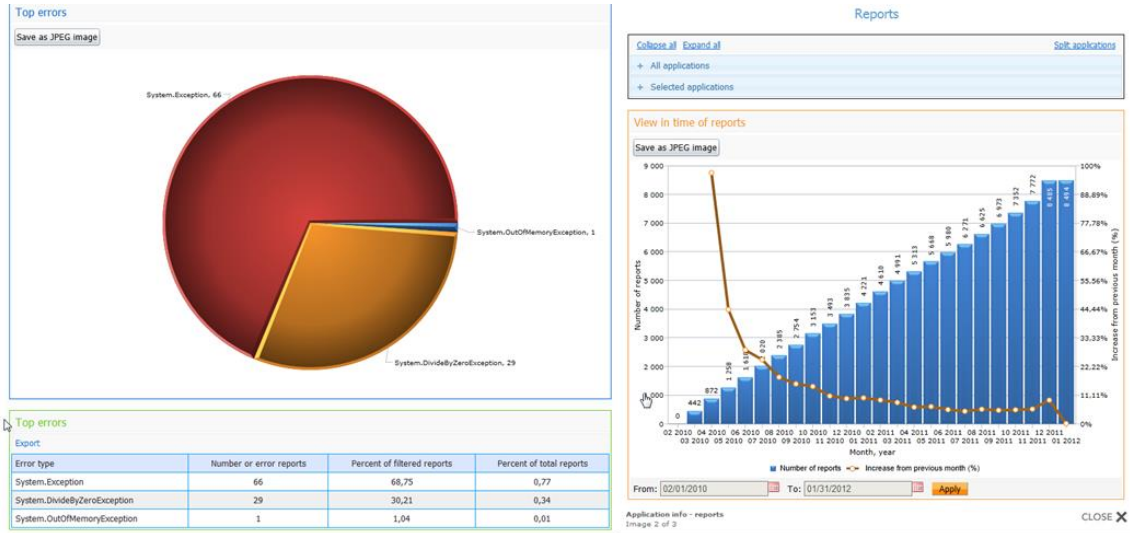
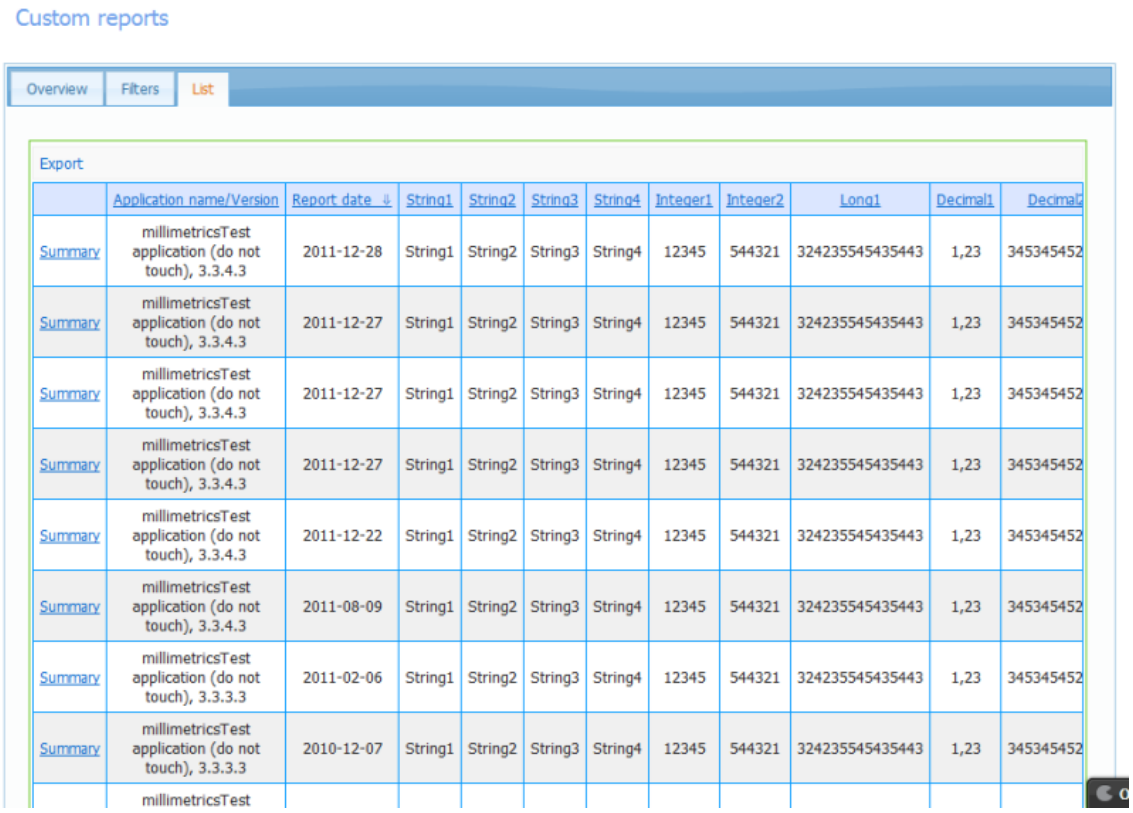


Figure 5 - “Top errors” and “View in time of reports” (*Millimetrics*)



	Application name/Version	Report date	String1	String2	String3	String4	Integer1	Integer2	Long1	Decimal1	Decimal2
Summary	millimetricsTest application (do not touch), 3.3.4.3	2011-12-28	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.4.3	2011-12-27	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.4.3	2011-12-27	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.4.3	2011-12-27	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.4.3	2011-12-22	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.4.3	2011-08-09	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.3.3	2011-02-06	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
Summary	millimetricsTest application (do not touch), 3.3.3.3	2010-12-07	String1	String2	String3	String4	12345	544321	324235545435443	1,23	345345452
	millimetricsTest										

Figure 4 - Custom reports (*Millimetrics*)

3.3 DeskMetrics

Deskmetrics is set around four main goals: to know users better – get information about the user and environment in order to guide decisions; to identify most used features – identity which features are vital. Discover the user's path within the application and the most used features; to grow user's engagement – make improvements in the product based on user's behavior and to grow customer satisfaction; to obtain new insights – recognize new opportunities comparing data over periods and track trends that can help understanding user's actions better. These four principles are well aligned with iUSE requirements. Therefore analyzing this product was imperative.

The API is organized around a general model: Track user events with associated metadata that sets the context for relevant runtime properties (e.g. DeskMetricsTrackEvent('Feature', '{"Real-time Module": "enabled"}')). This type of registering custom values (json array of key value pairs) is very flexible and has been adopted by numerous providers.

Deskmetrics records environment information such as OS and monitor count, and distinguishes between execution and installation metrics. There is no support for exceptions which is an important aspect in a desktop application (it existed in a former version). Desktop applications can run on multiple environments and an exception can be related to some recorded environment factor or context data. Therefore, desktop analytics should provide the means to collect and analyze exceptions.

The interface presents information in the form of tables, line and pie charts. The first thing that comes to the eye is that there is no legend (color coding). The user has to mouse hover around graph elements in order to know which property it is representing. The startup page is a dashboard that quickly shows some current parameters, such as sessions, users, top environment (OS, language, architecture, and memory size) and top countries. One interesting aspect is the possibility to show trends of a particular event or events and then, by selecting a specific event, the trend of its associated custom properties.

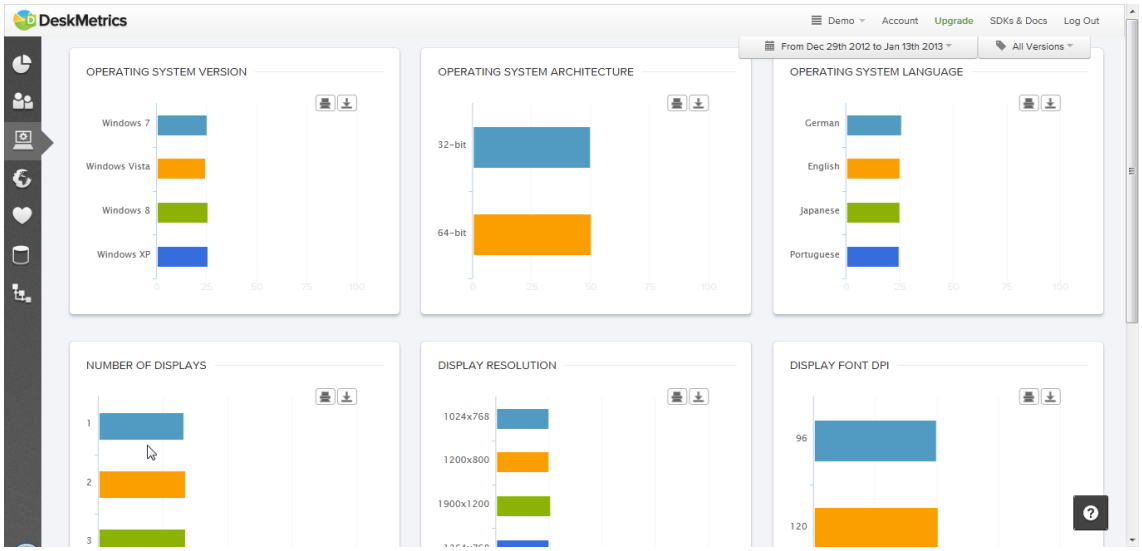


Figure 6 - Environment information (*Deskmetrics*)

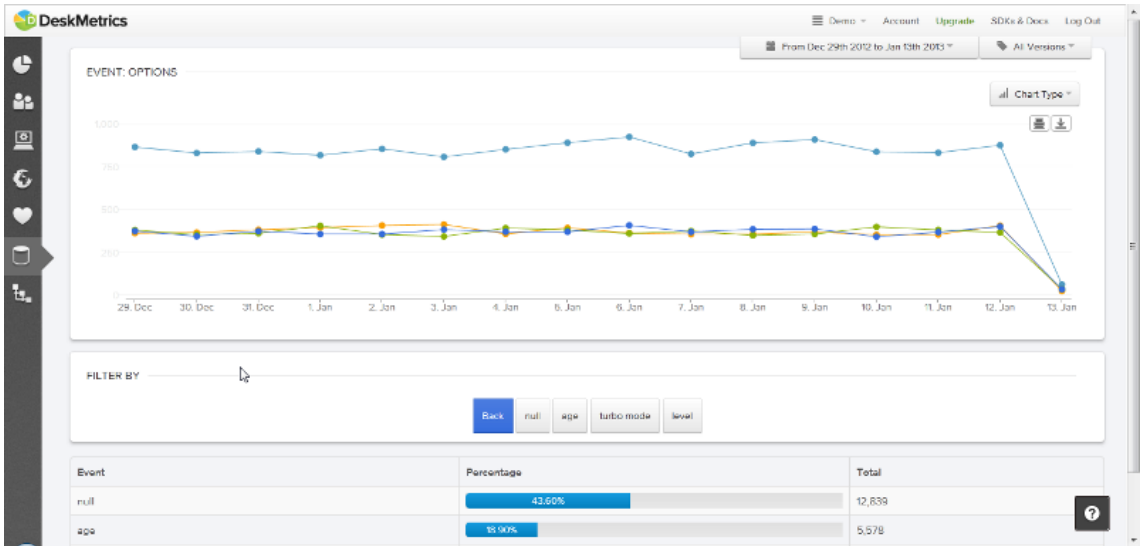


Figure 7 - Event trends (*Deskmetrics*)

3.4 TrackerBird

TrackerBird presents a dashboard with product activity (active users), top 10 versions, OS distribution, top 5 countries and the choice to view other analytics using line charts for trends, tables and pie charts for showing proportions (e.g. OS distribution). It has the enhanced ability to filter all presentation graphs by Country, Application Version, Language, OS type, OS language and License.

The new features, if compared to previous providers, are the tracking of the installed software license and the possibility to send messages (announcements, promotions or surveys) directly to the user's desktop. Messages are sent to all users that match to a target application usage profile. You may select a specific target audience using over 20 different filtering criteria such as geographic location, language, version, license status, application running time, days since installation, OS type, hardware profile, and so on.

Application Exceptions can be tracked but the provided analytics is a modest list of exceptions with context information (Product details, Operating System details, and Architecture details). There is no attempt to find patterns that could help to determine the root causes of an exception using any of the recorded context information. That inference has to be done manually by inspecting all logged exceptions. Not very helpful!

A useful configuration attribute in the integration API is the possibility to set different Privacy Modes (off, low, and high). These modes enable to collect from architecture and usage data, to architecture only, and don't collect. The Privacy Mode is selected by the user.

There isn't much support for custom data (e.g. App.EventTrack(string customText, double? customValue), and I haven't found where to explore or analyze it in the front-end.

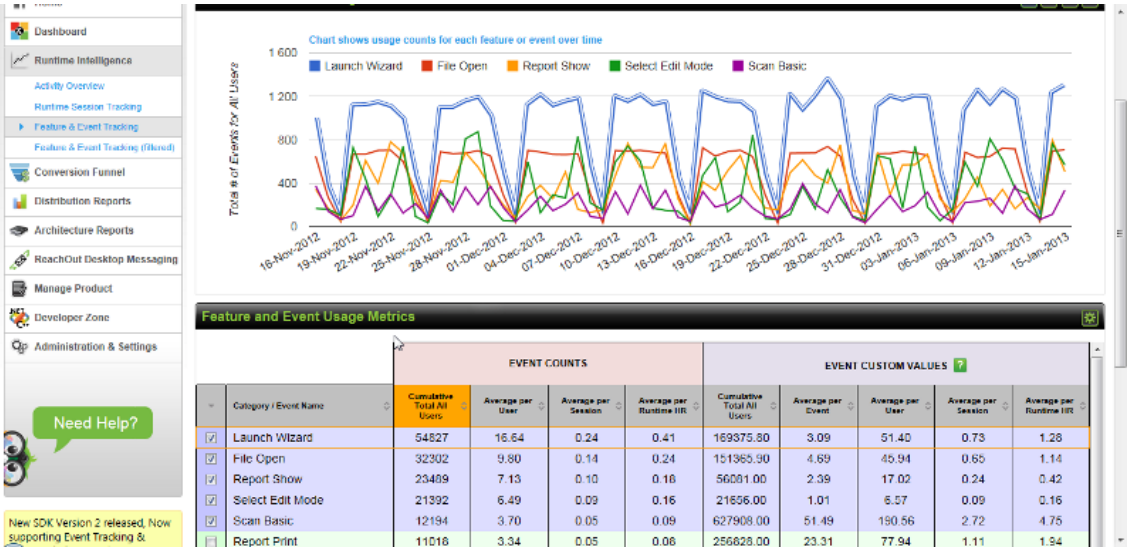


Figure 8 - Feature Events and OS report (*TrackerBird*).

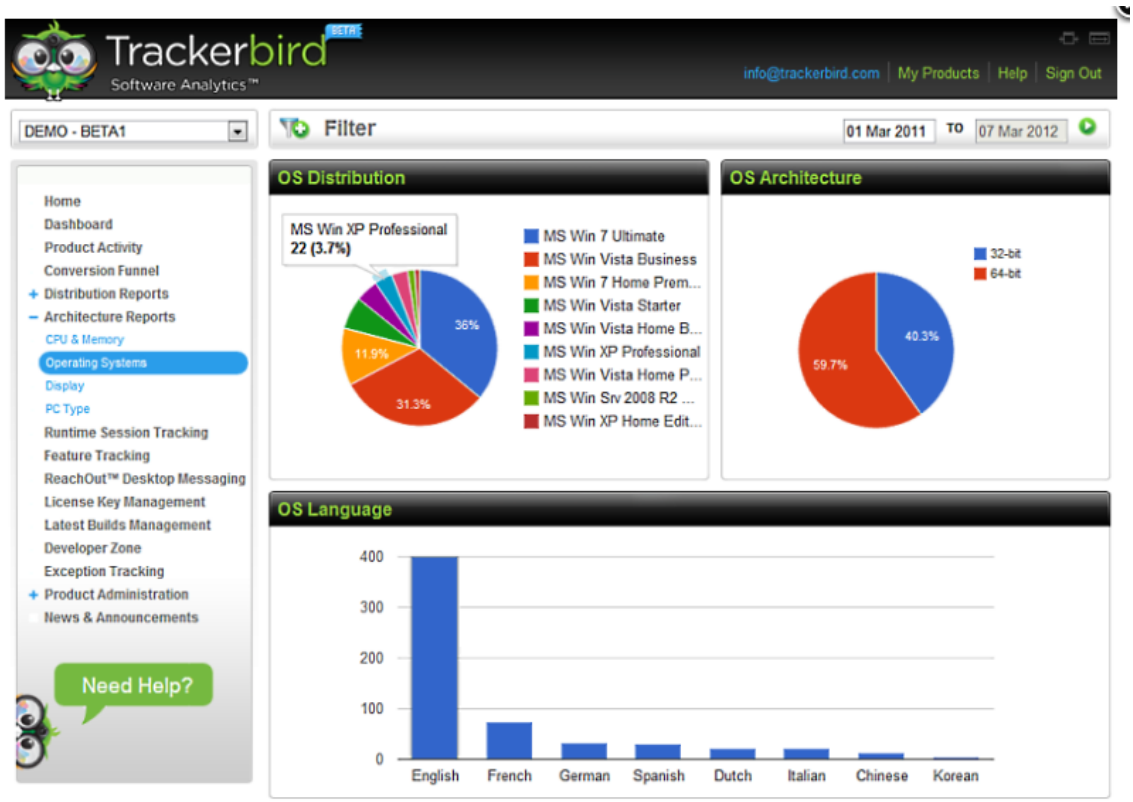


Figure 9 - Dashboard (*TrackerBird*)

3.5 UserMetrix

UserMetrix is perhaps the simplest provider in this set. The design is not polished but effective in communicating a snapshot of the implemented analytics. It fails though, in providing interactivity with the graphical elements. There isn't even the possibility to filter information by application version or date and the information granularity in trend is limited to monthly summaries. The initial dashboard presents summary information about total sessions, total users, trend in application usage (number of users/new users), most popular versions, distribution of OS represented in total sessions, most common errors and most common features.

