QualityMonitor: Improving Software Quality by Monitoring its Evolution

Alexandre Bergel, Marco Orellana, David Röthlisberger, Christian Palomares, Romain Charnay

University of Chile, Santiago, Chile

Abstract. The success of a software strongly depends on the ability of the software to adapt to new user requirements. However, modifying a software is not always an easy task. A high quality software is easier to adapt and improve. A software that is poorly conceived, independently whether it is functional or not, is costly to change and adapt.

Unfortunately, quality is difficult to achieve without proper monitoring tools and methodologies. Numerous tools have been proposed to assist software development, however they are either restricted to a specific problem range or are considered too "academic" (identified quality problems are not always understandable by a non-expert). Moreover, many of the available tools perform well to identify problems but provide little indication on how to remove those problems.

QualityMonitor is a product to monitor the quality of software source code. QualityMonitor innovates by delivering intuitive software visualizations to monitor quality. These "X-Ray" for software David ►*X Ray should be better introduced or omitted*◄ are accompanied with detailed but comprehensible indications on how to address quality deficiencies. QualityMonitor's visualizations are adjustable to the corporative programming conventions and particularities of the analyzed software, making it more flexible and agile than concurrent solutions.

In 2009, software maintenance costed the entire Chilean market 437 millions USD according to a survey sponsored by Microsoft and the Chilean government. The controlled experiments we realized in Europe and in South-America with our functional prototype suggest a significant reduction of maintenance cost.

We identified three large and prominent IT companies (Coasin[1], NIC Chile[2], Sonda[3]) that are expressing their interest in QualityMonitor. The problems these companies are facing are similar: a large software, developed over a long period, has to be maintained and enhanced with new features, however, the knowledge of its internal has evaded with changes of the development team. By drawing high level representations of software internals, QualityMonitor reengineers this knowledge David ►*what means "reengineering knowledge"?*◄, thus facilitating evolution and maintenance.

[1] `http://www.coasin.com`

[2] `http://www.nic.cl`

[3] `http://www.sonda.com/portada`

The product and services of QualityMonitor will be operated by Object Guidance, a recently created company. The international team behind Object Guidance is composed of 5 people. We are currently applying for a 90,000 USD grant, resources that will be used to shape the solutions of client requirements.

1 Problem: Difficulty to Bring Quality into Software

Software Quality by Example. During a workshop on software quality[4] in September 2008, A. Bergel demonstrated an early prototype of QualityMonitor. A quality assurance architect working for a major European car industry presented the following problem:

> *"I am in charge of the quality assurance for a large European automobile firm. The largest part of our production chain is controlled by a massive software. We have been working on this software for more than 15 years. The software has more than 2.5 millions lines of code and is developed in C (80% of the total amount of source code), C++ (19.5%) and few homemade languages (0.5%).*
> *Plotting the maintenance cost against the software change is an exponential curve: costs to maintain our software are increasing in a non linear fashion. It is becoming impossible to precisely predict and control the cost of each new release."*

This situation is found in any software producer that has not adopted a proper solution to monitor software development quality.

We addressed this problem with dedicated visualizations, using an earlier version of QualityMonitor for that very problem. Visualizations pinpointed components that are relatively easy to improve and without significant ripple effect. David ►*this paragraph is a bit in the air. Why mentioning an earlier version of QualityMonitor? Is it not better to mention what exactly QualityMonitor is doing now? Maybe it is better to replace this para by just saying that in the following we show how QualityMonitor can monitor software development quality.*◄

2 Concept and Product: QualityMonitor

Objectives. QualityMonitor is the product we envision to answer the following questions: *How to help development teams to easily maintain and continuously monitor the quality of their software programs?* and *What are the actions to be taken to improve the quality of software development?*. We address these questions using expressive visualization mechanisms. QualityMonitor produces for a given software a set of "radiographies" to immediately visualize code anomalies, suboptimal structure and assess the test coverage David ►*test coverage should once be*

[4] `http://www.systematic-paris-region.org`

explained◀. Visual patterns are associated with a given list of actions to efficiently react, in order to improve structure and test coverage of the software.

Visualization is often used in critical domains. For example in a hospital, body radiographies and analysis is the natural first step to seek the cause of symptoms. QualityMonitor is about doing radiography, not for human beings, but for software.

Example of Visualizations. Figure 1, Figure 2 and Figure 3 are three visualizations obtained from an open source software. These visualizations are called *test blueprint*, and they show which portion of the software is actually tested with the automatized and repeatable unit tests of the analyzed software David ▶*ok, test coverage is kind of explained here. Explanation could be extended and moved to where test coverage is first mentioned*◀.

