# An Ecological View on Software Reuse

Laura-Diana RADU
Department of Research, Faculty of Economics and Business Administration
Alexandru Ioan Cuza University of Iasi
glaura@uaic.ro

*The increase of consumption is an important motivation for the reuse of either physical or virtual products. As the software market has risen, software reuse has become a practice with favourable effects for software development companies and their clients. The most important benefits are increased productivity, reduced costs, better and easier maintenance, decreased development lead times and the improved quality of software products. Successful reuse depends on several technical and non-technical factors. The ecological impact of software is an important non-technical factor of software reuse that needs to be analysed in the context of the rapid evolution of optimization techniques. The main goal of this study is to identify ecological perspectives on software reuse. These will complete the framework of software reuse together with other technical factors, such as compatibility, and non-technical factors, such as economic and ethical implications.*
***Keywords:*** *Software reuse, Green software, Ecological impact, Domain engineering*

## 1 Introduction

Reuse is a component of the 3Rs (Reduce, Reuse and Recycle) promoted by environmental organizations around the world, such as the United States' Environmental Protection Agency (US EPA 2014) and the United Kingdom's Waste and Resources Action Programme (WRAP); it is also part of China's Circular Economy [1]. Human consumption has increased significantly both in terms of products as well as services. Most of products and services are designed, manufactured and delivered by using hardware and software products or they are hardware and software products. The explosion of these technologies brings new challenges for humanity [2] [3]. They influence climate change, biodiversity loss, and mineral scarcity, but their reuse could reduce waste and new production. In the case of software, in order to efficiently exploit the opportunities its reuse in multiple projects, it is very important to explicitly define this concept and its place in life cycle phases [4].

Software reuse is the use of previously results from all phases of their life cycle: product line requirements, functions, architecture, design patterns and codes [5] [6] [7] [8]. These practices should be anticipated from the conception and initiation phases of the project. Morisio et al. defined software reuse as "the systematic practice of developing software from a stock of building blocks, so that similarities in requirements and/or architecture between applications can be exploited to achieve substantial benefits in productivity, quality and business performance". Using existing components for building new applications reduces the effort of development [9]. This includes not only the development of new software, but also the extraction of reusable parts of existing applications [10]. The existence of the same requirements in more software project allows these practices. The availability of large numbers of open source projects is also a valuable source of reusable assets [11] and offers new possibilities for software reuse. The expected results are: increased productivity, reduced costs, better and easier maintenance, decreased development lead times and improved quality of software products. Software reuse is influenced by a wide variety of technical and non-technical factors. Projects evolve in order to meet new requirements or for maintenance reasons. In these circumstances, some changes will influence other projects that reuse the same components, especially in the case of library upgrades. According to Constantinou and

DOI: 10.12948/issn14531305/22.3.2018.07

Stamelos [11], software reuse is a recursive process that supposes the selection of reusable assets, upgrading libraries to the newest versions and fixing bugs. A feasibility and suitability analysis should be performed to ensure compatibility with the new application and compliance with customer requirements. Software reuse has important benefits for both companies and society. It would reduce negative environmental effects because companies would use less hardware and software resources in comparison with developing software from scratch. In order to ensure this, reusable components must satisfy quality criteria specifically for sustainable software.

The rest of this paper is organized as follows. Section 2 reviews the related work on software reuse; section 3 presents the recent approaches regarding green and sustainable software and its contribution to environmental protection; section 4 describes ecological dimension of software reuse. Finally, section 5 concludes the paper and provides future research directions.

## 2 Related work

Various studies on reusing software are presented in the literature [7] [10] [11] [12] [13] [14]. Additionally, software quality has been receiving a lot of attention, especially since the introduction of the concept of domain engineering, a component of software engineering [12]. The other component is application engineering. According to Harsu, the domain engineering has "to provide the reusable core assets that are exploited during application engineering when assembling or customizing individual applications" [13]. The same author calls domain engineering "engineering for reuse" and application engineering "engineering with reuse". The process is divided in three phases presented in Figure 1.

