RC25019 (C1006-024) June 29, 2010
Other

IBM Research Report

Measuring the Sustainability Performance of Software Projects

Felipe Albertao, Jing Xiao, Chunhua Tian
IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100193
P.R.China

Yu Lu, Kun Qiu Zhang, Cheng Liu
Northeastern University

Measuring the Sustainability Performance of Software Projects

Felipe Albertao, Jing Xiao, Chunhua Tian
IBM Research - China
{albertao, xiaojcrl, chtian}@cn.ibm.com

Yu Lu, Kun Qiu Zhang, Cheng Liu
Northeastern University
prothi@163.com, uniworldson@126.com, lectery@gmail.com

Abstract

Contrary to the common assumption that software is "environmentally friendly" simply because it is virtual, the processes and methods used to develop, maintain and deploy software do have an environmental, social and economic footprint. For example, e-waste could be greatly minimized if software vendors would take into consideration the lifetime of old hardware. Like so, software that is dependable minimizes waste of resources used to support the system. This paper introduces a set of software engineering metrics that can be used to assess the sustainability of software projects, and it demonstrates how these metrics were used to assess a real project.

1 Introduction

Information Technology plays a fundamental role in addressing environmental, economic and social concerns: IT can not only immaterialize activities that otherwise would consume resources (e-mail versus postal mail, virtual meetings versus travel) but also the capability to extract knowledge to optimize resource-intensive processes (such as analytics for water consumption and smart grids).

On the other hand, the impact caused by the development of IT products are very rarely accounted or even acknowledged across the industry. For example, it is estimated that one computer becomes obsolete for every new one put on the market [1]: In other words, the computer hardware itself is still usable, but the software rendered it useless or undesirable. If software developers took that fact into consideration, perhaps products would be designed from the ground-up to work on older hardware.

This paper outlines a method to quantify factors that (positively or negatively) impact the environment, the economy and society, with the objective of raising the awareness that software should be developed in a sustainable manner.

2 Previous Work

- This study is a follow-up of a methodology created by one of the authors of this paper [2].
- Eckart Wintzen has invented and implemented a concept called “Environmental Accounting” at Origin (an European outsourcing company) in the early nineties [3]. That was not a methodology per se, but ballpark estimates based on input from project managers. However, Eckart Wintzen was the first to recognize the environmental impact of “virtual” IT-based processes.
- IfPeople (an US-based IT company focused on non-profit organizations) has published guidelines for socially responsible IT practices [4]. However, the standard is focused on the adoption and operation of IT products, rather than the software development process.

3 Proposed Solution

The method outlined in this paper introduces a set of *Sustainability Performance Metrics* that can be used to improve the environmental, economic and social aspects of a software project, following three steps:

1. Assess the *Sustainability Performance Metrics* at the end of a software release cycle.
2. Analyze the *Sustainability Performance Metrics.*
3. Establish *Sustainability Improvement Goals* for the next software release cycle.

Most of the metrics do not have a “good” or “bad” result per se. The goal is to use the metrics as a basis for continuous improvement, where results are monitored and compared after each software release cycle.

4 Sustainability Properties

The *Sustainability Performance* is measured and analyzed against a set of properties [5], which if improved will bring economic, social and environmental benefits:

4.1 Development-Related Properties

Properties that impact the development process.

- *Modifiability*: The ability to introduce changes quickly and cost effectively.
  - Economic Benefit: Minimizes development and support costs.
  - Social Benefit: Enables system to be continuously adapted to meet societal demands.
  - Environmental Benefit: Minimizes environmental waste through less effort in producing and maintaining existing system.
- *Reusability*: Level in which system components can be reused in other systems.
  - Economic Benefit: Accelerates time-to-market.
  - Social Benefit: Enables the production of new products with less effort.
  - Environmental Benefit: Minimizes environmental impact through less effort in producing system.
- *Portability*: Ability of the system to run under different computing environments.
  - Economic Benefit: Increases potential market and system's lifetime.
  - Social Benefit: Reduced cost for technology adoption, by minimizing user's dependency on latest technology.
  - Environmental Benefit: Minimizes e-waste by extending the lifetime of old hardware.
- *Supportability*: System's ability to be easily configured and maintained after deployment.
  - Economic Benefit: Increased customer base due to reduced support costs.
  - Social Benefit: Vendor's independence increases the product usability and thus accessible to a larger population.
  - Environmental Benefit: Indirect benefit: minimizes resources required to provide support (transportation, physical material, etc...)