There are three operational objectives that stand out from its features:

- *Spend time fixing bugs, not reproducing them* - Whenever customers send feedback to software developers, it often results in a long laborious process to reproduce and diagnose the problem. UserMetrix combines application analytics with traditional error reporting to determine the most likely reproduction steps for software issues. This allows software developers to focus on actually fixing problems, rather than reproducing them.

- *Learn what frustrates users* - Many of the customary feedback approaches allow users to report problems only when software crashes. This often means that vital information is easily lost when error reports are not sent. When UserMetrix is integrated with a 'shake' gesture or 'panic' button it can even collect information about when people are frustrated or confused, allowing to engineer better user experiences.

- *Focus development on what matters most* - Developing an application and knowing what to fix or implement next can be a tricky business. Prioritizing issues by severity is time consuming and often involves lots of guesswork. UserMetrix helps Product Managers by prioritizing development on importance, for example, using the "Most common errors" feature.

This is the only provider to truly implement some actionable intelligence about reported application exceptions. It uses patterns in the recorded event workflow to infer the most likely error reproduction steps. The other analytic features are very limited in functionality.

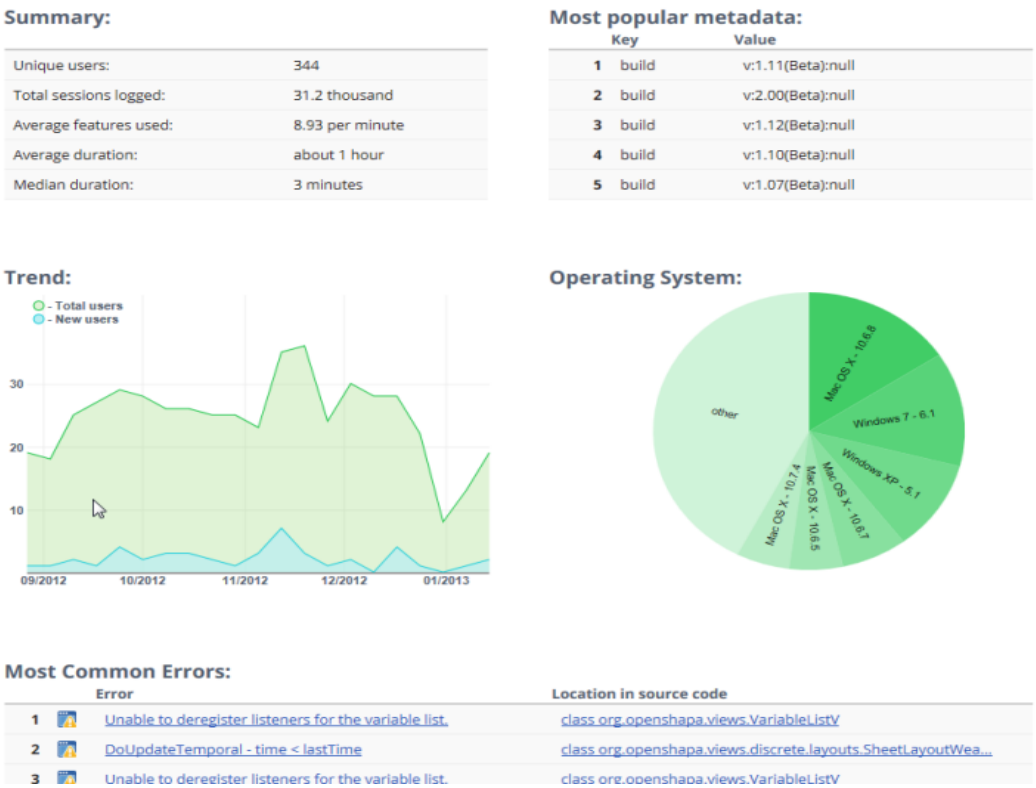


Figure 10 - Dashboard (*UserMetrix*)

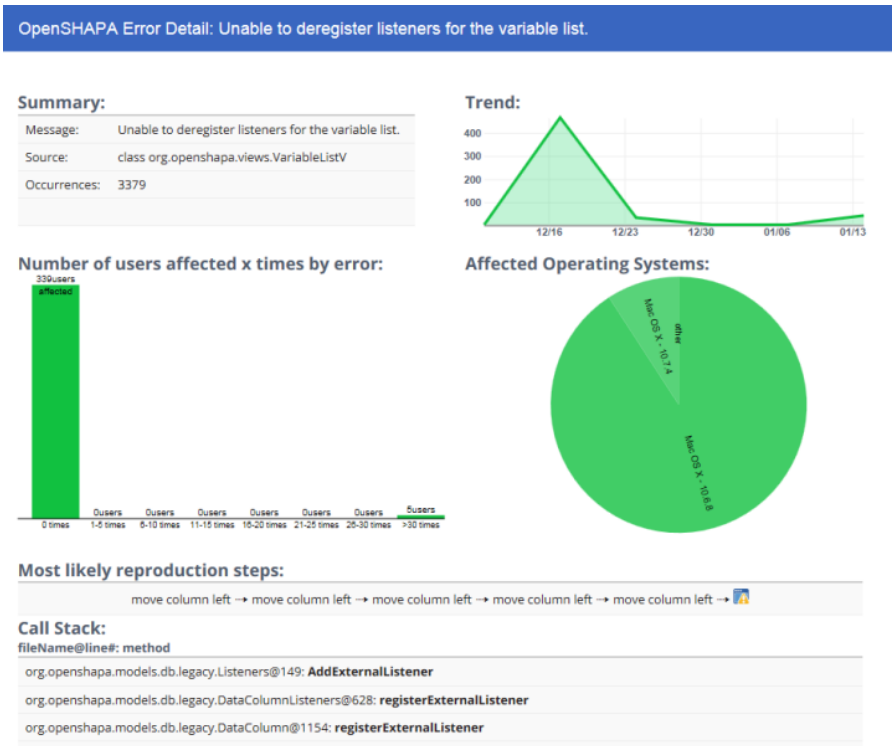


Figure 11 - Error detail (*UserMetrix*)

3.6 EQATEC

EQATEC is probably the most complete product in this market survey. It provides a full range of filter options and analytics, from Location, Version, Environment, Installations, New Version Notification, Log and Exceptions. It has the possibility to explicitly track an activity (event with duration) and if it was canceled.

One of the useful features is the session search where the system can find associated sessions by entering an installation ID. It could be useful to extend this functionality to search for values in other properties or to look for specific information about a user's session.

It has a flexible API that allows one to define categories of events using the dot notation (e.g. "Button.FeatureA" indicates that the feature "FeatureA" of category "Button" was used). The system will inspect this notation and create the different categories in the UI.

Exceptions are presented as a list of aggregated occurrences. For each exception type, a new case is open. An exception case can be closed or deleted, meaning respectively, that the exception has been fixed or the exception should not be presented anymore. However, if an older closed case is detected in a newer application version, the system reopens it, and it will reappear in the exception list. Generating a case for each exception type has the limitation of context unawareness. An exception type (ex.: System.ArgumentNullException) can be thrown from different code locations and for that reason it should be classified in distinct cases. For each listed exception case, there are details about the environment (OS, language, architecture) that can be inspected to help in finding a possible cause. Error reports can be viewed using line graphs to show total exceptions in period, total exceptions per session and the evolution of new exceptions in the period.

EQATEC supports other two interesting features to the world of desktop applications: *New Version* notification and *Remote Lookup*. The *New Version* feature can be used to inform users that a new version of the application is available and can be downloaded from a specific URL. For that to happen, the producer has to register a new version release in EAQTEC. The client API compares the current application version with the last registered version and fires an event to show the update information to the user. The *Remote Lookup* allows a producer to register a key/value pair and, in the client API, call the lookup method passing a key to receive a value stored at the EQATEC server. The returned value can then be used by the application as a configuration parameter or other.

One limitation compared to some of the other providers is the ability to record and analyze custom information. For example, Deskmetrics allows producers to store events and an arbitrary list of event properties (array of key/value pairs). EQATEC only provides an "EventValue" method to associate a long value with an event name. The sum of this value can then be visualized together with the event trend in time.

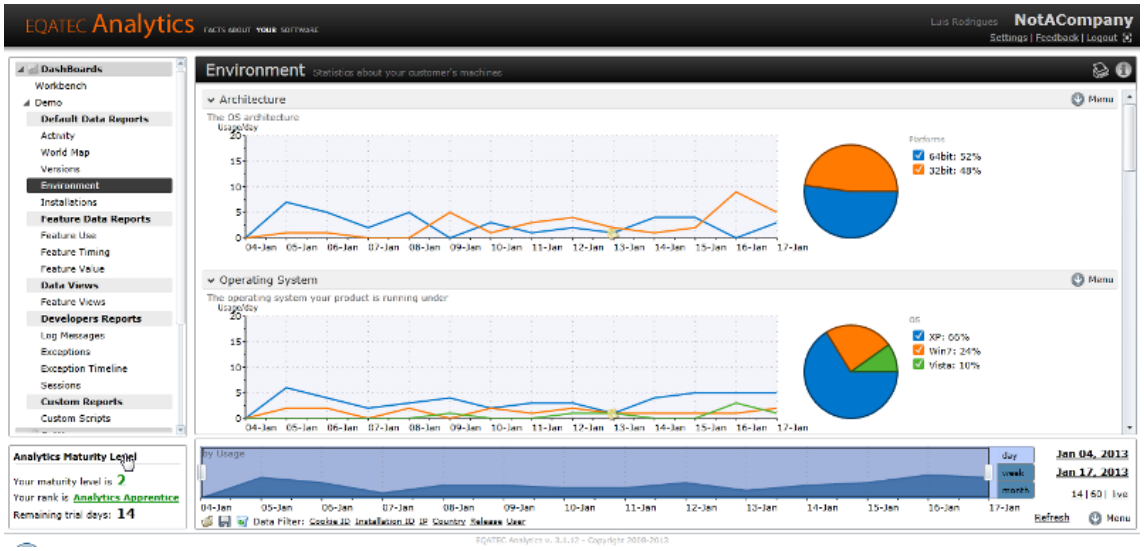


Figure 12 - Environment (*EQATEC*)

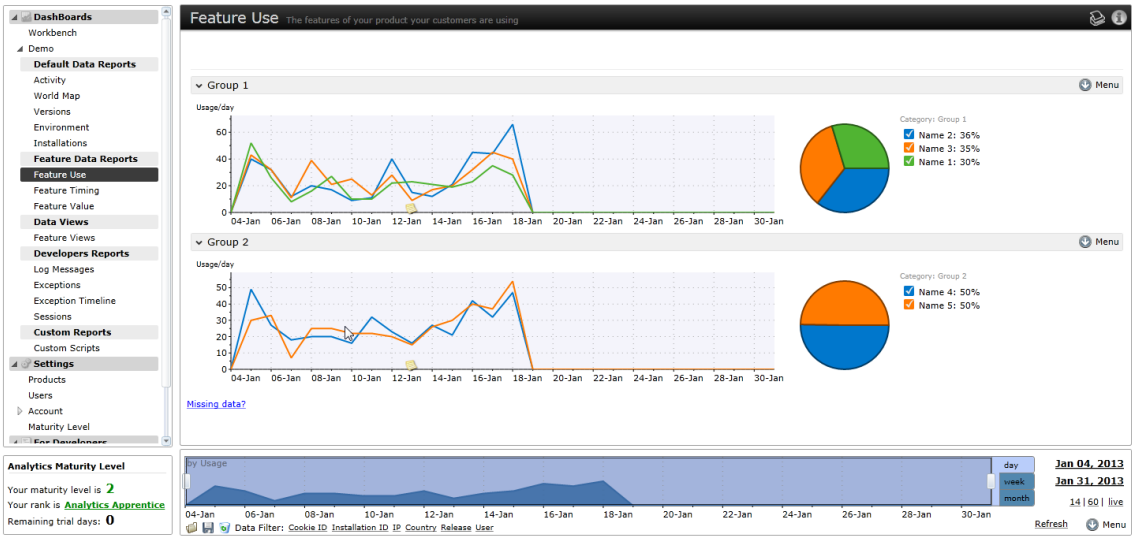


Figure 13 - Feature Use (*EQATEC*)

3.7 Mixpanel

Like *Google Analytics*, *Mixpanel* targets web applications. It offers funnels[5]; segmentation[6] and retention[7] tools that help to better understand user profiles and optimize applications for maximum user retention. The biggest difference between Mixpanel and Google Analytics is that Mixpanel emphasizes event tracking, and *Google Analytics* emphasizes page view tracking. The awareness that analytics based on actions and people is more important than page viewing is what makes it relevant to desktop analytics. It measures people's actions in the application.

Activity trends allow seeing user engagement week over week, month over month, and even hourly if you need the granularity. Similar to *Deskmetrics*, application usage is modeled as an event and each event can have a collection of associated properties that can be used to set context data. Trends can be visualized through well designed line charts, allowing a clear understanding of data points. Events can be filtered allowing for specific comparisons. By selecting a data point, a comparative table with values from earlier days and weeks can be used for enhanced period comparison. There is an added functionality when there's only one event filtered. In this case a trend of its associated properties is shown.

In this study, *Mixpanel* was the only provider that has shown an explicit section specifically designed for tracking user profiles. People analytics is a kind of analytics that reveals who the application customers are. It allows diving deep into a person's profile to see who he is and what he has done. There is also an important feature that is to push notifications to a filtered group of users based on collected profile properties (gender, age, location, and so on). This feature is different from *Trackerbird's* Richout Desktop Messages in the type of user profile. MixPanel uses people properties (e.g. age, gender), not like *Trackerbird's* that uses application properties usage (version, days since using application, etc.)

Other interesting feature is the ability to add notes and create bookmarks on data for later reference.

---

[5] Improves conversion rates by identifying where customers are dropping off with funnel analysis. This report allows you to answer questions like: "How customers that come from a certain ad campaign converting are?"

[6] Segmentation is a powerful and flexible way to slice & dice data. Segmentation can answer questions like: "What does age distribution (not average) of people that came from Twitter who uploaded a video look like?"

[7] Visitor retention is a metric that can help to identify if an application is "bleeding" users. It also helps to determine whether an application is valuable to them or not by showing you how often they come back.

Event	Today	Mon, November 28	Mon, November 21	Mon, November 14	Mon, November 7
Landing Page Loaded	158 +7.4%	190 +25%	152 +22.6%	124 +69.9%	73 -2.7%
Tutorial Viewed	100 +4.2%	124 +39.3%	89 +18.7%	75 +44.2%	52 +13%
Account Created	38 -1.8%	50 -5.7%	53 +76.7%	30 +25%	24 -20%
Song Played	15 +21.1%	16 +100%	8 -11.1%	9 +28.6%	7 -22.2%

Figure 14 - Events overview (*Mixpanel*)

Event	Today	Mon, November 7	Mon, October 31	Mon, October 24	Mon, October 17
Player Create	7,012 +73.1%	4,052 +100%	0 0%	0 0%	0 0%

Figure 15 - Event Period Comparison (*Mixpanel*)

3.8 Conclusions

One of the main goals of this comparative study was to see how analytics providers show quantitative and qualitative data evidence through graphical methods. A first look into the selected subjects shows a common approach to interface design and statistical presentation. All use the basic pie charts, bar charts and tables to present distribution of categories, and line charts to present trends in a time period. These standard methods of statistical presentation are well understood by the target users of this type of analytics framework. For that reason, they are very effective on showing information if limited to a small number of variables.

All providers show a maximum of two variables in distribution or trend graphs. In pie/bar charts, one variable (dimension) is shown using circle angles/bar length. In trends two variables are used, with data points distributed in an x-axis showing the time variable and quantitative values in the y-axis indicating a magnitude variable. Multiple lines are displayed simultaneously for comparative purposes. Color coding is used to associate a graphical element (bar, pie, slice or trend line) with a property or category name. Mixpanel is the only provider to implement some interactivity with graphics (see Figure 15) by showing a contrast from the selected data point value with values from different periods.

In terms of information visualization and in spite of good support on analyzing trends in time and top usages, the surveyed products did not provide any means to visualize usage patterns and relationship analysis. Also, information was visually scattered throughout multiple views, making it difficult to establish relations between views.

There is also a common denominator to the majority of the providers that is interface clutter – the extent of the interface occupied by the available analytic menu options is an example, almost asphyxiating information visualization graphics.