To keep this proposal under the imposed space constraints, only the *test blueprint* visualization is presented. Many more visualizations are available though[5].

Figure 1 illustrates the principle of the visualization by using a exemplary software written in an object-oriented programming language. Most software systems are nowadays written in an object oriented language. Each large box is a class of the system. Class inheritance is indicated with edges. A superclass is above its subclasses. Each small inner box is a method. Invocation between methods is indicated with edges. An untested method has a red border. The height of a method is proportional to its complexity. The width says how many methods invoke the method. The gray intensity says how many times a method has been invoked by the tests (dark = tested many times).

The figure indicates that the classes `MOAbstractGraphLayout` and `C3` are relatively well tested: all but one of its methods are tested. The class contains some tall and dark methods, indicating that the methods have been executed many times and in many different situations. Not all of the subclasses are well covered. For example, `C1` is not covered at all. `C2` contains 3 non-tested methods. One of them is tall, which indicates a high complexity. Leaving a complex software component untested is a risk for the general health of the application. David ▶*not sure whether is good to mix dummy class names (C1, C2, C3) with real class names (MOAbstractGraphLayout*◀

Figure 2 shows the evolution of the class `MOTreeMapLayout`. In the version 2.2 of the software, only three of its methods are covered. The presence of red methods indicates the software components on which the testing effort should focus. Gradually, the class went from a coverage of 27.27% to 100%, in version 2.5. It is widely acknowledged that having a high test coverage has a positive effect to reduce software maintenance costs [**?**,**?**].

Visualizations help developers assessing the inherent complexity of a software. The upper part of Figure 3 represents a central class of the software. It contains many methods, with many dependencies between them. The lower part represents an improvement of the class: many methods have been removed (especially

[5] `http://www.moosetechnology.org/docs/visualhall`

obsolete and unused methods), leading to a global improvement of the class. Version 2.17 of the class contains significantly less methods and dependencies.

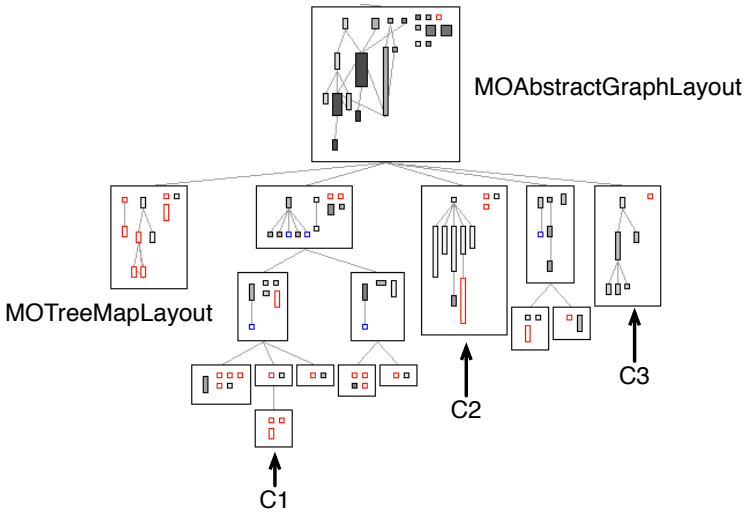

Fig. 1. Example of a test coverage visualization

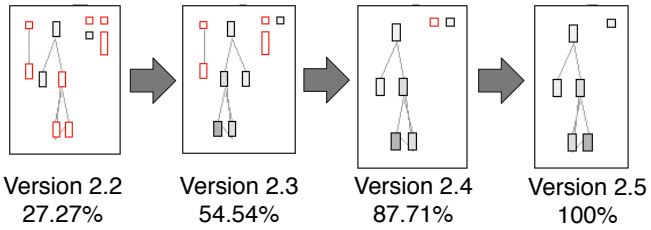

Fig. 2. Test coverage evolution

Remedy Actions. Visualizations identify deficiencies in the software source code. Each visualization comes with a set of actions to remedy the identified quality problems. Consider the example given in Figure 1 and Figure 3. Uncovered components are indicated in red. In that case, new tests may simply be written and the red boxes will turn gray.