4.2 Usage-Related Properties

Properties that impact the user at run-time.

- *Performance*: The time required by the system to respond to user requests.
  - Economic Benefit: Improves productivity.
  - Social Benefit: Minimizes dependency on latest technology.
  - Environmental Benefit: Minimizes e-waste by extending hardware lifetime. Minimizes energy consumption through less computer usage time.
- *Dependability*: The ability of a system to function as expected at any given time.
  - Economic Benefit: Minimizes support and maintenance costs.
  - Social Benefit: Increases societal productivity.
  - Environmental Benefit: Indirect benefit: Minimizes energy waste.
- *Usability*: Features that enable a system to be user friendly.
  - Economic Benefit: Increases customer satisfaction. Minimizes support costs.
  - Social Benefit: Contributes to the digital inclusion, by eliminating barriers (learning curve) and making system more accessible to a broader number of users.
  - Environmental Benefit: Indirect benefit: Less waste of resources used in training (books, training rooms, energy, etc...).
- *Accessibility*: The system's ability to serve people regardless of location, experience, background, or the type of computer technology used.
  - Economic Benefit: Increases potential market and/or audience.
  - Social Benefit: Enables technology to minorities, elderly, people with disabilities, non-English speaking communities, and illiterate population.
  - Environmental Benefit: Increases multicultural awareness and provides equal opportunities. Indirect benefit.

4.3 Process-Related Properties

Properties that impact project management

- *Predictability*: The team's ability to accurately estimate effort and cost upfront.
  - Economic Benefit: Minimizes risks of budget overrun.
  - Social Benefit: Increases team's conditions of work (avoid long workhours).
  - Environmental Benefit: Optimizes use of environmental resources.
- *Efficiency*: The overhead of production processes over the bottom-line value perceived by the customer.
  - Economic Benefit: Maximizes product value.
  - Social Benefit: Minimizes effort waste.
  - Environmental Benefit: Optimizes use of environmental resources.
- *Project's Footprint*: Natural resources and environmental impact used during software development.
  - Economic Benefit: Indirect benefit.
  - Social Benefit: Indirect benefit.
  - Environmental Benefit: Reduces fuel consumption and emissions, officespace utilization, and maximizes use of shared resources.

5 Metrics and Analysis

The metrics used to assess each one of the properties are described below, along with the Sustainability Performance findings for the Urban Water Management Platform (UWMP), a software project developed by IBM Research - China, Northeastern University and the Shenyang Eco-city Research Institute (SERI).

5.1 Modifiability and Reusability

Systems with a high number of interdependencies are hard to maintain because the impact of a given change is hard to assess. In order to address this problem, a common practice in object-oriented design is to measure the dependency among the classes of a given system[6]. The first metric to be analyzed is the *Instability*, which measures the potential impact of changes in a given package:

$I = Ce/(Ca + Ce)$

Where:

- *Afferent Couplings* (Ca): The number of classes outside a package that depend upon classes within the package.
- *Efferent Couplings* (Ce): The number of classes inside a package that depend upon classes outside the package.

Instability ranges from 0 to 1, where 0 means that the package is maximally stable, and 1 means that the package is maximally unstable.

However, a system maximally stable is also unchangeable, and in fact some packages must be unstable enough to allow changes. Therefore it is also necessary to measure how much a package can withstand change, which in object-oriented design is accomplished by the use of abstract classes. This metric is called *Abstractness*:

$A = Na/Nc$

Where:

- Na: Number of abstract classes in a given package
- Nc: Number of contrete classes in a given package

Abstractness ranges from 0 to 1, where 0 means the package is completely concrete and 1 means completely abstract.

We can now measure the relationship between Instability and Abstractness: The two extremes (a maximally stable and concrete package, versus a maximally unstable and abstract package) are both undesirable: The ideal package is one with a "balanced" Instability and Abstractness. The final metric measures how far a package is from the idealized balance, or the *Distance From Main Sequence*:

$D = |A + I - 1|$

Distance ranges from 0 to 1, where 0 means the package has the ideal balance and 1 means the package requires redesign and refactoring.

