Building Sustainable Software for Sustainable Systems: Case Study of a Shared Pick-Up and Delivery Service

Christophe Ponsard
CETIC Research Centre
Belgium
christophe.ponsard@cetic.be

Renaud De Landtsheer
CETIC Research Centre
Belgium
renaud.delandtsheer@cetic.be

Fabian Germeau
CETIC Research Centre
Belgium
fabian.germeau@cetic.be

ABSTRACT

Sustainability has become a major concern of the 21st century. With the digital revolution, ICT is actually part of both the problem and the solution. Despite this, sustainability-related design decisions are often left implicit and result from a process involving synergies and trade-offs among many non-functional requirements, including some constraints related to the software development and operation itself. The purpose of this paper is to identify such decisions and analyse the process that resulted into them based on a real-world industrial case with strong sustainability goals: a shared pick-up and delivery service. We also show how available methods for green software engineering help in better shaping this process and highlight some interesting lessons learned from our experience.

CCS CONCEPTS

• **Social and professional topics** → **Sustainability**; • **Software and its engineering** → **Requirements analysis**; *Software architectures*; • **Computing methodologies** → *Optimization algorithms*;

KEYWORDS

Sustainability, Goal Analysis, Software Architecture, Optimisation, Caching

ACM Reference Format:
Christophe Ponsard, Renaud De Landtsheer, and Fabian Germeau. 2018. Building Sustainable Software for Sustainable Systems: Case Study of a Shared Pick-Up and Delivery Service. In *GREENS'18: GREENS'18:IEEE/ACM 6th International Workshop on Green And Sustainable Software , May 27–26, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3194078.3194083

1 INTRODUCTION

Sustainability has become a major challenge of our century as we are now facing the reality and even consequences of reaching the fundamental limits of our planet's resources. In 1987, the United Nation Brundtland Report already had defined sustainable development as a "development that meets the needs of the present without compromising the ability of future generations to meet their own needs" [17]. Since this report, the tremendous development of Information and Communication Technologies (ICT) has introduced a new factor that can be an enabler to better control our growth but can also be a threat, given the large amount of resources it requires [24]. Within the ICT impact, as shown in Figure 1, software plays an important role and software sustainability was also identified amongst the key non-functional requirements of the 21st century [19].

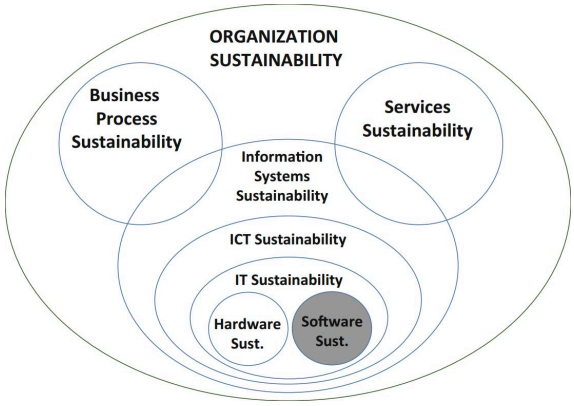

Figure 1: Different Sustainability Levels [5]

Designing "sustainable" software is not an easy task as different aspects, related to different facets of sustainability, need to be considered. In this paper, we refer to the terminology defined by [5] about Green "in" and "by" IS (Information System), IT (Information Technology). Those nested levels are depicted in Figure 1. In the scope of this paper, we are more specifically interested in:

- How the software helps in supporting the sustainability of the system in which it is being used, i.e. "green by software"
- How sustainable the software is, from the software engineering/execution/maintenance aspects, i.e. "green in software"
- How sustainable the IT infrastructure is (mostly in terms of energy efficiency), i.e "green in IT"

This paper reports on our industrial experience in building and putting in operation a shared pick-up and delivery service called Sam-Drive (or just "Sam" in short) in the suburbs of Brussels in Belgium [6, 14]. The Sam system has strong sustainability goals as mobility is a big challenge [1]. Before introducing the software, it was managed almost entirely manually. As we developed the software itself, different design decisions had to be taken by research engineers quite aware of good software development practices, including sustainability aspects. However, there was not a strong and

GREENS'18, May 27–26, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5732-6/18/05…$15.00
https://doi.org/10.1145/3194078.3194083

permanent focus on sustainability compared to other requirements, it was more implicit. This paper is also a look backward reporting about interesting achievements but also points out some areas of improvement, so others can learn from our experience.