Visualization play an important role in QualityMonitor. Instead of simply saying whether a component is covered or not, the visualization gives the *context* in which the component has to be considered. This is an important piece of

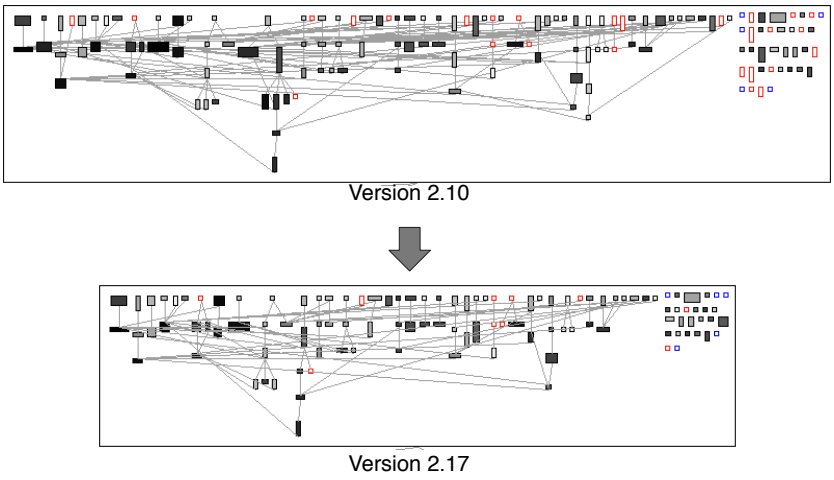

Fig. 3. Complexity reduction

information to decide whether or not the component is worth testing. Not all components require the same effort to test and not all of them are worth testing. QualityMonitor offers a set of interactions to drill down from the visualization to the source code, and tools to precisely identify source code of the visually represented components. Another benefit of visualizations is to monitor the reduction of complexity (Figure 3).

Related Work. Software quality has attracted the interest of numerous companies, including major software producers. Numerous tools have been produced to assist developers to control the quality of their code.

Microsoft produced Pex and Visio. Pex[6] automatically generates unit tests with a high coverage by employing sophisticated mathematical principles. Pex and the visualization presented in Figure 1 address the same problem: increasing the test coverage. They however diverge on the way how to achieve this. Pex produces executable and ready to use unit tests for a given program input. Pex however does not determine which components are already tested. Pex is also ineffective in reducing internal software complexity.

Visio[7] intensively uses visual diagrams for building data and control flow. It offers sophisticated capabilities to use a data base as input and Visio drives it along visual diagrams David ►*I do not understand this sentence*◄. Visio is advertised as a diagram tool constructor and as a way to quickly and easily manipulate data bases. Its focus is hence rather different than what QualityMonitor proposes. For example, Visio cannot identify complex software components, therefore it cannot help reducing software complexity.

[6] `http://research.microsoft.com/en-us/projects/pex`
[7] `http://office.microsoft.com/en-us/visio`

Fortify[8] is "a suite of tightly integrated solutions for identifying, prioritizing, and fixing security vulnerabilities in software. It automates key processes of developing and deploying secure applications." Fortify processes source code and applies some rules that check proper usage of APIs, programming conventions and security. It also provides dynamic trace David ►*dynamic trace might no be clear to the reader*◄ analysis capabilities. Fortify is about identifying and resorbing vulnerabilities. QualityMonitor has a different and complementary focus, which is monitoring the quality. The primary focus addressed by Fortify is "how to make software less vulnerable", QualityMonitor focuses on "how to make software easier and less costly to maintain".

It is well known that monitoring the evolution of software structure is important for the maintainability of it. Lattix[9] is a tool that graphically represents dependency cycles among software components. It uses a sophisticated "dependency structural matrix" to visually represent dependencies and their anomalies. The objective of Lattix is about identifying wrong dependencies. This is indeed within the range of QualityMonitor. QualityMonitor solves this very problem by using a different visual presentation (a graph instead of a matrix). Lattix is made for a one-shot analysis. Monitoring with Lattix has to be done manually, thus becoming an additional task to be manually performed on a regular basis. QualityMonitor automatizes this process by periodically and automatically sending code quality reports based on our visualization David ►*the fact that QualityMonitor sends reports is important, thus this should probably me mentioned at a more prominent place for the first time than in related work. What about adding it to the objectives paragraph at the beginning of this section?*◄. In addition, QualityMonitor enables visualizations to be tailored to a particular application, according to the culture and convention adopted in the company.

The solution for monitoring software quality offered by QualityMonitor differs from traditional approaches by a number of essential points. QualityMonitor is based on adaptable visualizations that are frequently and regularly generated by automatically assessing code quality. QualityMonitor unifies, as a single platform, effective ways to visualize the architecture and the testing of a software, unlike competitor's solutions that focus on only the visualization or only the testing. This unification makes QualityMonitor unique against the current solutions found in industry and academia.