Each of the providers has a feature in which it stands stronger when compared to the others. Next are listed the features that were collected from all subject providers, during this study, and that were considered the most useful in a desktop analytics framework:

- Use key/value pairs for custom values. (*Deskmetrics*)
- Show trends of a particular event or events, and also, the trends of its associated properties. (*Deskmetrics*)
- Add the installed software license as a tracking attribute. (*Trackerbird*)
- Send messages (announcements, promotions or surveys) directly to the user's desktop. (*Trackerbird*)
- Add a privacy mode property into the API (off, low, and high). (*Trackerbird*)

- Use patterns in the recorded event workflow to infer the most likely error reproduction steps. (*UserMetrix*)
- Track Activity duration and if it was canceled. (*EQATEC*)
- Provide a Search option for a specific session ID. (*EQATEC*) – Useful to add the capability to search other properties.
- Manage exceptions as "cases". An exception case can be closed or deleted, meaning that the exception has been fixed or the exception should not be presented anymore. (*EQATEC*)
- Show comparisons with values from earlier days or weeks for enhanced period comparison. (*Mixpanel*)
- Push notifications to a filtered group of users based on collected usage profile. (*Mixpanel*)
- Create bookmarks on data for later reference. (*Mixpanel*)

Chapter 4

Requirement Specification

4.1 Introduction

This chapter provides the reader with the requirements specification that supported the implementation of *iUSE* Runtime Intelligence Framework and its objectives (see 1.2). It starts with a brief description about the stakeholders, exemplifying in what way *iUSE* analytics could help with their responsibilities (4.2). Section 4.3 shows the overall system architecture, describing each component in its responsibilities and interactions. Functional requirements are enumerated in section 4.4 and include data point specification, questions the system must address, and the integration of *iUSE* with the semantic world and its linked-data world. Section 4.5 contextualizes the project in terms of a Visual Analytics Agenda and states the technology preconditions and recommendations. It concludes with Visual Analytics Dashboard that describes the visual models, their interactive and visual mappings, usability and interfaces (section 4.6). The next figure shows an overview of *iUSE* framework.

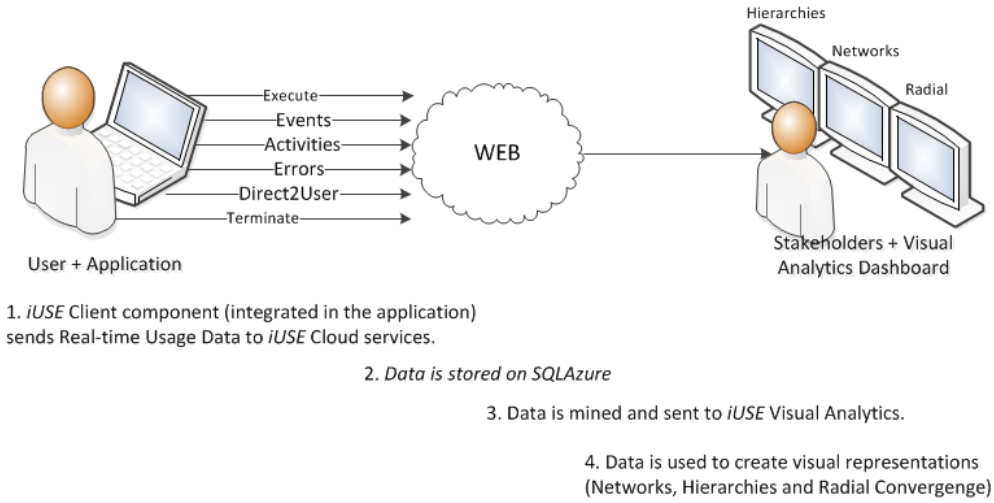


Figure 16 - *iUSE* overview.

4.2 Stakeholders

The stakeholders are the "actors" of the R&D department of software development organizations – Managers and Developers. These stakeholders are also the users of *iUSE* analytic features.

- *Managers* – Are responsible for the Research & Development activities within the company and for managing the product catalog and feature set. A manager is expected to take strategic decisions affecting the product roadmap. Thus it is crucial to have a clear and current view of the product usage to take informed decisions. *iUSE will assist in decision making and provide the means to better know users, their environments and profiles, and assist on R&D priority and investment decisions. It will*

provide insights about product activity and feature usage trends and provide the organization with a direct communication channel with its users via their application.

- *Software Developer* – Is technically responsible for the implementation of the product features and bug fixing. *iUSE will alert to existing application exceptions and suggest the most likely environment factors and reproduction steps for those exceptions*.

4.3 System Architecture

The system will be divided into Client and Server services as follows:

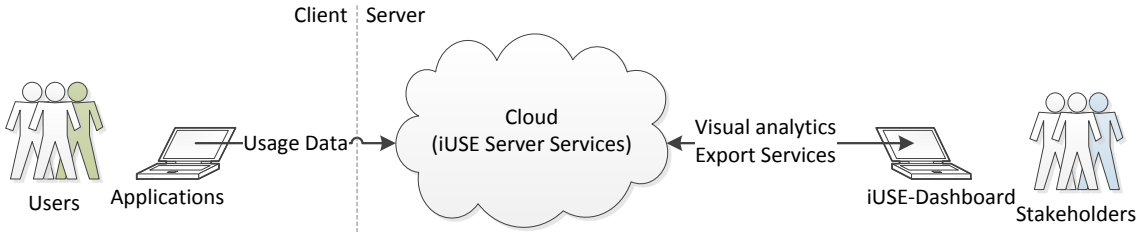


Figure 17 - *iUSE* architecture.

Client component provides a utility to unobtrusively collect and send runtime usage data (user, session, and device information) to cloud services. Integrators will incorporate this component on their applications to submit information (in real-time) and to capture environment information for each user session. It is vital that application performance and user experience are not affected by the data collector execution utility. For that reason, the collector utility will run on a separate application thread.

Cloud Services are the server components responsible for data and visualization mining services. Data services provide the entry point to collected runtime data persistence, mining and exporting services. The stakeholders will access the analytical features of *iUSE* anywhere and anytime by accessing a web dashboard. Cloud services will provide data services to *iUSE* – such as:

- *Data Storage* – Data storage model and technology will take into account scalability, performance and storage costs associated with large data sets. It will be considered as a requirement that, where conceivable, data processing (e.g., aggregations) be performed upfront on receiving data to have a minimal impact on responses to the end user analytics experience.

- *Web endpoint* – Integrators will submit data from their applications to a web server, and *iUSE* Visual Analytics Dashboard will request mined information to feed the visualization models.

- *Export services* – Organizations will access collected raw data in a standard format. The adopted standard will provide integration with the semantic web and its linked-data world (see 4.4.3).

4.4 Functional Requirements

Data Point Requirements (see 4.4.2) (see Market Survey)

4.4.1 Core Features

iUSE core features are grouped into five major functional categories: *Product*, *Environment*, *Features, Exceptions* and *Messaging*. This grouping reflects the different set of questions stakeholders expect the system to answer. Questions, such as "*What is the Operating System? Is it a 64 or 32 bit OS?*" are frequent among developers when reported issues escalate to R&D. These and other questions were identified by interviewing the stakeholders and documented in their profiles (see Personas).

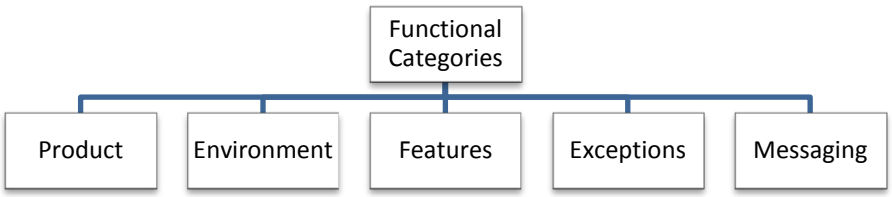


Figure 18 - Functional Requirements.

Monitor Product Activity – Tracks how many unique users have installed/uninstalled or actively use the software; compares product activity between different Versions/Editions of software; analyzes how often and for how long users interact with the software to identify their behavior and reliance to the product. *iUSE* must address the following product related questions:

- What versions of the application are used and how popular is each version?

- How many times per day do users run the application and how long is a typical runtime session?

- Are users switching to the newly released versions fast enough?
- How many users are stuck using an old build?
- How many customers would be affected if you had to stop supporting a particular build or product version?
- How many users are being affected by that bug you have found in version X?

Collect Environment Data - Get distribution insights on Operating systems, languages, hardware architecture, display resolutions and machine types. Environment data helps on prioritize development and testing for customer base platforms and architectures. *iUSE* must address the following environment related questions:

- What machines and platforms is the software running on?
- Is it worth fixing a feature affecting Win XP or are there only a few users using it?
- Should you adjust your UI to better support notebooks or should you focus on widescreens and dual monitors?

Feature Usage Trends – Track which product features are more popular among customers and which are underused. Define where to focus development efforts either on improving or removing unused features. *iUSE* must address the following feature related questions:

- Which features should get higher priority in development?
- What product features are left undiscovered by evaluation users?
- Are customers using the software only for a specific feature-set?
- If you had to stop maintaining a particular feature, would anyone be affected?

Track Application Exceptions – When running, software encounters scenarios never tested or imagined, therefore exceptions will most likely occur. Proactively problems should be identified and fixed before they could be reported. *iUSE* will allow reporting on exceptions by collecting critical information about the software such as version/edition, the classes and methods which generated the exception as well as the running environment such as machine architecture and operating systems. *iUSE* must address the following application exceptions related questions:

- What are the most common exceptions?

- What unhandled errors does the application cause?
- Which customers are experiencing a specific problem?
- What are the most likely environmental/reproduction steps for a specific software issue?

Direct-to-desktop Messaging – It is a communication channel through which producers can easily deliver new updates, marketing announcements, informational and promotional messages or even surveys to end-users who are running the software. Producers may select a specific target audience using different filtering criteria such as language, version, edition, license status, OS type, hardware profile, etc. Messages are delivered to end-users with full control on how and when users see messages (e.g., embedded within application dashboard). Producers use the *iUSE* Client API to retrieve the message contents and display it within the application. It should be possible to configure messaging to target specific profiles, making the following scenarios possible:

- Shorter and targeted surveys. With direct-to-desktop surveys and collected runtime intelligence (software, OS, language, and hardware profile they are running) producers will be able to customize a survey to a specific audience.
- Send out bug fixes or new version announcements only to users who are running affected builds.

4.4.2 Data Point Requirements

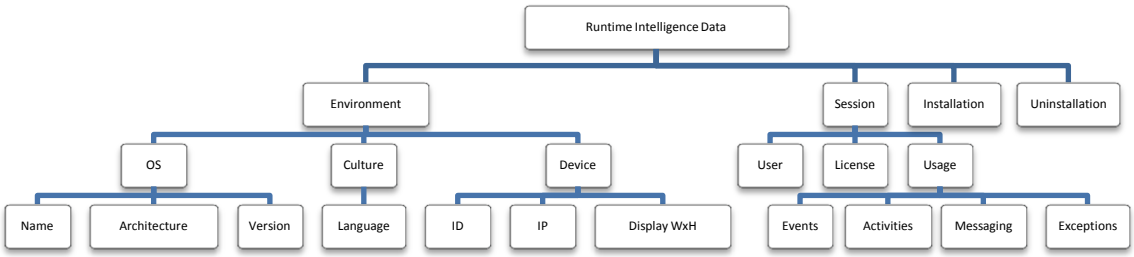


Figure 19 - Runtime Intelligence Data.

In order to measure development success, there are two things of key importance: implementing the means to collect application runtime data in a way that don't interfere with normal application execution; offering smart data, which means that the resulting analysis should contribute to increase application value and customer satisfaction.

To create valuable analytics, the client services will collect two types of data – context and execution data. The context data collects information about the device environment in which the client application is running. The execution data is information about application usage.

The defined analytic core features (see 4.4.1) together with tables Table **2** and Table **3** worked as a blueprint to the defined data point requirements. Table 2 maps objectives, data requirements and the instrumentation and analytics technologies selected for "developer as the customer". Table 3 summarizes the key features between Web and Desktop analytics flavors.

Table 2 - Common runtime data points and pivots.[8]

Objectives	Sample Data Points (Data Worth Collecting)	Sample Filters (Data Filter and Query Criteria)
Adoption and Activity	• Unique users • Sessions • Session Duration • Installation • Uninstallation • Geolocation (by IP)	• User • License • Device • OS • Application version • Geolocation
UX Improvement	• Features used • Feature duration • Feature canceled • Feature usage sequencing • Display resolution • Data values (user-entered values or other runtime data bindings)	• Feature used • Feature canceled • Data values
Quality of Service	Exception reports/stack trace	Exceptions by version, stack and feature usage
User Profiling and Support	• Integration with CRM service • Direct-to-Desktop Messaging	• License • Device

[8] Based on Holst, S. (2011). "Application Analytics: Why Is the Developer Always the Last to Know?" from http://visualstudiomagazine.com/Articles/2011/07/01/pfven_App-Analytics.aspx?Page=1.

Table 3 - Application vs. Web analytics focuses and features.[9]

	Application Analytics	Web Analytics
The Customer	Development	Sales and Marketing
Representative Use Case Scenarios	• Measure adoption and activity • Improve UX • Track exceptions and other Quality of Service indicators • Simplify and improve support • User profiling	• Measure page views and user clicks • Target advertising • Track user conversions • User profiling • Usage metering
Platform Support	All runtime surfaces including mobile devices, desktops, server-side and cloud-based runtimes	Browser and/or mobile device only
Data Requirements	Complex objects and application-specific data, including custom types and stack traces	Primitive types with defaults focused on sales and marketing requirements
Precision	Method-level	Presentation layer events (clicks, page views)
IDE and ALM Tool Integration	High priority	Low priority
Privacy, Identity and Security Considerations	• Opt-in policy enforcement • Developer's own content • Repository can be local or hosted	• NA • Ad and analytics service provider's own content • Repository is hosted by ad and analytics providers

[9] Holst, S. (2011). "Application Analytics: Why Is the Developer Always the Last to Know?" from http://visualstudiomagazine.com/Articles/2011/07/01/pfven_App-Analytics.aspx?Page=2.

The following tables describe the resulting data point requirements: Table 4 defines context data and Table 5 defines execution data *iUSE* will collect in order to support the functional requirements.

Table 4 - Context data points.

Data	Description
OS	Identifies operating system platform (e.g., "Windows 7"). *iUSE* client component will collect environment information from the device the application is running on. Environment information includes properties of the operating system that can influence the execution of the application, such as: • **Name** - Operating system name (e.g., "Windows 7"). • **Architecture** - Operating system architecture (e.g., 32/64). • **Service Pack** - Operating system Service Pack (e.g., "Service Pack 1").
Culture	Identifies system culture (deduces user culture and other regional settings). Environment information includes culture properties that can influence the execution of the application, such as: • **Language** - Identifies language using the LCID string table (e.g., "pt-PT").
Device	Identifies applications environment. Environment information will include device properties that can help on UX improvement, such as: • **Display WxH** - Main display resolution. (e.g., "1024x784"). Environment information will include device properties that can help on User profiling and Support, such as: • **Device ID** - Identifies the device. The default Client implementation will use disk serial number as the Device unique identifier. • **Device IP Address** - Identifies the device IP address, used to estimate the user location if geolocation is not available. • **Geolocation** - Location where the user is executing the application. If longitude and latitude are not specified, the system will use the device IP address to estimate user's location (e.g., using freegeoip.net web service[10]).

[10] http://freegeoip.net/ is a public RESTful web service API for searching geolocation of IP addresses and host names.

Table 5 - Execution data points.

Data		Description
API Key		Uniquely identifies the owner of an *iUSE* account. *iUSE* provides the apiKey to the integrator when a client creates an account. The integrator will specify this key when communicating with cloud services.
Session		Uniquely identifies each session that is being tracked by *iUSE*. Client component creates a session token (GUID) when application starts, and all subsequent collected data is associated with it.
User		Identifies the user running the session. Integrator is responsible for setting the user identifier according to business and analytical requirements. Examples of user identifiers are: custom application user identifier, email, OpenID, License User Serial Number, etc. This property will work in combination with session ID to determine unique executions. If the integrator is able to provide a user identifier that uniquely identifies the user on each session, then *iUSE* will be able to differenciate between executions and unique executions. For example, on a specific day the system could log 100 sessions, but a relevant question is: from how many different users?
License		Identifies the license of the running session. Integrator is responsible for setting the license according to business and analytical requirements. Examples of license identifiers are: license serial number, customer id, etc.
Application		Identifies the running application. Application information will include properties that can help on measure adoption, such as: • **Name** - Identifies the application being tracked. The integrator must register application on *iUSE* to be able to log tracking information. This property will be the main analytical filter (e.g., "NGCO"). • **Version** - Identifies the application version that the user is executing. This property will be used to track version adoption trends (e.g., "12.1").
Usage		Measures adoption and activity by tracking how users interact with the application and its features. Usage information will include:
	Events	Identifies an application event. An event occurs at some point in time and does not have duration. Application events will be characterized by the following properties: • UTC **Date/time** • **Category** (e.g., "menu") • **Label** (e.g., "invoices_click") • **Feature** (e.g., "invoice")

	Activity	Identifies an application activity. An event occurs at some point in time during a period of time. An activity can be cancelled, started by an event or other activity or process. Application activities will be characterized by the following properties: • UTC **Date/time** • **Category** (e.g., "e-commerce") • **Label** (e.g., "send invoice pdf") • **Feature** (e.g., "#invoice") • UTC **Date/time ended** • **Canceled** (true/false)
	Messaging	Identifies a direct-to-user message. The message can be defined to target a specific user or application profile. Application messaging will be characterized by the following properties: • UTC **Date/time** • **Category** (e.g., "UX Survey") • **Label** (e.g., "New Invoice UI Survey") • **URL** (e.g., "https://pt.surveymonkey.com/UXSurvey")
	Exception	Identifies an application error. Exception information will include properties that can help mitigate or fix quality issues, such as: • UTC **Date/Time** • **Source** (e.g., "business.createInvoicePDF()") • **Message** (e.g., "out of memory") • **Stack Trace**
Installation Uninstallation		Identifies when an application was installed or uninstalled. The only information captured during an installation or uninstallation session is device information.