In addition to providing the community with a new case in the same spirit as [7], we identified and partly addressed the following interesting research questions:

- RQ1 - How did we "decide" about SW sustainability objectives in a consistent way at system level ?
- RQ2 - How did we implement green by/in IT/SW in a coherent way ?
- RQ3 - How can we assess the sustainability of the resulting system ?

The structure of this paper is as follows. Sections 2, 3 and 4 respectively address the three research questions based on our case with some relevant literature. It is followed by a discussion in Section 5. Finally, some conclusions and perspectives are presented in Section 6.

2 RQ1 - SUSTAINABILITY GOAL ANALYSIS

Sustainability aspects are often classified into three interdependent dimensions: economic, social and environmental already identified in [17]. We confined ourselves to those classical dimensions although some extra dimensions can also be considered [2].

The Sam goals can be summarised and analysed using those criteria:

- *Encourage ride sharing with a degressive pricing model for people accepting to travel with others with some limited detour.* The three dimensions are covered as the car sharing reduces the environmental footprint, the user is economically rewarded. On the social side, people can travel together. Actually, the service is mainly organised based on recurrent drives which typically cover the needs to drive kids to school or to after-school activities. It also includes additional drives for older people not willing/able to take their care and for people travelling to/from the nearby airport.
- *Use low-qualified labour, often unemployed people as drivers.* Such people get the chance to find a secure job, in contrast to precarious jobs proposed by the "app-driven" new economy, like Uber [25]. This dimension is social.
- *Improve the work of the dispatching team.* The previous situation was a poor IT solution (i.e. Excel sheets) putting stress on the team and also limiting the operational and growth capabilities. So, this dimension is both economic and social.
- *Optimise the planning* to match driver's constraints and minimise time on road, in traffic jam and carbon emissions. This goal has both environmental and social dimensions.
- *Renewal of car fleet.* The new fleet uses the latest car generation with lower consumption and CO_2 emission. This is both an environmental and economic goal.

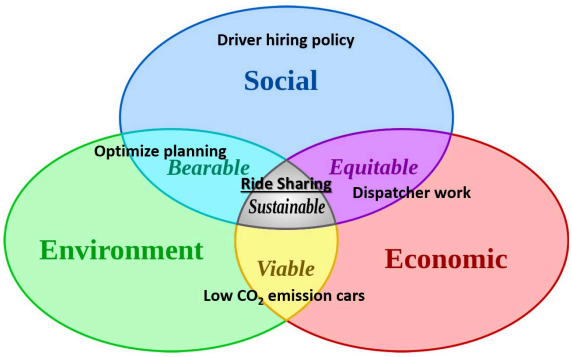

Figure 3: Sam sustainability goals represented on the three sustainability dimensions

Such goals can be modelled and decomposed using goal-oriented requirements engineering like KAOS [23], i* [26] or URN/GRL [11]. Different techniques have been proposed for carrying our sustainability analysis like KAOS [13, 21] and i* [4], also including a taxonomy. Figure 2 represents our structuring of the above goals using