What makes us attractive to our customers. We have identified three companies that expressed their interest in QualityMonitor: they are ready to let us assess their software source code and they will be responsive to our recommendations for improvement. We have tried to identify what are the reasons for their interest in QualityMonitor. According to the reactions we got when we demonstrated an early version of QualityMonitor, the following points play an important role in the future of QualityMonitor:

[8] `https://www.fortify.com/products/fortify360/index.html`
[9] `http://www.lattix.com`

- QualityMonitor is backed up with a significant number of scientific results. A. Bergel and D. Röthlisberger published in the most prestigious conferences (Section 6).
- People behind QualityMonitor are working at the University of Chile, a prominent institution: University of Chile is considered as the "top" Chilean University and is internationally well ranked[10].
- International composition of the Object Guidance team. An important socio-cultural factor in Chile is to positively perceive the interaction with Europa and North-America.
- Experience of the Object Guidance team. The presentations we give to our potential customers are driven by the problems we identified during our extensive experience in the field. Customers find themselves in the problems we exposes during our presentations.

Robust prototype. Moose[11] is an open-source software analysis platform. Moose is the result of an international effort, in which the University of Chile, INRIA and the University of Bern are the principal actors. Moose is a robust prototype, principally used for research purposes. Moose has been under constant development for more than 12 years. Based on numerous controlled experiments, we judge that Moose has now reached a level of maturity to conduct analysis on industrial and commercial software.

Moose is distributed under the MIT license[12], an open source license. This implies that anyone can take Moose and build a software similar than QualityMonitor. In practice, this represents a minimal risk. According to our experience in training people for Moose, it takes approximately 6 months for an engineer to be trained before being able to conduct software quality analysis with Moose. Based on the average salary of a software engineer in Chile (2,000 USD), this represents a cost of 12,000 USD, which is way above the cost of adopting QualityMonitor, estimated at 2,200 USD for a 6-month project (cf. next section).

Even if one would clone QualityMonitor, a significant amount of effort is necessary to be recognized in Chile:

- QualityMonitor and Moose have been regularly presented at SPIN Chile[13], the Chilean network about software process improvement and process quality. Presenting QualityMonitor ideas under a different product will need to be strongly motivated to be credible.
- Thanks to SPIN Chile, the QualityMonitor team has established links with several companies in Santiago that cannot be credibly undone by a newcomer with a clone of Moose.
- More importantly, we have an extended human expertise, backed up with numerous scientific publications in international scientific venues.

[10] `http://www.topuniversities.com/institution/universidad-de-chile/wur`
[11] `http://www.moosetechnology.org`
[12] `http://www.opensource.org/licenses/mit-license.php`
[13] `http://spinchile.wordpress.com`

For this reason, even if the possibility of cloning QualityMonitor exists, we believe this imposes a minor threat.

Moose has been employed to analyze the software of several large companies in France (Renault, Airbus, France Telecom), Argentina (Caesar Systems) and Chile (NIC Chile). Even though these analyses were successful (*i.e.,* problems were identified and plans for addressing them were proposed), Moose remains a research prototype and not a product: documentation is missing and its implementation is detached from practical considerations. Currently, conducting an analysis requires many technical steps, which represents a cost that a company cannot easily afford according to the experience we gained when realizing controlled experiments.

The purpose of this Intel Challenge proposal is to give the necessary resources to build QualityMonitor, a facade for Moose to ease its use and exploitation. QualityMonitor will offer a web interface to easily upload software source code and generate detailed reports.

QualityMonitor is a stand-alone application, independent from any programming environment. QualityMonitor analyzes software source code. The programming languages supported by the current prototype of QualityMonitor are Java, C David ▶*not C++ or C#? This is strange that C as non-OO language is supported*◀ and Smalltalk. Section 3 details the technological factor against the current market.

Benefits for the clients. Consider a software company consisting of 6 employees. In Chile, the average monthly salary per engineer, including taxes, is about 2,000 USD. The total cost for the salaries is therefore 12,000 USD per month. Over a seven-month project, the cost is therefore 84,000 USD. By using QualityMonitor, we forecast a reduction of the project length by one month. The cost of QualityMonitor for a 6 months period is 2,200 USD (200 USD per month plus 1,000 USD for the initial setup cost). **The total savings for the company for a seven-month project is 9,800 USD**. This forecast has been made according to our consulting activities previously realized, as described below.

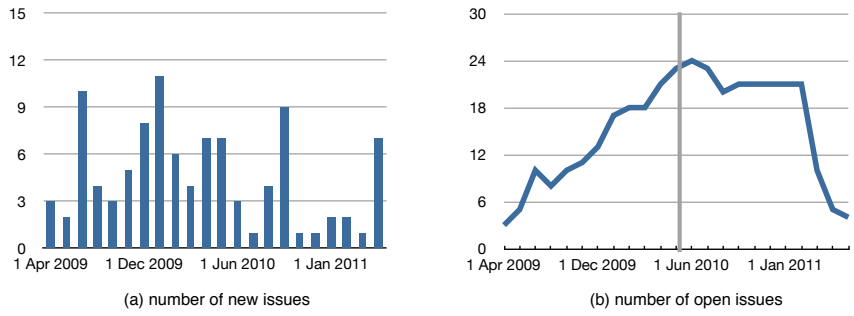

Fig. 4. Evolution of software issues.