The table below shows the metrics for the UWMP source code:

Package	Ca	Ce	I	A	D
Java: map.bean	8	2	0.20	0	0.80
Java: map.io	2	3	0.60	0	0.40
Java: common	1	0	0	0	1
Java: kpi.servlet	0	1	1	0	0
Java: map.business	0	4	1	0	0
Flex: comp	2	19	0.90	0	0.10
Flex: comp.key	1	1	0.50	0	0.50
Flex: kpi	1	1	0.50	0	0.50
Flex: beans	21	1	0.04	0	0.96
Flex: maps	1	18	0.95	0	0.05
Flex: business	1	1	0.50	0	0.50
Flex: events	20	1	0.05	0	0.95

The metrics indicate a high number of packages that require redesign (D close to 1). The fact that packages were organized by design patterns (beans, business, servlets, io) rather than their actual functions are the most likely cause for such instability. The key conclusion is that the packages should be reorganized by system functionality.

5.2 Portability

The goal is to maximize the hardware's lifetime by its actual physical durability rather than forcing its obsolence by software platform requirements. Therefore it is desirable to measure the *Estimated System Lifetime*, or the estimated number of years the minimum hardware required by the system reached the market.

UWMP depends on the following platforms and components: Adobe Flash Player 10, Java 6 on Windows XP, IBM SPSS Statistics 18, Geoserver and PostgreSQL. By investigating its minimum hardware requirements, we have estimated that the system can run in hardware as old as from 2003, giving it a lifetime of 7 years. We concluded that 7 years is an acceptable timeframe based on the types of PCs used by the UWMP's target users, and therefore the system is not going to require unecessary hardware upgrades.

5.3 Supportability

The metric *Support Rate* was used, which is calculated by the number of user questions that required assistance divided by the number of minutes the system was used in a given session. Because UWMP is a new system with no support history, we have used the results of the usability study to calculate the Support Rate, as such: The usability test subject performed 4 tasks that required assistance / 8.3 minutes = 0.48

UWMP is a web-based system, not requiring installation on the user's desktop. Desktop-based systems can also use the *Estimated Installation Time* metric, which is the amount of time the user takes to install the product without assistance.

5.4 Performance

A measure of performance is relative to the objective of the system. Since our goal is to improve user's productivity, we use the *Relative Response Time* metric, which is the number of tasks with an unnaceptable response-time divided by the total number of tasks tested. The usability study indicated one task with an unnaceptable response-time from a total of 11 tasks, or a Relative Response Time of 0.09.

5.5 Dependability

Defect Density (that is, how many defects a system has relative to its size) is a common software engineering metric, however that metric alone could be misleading because an inefficient testing process could cause the defect density to be low. Therefore we used the following metrics:

- *Defect Density*: Known defects (found but not removed) / LOC (lines of code):
  4 known defects / 2,111 LOC = 0.002

- *Testing Efficiency*: Defects found / Days of testing:
  9 defects found / 16 hours of testing = 0.56

- *Testing Effectiveness*: Defects found and removed / Defects found:
  5 defects removed / 9 total defects found = 0.55

5.6 Usability

Three of the Nielsen's usability attributes[7] were used to assess the system's usability:

- *Learnability*: Number of minutes to accomplish first critical task without assistance / Number of minutes system was used by user
  0.43 / 8.3 = 0.05

- *Effectiveness*: Number of tasks completed without assistance / Total number of tasks
  5 / 11 = 0.45

- *Error Rate*: Number of tasks which were completed but deviated from normal course of action / Total number of tasks
  2 / 11 = 0.18

5.7 Accessibility

We reviewed accessibility based on a list of requirements[8] and using the following score: 0=Non Existent, 1=Not Adequate, 2=Acceptable, 3=Adequate.

- *Support for motor impaired users*: 1 = Flash Player 10 supports motor control, but UWMP does not provide shortcuts

- *Support for visual-impaired users*: 1 = Flash Player 10 supports color adjustments, however it does not include features for low-vision users, such as zoom-in or font sizing.

- *Support for blind users*: 2 = Flash Player 10 supports screen readers.

- *Support for users with language and cognitive disabilities*: 0 = UWMP uses very technical language, and no investigation was carried to find out what professionals with such disabilities in the water management industry do to overcome such challenges.