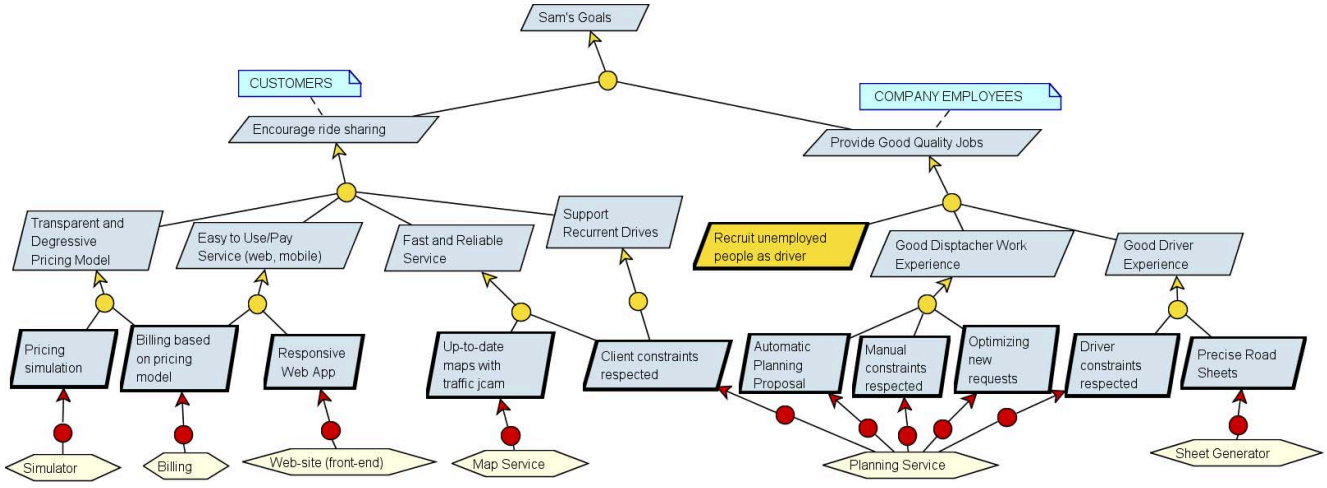

Figure 2: Goal Refinement Tree for the Sam System

KAOS. Actually, it is a classical decomposition using a stakeholder-based decomposition as first level (end-users vs employees) and does not specifically reflect the sustainability dimension. Providing such a view is however interesting because, as identified above, our goals have most of the time more than one sustainability dimension reinforcing each others and resulting in a win-win situation. To represent this, we propose to locate them on a Venn diagram reflecting the three dimensions and their different intersections. For more complex classification, there are some other interesting possibilities like a spider diagram as in [2].

3 RQ2 - ACHIEVING GREEN IN/BY SOFTWARE

In order to enforce the goals identified in the previous section, a refinement into requirements on the software and IT infrastructure, or expectations from the user interacting with the IS needs to be done. The two bottom layers of Figure 2 represent the main requirements and responsible agents, respectively. A collection of interactions modules is necessary to cover the functionalities like registering users (for a single trip or recurring trips), computing paths on the map, computing the corresponding price, either in simulation mode or for billing, and planning the drives.

Two key points detailed in this section are the optimisation technology used and the specific attention to the application architecture which enables its efficient deployment on a lightweight virtualised infrastructure.

3.1 Efficient and Effective Optimisation Engine

On the Green BY IT side, in Figure 2, the planning service is responsible of many constraints linking customers and drivers, possibly in a recurring manner (e.g. each day of the week for driving to school). Our approach for an effective engine (i.e. fulfilling system level sustainability goals) allows to cope with large problems and domain constraints. However, our main interest is not to hunt for the optimal solution but rather to improve over dispatcher jobs and help them cope with larger fleets. To support this, we compute indicators assessing total distance reduction, load on drivers, etc.

On the Green IN IT side, we also paid a lot of attention to the efficiency of the optimisation technology used. Rather than looking at engine implemented in C/C++ which compile directly to machine level, we have chosen more powerful languages supporting efficient algorithmic constructs able to provide lower complexities when exploring the search space while providing the developer with more expressive constructs to solve a complex set of constraints. The selected technology is the OscaR.CBLS engine, developed in Scala [8, 18]. It uses constraint-based local search [22] and has dedicated data structures for efficiently dealing with routing problems [12]. Although there is some overhead for small problems, given the complexity of the pick-up and delivery problem we have to deal with, our approach proved more efficient than other options like Google OR-Tools [9].

3.2 Green IN IT through a Clean Architecture

The architecture of the application is depicted in Figure 4. The dispatching user interface is relying on our OscaR solver which in turn requires to be able to access a distance matrix connecting all pairs of points that are potentially involved in a driver tour. In addition to its very large size, this matrix can also return travel time functions instead of a single number as the time might vary during the day. Very specific algorithms are available for this purpose: through "contraction hierarchies" they are able to compute paths in a few milliseconds at the expense of some storage. In addition, the need to call such a service can even be further reduced by using a cache mechanism which was implemented in our case, again at the expense of more storage space.