To effectively measure the benefit of QualityMonitor, we use as a case study the development of a medium-sized software system[14] from April 2009 until January 2011. The software is architected and developed by 4 people, remotely distributed (1 developer in Chile, 2 in Switzerland and 1 in France). By being successful, the number of users of this software has rapidly increased, thus leading to frequent bug reports and requests for enhancements (what we call "issue" in general). The left-hand part of Figure 4 shows the distribution of software issues over the development period. Each of the 4 peaks (June 2009, January 2010, April 2010, April 2011) represents a deadline met by the customers of the monitored software: issues were found in the rush to meet their deadline.

The right-hand side of Figure 4 shows the number of open issues against time (less is better). The monitoring using QualityMonitor begun in May 2010. A first effect appeared in June 2010 with a reduction of the number of open issues: Visualizations and analysis given by QualityMonitor helped developers addressing the issues by improving the quality of the software in a step-wise fashion (as illustrated in Figure 2 and Figure 3). Since QualityMonitor is employed the number of open issues has been in a constant reduction, demonstrating that the development process is able to quickly absorb customer requests and bug fixes.

The total length of the project is 22 months. By discussing the benefit of QualityMonitor with the 4 developers of the monitored software, it appears that our expertise helped them shortening the development time by 3 months, thus 1 month for 7 months of development.

QualityMonitor and quality certification. Our primary target is the Chilean market as a short term penetration. The Chilean software industry is avid to quality certification in order to gain international credibility. This is particularly relevant when producing software system for governmental agencies, military and exporting outside national boundaries.

The *Capability Maturity Model Integration* (CMMI) is a process development approach for companies producing software systems. Obtaining a CMMI level 2 or higher is often the ambition of Chilean companies who wants to broaden their market segments.

Associating QualityMonitor to the CMMI certification is a perspective we started to investigate. The unique Chilean company to deliver CMMI certification is America XXI[15]. The QualityMonitor team visited America XII twice, in Oct 2009 and Apr 2011. The executive direction of America XXI welcomes QualityMonitor and they expressed their interest in collaborating. We plan to visit them again in February 2012, after the propulsion of QualityMonitor. Their interest in QualityMonitor stems from their weakness to analyze application source code. Most of their evaluation effort is concentrated on the process and the formalization of the requirements. Our expertise complements well theirs.

[14] The system is developed in an object-oriented programming language and is large of nearly 200 classes and 2,000 methods.

[15] `http://www.americaxxi.cl`

3 Market Analysis

IT Market in Chile. According to a study realized by the GECHS[16] and CETIUC[17] and sponsored by Microsoft and InnovaChile, the Chilean industry of information technology comprises 1,871 companies, in which only 350 are involved in producing software and IT services. These 350 companies are developing and selling custom software and general IT solutions. The average income in 2009 for each of the 70 small and medium companies part of the GECHS is evaluated at 2.5 millions USD.

The study further shows that the necessity of IT solutions has generated an annual increase of 15% of income for the Chilean market, since 2008. The main services offered in Chile for the international market are software license David ►*can you elaborate on that? it is not clear to me what "software license" is*◄ (representing a share of 31% of the total income) and general project administration (17% of the total income). According to the GECHS, small and medium companies spend 52 millions USD to maintain their software, every year.

These figures emphasize the importance of controlling the quality of what is being produced and consumed, especially in an emerging country such as Chile.

Software Quality and Software Engineering. Software engineering is expensive and large resources are necessary to deliver software of quality (*i.e.,* bugfree, robust, extendable with new features at low cost). Sommerville [**?**] and Davis [**?**] estimate that the cost of software maintenance accounts for 50% to 75% of the overall cost of a software system. Applied to the Chilean industry, these figures reveal that each of the 70 surveyed companies spends between 1.25 and 1.87 millions USD. We estimate the lower bound of the cost of software maintenance in Chile to be 437 millions USD, extended to the 350 companies involved in producing software and IT services.