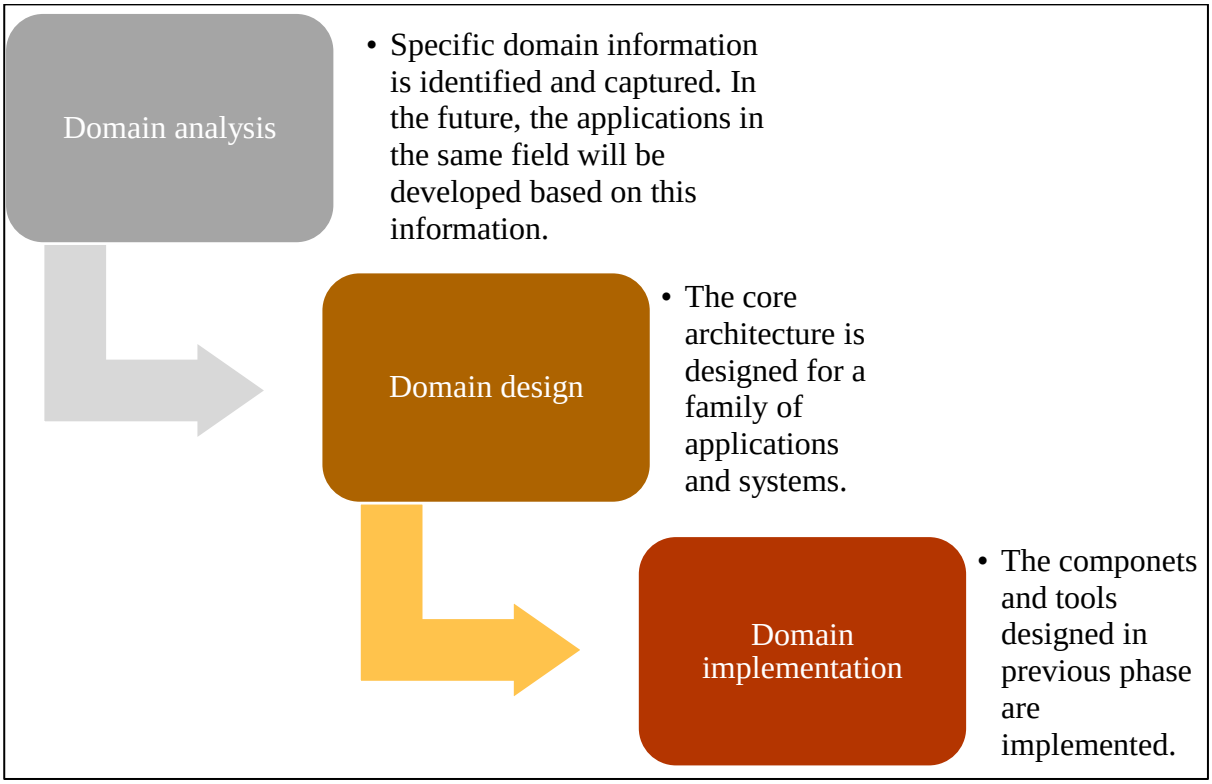


**Fig. 1.** Domain engineering phases (adapted from [13], [14])

The evolution of software engineering has increased concerns regarding the development of sustainable software. Creating reusable components could positively contribute to these concerns. However, reusing software takes different forms. Jie et al. [12] identified the following two categories: product reuse and process reuse, depending on the reused

DOI: 10.12948/issn14531305/22.3.2018.07

item. Product reuse is defined as the reuse of previously built software components by integrating them into the new system, while process reuse is defined as the development of an efficient software process and repository that produces base knowledge. Another classification, proposed by the same authors, divides reuse into black-box reuse and white-box reuse [10] [12]. In black-box reuse mode, the components are reused directly without any changes. In white-box reuse mode, the components must be modified according to new requirements. The components could be libraries, software specifications, software design, interfaces, prototypes, planning, documentation, frameworks, test cases or templates. Some components are interrelated. For example, the reuse of source components implicitly involves the reuse of analysis and design.

Xin and Yang identify four types of software reuse processes: reuse without modifying, reuse with modifying parameters, reuse with modifying code, design or critical level and new software without reusing [7]. Table 1 presents the definitions of the first three types of software reuse, since the fourth type is not reuse, but the development of new software from scratch.