4.4.3 Linked Data World

Wikipedia defines Linked Data as "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs[11] and RDF[12]." Linked Data has one remarkable property: It may be easily combined with other Linked Data to form new knowledge.

Ontologies are developed to facilitate knowledge sharing and reuse by people and software. Gruber (1993) defines ontology as a "formal, explicit specification of a shared conceptualization". They are a commitment to an abstract representation of knowledge in the

[11] URI is used to unambiguously identify a resource (for example a person).
[12] Resource Description Framework, an official W3C Recommendation for Semantic Web data models.

domain of interest, a commitment to how a thing is represented: a class (general things), the relationships between those things and the properties (or attributes) those things may have.

iUSE data is already on the Web, but in what regards its visibility to the Web of Data (World Wide Web global database) it is has if it continues isolated in its proprietary container. By adopting Linked Data standards *iUSE* frees its data from its silo (relational database) so that it may be found, shared and combined with other people's data. Entities such as Companies, Countries, Industry Terms, Organizations, People, Products and Technologies stored in *iUSE* could be combined by smart applications with other information on the Web of Data, for example using the Linking Open Data project. The Linking Open Data project is a community activity started in 2007 by the World Wide Web Consortium's Semantic Web Education and Outreach (SWEO) Interest Group. The collection of Linked Data published on the Web is referred to as the "LOD cloud" and currently consists of more than 300 datasets from various domains[13].

iUSE will formally describe the entities and relationships that underlie the framework to share a common understanding of its information semantics among people or software agents (Musen 1992; Gruber 1993): with RDF serving as the foundation, RDFS and Web Ontology Language[14] as the core representation languages of the Semantic Web.

From the surveyed Ontology stores[15] no suitable match to accommodate the domain of *iUSE* was found. However, some ontology models (i.e., FOAF, OGP, GEO, EVENT, and OPENID) are relevant to *iUSE* because they are close related to its entities. The Enterprise Ontology[16] contains some intersecting concepts that will be integrated (see Table 6).

Table 6 - *iUSE* and EO common concepts.

Term	Description
ACTIVITY	This is intended to capture the notion that involves actual doing, in particular including action. An ACTIVITY can have happened is the past and may be happening in the present. The concept of activity is closely linked with the idea of the DOER, which EXECUTES an ACTIVITY SPECIFICATION. An ACTIVITY is linked to a TIME INTERVAL to refer to when ACTIVITIES are performed.
EVENT	This is something that happens in a TIME POINT, a particular, instantaneous point in time. A DOER triggers the EVENT.
DOER (*iUSE* User)	A DOER may be a PERSON, ORGANIZATIONAL UNIT or MACHINE.

[13] See http://www4.wiwiss.fu-berlin.de/lodcloud/state/ for details on the LOD cloud.
[14] *OWL Web Ontology Language Guide: W3C Recommendation 10 February 2004*. W3C (2004-02-10).
[15] DMOZ, OOR, Protégé Ontology Library, LearningStation, TONES, NEPOMUK, Mathieu d'Aquin, Natalya F. Noy, Where to publish and find ontologies? A survey of ontology libraries.
[16] Develop within the Enterprise Project, a collaborative effort to provide a framework for enterprise modeling.

4.5 Technology

Considering the nature of software development organizations, there isn't any limitation on technology requirements. Technology should be the most appropriate to successfully implement the product's functional requirements and to guarantee that future enhancements and support are easily implemented.

The NVAC's R&D agenda for visual analytics addresses technical needs for multiple areas, as well as recommendations for speeding the movement of promising technologies into practice (Thomas & Cook, 2006). The Visual Analytics Agenda addresses technical needs, such as scalability, that will be considered for the implementation of *iUSE* Visual Analytics Dashboard:

- *Information scalability* – Information presentation will scale and adapt to the audience. Relevant information may appear at a variety of scales, and the user will be able to change between scales in a way that is easy to understand and track.

- *Visual scalability* – Visual scalability is the capability of visualization representation and visualization tools to effectively display large data sets, in terms of either the number or the dimension of individual data elements (Eick, S. G. & Karr, A. F., 2002). Implementation will investigate adequate support for quality of visual displays, the visual metaphors used in the display of information, the techniques used to interact with the visual representations, and the perception capabilities of the human cognitive system.

- *Display scalability* – Implementation will develop techniques that scale to a variety of display form factors to take advantage of whatever capabilities are available to support analysis. One major challenge is to use consistent visualization and interaction techniques regardless of display's size.

Implementation will use the stack of Microsoft technologies spanning from client to server tools and frameworks. This particular choice of technology will not hamper adoption of *iUSE* because of its architecture. A REST Web API will be ultimately the interface between integrators and *iUSE* cloud services (see 4.3) – the only requirement is to be able to use internet services using HTTP.

Client services will be implemented as .net 2.0 components. The .net framework has a broad adoption and support on desktop operating systems, and easy integration with legacy development tools, such as COM components. Application integrators can implement proprietary client components, on any technology; as long as they follow the *iUSE* REST Web API specification.

iUSE server components will be implemented using the Azure cloud computing platform. Windows Azure is a Microsoft's cloud-based platform for developing, managing, and hosting applications off-site. Azure supports open standards and Internet protocols, such as HTTP, XML, SOAP, and REST. There are SDKs for Java, PHP and Ruby, for applications written in those languages, and Azure tools for Eclipse.

Collected data will be stored on SQL Azure (a component of Azure platform). Azure Storage provides high availability and reliability with redundant copies and automatic failover.

The web end-point will be implemented using ASP.NET Web API[17]. HTTP is simple, flexible, and ubiquitous. Almost any platform has a HTTP library, so HTTP services can reach a broad range of clients, including browsers, mobile devices and traditional desktop applications.

In what concerns *iUSE* Visual Analytics Dashboard, one of its requirements is to be an application for multi-platform audience with access anytime and anywhere. Also, it plans to reach its audience on the go with devices such as tablets. With HTML web based applications, the application is reachable from anywhere at any time and, with HTML web-based mobile applications, the application consistently displays across mobile web browsers, including future devices.

HTML5 is the next generation on web technologies, enabling web applications to be built with rich user interfaces and no plug-in requirement for rich multimedia experiences. With broad support from Apple, Mozilla, Microsoft and Google, all the major browsers have rapidly incorporated HTML5 features. HTML5 adds new audio and video capabilities, an immediate 2d bitmap drawing area and 3d rendering using WebGL and a set of API's to access device environment, such as local storage and audio synthesis. CSS3 modules have an impact on every aspect of presentation (such as rotation and scaling) and developers can use CSS3 to specify a style by device – for example to differentiate styling between PC and mobile devices using @media rules. Some of the most powerful CSS3 modules for application developers are "transitions" and "animations", often supported by GPU acceleration, resulting in an enormous performance gain when compared to equivalent JavaScript animations.

iUSE Visual Analytics Dashboard will be implemented with HTML5 technologies. Visualization models will be implemented using one important addition to the HTML5 specification – integrated SVG technology. SVG is used to describe Scalable Vector Graphics, a retained mode graphics model that persist in an in-memory model that can be manipulated through code resulting in re-rendering. Similar to HTML, SVG is built into the document using elements, attributes, and styles. When the <svg> element is first introduced into the document, it behaves much like a with presentation attributes that can be styled with CSS styling rules. Another key differentiating factor of SVG is the ability to code interaction without complexity. Just as SVG has a programmable DOM like HTML, it also has an event model. This integration with the DOM enables high interactivity with visual elements.

[17] ASP.NET Web API is a framework to build HTTP services on top of the .NET Framework.

Figure 20 provides a performance comparison between the two 2d drawing technologies of HTML5, the Canvas and SVG. Considering that between the goals of Visual Analytics Dashboard are high information density in full-screen mode and high interactivity with visualization models and display scalability, SVG will be the presentation technology.

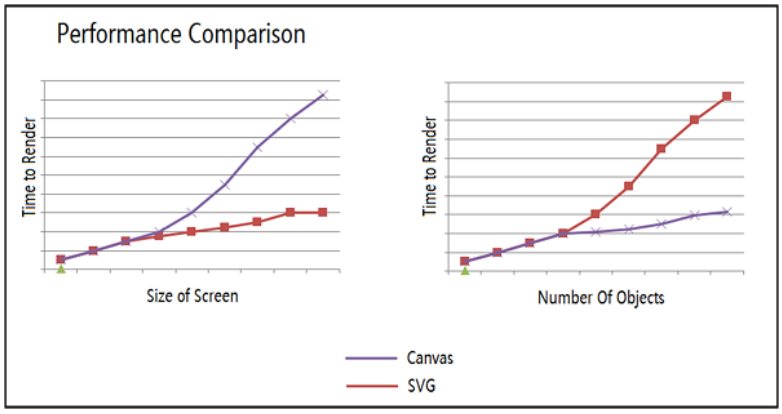


Figure 20 - Canvas vs. SVG performance comparison (MSDN, 2013).

In what concerns the implementation of data visualization (see 4.6), Data-Driven Documents (D3) will be used. D3[18] is a JavaScript library with the central tenet to make visualization easier without introducing a new way of representing an image. D3 uses existing standards – namely HTML, CSS and SVG. With D3, designers selectively bind input data to arbitrary document elements (e.g., Circle, Path) applying dynamic transforms to both generate and modify content, typically using JavaScript (Michael Bostock, Vadim Ogievetsky, & Jeffrey Heer, 2011). D3 is different from other graphical libraries, besides ingeniously architected, it separates data from presentation. D3 provides a series of mathematical models that can be used to graphically represent data, using the developer choice of presentation technology (e.g., HTML, CANVAS, WEBGL, SVG, etc.). It also has a great support for defining Colors (RGB, HSL and other color spaces) and Scales (Quantitative and Ordinal Scales), used to define color ranges (integrates ColorBrewer sets) and important in defining the range of graphical dimensions in the domain of the data they represent. *iUSE* will use three of those mathematical models to help depicting its visual analytics models:

- *Force Layout* –Will be used to create network visualizations (see 4.6.1). A flexible force-directed graph layout implementation using position *Verlet integration*[19] to allow simple constraints. This implementation uses a *quadtree*[20] to accelerate charge interaction using the *Barnes–Hut approximation*[21]. In addition to the repulsive charge force, a pseudo-gravity force keeps nodes centered in the visible area and avoids

[18] Introduction to D3: http://mbostock.github.io/d3/talk/20111018/#0
[19] Verlet integration is a numerical method used to integrate Newton's equations of motion.
[20] A quadtree is a two-dimensional recursive spatial subdivision.
[21] The Barnes–Hut simulation (Josh Barnes and Piet Hut) is an algorithm for performing an n-body simulation.

expulsion of disconnected subgraphs, while links are fixed-distance geometric constraints.

- *Pack Layout* – Will be used to create the hierarchical visualizations (see 0). Produces a hierarchical layout using recursive circle-packing. The size of each leaf node's circle reveals a quantitative dimension of each data point. The enclosing circles show the approximate cumulative size of each subtree.

- *Chord Layout* - Will be used to create the Radial Convergence visualizations (see 4.6.3). Chord diagrams show relationships among a group of entities and are produced from a matrix of relationships.

4.6 Visual Analytics Dashboard

Although the analytical challenges of a Runtime Intelligence Service are vast in what concerns Visual Analytics, the scope of *iUSE* Dashboard is on finding *usage patterns and relationships around features and environment*.

Multidimensional data is a challenging aspect in Information Visualization, because some properties of images have to be explored to distinguish between several variables in a 2D drawing plane. For this purpose several methods have been proposed. Sachinopoulou (2001) suggested a classification into six groups summarized in the following table:

Table 7 - Techniques for representing multivariate linear data.

Methods	Description	Some Known Techniques
Geometric	Transforming and projecting data in a geometric space.	Scatterplot matrix, Hyperslice, Prosection views, Surface and volume plots, Parallel coordinates, Textures and rasters.
Icon	Relies on a geometric figure (the icon) where the values of an attribute is associated with one features of this, such as the color, a shape, the orientation.	Chernoff faces, Stick figure, Color icon, Glyphs and Autoglyph.
Pixel	Use pixel as basic representation unit, and manipulate pixels to represent data.	Space fillings and Mosaic plots.
Hierarchical	Include trees and hierarchies and are useful when the data has some hierarchical or network structure.	Hierarchical axes, Dimension stacking, Threes, Worlds within worlds, Infocube.
Distortion	Propose to distort the tree-dimensional space to allow more information to be visualized.	Perspective Wall, Pivot table and table lens, Fish eye view, Hyperbolic trees, Hyperbox.

Graph based	Represent data using nods and edges and is adopted when the large graphs should be represented.	Basic graph, Hyperbolic graph.

iUSE will focus on two of them: *Hierarchical* (Include trees and hierarchies that are useful when the data has some hierarchical or network structure); and *Graph based* (Represent data using nodes and edges and is adopted when the large graphs should be represented). Radial Convergence (Lima, 2011, p. 196) will be used as an analytical complement to the Graph based representation. The next figures show examples of the three visual taxonomies that will be implemented in *iUSE:*

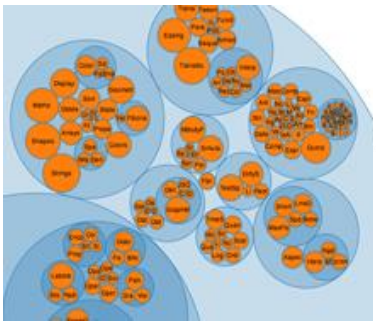


Figure 23 - Hierarchical (D3.js).

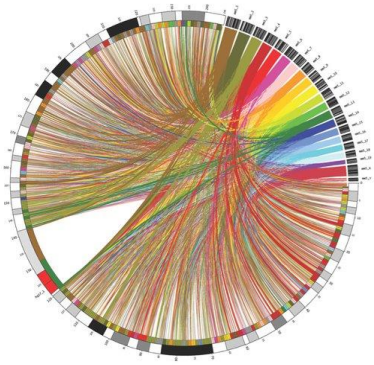


Figure 23 - Radial (Lima, 2011, p. 197).

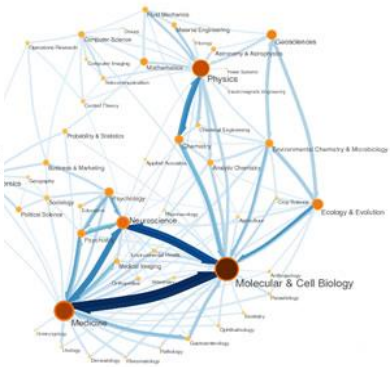


Figure 23 - Networks (Lima, 2011, p. 102).

There are additional techniques for multidimensional data representation that should be used together with the techniques cited in the table above. Their purpose is to highlight relationship on a subset of variables – composition, layering and separation, micro-macro readings, and small multiplies. *iUSE* will implement layering and separation, micro-macro readings and small multiples:

- *Layering and separation* is a technique illustrated by (Tufte, 1990), among others, and concerns the visual differentiation of various aspects of the data. It is achieved by distinction of color, shape, size, addition of elements that direct the attention via visual signals, or ordering data to emphasize layer differences.

- *Micro-macro* reading is a method for presenting large quantities of data at high densities in a way that a broad overview of the data is given and yet immense amount of detail is provided (Ruddle et al., 2002). It encodes information at different levels of detail. As an example, one same image can be used to detect fine-grained level on information encoded (micro processing) as well as large-grained level of information (macro processing). Micro/macro designs enforce both local and global comparisons and, at the same time, avoid the disruption of context switching. High-density designs also allow viewers to select and personalize data for their own uses (Tufte, 1990).

- *Small multiples* technique consists of the same graphical design structure repeated several times (Tufte, 1990). It is used to compare at a glance series of graphics showing the same combination of variables while another variable changes.

The proposed visual taxonomies – *Networks (Graph based), Hierarchical and Radial convergence* – all provide good micro-macro capabilities that will be enhanced by interactivity. The user will be given the option of zoom and detail on demand. Layering and separation will be implemented mainly by the use of color, size, transparency and visual signals to direct the user's attention. The technique of small multiples – to maintain a design structure while changing the variables – will be applied by reusing the visual taxonomies for presenting different types of data. For example, in *iUSE* Dashboard, networks will be used to represent workflows between features, collected data points and to highlight error related patterns of usage and extract the most likely reproduction steps.