There are currently no global (both international and national) solutions to reduce the cost of software maintenance and controlling the quality of solution development David ►*you mean for the Chilean market or in general? There are many means to reduce software maintenance costs, so maybe you should re-formulate the sentence a bit*◄. Symptoms of software quality problems are:

- The author of a poorly written, but functional, critical component may gain an excessive importance in the development team. We have seen numerous situations in which the departure of an engineer may endanger the whole company. By being poorly written we mean that the component is hardly understandable by other developers.
- In the case of a departure of a central software engineer, completely rewriting parts of the software is often the only feasible way to address new customer requirements.

[16] Sociedad Chilena de Software y Servicios, `http://www.gechs.cl`
[17] Centro de Estudios de Tecnologias e Información, `http://www.cetiuc.cl`

– Outsourcing software development has a tendency to significantly increase the cost of software maintenance David ►*reference needed here as this saying is against common belief in IT industry*◄. For example, a number of critical software programs of Banco de Chile have been developed in India. As a result, the way to fix software problems is often to send Indian experts to Chile, which incurs high cost and long delay.

These problems are well known. Solutions that are commonly employed to prevent or address them are:

– *Training.* Many Chilean universities offer professional training programs to level up engineers and developers. These programs are usually costly and very time-consuming with a uncertain return on investment.
– *Metric control.* ISO-based quality models employ metrics to measure software development progress. ISO models are employed in several large European companies[18]. Similarly to the field of economy, metrics can be aggregated in a dashboard to monitor the software evolution[19]. These models are efficient to detect lack of quality in a software's source code, however they are of little help on how to fix the situation since metrics output (*i.e.,* numbers) are not directly mappable to defects.
– *Dedicated consulting.* Externalizing the control quality is a common practice[20]. This is often a punctual effort that cannot be easily used for a continuous monitoring and improvement.

Technological factor. The current prototype of QualityMonitor analyzes applications written in the Java, C, and Smalltalk programming languages. In 2008, about 40% of software produced in Chile used the Java programming language and 17% used .Net[21]. The share of Java is estimated at 25% David ►*in the sentence above it says 40%?*◄ and 50% for .Net. Targeting .Net languages, including C# and Visual Basic, are critical for a deep penetration of QualityMonitor. David ►*again, how fits C in this picture here? No numbers are given for C (neither for Smalltalk) — do you actually mean C#? It also does not become clear whether .Net / C# is already (partially) supported by QualityMonitor or whether one of the goals of this proposal to add such support.*◄

Expected grow. South-America is often seen as an outsourcing place for software development. Our participation in the SPIN Chile, an industrial network for companies having an interest in software and process quality, revealed that a large proportion of companies are willing to gain access to the European and North-American market. In addition, the number of students in universities

[18] `http://www.qualixo.com/Squale/squale.html`
[19] `http://www.castsoftware.com`
[20] Many companies offers service for testing and assessing the quality of architecture, *e.g.,* `http://www.americaxxi.cl`
[21] Source: "Sexto Diagnostico de la Industria Nacional de Software y Servicios" realized by the GECHS.

taking Computer Science lectures is slightly increasing, which is contrary to the global trend: most European universities see the number of students in Computer Science diminishing. All this provides solid guarantees that the Chilean IT market will continue to grow, as it did over the last few years.

Market segment. The core market segment we are targeting are companies that produce software in Chile. We estimate the number of potential companies at 350, as mentioned earlier.

Customer profile. The three companies that expressed their interest in QualityMonitor have the same problems to solve: a critical legacy application that has to be maintained and possibly enhanced. Developers that produced the original versions moved into different projects or companies, leaving the software maintainers without an adequate knowledge base.

Our ideal customer profile includes producers of software that either develop *locally* or *externalize* the development.

Government regulations. The Chilean political context is favorable to QualityMonitor objectives. Currently, there is a regular tax of 15% on selling software. This tax is usually perceived on software delivered with hardware purchases. This tax will be removed in 2011 David ►*you know the exact date?*◄ as an effort to favor the Chilean industry (which also consumes software) and fight against illegal software copies.

4 Business Model Canvas

We present the business model canvas [**?**] associated to QualityMonitor.

Customer segment: companies developing software, either for themselves or for tierce clients.

Value proposition: (i) controlling the quality of software; (ii) increase the performance of the software David ►*this point has been mentioned in the text above?*◄; (iii) reduce the dependency between engineers and the produced software; (iv) real-time monitoring of the software development.

Channels: (i) The QualityMonitor product will be distributed via the Web (the software to be analyzed is uploaded on a website) or (ii) installed onsite at the customer's headquarters. (iii) Adequate capacitation David ►*can you elaborate on this term? not clear whether you mean consulting, tools, training, etc.*◄ will then be offered to addressing defaults found by QualityMonitor in the analyzed software applications.

Customer relationships: personal assistance and product sell.

Revenue stream: per use, license, subscription.