**Table 1.** Types of software reuse [7]

| Name | Description | Reuse process decision | Reuse process management |
|---|---|---|---|
| Reuse without modifying | Software previously developed for other system(s) that could be reused without any modifying. | Reuse feasibility analysis<br>Reuse type and process choice<br>Software technical status check | Software system test<br>Software confirming review<br>Software maintenance |
| Reuse with parameters modifying | Modifying parameters only and reusing software in the same form. | | Software modifying influence analysis<br>Software regression test<br>Software configuration test review<br>Software system test<br>Software confirming review<br>Software maintenance |
| Reuse with code modifying | Modifying code, design or raising critical level and reuse software in this form. | | Software requirement modifying review<br>Software design modifying<br>Software implementation modifying<br>Software regression test<br>Software configuration test review<br>Software system test<br>Software confirming review<br>Software maintenance |

DOI: 10.12948/issn14531305/22.3.2018.07

Software reuse has advantages and challenges. Dabhade et al. [14] identified the following major benefits: increasing reliability and effectiveness, accelerating the development stage, increasing the productivity of software, minimizing operational and maintenance costs, enhancing system interoperability, developing software with less manpower, producing standardized software, delivering good quality software and obtaining competitive advantages.

The most important challenge is technical compatibility with the system in which the components are integrated or reused. Technical compatibility is influenced by the level of reuse [15]: at a low level (reuse of design patterns), at a medium level (software reuse) or at a high level (when product line techniques are used to identify reusable components). It is also influenced by the architectural stability and evolution of the software. In order to measure these features, Constantinou and Stamelos have proposed six metrics for two project types: those intended to serve as reusable libraries and those that were not designed for reuse [11]. Other technical challenges include increased maintenance costs when source code is not available, the lack of tool support, the expense of creating and maintaining reusable components in libraries, knowledge of requirements and reusable components, and legal issues [10] [14].

Technical factors are crucial for software reuse, but non-technical ones are also very important. There are at least three non-technical implications: economic, ethical and ecological. Economic and ethical implications are more evident and subject to study because they influence business decisions. The problem of environmental protection in the software industry is relatively new, even though information and communication technology (ICT) researchers and practitioners have been interested in environmental protection since the end of the last decade [16]. These concerns should increase proportionally with the level of infiltration of ICT in economic and social life. They need to be analysed in accordance with the definition and characteristics of green software [17] [18] [19].

In order to manage software reuse efficiently, the next section identifies and analyses ecological elements that must be considered in software reuse processes. These will complete the framework of software reuse.

## 3 Green and sustainable software

The interest in environmental protection through ICT has received particular attention over the last ten years. In 2008, the concept of green ICT was adopted in literature and practice to reflect these concerns. Although it has evolved and has been divided into specific areas, the most comprehensive definition remains the one given by Murugesan in 2008. According to this author, green ICT is "the study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems—such as monitors, printers, storage devices, and networking and communications systems—efficiently and effectively with minimal or no impact on the environment. Green IT also strives to achieve economic viability and improved system performance and use, while abiding by our social and ethical responsibilities" [20]. According to Vickery [21], the interaction between ICTs and the natural environment can be categorized into three levels:

- *Direct impacts* – positive and negative effects due directly to ICT goods and services and related processes;
- *Enabling impacts* – ICT use that reduces environmental impacts across economic and social activities outside of the ICT field; and
- *Systemic impacts* – individual and collective behavioural change.

Most theoretical and practical initiatives have focused on direct impact, with a particular focus on positive and negative effects of the hardware, as their influences on the environment are more evident. However, in the past few years, new initiatives have emerged in research and practice to promote green and sustainable software. This software dictates how energy is consumed by any

DOI: 10.12948/issn14531305/22.3.2018.07

programmable device, although the ultimate responsibility is attributed to physical equipment [22]. Optimization techniques aim to develop software to improve resource efficiency, including energy sources.

The positive effects of software on the environment can be direct, by reducing negative ICT impacts, or indirect, by using software to support other business initiatives in reducing negative environmental impacts. In order to generate these influences, software must be green and sustainable. According to Dick et al. [23], green and sustainable software is "software, whose direct and indirect negative impacts on economy, society, human beings, and environment that result from development, deployment, and usage of the software are minimal and/or which has a positive effect on sustainable development". Sustainability is considered from three dimensions: social, economic and environmental [24]. The last dimension corresponds to green software and can be divided according to its direct or indirect influences on the environment in "green by software" and "green in software". Table 2 presents these concepts and their impact on the environment.