4.6.1 Networks

Figure 24 depicts the base structure of a network, with its nodes and links mapped to runtime intelligence data. The mappings are described next:

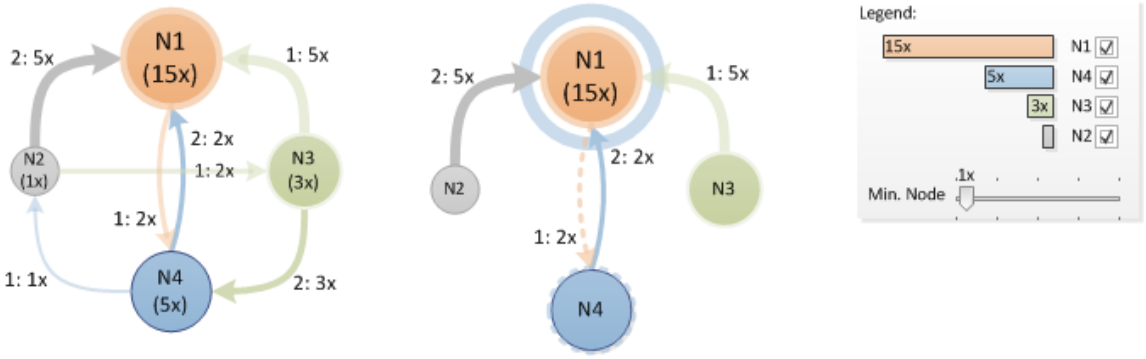


Figure 24 - Network topology.

Node – Node will represent the frequency of a specific data point or class such as Device, Event, Activity, Feature, Error or Direct2User (see 4.4.2). A typical data point is represented by *iUSE* as *"{class:'Event', category:'menu', label:'invoice_click', feature:'invoice', date:'2013-06-28T10:23:57.2340'}"*. Nodes will encode the following properties:

- *Color* – Color is used to represent data classes (e.g., "Event", "Error"). In the workflows represented by *iUSE* (Data, Features, and Errors), the nature of data is qualitative, therefore adequate color will be chosen and consistently applied so the user

can easily recognize the specific type of data being represented. ColorBrewer[22] will be used for assigning qualitative colors. Red will be reserved for class Error because of its association with warnings.

- *Text* – Label of the specific data represented by the node. For example, when depicting data or error workflows, the text of the node is the data point label property (e.g., "invoice_click", "out of memory"). When in feature workflow, the text of the node will be its associated feature property (e.g., "invoice").

- *Size* – The area of the node is proportional to the represented data point frequency within the data represented, which is the same as the sum of all links strengths converging into the node. Figure 24 shows clearly that the node with more occurrences is N1. It could signify that feature N1 is at the top of user's preferences or that N1 represents a specific error that is having the biggest impact on users. It also shows that for N1, as stated before, its value (15x) is equal to the sum of links whose destination is N1 – (N3→ N1) = 5, (N2 → N1) = 5, (N4 → N1) = 2, and (N1 → N1) (see *Stroke-width* below) = 3 – totalizing 15.

- *Stroke* – Stroke-width is proportional to the frequency of a node where source equals target (link to himself). For example, node N1 has a stroke-width larger than N3, which visually represents the fact that occurrences of type (N1 → N1) occur more frequently than (N3 → N3) or any other nodes.

Links – Represent specific workflow between data elements (source → target) and its frequency. Links will encode the following properties:

- *Color* – Link color is related to the source node. For example, link (N1 → N4) is orange, the same color of node N1 (source). The color-coding used, having a specific node as a reference, enables looking at its links and realize if they are outbound links or inbound links. Links of different color connected to the reference node mean for sure different types of data inbound to the node, while links of the same color represent connections to the same data type (inbound or outbound). In order to reduce display clutter when representing heavily linked networks, arrows from extremities representing orientation will not be used. When the node has the same color coding as the links connected, link orientation will have to be disambiguated by selecting the node. When a node is selected all outbound links, besides the color, are coded has dashed lines (see *Selected Node,* below).

[22] ColorBrewer, from http://colorbrewer2.org/, is a tool that depending on the number of data classes, and the nature of data, suggests appropriate color schemes, including colorblind safe colors.

- *Stroke* – The stroke-width is proportional to the frequency of the link. For example, link (N2 → N1) has the same stroke-width than link (N3 → N1) – both have the same occurrence (5x) – as opposed to link (N2 → N1: 5x) that is 5 times thicker than (N2 → N4: 1x).

- *Opacity* – Represents the relevance of a connection. Less opaque (more transparent) links are less relevant to the analysis. Relevance is connected to the number of runtime sessions a link occurred. Considering an example of feature workflow, based in Figure 24, the link (N3 → N1) occurred 5 times and the link (N3 → N4) occurred 3 times. Nevertheless, the link (N3 → N1) is depicted with transparency because it has a relevance of 1 ("1: 5x") and (N3 → N4) is fully opaque because of its relevance of 2 ("2: 3x"). The opacity levels will follow a logarithmic scale to create a more effective layering of links, reducing display clutter by concealing the less relevant links from the visual representation.

Selected Node – Figure 24 (center) shows the network adapting to a change in context, in this case the selection of a node. When a node is selected, its entire outgoing links will be dashed to help distinguish between inbound and outbound links, when networks are more complex. Also, any other nodes or links that are not connected to the selection are dimmed (irrelevant to the analysis) and connected node texts will maintain only the name and not the value. Each connected link will present a small text indicating its frequency.

Pinned Node – To help in network analysis, the user will select and drag nodes to reveal more information and to force repositioning of the network. But, because the physics engine will try to bring together heavy linked nodes, a scheme to pin a node in a position will be implemented. The user can press CTRL key or activate an UI element to pin a node after dragging it. This will enable the user to distribute network nodes to analyze relationships between nodes of interest better (see N4 from Figure 24 (center)).

Filters – Networks will provide better analytics when application usage generates patterns of usage – which generally do. However, there are also less used workflows that may be irrelevant to the analysis but create display clutter. For this reason, a filter will be implemented allowing users to select the range of node values. Nodes bellow the defined threshold will be considered irrelevant and not shown in the visualization. Users will also be able to filter by node class, such as Activities or Events, etc. (see Legend in Figure 24)

4.6.2 Hierarchical model

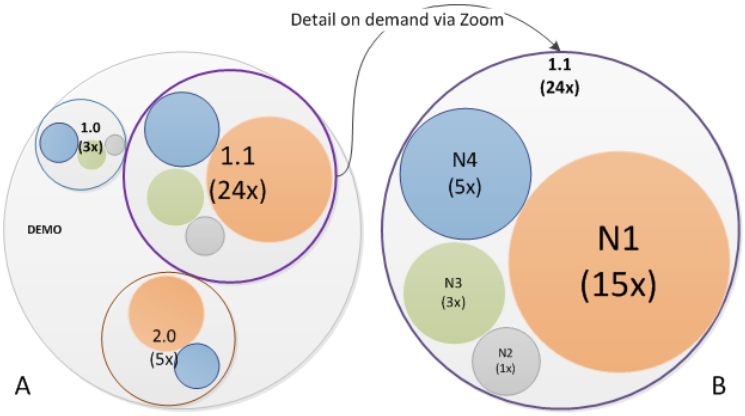


Figure 25 - Hierarchical model.

Figure 25 shows the hierarchical visualization that will be used to identify top product usages. This model produces a cluster layout using recursive circle-packing. Details are on demand – user selects a node and the system will zoom-in to show enclosed details. The cluster view implements the small-multiples technique where a fixed structure (colored circle) is repeated with variations in data, represented by size and color.

Clusters will be used to show application execution trends, such as: Sessions (cluster by Application → Version), Application Data (cluster by Application → Version → Label of data), Application Features (cluster by Application → Version → Feature) and Device Environments, such as Operating System (cluster by Application → Version → OS Name → OS Language → OS Architecture 32/64 bits).

Considering a cluster of Features as an example, Figure 25 (left) shows an overview of feature usage by application and version. The overview shows that version 1.1 of application "DEMO" is clearly the one that contains more assorted feature usage and, in overall, the Orange tinted feature is the most used, except in version 1.0 – probably because it was a feature introduced later. From the detailed information *Figure 25 (right)*, the user can visualize individual feature usage (name and frequency). The mappings are described next:

- *Color* – Identical to network usage (see above). Application, version and specific classes of data (Events, Activities, Direct2user and Errors) are qualitative values represented by color that will be consistently used throughout the Dashboard visualization models. Each pack (parent node) will have a stroke color that corresponds to its super-class value (Application, Version) and each data (child node) will be filled-in with the color of its data class. Data class can be one of the following: label of data point when showing all collected data classes (i.e., events, activities, errors, etc.), feature name when viewing feature trends, or device properties when

viewing operating system distribution (i.e., OS name, OS language, OS Architecture, OS Service Pack).

- *Text* – Implemented in the same way as in networks (see above).
- *Size* –The size of the child nodes represents the frequency the specific data occurred in the visualized data sample. The size of each parent node is recursively calculated to accommodate all its children. Also, the size of text is proportional to the size of the node.
- *Opacity* – Implemented in a similar way as the network links. But in this case, the relevance is proportional to the frequency, hence its size (see above).

4.6.3 Radial Convergence

Radial convergence uses a circular layout that represents relationships between elements of the data. The next illustrations[23] depict the structure of the Radial Convergence model.

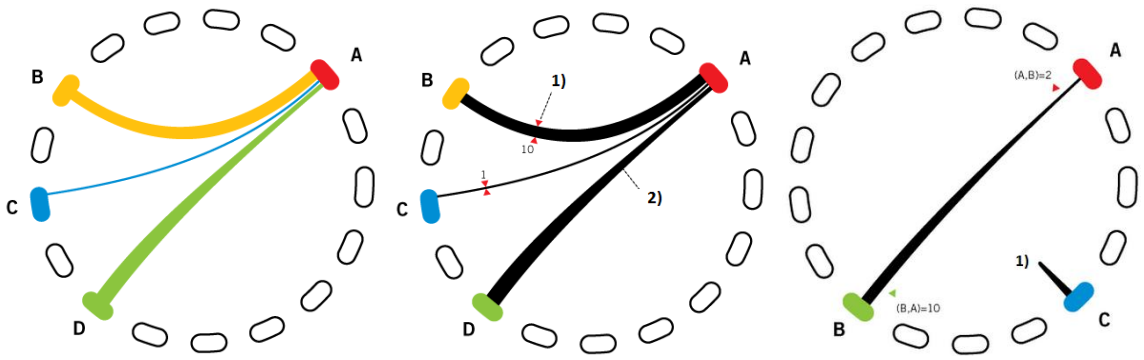


Figure 26 - Radial Convergence.

In Figure 26, links with variable thickness represent the extent of the relationship between elements. The quantity of the associated relationship (e.g., frequency of a specific Feature), will be represented by the thickness of the link. A link will have variable sized ends to indicate a ratio A:D = 1:5. Figure 26 (left) shows that when links are colored based on the elements that they relate, spotting patterns is easier. The direction of relationships (links) will be colored by source. For example, in Figure 26 (left), when considering the cells A and B, according to color-coding the direction of their link is (B → A). By coloring the links based on one of the elements, it becomes easier to follow relationships to/from an element. In Figure 26 (right) Data links (A → B) and (B → A) are shown by a single ribbon whose ends are of variable thickness,

[23] From http://circos.ca/presentations/articles/vis_tables1/
Circos is a software package for visualizing data and information.

which are the data values from source and target. For example, if (A → B) = 2 and (B → A) = 10 then the ribbon's end touching A is thickness 2 and the ribbon's end touching B is thickness 10.

The Radial Convergence is a good analytic complement to *iUSE* networks. Each node in the network, whose value corresponds to the strength of inbound links, will be related to a cell in the Radial, whose value corresponds to the strength of outbound links. There will be a connection between the two visual models in a way that when a user selects a node (network model) or a cell (radial model) the other model will reflect the same selection, offering two complementary views, one dedicated to inbound links and the other to outbound links.

4.6.4 Usability

Tufte (1990) states that "Escaping the flatland of two-dimensional computer screen and enriching the density of data displays are the essential tasks of information design." In what concerns data density, the *Dashboard will be optimized for full-screen mode*. This will allow taking advantage of the maximum display resolution to create high-density visualizations.

Also, taking advantage of the increasing multiple-monitor environments is important. The user will be given the option to see overview on one monitor and detail on another. The two views must be connected together to enable that interaction in one monitor produces changes on the second monitor view. This possibility increases display scalability (see 4.5).

As stated in the previous section, *iUSE* targets a multi-platform audience with access anytime and anywhere. Also, it plans to reach its audience on the go with devices such as tablets. For this reason, the UI must be designed with the focus on touch interaction.

Stakeholder's (users of the Visual Analytics Dashboard) work in silent environments, therefore sound will be limited to engagement, interaction and feedback purposes (Action → Sound). In the context of networks, the movement produced by node dragging will generate sound based on multidimensional information, derived from the node value and the number of network links. Figure 27 shows the process of synthesizing sound for *iUSE* networks. The process is a variation of "Ambient Drone"[24] that uses Web Audio API[25] to synthesize ambient sound textures in real-time using filtered noise.

Figure 27 (A), (B) and (C) represent real-time JavaScript manipulations of the different audio nodes. First the system will map the number of generators producing sound (n) to the total links in the network. Then, when the network is moving by the dragging of one of its nodes, the base note of the generated sound will be proportional to the selected node size (A), panner will be updated with random (x,y,z) positioning (B), and audio gain will be proportional to the network internal alpha cooling parameter (C).

[24] See "Ambient Drone" details at http://matt-diamond.com/drone.html.
[25] W3C Web Audio API Draft: https://dvcs.w3.org/hg/audio/raw-file/tip/webaudio/specification.html.

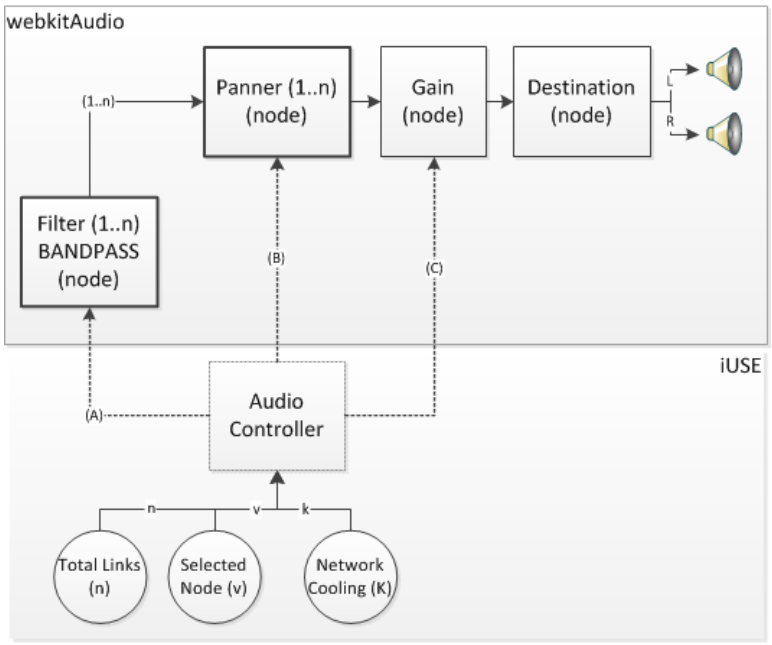


Figure 27 - Real-time synthesized audio.

Internally, the network layout uses a cooling parameter alpha (k) which controls the layout temperature: as the physical simulation converges on a stable layout, the temperature drops, causing nodes to move more slowly. Eventually, alpha drops below a threshold and the simulation stops completely.

The overall audio synthesis simulates movement friction and each interaction with a network specific node will reproduce a one-off sound, because although it uses the same base note, the resulting sound will be slightly transformed by the random noise filter.

Chapter 5

Implementation

5.1 Data Storage

This section reveals details about data storage implementation, presenting an overview of database schema and design decisions. Resulting from data storage technology requirements (see 4.5) the database is hosted on a cloud server running SQL Azure that provides the necessary IT infrastructure and QoS: scalability, availability and reliability.

Table 8 briefly describes the entities being modeled and Figure 28 shows an overview of their relationships.

Table 8 - Database entities.

Entity	Description
Project	The software application to be tracked. A project can have multiple versions. The version is defined at runtime by the client API.
Member	A person registered in *iUSE* as a member (e.g., Developer), using the framework to record usage of a Project. A member has a key which authorizes him to communicate with *iUSE* REST Web API.
User	The person that is interacting with the Project at runtime. The user ID should be the same between sessions in order to capture distinct usage in time.
Session	Session represents all collected data points from a User since the beginning of a Project execution until application termination. A data point is a specific type of data collected in a point in time. The client API should create a GUID that uniquely identifies each new session.