Key resources: (i) qualified human capital (excellent software programmers and engineers are essential for the business since they will have to report detailed

analysis David ►*unclear. report on the detailed analysis of the tool, perform the detailed analysis?*◄; (ii) non-disclosure agreement of the software analyses.

Key activities: analyzing software and proposing improvement to remove quality-associated defaults. Improvements will have to be quickly proposed to the clients.

Key partnerships: (i) Community around the Pharo programming language[22] (Pharo is used to build Moose); (ii) Community around the Moose software analysis platform.

Cost structure: (i) proposition for a premium-level quality; (ii) motivation for the value of the service; (iii) principal cost associated to the human capital.

5 Timeline and budget

Extending the range of supported programming languages. C#, Visual Basic, PHP. David ►*this sentence is quite in the air?*◄

We envision the following milestones for the first year:

- August 2011 - Creation of Object Guidance
- October 2011 - Government sponsored 90,000 USD grant, *Capital Semilla*. Two full time engineers will then be enrolled.
- March 2012 - Implantation of QualityMonitor in two of our customers
- June 2012 - Online services publicly offered

Figure 5 presents the income and cost of Object Guidance. We expect an initial cost of 6,000 USD for computer acquisition. On Month 4 we will then have a third employee, in charge of the marketing. This will raise the cost up to 6,000 USD per month for the salaries.

Each new client will bring an income of 1,000 USD, plus a month subscription of 200 USD. This subscription will cover the cost and maintenance of the project monitoring. The initial fee for the clients and the subscription is the reason why the curve of income is not smooth over the time.

Income will be greater than the cost on Month 14. Once QualityMonitor will be reach a stage of commercialization, for 8 consecutive months we will have between 4 and 5 new clients per months.

We expect the following figure for the first two actives years of QualityMonitor:

Months	Income	Cost
1 - 12	3,600	72,000
12 - 24	88,000	72,000

Appendix A details the prevision made above.

[22] `http://www.pharo-project.org`

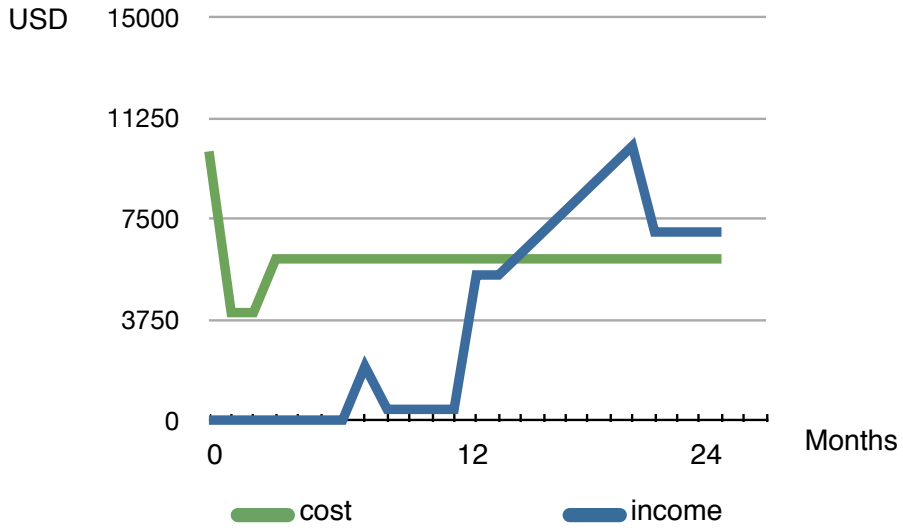

Fig. 5. Cost and income of Object Guidance

Grant. Capital Semilla is a program sponsored by the Chilean Government (CORFO) to ease the creation of startups[23]. Together with the Novos incubator, we are postulating to the Capital Semilla program. After being interviewed and having our description reviewed by a selective panel, we have been selected for the final phase. The incubator is confident we will obtain the grant since we comply with the priorities set by CORFO:

- we have a strong research result that has been scientifically recognized[24] and validated in European companies
- we have a strong and stable contact with international research institutions (A. Bergel is the principal investigator of a major project with INRIA[25], the French Research Institute in Computer Science)
- there is a realistic perspective to export abroad QualityMonitor, a technology developed in Chile. QualityMonitor has been tried in a company in Argentina, and contributors to the Moose open source project (located in Switzerland and in France) are positively welcoming our effort. The situation is favorable for using companies associated to Moose contributors as a testbed for QualityMonitor.