**Table 2.** Green software [17] [25] [26] [27]

| Concept | Definition | Environmental impact |
|---|---|---|
| Green by software, Green 1.0 or software with indirect impact on environment | Software that provides efficient resources management for other applications and software dedicated to environmental protection and monitoring. In this case, software is the tool used to support sustainability goals. | The software influences green behaviour:<br>- Energy-aware software;<br>- Software supporting sustainable processes;<br>- Sustainable software development. |
| Green in software, Green 2.0 or software with direct impact on environment | Minimizing negative influences of software development on the environment and supporting the complete life cycle of sustainable software system engineering. | - Energy efficient software |

The concept of environmental protection in the software development life cycle has two main phases: software development and software execution [28] [29]. In both phases, the main form of environmental protection is energy saving. This will have other positive influences, such as the reduction of $CO_2$ emissions. In software development, energy can be saved by using more responsible ICT equipment, identifying existing modules or components and reusing them in new software, avoiding recurring work [28], switching equipment off or putting it in standby mode when not in use and employing refactoring strategies and self-adaptation techniques [22]. In the design phase, the following methods exist to increase energy efficiency: efficient database queries (-25% energy consumption), optimized data management (+70% performance), flexible computation offload (-40% power consumption), smart use of web resources (-8.5% energy consumption), website content delivery (-45% energy consumption) and software refactoring (-50% energy consumption) [30]. In software execution, energy can be saved by updating and reusing applications instead of developing new ones, selecting the most energy efficient software on the market and monitoring and optimizing energy consumption for all programmable devices. In both cases, software reuse could improve the process of software development from both a technical and non-technical perspective. Environmental protection is a non-technical motivator with long-term implications for all of society.

DOI: 10.12948/issn14531305/22.3.2018.07

## 4 Ecological dimensions of software reuse

Software reuse is not new, but it still fails frequently in commercial companies, most often for human and organizational reasons and sometimes for technical reasons [31]. Software engineers must find their own benefits of reusing software. Their personal values on environmental protection, together with organization strategies in the same field, could be sufficient incentives for adopting these practices. However, previous research in software reuse has focused most often on its economic and technical benefits; environmental implications have not been studied.

Reusability is a criteria for sustainable software [32]. Before deciding on the software components that should be reused at various stage of the software development life cycle, a feasibility study on their environmental impact is required. Even if reuse could be technically and economically favourable, from the viewpoint of influences on the environment the reuse of some software components could be less advantageous, due to the rapid evolution of optimization techniques. According to Beghoura et al. [18], green constraints fit into the category of non-functional requirements, so they need to be explicitly defined in the requirements specification phase and supported by software design and implementation. Based on their opinion, the opportunity to reuse software must be analysed starting with initial requirements specifications. These could include specifications on reducing energy consumption or using more environmentally friendly hardware or tools, if the client or the company developing the application has and applies strategies for environmental protection. These specifications could limit software reuse or could allow its reuse with modifying parameters or with modifying code and design. These practices consume more resources in the development stage, but are more eco-effective post-implementation. In addition, the new components that consume fewer resources will be reused later in other systems. Johann et al. [33] proposed the following activities in order to enhance an environmentally favourable impact over the whole life cycle: sustainability reviews and previews for every phase, process assessment and the sustainability retrospective.

Before reusing a component, it must be evaluated according to environmental quality criteria. A presentation of the environmental impact, along with other aspects of software sustainability, need to be described in a sustainability journal [34]. The environmental quality criteria must follow the same stages and activities as quality software evaluation: acceptance criteria, an environmental project quality plan, environmental quality criteria and the environmental quality assurance role that could be achieved by auditors [26]. One solution is the labelling of software components and software products according to their influence on the environment. At present, there are some initiatives to label green software [35], but a standardized eco-label is still missing [36]. Previous studies show that neither programmers nor users request energy efficiency for their product and this concern is missing during maintenance [37] [38]. In order to encourage the reuse of only green software components, these should be evaluated according to environmental criteria. Kern et al. proposed the following criteria for software evaluation: efficiency, feasibility and perdurability [35].