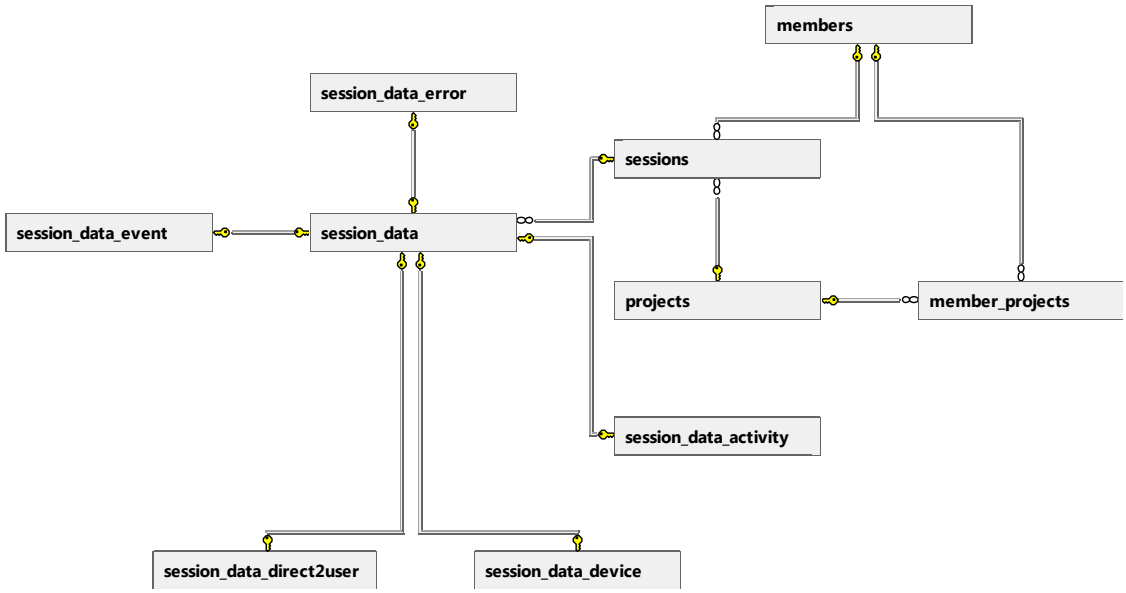


Figure 28 - Entity relationship.

Implementation

iUSE databases can rapidly grow larger, depending on the quantity of data points an organization is collecting. Therefore, a major concern when designing the database was to properly normalize tables, saving storage space and storage cost.

The main tables are *[session]*, *[session_data]* and *[session_data_*]*[26]. Each individual session is represented by one row of *[session]*, with all common data stored in *[session_data]* (Super-table) and specific types of information in *[session_data_*]*. Special care was taken on defining indexes for foreign keys for search fields in order to optimize response time to analytic requests.

Figure 29 shows a typical request for analytical data when client applications use the web API to query information. For performance reasons, all queries to the database are implemented as stored procedures so that they be pre-compiled and optimized. Stored procedures then access information by using views that abstract internal representation of fragmented tables. For example, a query to a user's workflow calls the stored procedure "*sp_user_flow"* that queries the view "*view_user_flow"*, ordering its results by data point date. This view "*view_user_flow*" encapsulates a union between all relevant specific session data ("*session_data_event"*, "*session_data_activiy"*, "*session_data_error*", etc).

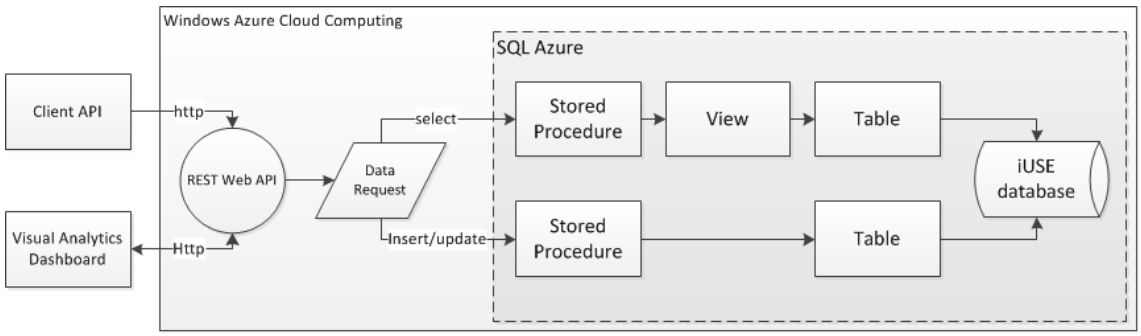


Figure 29 - Anatomy of a data request.

Some analytics require mined data; some of this computation is executed at runtime, at the client web dashboard. But, to limit the amount of data returned to the client and for performance requirements, some aggregations must be made at the server' side either directly at the database server or by the web role. For example, data inserted into tables related to features (*session_data_event*, *session_data_activity*) are monitored by database triggers. On inserting, the trigger updates a session *feature_flow* table that maintains the number of occurrences between features (important to monitor feature workflow trends). This data is then mined by the web server to create global community usage information. The aggregated information is then consumed by *iUSE* visual analytics models.

[26] *represent sub-tables of specific type, such as Event or Activity.

5.2 Web endpoint

iUSE Web endpoint is built with ASP.NET Web API – a framework for building HTTP services on top of the .Net Framework. HTTP is simple, flexible, and ubiquitous. Almost any platform has an HTTP library, so HTTP services can reach a broad range of clients, including browsers, mobile devices, and traditional desktop applications.

This service provides insert and query interfaces to the data storage, and computes mined data to be consumed by visual analytics models. Figure 30 illustrates the six types of data-mining accessed through the Web endpoint.

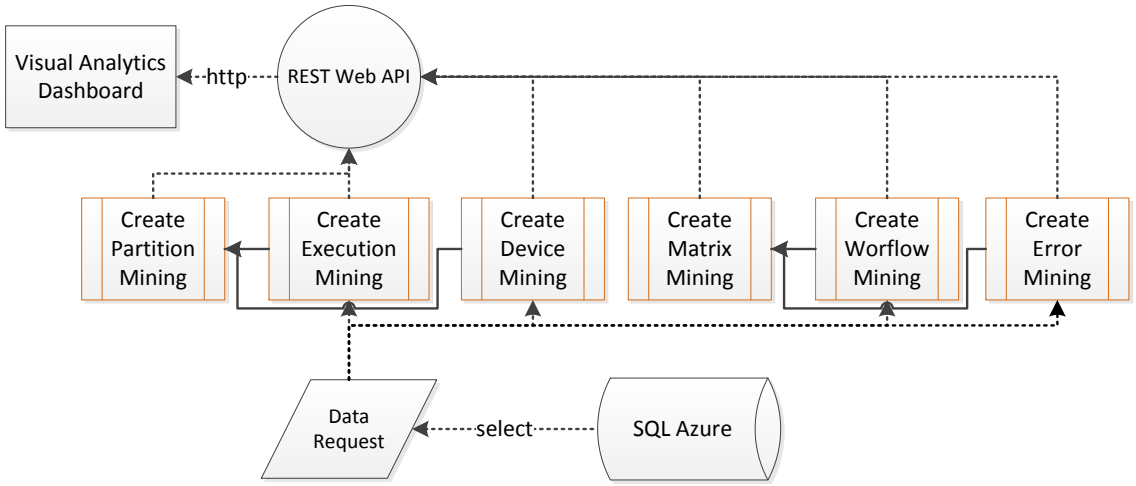


Figure 30 - Analytics data-mining.

The different data mining models are optimized for a particular visualization model and consumed by the Visual Analytics Dashboard.

All requested data can be filtered by date interval, application name, application version and category. Each element of the aggregated response includes a *Range* value – total subject occurrences; and a *Domain* value – total of distinct subjects. For example, an element of the Session execution analytics with a *Domain* value = 3, and *Range* = 10 reveals that 3 distinct users executed 10 sessions (in the context of the specified filter).

The next list is a brief description about each analytic model:

- *Partition Mining* – Creates parent-child aggregations. Partition analytics can be requested for Device usage overview, particularly OS distribution (e.g., OS > Language > Architecture) and Application usage distribution (e.g., Application > Version). The subject of *Domain* is *distinct devices*.

- *Execution Mining* – Creates data point aggregations by month, day and hour. Execution analytics can be requested for *Sessions*, *Events*, *Activities*, *Direct2Users*,

Errors, *Installations* and *Uninstallations*. The subject of *Domain* varies according to context:

- *Sessions* – Subject of *Domain* is *distinct users*.
- *Events*, *Activities* – Subject of *Domain* is *distinct labels*.
- *Direct2User* – Subject of *Domain* is *distinct URL's*.
- *Errors* – Subject of *Domain* is *distinct error messages*.

- *Workflow Mining* – Creates aggregations of sequence data points. The collected usage data includes *Events*, *Activities*, *Direct2User* and *Errors*. Each element on the workflow represents a link between two data points. Subject of *Domain* is *distinct Sessions*.

- *Matrix Mining* – Structures the workflow analytics as a matrix [MxM] where M = total distinct workflow links.

- *Error Mining* – Creates aggregations on error sequences. The system searches for previous interactions with the system before errors. The analysis will create a workflow between collected data points with the intent to highlight eventual error related patterns of usage and extract the most likely reproduction steps.

Mined data is not limited to feed *iUSE* Visual Analytics but also any other HTTP Client. For example, the organization could integrate that information on a SharePoint portal in the format, or a list, or other visual graphic.

5.3 Modeling with OWL

OWL is the W3C recommended language for describing Ontologies. The model was built using Protégé-2000[27] following an iterative process.

iUSE ontology is based on simple hierarchy of concepts and relations captured from the database schema (see 5.1) and linked data requirements (see 4.4.3) enriched with axioms used to fix the semantic interpretation of concepts and relations. In terms of modeling one of the goals was to keep it simple to allow easy extension – *Creeping conceptualization*[28] (Antipattern). Nevertheless, extension will depend on how *iUSE* integration is being architected and the type of information that is collected.

[27] Ontology tools survey, Revisited (Denny, 2004)
[28] Semantic Web for the Working Ontologist (Allemang & Hendler, 2008)

The design of software inherently involves a model of the commonality and variability in its domain. The object model (API) of *iUSE* was implemented using OOP that uses classes and sub-classes to represent hierarchies. Classes high up in the hierarchy represent common functionality while classes farther down represent more specific functionality. The idea of class hierarchies is transversal to semantic web standards. High-level classes represent commonality among a large variety of entities, whereas lower-level classes represent commonality between a small, specific set of things. The model hierarchy was consistently implemented throughout the *iUSE* database model, OOP model and the Ontology.

RDFS inference is the mechanism that enables a system to determine other information, related to stated information, as if it had been stated. This inference mechanism greatly reduces the quantity of information needed to be export by *iUSE*.

5.3.1 Domain and Scope

Requirements specification (see 4.4), database entities, properties and relationships (see 5.1) helped to define de domain and scope of the Ontology: *Tracking usage of desktop software*. The main purpose is to expose *iUSE* data, by providing a mechanism to export data into any RDF data store and creating the basic axioms that provide inference mechanisms to facilitate the query[29] of simple questions about the usage of software application. *Competency questions* such as:

- *What is the workflow in the application*? This can be answered by querying all instances of type "Data".

- *What is the usage of a specific type of data*? This can be answered by querying all instances of the specific type (e.g., "Activity", gives all tracked activities).

- *Is there any Data that should trigger an alert or organizational procedure?* There are some types of data that are tracked with the intent to trigger some kind of procedure in the software organization. For example, error and warnings could trigger an alert on development to proactively investigate the possible causes. This can be answered querying all instances of type "Error".

- *How can I merge usage data with other ontologies?* In terms of modeling, the primary goal was to keep it simple so it could be easily extended or merged with other ontologies. Linking data to other ontologies depends on how software organizations architect *iUSE* framework integration and the type of information that is being stored. For example, the proliferation of social network web applications is putting the social

[29] SPARQL is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format.

graph concept at the center of the scene. To merge *iUSE* with social network ontologies, the user of the application should be given a unique identifier capable of making the bridge (OPENID, foaf:email, etc.). In such scenarios, data-link requirements should be defined when architecting *iUSE* integration.

5.3.2 Asserted model

The main terms were imported from the terms created when designing the database schema. A top-down approach was used (Uschold & Gruninger, 1996) to specialize some new classes: "*CompletedActivity"*, "*CanceledActivity"* (sub-classes of "*Activity"*). Specific "*Data*" classes like "*Event*" or "*Activity*" were already specialized when creating the database schema and ported to this model (see Table 9). Figure 31 shows the asserted OWL model.

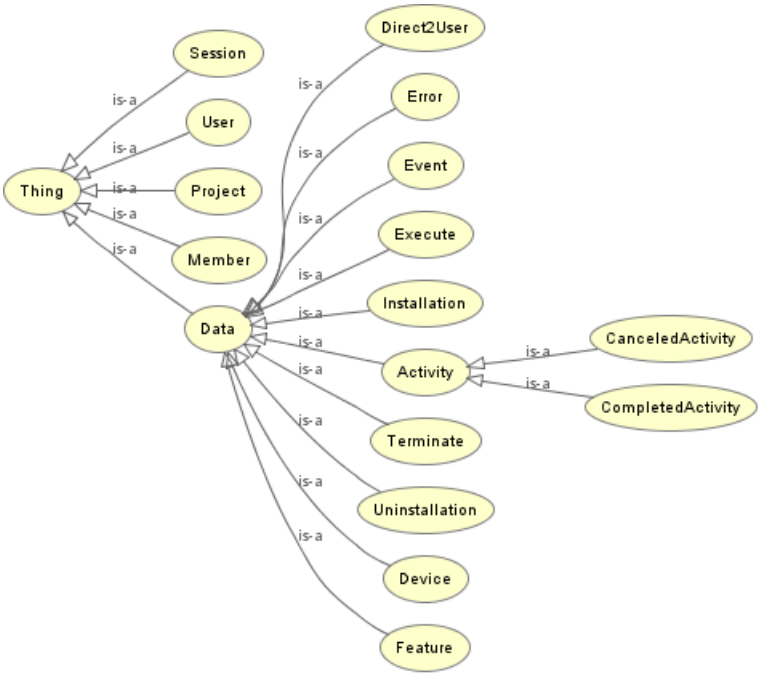


Figure 31 - OWL model.

Table 9 - OWL classes.

Class		Description
Project		Software application that is being tracked.
Member		A person, registered in iUSE as a member (e.g. Developer) that is using the framework to record usage of a Project.
User		A person that is using the Project.
Feature		Represents a functional unit that is being used by the User.
Session		Encloses all information recorded by a Member about a User interaction in a Project.
Data	Disjoint Classes[1]	Encloses all different data types recorded in the Session.
Device		A specific type of data. It represents information about the device where the Session is executing.
Execute		Specific type of data. It represents the start of a Session.
Terminate		Specific type of data. It represents the end of a Session.
Event		A specific type of data. Something triggered in the application. An event has no duration, it happens in a point in time.
Error		Specific type of data. It represents an exception during the execution of a Session.
Direct2User		Direct2User represents interaction with the user via redirect to a URL.
Activity		Activity is a specific type of data related to some task the user has performed. An activity is associated with a time-span. An Activity can be cancelled.
CompletedActivity (Activity)	Disjoint[2]	CompletedActivity is a specific type of Activity data. It represents all activities that were completed.
CanceledActivity (Activity)		CanceledActivity is a specific type of Activity data. It represents all activities that were cancelled.

[1] A specific Data instance can only be of one type.

[2] An Activity instance can be classified (typed) as Completed or Canceled (Exclusive)

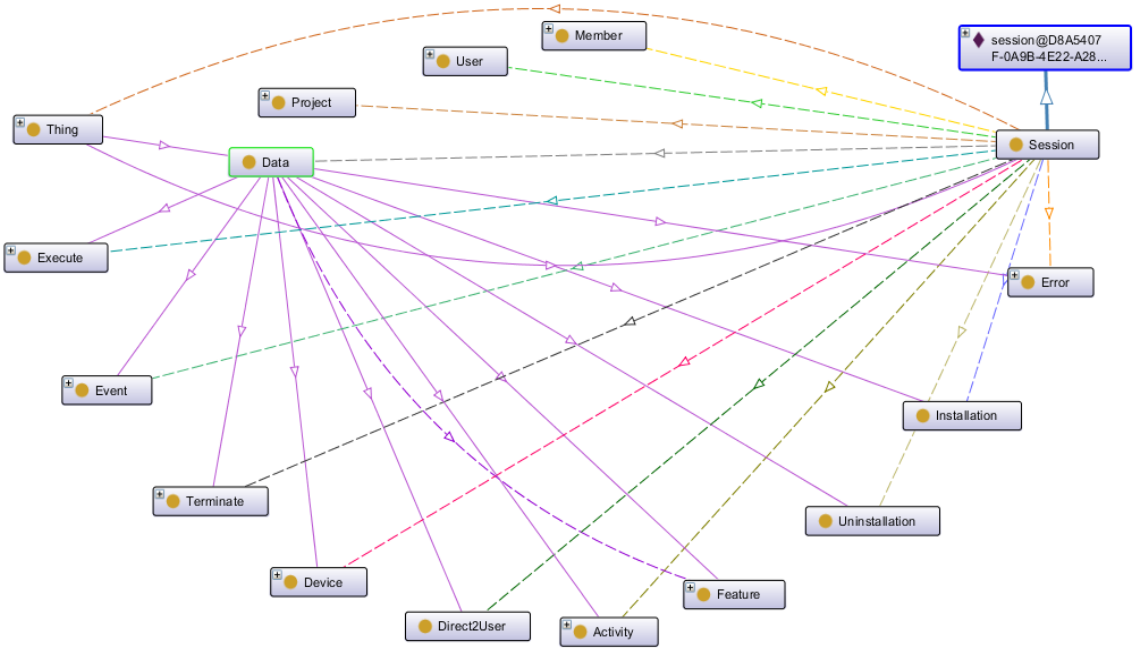


Figure 32 - OWL session.