[23] `http://www.corfo.cl/lineas_de_apoyo/programas/capital_semilla_apoyo_a_la_puesta_en_marcha`
[24] `http://bergel.eu/publications.html`
[25] `http://inria.fr`, `http://pleiad.dcc.uchile.cl/research/plomo`

Initial engineering force. Object Guidance will be entitled to receive internship students from the University of Chile. This mechanism will be used to identify potential engineers.

The average salary in Chile is about 2,000 USD per month for an experienced programmers. We plan an initial investment of 6,000 USD for the necessary infrastructure for software development (two laptops and one server). Even if no agreement has been signed already, local and internet connection are likely to be provided by the University of Chile and the Novos incubator.

The 90,000 USD grant will therefore cover the initial 6,000 USD, leaving 42 man month, which will be used to develop QualityMonitor.

The first prize of the Desafio Intel competition is 15,000 USD. The award will be used to initiate a marketing force by hiring a dedicated person for 6 months, full-time, totaling 12,000 USD. The remaining 3,000 USD will be used to development the corporative image of Object Guidance, which includes creation of a website and an advertising strategy.

6 Object Guidance: The Company and the Team

Object Guidance is the company that will sell the QualityMonitor products, sell services around the product and provide consulting in software quality. The team is composed of 5 people, including 2 academics for the scientific consulting, 1 civil engineer to define the development strategy, 1 engineer in computer science and 1 executive at the incubator.

The current Team involved in the creation of Object Guidance is:

- Prof. Dr. Alexandre Bergel – PhD University of Bern, Switzerland. Assistant Professor at the University of Chile. Expert in Software Quality and Software analyzes. Website: `http://bergel.eu`
- Dr. David Röthlisberger – PhD in computer science and Bachelor in business administration from the University of Bern, Switzerland. Postdoctoral in the Pleiad laboratory at the University of Chile. Expert in Software Quality and Software analyzes. A. Bergel and D. Röthlisberger know each other for more than 8 years: both did their PhDs at the University of Bern. Website: `http://www.droethlisberger.ch`
- Marco Orellana Fuenzalida – Civil Engineer in industry and Computer Science. University of Chile. Currently in charge of the business plan of QualityMonitor and contacting clients. Marco has worked with R. Charnay and A. Bergel since October 2010. Marco will be full time employee at Object Guidance.
- Christian Palomares – Civil Engineer in Computer Science. University of Chile. Experimented programmer. C. Palomares is a student of A. Bergel. They know each other for more than 1.5 years. Christian will be full time employee at Object Guidance.
- Romain Charnay – Executive at the Novos[26] incubator. R. Charnay and A. Bergel have been working for more than 2 years on defining the strategy to

[26] `http://novos.cl`

create Object Guidance and launch QualityMonitor. R. Charnay has provided invaluable advises.

Object Guidance will have strong links to the University of Chile since Alexandre Bergel and Dr. David Röthlisberger will maintain their positions at the University of Chile. However, the company will be legally independent of the University of Chile. No legal mention of the benefits will therefore be made.

Object Guidance will be created in August 2011 as a stock-based company. At the moment of writing this business plan, all the paperwork has been provided to Jaime Luarte Correa, lawyer at Brokering & Luarte, a company that provides legal services[27]. The corporative identity has been realized by Volta design[28], it is available online[29].

Object Guidance is advised by Novos, the incubator associated to the University of Chile. Novos is helping the creation of Object Guidance by establishing the contact with designer, lawyer and the Chilean public funding agency for the Capital Semilla.

Object Guidance will be located in Santiago, Chile. Each member of the team will be physically located in Santiago as well.

A Cost and income for Object Guidance

All figures for the **income**, **cost** and **expenses** are given in USD.

[27] `http://bylabogados.cl`
[28] `http://www.estudiovolta.com`
[29] `http://www.estudiovolta.com/portafolio/quality-monitor`

Month	Income	Cost	New Clients	New employees	Expenses
1	0	10000	0	2	6000
2	0	4000	0	0	0
3	0	4000	0	0	0
4	0	6000	0	1	0
5	0	6000	0	0	0
6	0	6000	0	0	0
7	0	6000	0	0	0
8	2000	6000	2	0	0
9	400	6000	0	0	0
10	400	6000	0	0	0
11	400	6000	0	0	0
12	400	6000	0	0	0
13	5400	6000	5	0	0
14	5400	6000	4	0	0
15	6200	6000	4	0	0
16	7000	6000	4	0	0
17	7800	6000	4	0	0
18	8600	6000	4	0	0
19	9400	6000	4	0	0
20	10200	6000	4	0	0
21	7000	6000	0	0	0
22	7000	6000	0	0	0
23	7000	6000	0	0	0
24	7000	6000	0	0	0

Fig. 6. Cost and income for Object Guidance.