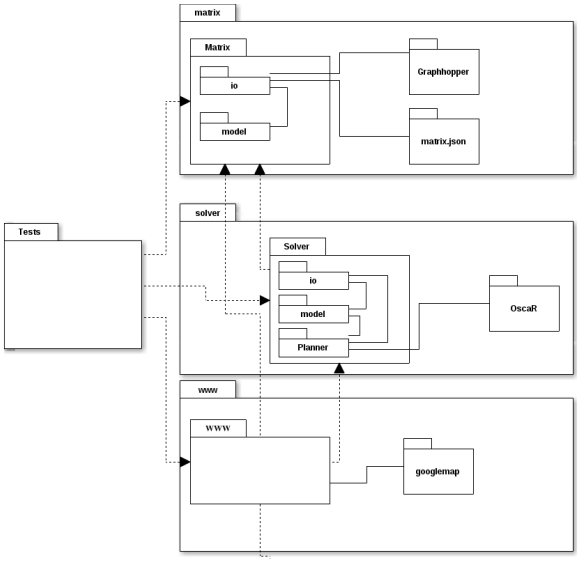

Figure 4: Container-based Architecture of the Sam System

At IT level, each service of the resulting architecture was deployed as a container based on the Docker technology [10]. Our architecture is quite easy to match with container at the top level of the packages shown in Figure 4. From a resource usage point of view, it is much more lightweight than using a set of virtual machines and it is also much easier to reproduce and to maintain in different environments (like dev, test and production) [16].

4 RQ3 - ASSESSING SUSTAINABILITY LEVEL

As pointed hereabove, early in the Sam development, we felt the necessity to define Key Performance Indicators related to our business goals (including sustainability aspects). This includes global indicators relating to customers, requests, distance, costs but also more precisely related to optimisation operations (gain on distance, time, costs). At technical level, a similar need was raised, especially for the optimisation part (e.g. the case of over constrained problems) and the matrix part (matrix cache hit rate and performance of computation of new values).

Different dashboards were implemented based on an Open Source monitoring system called Promotheus [20]. This component is easy to configure thanks to the ease to add connectors, many of them being available off the shelf. It can collect data in an efficient way and also provides good dashboards as illustrated in Figure 5 which relate to the technical indicators.

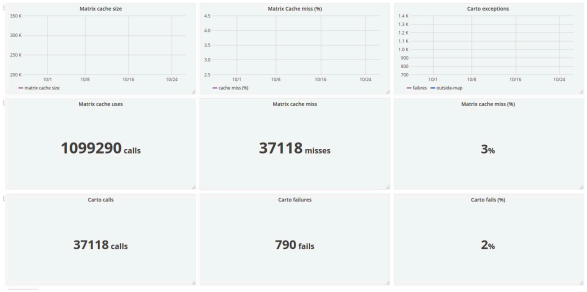

Figure 5: Monitoring system

Although the Sam system is only deployed since a few months, the monitoring has already shown its limitations of our current maps compared to real travel times. Some inefficiencies have also been identified in our cache which is storing too many points.

5 DISCUSSION

As this work is anchored in a specific case, the previous analysis is rather specific. However, we believe a number of interesting topics have a wider impact such as:

- The importance of the requirements analysis, especially to identify how well sustainability goals interact with goals of all stakeholders. Our case has a very positive configuration with a lot of synergies but, in general, one should also expect conflicts. Dealing with them is important to avoid discarding sustainability because of a lower priority ranking.
- Refining the goal analysis can also point out which software components are supporting sustainability and thus help in better focusing on the Green by software dimension.
- At the IT level, achieving a good Green IN IT vision definitely requires to have a strong architecture. At the software level, our main concern was on the algorithmic side but this issue is more related to the computationally intensive nature of our application. Data intensive application will probably raise the need of adequate data structures. A number of guidelines and patterns for green coding are available, e.g. [3, 15].
- Defining and monitoring relevant indicators as soon as possible is definitely worth the overhead of such a component.

6 CONCLUSIONS AND NEXT STEPS

In this paper, we presented a case study focusing on sustainability aspects, both at system and software levels. We addressed it based on three research questions coping with three key steps: requirements, implementation and exploitation. We tried to make use of a number of published techniques or to analyse our work in the light of such techniques as the process was probably not (and never is) perfect. Our hope is that this feedback can be useful to the community working on sustainable software based systems, both for the case it provides and the partial analysis we carried out. The Sam system is fully available on-line (in French) at https://www.sam-drive.be. We expect to learn more things as the project is still young and evolving. Our monitoring system will enable to assess how well the software fits the sustainability goals and to confirm the architecture is able to cope with the anticipated evolution in size and type of services. We also plan to analyse our development process from the sustainability point of view.

ACKNOWLEDGMENTS

This work was partly funded by the SAMOBI CWALITY Project of the Walloon Region (grant nr. 1610019). The authors would like to thank Sam-Drive for sharing information about his project organisation and his business model, especially in relation with the sustainability dimensions.