The instances are created by the export mechanism included in *iUSE* Web API, which lets the user export session data in RDF/XML or Turtle. Figure 32 shows a representation of a session instance.

A Session carries all the Data recorded by a Member about usage of a User in a Project running in a specific Device and time-frame that spans from Execute to Terminate.

Project, User, Member and Feature are classes connected to Session by object properties – modeled as classes better reveal the organic of a session. When exporting RDF data, different Sessions may point to the same instance of a Project, User, Member and Feature, as long as they represent the same thing. Also, considering RDF queries, this structure better serves the visual representation and inference engine to obtain answers about specific entity individuals, such as:

- What are the Sessions running the Project (id="NGCO", version="12.00")?
- What Sessions have been used by User (id="luismiguelfr74@hotmail.com")?
- Who is using Feature (id="#320")?

Properties were imported from database schema, although not used in the same way. A property, unlike OOP or database schema, is not part of the class; it connects a class or classes to a value property or object property. For example, a "*hasFeature*" (object property) is associated to the domain of "*Data*", so it can be referenced by any of its sub-classes (In current implementation to "*Activity*" and "*Event*"). Table 10 and Table 11 (below) show the complete description of object and data properties.

Table 10 - OWL object properties.

Object Property	Inverse	Domain	Range
hasCreator[1]	isCreatorOf	Session	Member
hasProject[1]	isProjectOf		Project
hasUser[1]	isUserOf		User
hasData	isDataOf		Data
hasFeature[1]	isFeatureOf	Data	Feature
hasActivity[2]	isActivityOf		Activity
hasEvent[2]	isEventOf		Event
hasDevice[1,2]	isDeviceOf		Device
hasDirect2User[2]	isDirect2UserOf		Direct2User
hasError[2]	isErrorOf		Error
hasExecute[1,2]	isExecuteOf		Execute
hasTerminate[1,2]	isTerminateOf		Terminate
hasInstallation[1,2]	isInstallationOf		Installation
hasUninstallation[1,2]	isUninstallationOf		Uninstallation

[1] Functional property.

[2] Inverse functional properties.

Table 11 - OWL data properties.

Data Property	Domain	Range	Description
hasArchitecture	Device	String	Device OS Architecture (e.g. "64")
hasCity		String	Device City (e.g. "Porto")
hasCountry		String	Device Country (e.g. "Portugal")
hasRegion		String	Device Region (e.g. "Porto")
hasIP		String	Device IP (Internet Address)
hasLanguage		String	Device OS Language (e.g. "pt-PT")
hasLatitude		String	Device Latitude (Geographic location)
hasLongitude		String	Device Longitude (Geographic location)
hasOS		String	Device OS (e.g. "Windows 7")
hasResolution		String	Device Resolution (e.g. "1024x748")

hasDeviceID	Data	String	Device Identification (e.g. device serial number)
hasDataID		Decimal	Session data ID. Each data point stored in iUSE has a unique number ID.
hasCategory		String	Identifies a category of data. Is used to categorize data points of type "Event", "Activity", "Error", "Log", "Direct2User".
hasEmail	Member	String	Member registered email.
hasSource	Error	String	Error source.
hasStackTrace		String	Error Stack Trace.
hasURL	Direct2User	String	Direct2User URL that the user visited.
hasVersion	Project	String	Project runtime Version defined via Client API.
hasID	-	String	Generic ID.
hasDt	-	Datetime	Generic datetime. Can be used to define a point in time, or start of a time period.
hasDtEnded	-	DateTime	Generic datetime. Can be used to define the end of a time period (Session or Activity)
hasDuration	-	Decimal	Generic time duration in seconds.
hasLabel	-	String	Generic description.
hasMessage	-	String	Generic message.
hasToken	-	String	Generic client unique identifier (Session, User)

Note: All data properties are "Functional"

5.4 Dashboard

In this section screenshots of the prototype will be commented. Figure 33 shows the structure of the dashboard menu and the orientation for touch devices. The structure of the menu will be created so the user doesn't have to navigate on small menus and sub-menus. The menu will follow a network inspiration with options grouped by category and moved to the corners, where it is easier to select when used in a tablet or touch device.

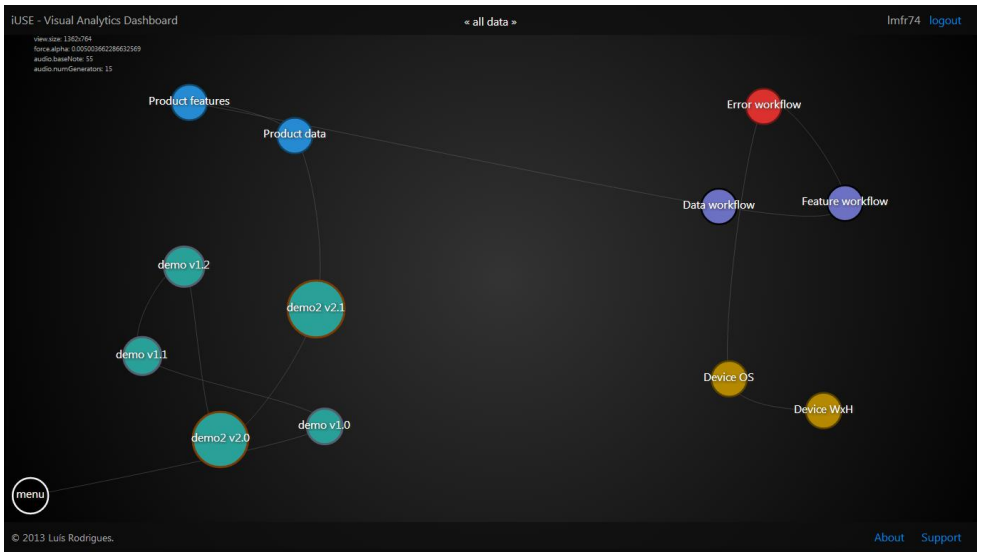


Figure 33 - Menu.

Figure 34 shows an overview of the Hierarchical Model and its zooming behavior (details on demand). The use of color creates a cluster from where patterns can be viewed.

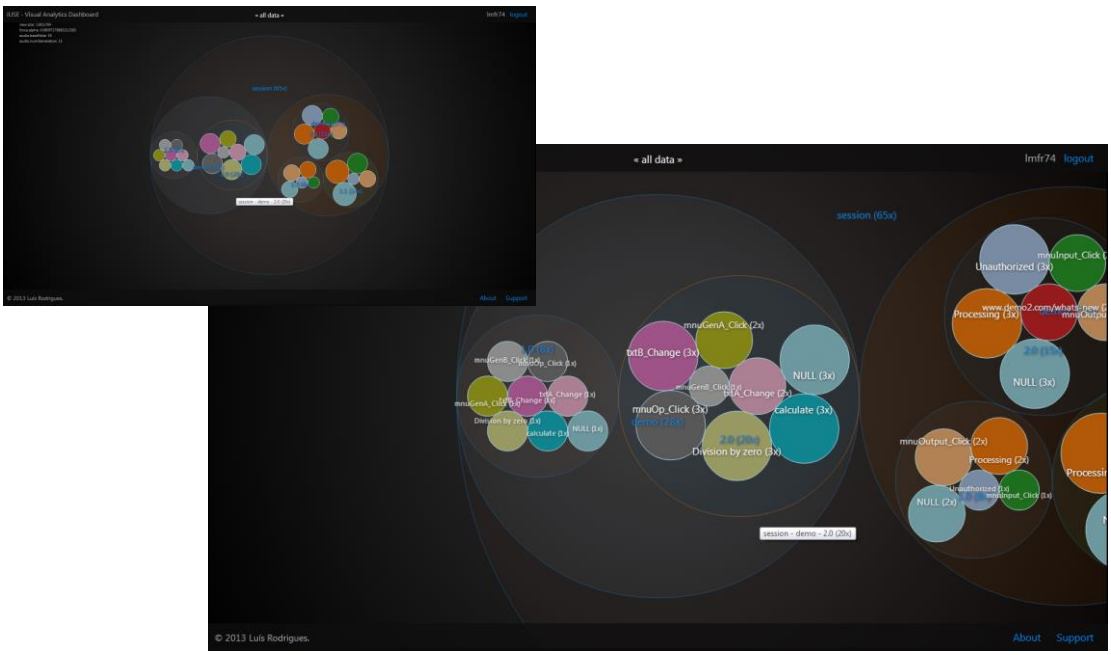


Figure 34 - Hierarchical Model.

Implementation

The next figures show the visualization models of Networks together with the Radial Convergence. The interface provides the ability to zoom out models in order to focus attention either on network or radial convergence. Legend works as a bar chart providing indications about the color coding and scale. Figure 35 shows the network adapting to the selection of a node. Nodes not connected to the reference node are dimmed out, outbound links are dashed and link strength detail is visible. All networks follow the same visual representation.

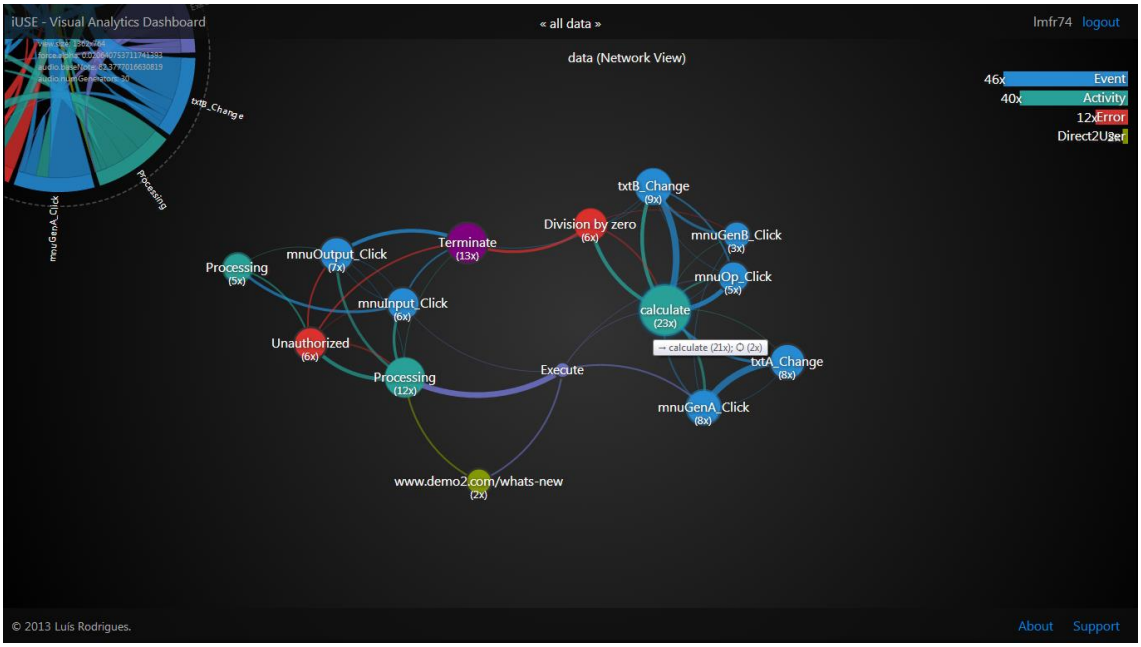


Figure 36 - Networks.

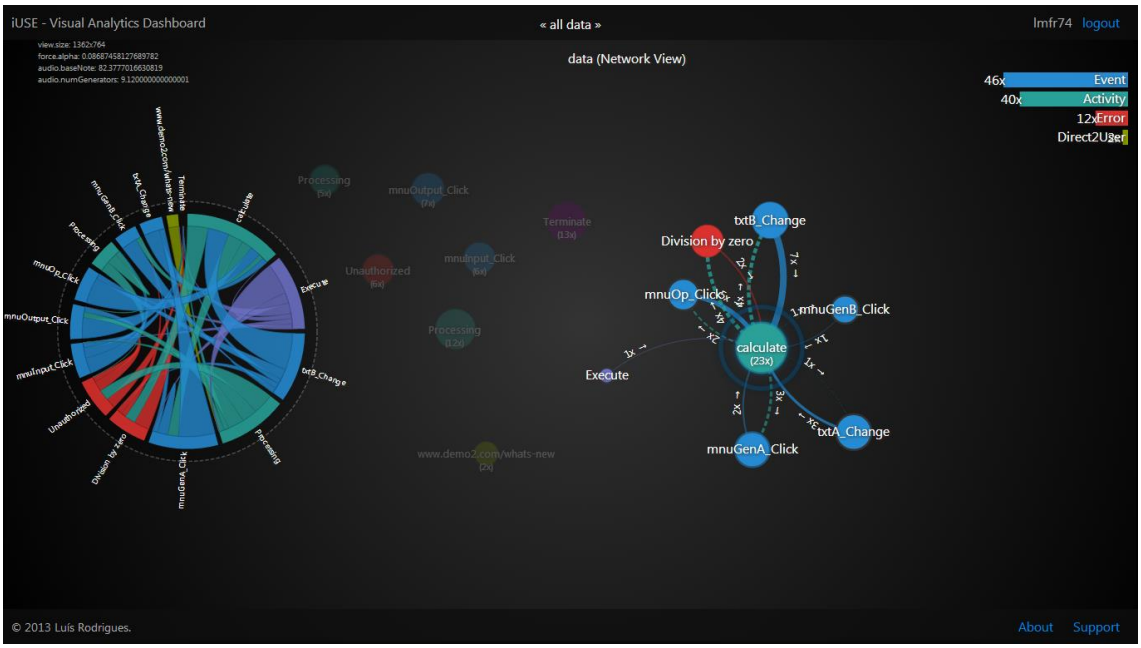


Figure 35 - Networks (Selected Node)

Network topologies are used to show error patterns and to view their most likely causes (see Figure 37). *iUSE* processes the last 5 events before an error occurred and represents their frequency and relationships together with device properties (OS, Architecture and Language) that may have caused the error.

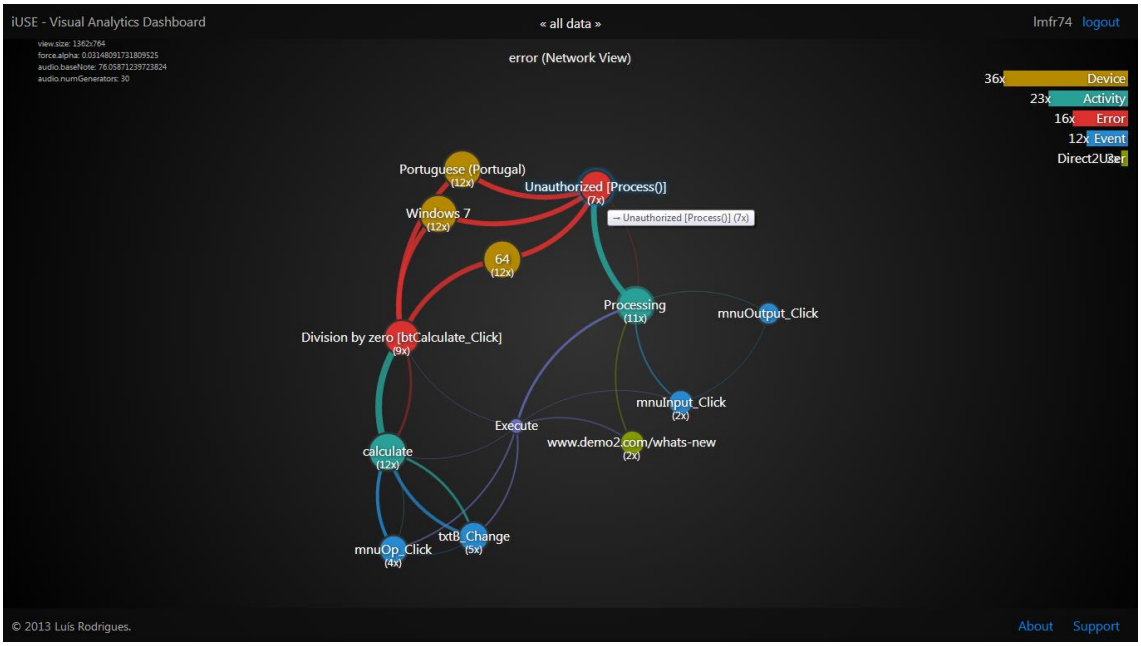


Figure 37 - Error analysis using networks.

To test the prototype, the method of Talk-Aloud protocol was used (see Usability testing). The listed report concludes that the three visual models, their mappings and coding were well understood. It identifies that when interacting with networks, the user entered a much higher engagement state. Complex networks are intriguing and stimulating.