- *Support for illiterate users*: 2 = UWMP has no support for illiterate users, however it is extremely unlikely that the target user (water-management professionals) will be illiterate.
- *Internationalization and localization support*: 2 = UWMP supports English and Standard Mandarin, the most common languages spoken by the target users. UWMP does include a few localized features such as dates, however additional changes are necessary, such as correctly translating indicators that use the chinese character "wan" (which means "10,000") to the English "thousand" or "million" (for example: "78 millions" instead of "7800 ten-thousands").

5.8 Predictability

The goal is to measure how effectively the development team is able to estimate effort and cost. Normally this is measured by simply counting the number of days a project was delayed. However, that metric is not effective for Agile-based projects because a project is never "late" per se. Instead, Agile practitioners prefer to remove features and tasks from the plan.

The UWMP team uses a variation of the Extreme Programming methodology, based on points instead of hours worked. First we have compared the number of points initially estimated at the beginning of each iteration with the number of points used at the end of each iteration. Then, the *Estimation Quality Rate* metric was calculated by dividing the number of iterations where the difference was +/- 20% by the number of total iterations in the project. The result was zero, given that no iteration had more or less than 20% difference in points.

5.9 Efficiency

The *Project Efficiency* is measured by the effort towards deliverables that add direct value to the customer (such as coding, manuals, etc. . .) versus project-related effort (such as infrastructure maintenance, project management, etc...). In the UWMP project, 38 points were used in infrastructure-related tasks, out of a total of 310 points (12.25%). Therefore the Project Efficiency is 88%.

5.10 Project's Footprint

It is hard to accurately calculate the amount of natural resources used by the team, such as fuel or paper. However, the goal is not to extract an exact amount, but instead to allow the team to progressively improve the metrics over time. We have chosen two metrics that reflect resource-intensive activities, such as transportation from/to the office, and long-haul trips:

- *Work-From-Home Days:* 2 days out of 165 total team-days (33 project days * 5 team-members) = 1.21%
- *Long-Haul Rountrips*: By airplane: 6; By train: 0.

6 Results

Based on the analysis above, the following problems were identified:

1. *Modifiability and Reusability:* Packages organized by design instead of functionality.
2. *Portability:* No action required.
3. *Supportability:* No action required.
4. *Performance:* Low response time of the "Data Audit Analysis" function.
5. *Dependability:* No action required.
6. *Usability:* No action required.
7. *Accessibility:* Short-cuts not available; Zoom-in or font-sizing not available; Chinese "wan" not correctly translated.
8. *Predictability:* No action required.
9. *Efficiency:* No action required.
10. *Project's Footprint:* Team-members rarely work from home; Zero train trips.

The list above was prioritized based on 3 attributes (customer value, effort, and level of improvement) and the following specific *Sustainability Improvement Goals* were defined for the next release of the software:

- *Goal #1:* Improve the "Data Audit Analysis" response time by 50%
- *Goal #2:* Redesign the package structure and improve the Distance metric on at least 5 packages.
- *Goal #3:* Prioritize train trips whenever possible.

7 Conclusion and Future Work

This study demonstrates that is feasible to incrementally improve the sustainability of software projects by measuring a set of metrics over several releases of a software product. Further work is necessary in order to develop benchmarks for the metrics, by applying this methodology on several other projects of different sizes. Such effort would be greatly facilitated if the collection of the metrics were automated.

References

[1] Poison PCs and Toxic TVs, Silicon Valley Toxics Coalition, 2001.

[2] Felipe Albertao, Sustainable Software Engineering, Carnegie Mellon University, 2004. http://www.scribd.com/doc/5507536/Sustainable-Software-Engineering

[3] Rogier van Bakel, Origin's Original, Wired Magazine, Issue 4.11, 1996. http://www.wired.com/wired/archive/4.11/es_wintzen.html

[4] IfPeople, ResponsibleIT Standard. http://www.ifpeople.net/learn/resources/sustainability/responsibleIT-ifpeople.pdf

[5] Most properties are in fact Quality Attributes. See Len Bass, Paul Clements, and Rick Kazman, Software Architecture In Practice - CMU SEI Series, Carnegie Mellon University

[6] Robert Martin, OO Design Quality Metrics: An Analysis of Dependencies, 1994. http://www.objectmentor.com/resources/articles/oodmetrc.pdf

[7] Usability Engineering, Jakob Nielsen, 1994

[8] List partially adapted from the Web Accessibility Initiative, Ball State University. http://www.bsu.edu/web/bsuwai/use.htm