REFERENCES

[1] David Banister. 2008. The sustainable mobility paradigm. *Transport policy* 15, 2 (2008), 73–80.
[2] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S. M. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters. 2016. Requirements: The Key to Sustainability. *IEEE Software* 33, 1 (Jan 2016), 56–65. DOI: http://dx.doi.org/10.1109/MS.2015.158
[3] F. Bordage. 2015. *Eco-conception web / Les 115 bonnes pratiques: Doper son site et réduire son empreinte écologique.* Eyrolles. https://books.google.be/books?id=B2WJCgAAQBAJ
[4] J. Cabot, S. Easterbrook, J. Horkoff, L. Lessard, S. Liaskos, and J. N. Mazon. 2009. Integrating sustainability in decision-making processes: A modelling strategy. In *31st Int. Conf. on Software Engineering - Companion Volume.* 207–210.
[5] C. Calero and M. Piattini. 2015. *Green in Software Engineering.* Springer International Publishing.
[6] CETIC and Sam-Drive. 2016. SAMOBI - The Next Generation Shared Taxi. https://www.cetic.be/SAMOBI-3055. (2016).
[7] R. Chitchyan, S. Betz, L. Duboc, B. Penzenstadler, C. Ponsard, and C. Venters. 2015. Evidencing Sustainability Design through Examples. In *Proc. of the 4th International Workshop on Requirements Engineering for Sustainable Systems, RE4SuSy), Ottawa, Canada, August 24.* 45–54.
[8] Renaud De Landtsheer and Christophe Ponsard. 2013. OscaR.cbls : an open source framework for constraint-based local search. In *Proceedings of ORBEL'27.*
[9] Google. 2012. OR Tools: Operations Research Tools developed at Google. https://code.google.com/p/or-tools/. (2012).
[10] Solomon Hykes. 2013. Docker - Build, Ship, and Run Any App, Anywhere. https://www.docker.com/. (2013).
[11] International Telecommunication Union. 2012. Recommendation Z.151 (10/12), User Requirements Notation (URN) âĂŞ Language Def. (2012).
[12] R. De Landtsheer, G. Ospina, Y. Guyot, F. Germeau, and C. Ponsard. 2017. Supporting Efficient Global Moves on Sequences in Constraint-based Local Search Engines. In *ICORES.* SciTePress, 171–180.
[13] M. Mahaux, P. Heymans, and G. Saval. 2011. Discovering Sustainability Requirements: An Experience Report. In *REFSQ (Lecture Notes in Computer Science)*, Vol. 6606. Springer, 19–33.
[14] Michel Reynders. 2012. SAM S. https://www.sam-drive.be. (2012).
[15] S. Murugesan and G.R. Gangadharan. 2012. *Harnessing Green IT: Principles and Practices.* Wiley. https://books.google.be/books?id=xAtJK2L7O5UC
[16] R Nagler and others. 2015. Sustainability and Reproducibility via Containerized Computing. *Synchrotron Radiation* 4769 (2015), 145.
[17] United Nations. 1987. World Commission on Environment and Development: Our Common Future. Oxford Univ. Press http://www.un-documents.net/our-common-future.pdf. (1987).
[18] OscaR. 2012. OscaR: Scala in OR. https://bitbucket.org/oscarlib/oscar. (2012).
[19] B. Penzenstadler, A. Raturi, D. Richardson, and B. Tomlinson. 2014. Safety, Security, Now Sustainability: The Nonfunctional Requirement for the 21st Century. *IEEE Software* 31, 3 (May-June 2014), 40–47.
[20] Promotheus. 2018. From metrics to insight. https://prometheus.io. (2018).
[21] David Stefan, Emmanuel Letier, Mark Barrett, and Mark Stella-Sawicki. 2011. Goal-oriented system modelling for managing environmental sustainability. In *3rd Int. Workshop on Software Research and Climate Change. Lancaster, UK.*
[22] Pascal Van Hentenryck and Laurent Michel. 2009. *Constraint-based Local Search.* MIT Press.
[23] Axel van Lamsweerde. 2009. *Requirements Engineering - From System Goals to UML Models to Software Specifications.* Wiley.
[24] Graham Vickery. 2012. Smarter and Greener? Information Technology and the Environment: Positive or negative impacts? International Institute for Sustainable Development. (2012).
[25] Rachelle Younglai. 2015. Rise of sharing services Uber, Airbnb points to a precarious labour climate. The Globe and Mail http://bit.do/precarious-sharing-economy. (2015).
[26] Erick S. K. Yu and John Mylopoulos. 1997. Enterprise Modelling for Business Redesign: The I* Framework. *SIGGROUP Bull.* 18, 1 (April 1997), 59–63.