Chapter 6

Conclusions and Future Work

6.1 Objectives Accomplishment

This thesis intended to achieve three main goals: The first one was to create a high-fidelity prototype, using state of the art visualization models and technology, likely to evolve into a working product, which could then be used in the context of organizations; secondly, the project was considered as an opportunity to grow as a professional in areas where personal knowledge was sparse or even null. As a desktop business software developer, web technologies were not a priority for many years, so the selection of the state of the art web technologies related to the *Semantic Web, HTML5, SVG* and *Web Audio* were the opportunity to develop new skills; lastly, there was a personal interest in information visualization and the urge to develop and explore new models of visualizing complex information. The selected visual models showed to be efficient for the task of Visual Analyzing Runtime Intelligence Data.

The research on the domain of *Runtime Intelligence* requirements, the proficiency in new web technologies and the integration of innovative aspects in the domain of *Runtime Intelligence* encourage the author about *iUSE* as a starting point of a simple but powerful *Visual Analytics* tool.

6.2 Future Work

The project addressed several aspects of *Runtime Intelligence* services, from C*lient to Server* services to *Visual Analytics* that focused on network visualizations as a tool to detect usage patterns and relationships in data. In order to be a fully functional product, *iUSE* needs to embrace the richness of time dimension, either in the proposed models or with the more traditional trend graphics (e.g., line charts), and to provide a broader range of analytics and enhance existing ones with further usability testing. As an example, in order to find error workflows and the most likely reproduction steps, a tree with branches representing the order in which the user navigates in the application would help seeing the sequence of steps.

In *Visual Analytics* requirements, many scalability objectives were documented but not implemented in the prototype - multi-monitor support is an example that could be additional innovation to *Runtime Intelligence* services. It is technically possible to implement by using web sockets as the communication channel between independent and isolated browser windows that could then work as an extension to each other. Other scalability objectives like semantic zoom were implemented in the hierarchical view by showing more detailed information while zooming to child packs. Semantic zoom will be a valuable addition to implement *details on*

demand and reduce clutter on network visualizations. For example, in dense network visualizations the user could zoom-in using the mouse or touch, and while the network scaled up, more information would be displayed.

The proposed network model has high processing requirements for the client's browser, because of its physics engine and graphical fidelity. The browser needs to support the most recent *HTML5* specifications and be able to use *GPU* acceleration so that more dense visualizations are fluid. To minimize performance degradation on denser networks, browser multithreading is a possible and recommended enhancement. This would benefit the audio synthesizer, because the physics engine runs on parallel with the audio generator on the same browser thread. Performance tests with denser networks resulted in poor frame rate when some visual elements were used, such as *SVG* filters and dashed lines. This is expected to become less of a problem in a near future with the growing support for hardware acceleration in web browsers.

The defined architecture of *iUSE* and the adoption of a cloud computing platform, such as *Windows Azure* enable the creation of asynchronous services that can scale to handle more demanding mining algorithms and process more information. *Runtime Intelligence* data has an interesting feature: the collected data is a snapshot in time that doesn't change; therefore it can be used to provide prepared information. For example, a worker's role running on the cloud computing could mine and prepare data at 00:00 each day, so the next working day data is ready for rendering. Also, because *HTML5* specification includes local storage, client browsers could store received data that feed visualizations, and reused it each time an analytical task is needed, with no server requests. The user could then refresh data on demand.

In conclusion, the prototype revealed interesting analytical features of network topologies in the domain of *Runtime Intelligence* and, because it is built on standard and evolving web technologies, future enhancements will inevitably benefit *iUSE*.

References

Allemang, D., & Hendler, J. A. (2008). *Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL* (2nd ed.). (Elsevier, Ed.) Morgan Kaufmann.

Andrew W. Donoho, David L. Donoho, & Miriam Gasko. (1988). MacSpin: Dynamic Graphics on a Desktop Computer. *Computer graphics & Applications*, 58.

Ankerst, M. (2000). Visual Data Mining. *Visual Data Mining, Dissertation (Ph.D. thesis)*. Faculty of Mathematics and Computer Science, University of Munich.

Ankerst, M. (December de 2002). The perfect Data Mining Tool: Automated or Interactive? *ACM SIGKDD Explorations Newsletter*, pp. 110-111.

Berners-Lee, T. (March de 2009). *Tim Berners-Lee on the next Web.* Obtido em 30 de June de 2013, de TED: http://www.ted.com/talks/tim_berners_lee_on_the_next_web.html

Bertin, J. (1984). *Semiology of Graphics: Diagrams, Networks, Maps.* ESRI Press (November 1, 2010).

Bostock, M. (2011). *D3: Data-Driven Documents*. Obtido em 30 de June de 2013, de http://mbostock.github.io/d3/talk/20111018/#0

Deleuze, G., & Guattari, F. (1972-80). *Capitalisme et Schizophrénie.* Paris: Les Editions de Minuit.

Denny, M. (2004). *Ontology Tools Survey, Revisited*. Obtido de http://www.xml.com/pub/a/2004/07/14/onto.html

Dix, G. E. (2007). A taxonomy of clutter reduction for information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 1216–1223.

Dondis, D. A. (1974). *A Primer of Visual Literacy.* MIT Press.

Eick, S. G., & Karr, A. F. (2002). Visual Scalability. *Journal of Computational and Graphical Statistics*, 22-43.

Gruber, T. R. (1993). A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*(5), 199-220.

References

Holst, S. (2011). *Application Analytics: Why Is the Developer Always the Last to Know?* Obtido em 30 de June de 2013, de http://visualstudiomagazine.com/Articles/2011/07/01/pfven_App-Analytics.aspx?Page=1

Keim, D. A., Kohlhammer, J., Ellis, G., & Mansmann, F. (Edits.). (2010). *Mastering The Information Age - Solving Problems with Visual Analytics.* Eurographics Association.

Keim, D. A., Mansmann, F., Schneidewind, J., & Ziegler, H. (5-7 de July de 2006). Challenges in visual data analysis. *Information Visualization, IV*.

Köhler, W. (1947). *Gestalt Psychology.* New York: Liveright.

Lewis, C. H. (1982). *Using the "Thinking Aloud" Method In Cognitive Interface Design.* IBM.

Lima, M. (2011). *Visual Complexity: Mapping Patterns of Information.* New York: Princeton Architectural Press.

Michael Bostock, Vadim Ogievetsky, & Jeffrey Heer. (2011). D3: Data-Driven Documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis).*

Moere, A. V. (s.d.). Obtido em 30 de June de 2013, de Information aesthetics: http://infosthetics.com/

MSDN. (2 de 5 de 2013). *How To Choose Between SVG and Canvas*. Obtido de http://msdn.microsoft.com/en-us/library/ie/gg193983(v=vs.85).aspx

Niggemann, O. (2001). Visual Data Mining of Graph Based Data. 392-396. Germany: University of Paderborn.

Ruddle, R, Brodlie, K., & Dimitrova, V. (2002). *Communication, visualisation and interaction.* University of Leeds, School of Computing.

Sachinopoulou, A. (2001). *Multidimensional Visualization.* Technical Research Centre of Finland. VTT Publications.

Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualizations. *IEEE Symposium on Visual Languages*, 336–343.

Spenc, R. (2007). *Information Visualization - Design for Interaction* (2nd ed.). Pearson Education Limited.

References

Sperber, D., & Wilson, D. (1995). *Relevance: Communication and Cognition.* Oxford/Cambridge: Blackwell Publishers.

Thomas, J. J., & Cook, K. A. (2006). A Visual Analytics Agenda. *IEEE Computer Graphics and Applications, 54*, 10-13.

Tufte, E. R. (1990). *Envisioning Information.* Cheshire: Graphics Press.

Tufte, E. R. (1997). *Visual Explanations: Images and Quantities, Evidence and Narrative.* Graphics Press.

Tufte, E. R. (2001). *The Visual Display of Quantitative Information* (2nd ed.). Graphics Pr.

Uschold, M., & Gruninger, M. (1996). Ontologies: principles, methods and applications. *The Knowledge Engineering Review, 11*(2), 93-136.

Weaver, W. (October de 1948). SCIENCE AND COMPLEXITY. *American Scientist, 36*(4), 536-544.

Wurman, R. S. (2000). *Information Anxiety* (2nd ed.). New York: Que.

Appendix A

Usability testing

The method of Talk-Aloud protocol (Lewis, 1982) was used to gather data from *iUSE* usability tests. The test was directed to a software developer of a LOB organization interacting for the first time with *iUSE*. He was instructed about the domain of the application he was going to test and the requirements of the Talk-Aloud protocol and then asked to perform the following tasks:

NOTE: *iUSE* Visual Analytics Dashboard was set to its menu page.

1. Enter the option “Product Data”
    a. Q: What are the trends in data usage?
    b. R: The user correctly identified the most frequent collected data by referring to the larger circles. But, although the mouse suggested interaction by highlighting the circles on mouse over, the user never initiated interaction other than mouse over to read tooltips. Clicking the circle would have initiated a zoom to show details.

2. Enter the option “Data Workflow”
    a. Q: What is the most executed activity?
    b. R: Again, by looking to the network, the user easily identified the most executed activity “calculate”. In this case the user naturally initiated exploration by interacting with the network nodes. Also, he felt comfortable with the Radial Convergence and made an initial exploration. But his primary visual model of exploration was the network.

    c. Q: After starting an application, what is the event or activity more frequently used by the community?

d. R: In this case the user was pointed to the “execute” node because there was no sufficient information in the UI to elucidate about the purpose of the “execute”. After that, the user successfully identified inbound and outbound links and determined the strongest connection from “execute”.

e. Q: Do you see any errors? What is the most frequent error?
f. R: The user responded “it must be the red node!” Then he was asked if there was any clue on the UI that indicated how the errors should look like. He wasn’t able to find the legend that showed the color coding, although well visible at the top right corner. The problem here seemed to be full engagement with the network that prevented him from notice the legend.

3. Enter the option “Error Workflow” and identify the “Unauthorized” error and state:
    a. Q: What is the most likely source of error?
    b. A: Again, the user easily found the node and identified the stronger link connecting to the error node.

    c. Q: In what environments are they occurring?
    d. R: The user inferred the device nodes by label name “e.g., Windows 7” and the overall network structure. Didn’t need the legend.

Conclusions:

- When interacting with the Hierarchical View the user had some difficult in identifying that he could zoom in by clicking the circles, but understood and analyzed the model easily. A simple “click to zoom” message or icon will mitigate this issue.

- When interacting with networks the user entered a much higher engagement state. He understood the mappings and coding easily. The exception was in noticing the legend. To mitigate this, when a user interacts with a node, the legend associated with the node class could show some highlight, at least during the first interactions.

- The Radial Convergence mechanics was well understood.

Appendix B

Personas

iUSE is a specialized system aiming a specific group of users. In order to obtain a more profound knowledge of those users and to improve the design of the system, the concept of *Persona* (Cooper, 1999) was built. Each *persona* is an archetypal user, based on real people (stakeholders from a company that develops LOB desktop software) – a Software Developer, a Research & Development Manager, a Product Manager and a Customer Service Manager:

- *Luís (Software Developer)* - Luís is a software architect of 38 years old who is working as a software developer for a company specialized in LOB applications widely deployed as desktop applications. As a "common components manager" he helps in developing complex systems, in different development platforms, that are integrated by other teams. Those components are the basis for a wide range of products and therefore quality must be at the top of Luís's priorities. Regardless the commitment to development best practices, after product release, Luís lacks a reliable communication channel that effectively provides him data concerning quality issues, in order to proactively address them and reduce impact on the installed base. Quality issues, such as unhandled software exceptions, are reported to the company's product support by users. When reported issues escalate to R&D, frequently lack context information, and therefore Luís consistently asks for the same questions: *What is the Operating System? Is it a 64 or 32 bit OS? What regional settings are being used?* When the problem fails to be replicated internally, it is necessary to access customer's running environment and if necessary create a specific version, in order to locally track the steps that reproduce the error and analyze the exception stack trace or log other context data.

- *Joaquim (R&D Manager)* - Joaquim is a 45 years old R&D manager. One of his primarily tasks is to make decisions based on the company's goals. For that matter he needs actionable data in which to rely. The software company has to cope with an enormous and heterogeneous usage and different desktop environments from all its users. As systems evolve and new operating systems reach user's desktops, development investments have to be balanced with user's installed base and questions must be asked, such as: *Can we stop supporting Windows XP? How many active*

clients would that affect? Can we integrate more sophisticated and demanding hardware solutions? What's the hardware stack of our users?

- *Carlos (Product Manager)* - Carlos is responsible for managing the product catalog and feature set. He faces a daily difficulty in measuring how core features are used by customers. Such information is vital to prioritize development efforts and answer some of Carlos's questions: *Can we drop this specific feature? Are users aware of the importance of the feature? Are users updating to the new version?* Presently, Carlos relies on the licensing model to obtain information about users. A license includes user's information (i.e., fiscal number, address) and application information (i.e., version, functional level and modules). Based on these data, he can assume some information concerning user's main functional requirements, but he cannot infer how user is using those modules, or how each of the individual features is being used. In short, he lacks a survey about community feature usage patterns.

- *Ana (Customer Service Manager)* - Ana manages customer services oriented teams which include support and training. The support team is responsible for answering customer's technical issues about software and to record any reported quality matter. The training team provides users with the necessary qualifications to use software properly (i.e., payroll, accounting), in order to manage their business. Customer Service Department is also responsible for publishing information about new releases, training opportunities and other relevant subjects to the customers, using different channels such as the company's web page, reseller's email channel, social networks like Facebook or Twitter, and so on. Although aware of the importance of this information, the company has not yet implemented any direct-to-user communication mechanism.

Appendix C

Stakeholders Survey

The survey was created and submitted through an online survey platform (SurveyMonkey) on the 22th of April, 2012, and the main goal was to get a first impression on the importance software development organizations give to the possibility of having usage information from their users in general, and in particular the degree of business interest about a first set of usage properties.

For the matter 14 individuals, representative of a software development organization (Sage Portugal), were used as a sample – Managers and Developers. From the invited universe 6 responded to all of the questions, 8 didn't accept the survey invitation. Next is the list of questions and answers:

Q1: Would you consider important to have a tool capable of analyze and visualize end- user's usage of your applications?

Options	Responses
Not important	**0%**
Little important	**0%**
Very important	**100%**

Q2: Which of the following do you consider relevant to know about user's IT equipment?

	Irrelevant	Little Important	Important	Total	Average
OS (Version, Service Pack, 32/64bits, Language)	**0%** 0	**16,67%** 1	**83,33%** 5	6	2,83
Memory (Total)	**0%** 0	**50%** 3	**50%** 3	6	2,50
Processor (Name, Type, Frequency, Colors, 32/64bits)	**0%** 0	**66,67%** 4	**33,33%** 2	6	2,33
Screen (Quantity, Main Resolution)	**16,67%** 1	**50%** 3	**33,33%** 2	6	2,17
Geolocation (Country, Region)	**0%** 0	**83,33%** 5	**16,67%** 1	6	2,17
Disk (Total)	**16,67%** 1	**66,67%** 4	**16,67%** 1	6	2,00
.NET Versions Installed	**16,67%** 1	**83,33%** 5	**0%** 0	6	1,83
Machine ID (e.g.: Disk Serial Number)	**83,33%** 5	**16,67%** 1	**0%** 0	6	1,17

Q3: In what concerns the usage of the application on the client, please evaluate the degree of importance of the following:

	Irrelevant	Little Important	Important	Total	Average
Versions in use	**0%** 0	**0%** 0	**100%** 6	6	3,00
Features used by the client	**0%** 0	**0%** 0	**100%** 6	6	3,00
Features usage flow in the application	**0%** 0	**16,67%** 1	**83,33%** 5	6	2,83
Installed features which were never used	**0%** 0	**33,33%** 2	**66,67%** 4	6	2,67
Number of users running those features	**0%** 0	**50%** 3	**50%** 3	6	2,50
Installations	**0%** 0	**50%** 3	**50%** 3	6	2,50
Uninstallations	**0%** 0	**50%** 3	**50%** 3	6	2,50
Possibility to interact with the user via desktop, in a predefined context	**16,67%** 1	**33,33%** 2	**50%** 3	6	2,33
License in use	**0%** 0	**83,33%** 5	**16,67%** 1	6	2,17

Analyzing the responses the following deductions were made:

Q1: All respondents clearly stated the importance of knowing how users interact with their applications, and the relevance of a tool that enables it.

Q2: Although running environment support is decided in advance by software producers, tracking user's environment information is vital for planning development investments – particularly operating system and architecture (32/64 bits) – because of its tight relation to software development tools. Hardware information features have less business value when compared to software information like OS. Usually, the importance of the former is mainly for knowing if development technology can be pushed forward without affecting the installed base.

Hardware features such as screen resolution and machine ID were less voted, but further discussion about these results suggested they should have higher importance. Screen resolution could have and immediate benefit to product development, because typically desktop applications lack display scalability and target fixed resolutions (e.g. 1024x784). Furthermore, because some license models are per machine, collecting the machine ID would be of business value when collected together with the license ID (see *Q3*).

Q3: Overall, all functional requirements were considered useful, but the spotlight was in product information (version) and feature usage. Any tracking tool must address these needs.

The less voted functional requirement was license ID. But, as previously discussed (*Q2* conclusions) license ID when combined with machine ID, in a per machine licensing model, both become of higher business value (e.g. tracking illegal use of software).