- *Efficiency* is defined as software behaviour when it can save resources and avoid waste [39];
- *Feasibility* refers to environmental impact – resource-oriented feasibility (energy type, energy consumption, energy management options and carbon footprint) and social impact – well-being oriented feasibility (sustainability support, accessibility and usability) [39] [35]; and
- *Perdurability* is "the degree to which a software product can be modified, adapted and reused in order to perform specified functions under specified conditions for a long period of time" [40].

Perdurability refers directly to software reuse. In Standards ISO: 25010, Calero et al. [40] identified three characteristics related to this

DOI: 10.12948/issn14531305/22.3.2018.07

criteria: (1) *reusability* – the degree to which an asset can be used in more than one system, or in building other assets; (2) *adaptability* – the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments; and (3) *modifiability* – the degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality [41]. In Figure 2, perdurability aspects are presented separately by the development and usage phase.

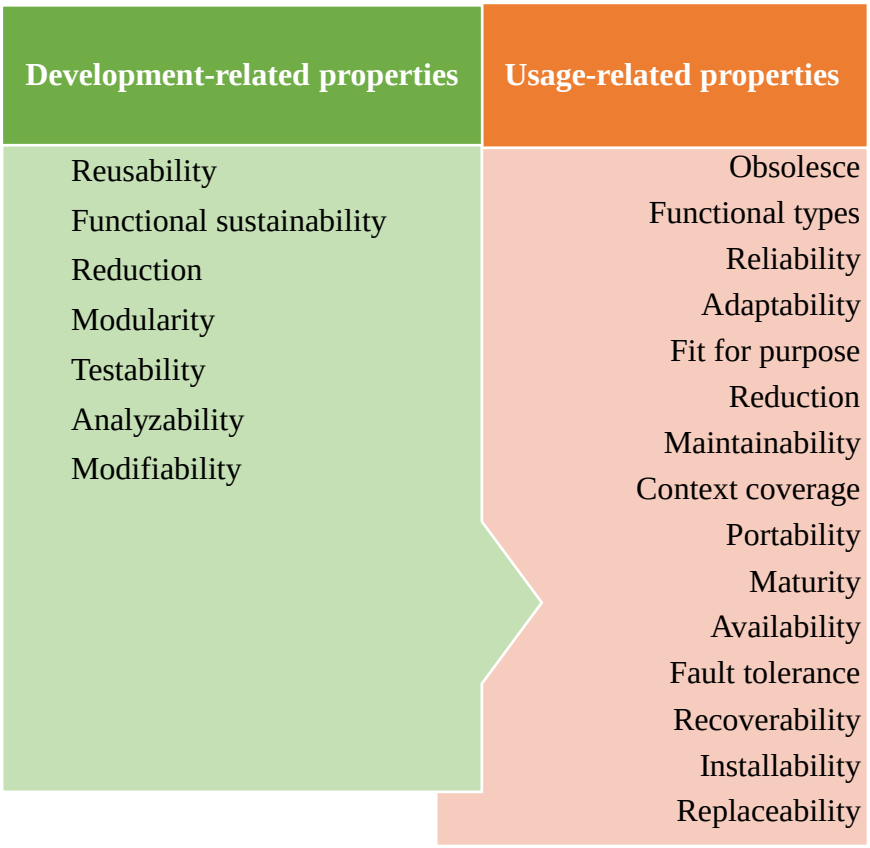


**Fig. 1.** Criteria for sustainable software products mapping to perdurability [35]

The possible environmental impact is not the only challenge of software reuse for decreasing negative ecological influences. Software development teams need to anticipate future environmental regulations, best practices, norms and market trends [26]. But awareness regarding environmental protection in the software industry starts with education. In general, green computing elements are missing in the training of programmers. According to Pang et al. [37], the influence of the algorithms used on energy consumption should be integrated into computer programming courses in university and school. These concerns should also be included in software companies' strategies regarding environmental protection and in specifications of products and services [42].

As a result of software reuse, the quality of the applications and systems will be improved. Efforts should be aimed at replacing or modifying components with high energy consumption or with other negative environmental effects. A classification of the components that could be reused according to ecological impact would be useful. Even in the absence of software component labelling standards, companies could have some internal criteria for environmental protection for these components. Figure 2 demonstrates how environmental concerns should be included in the software reuse process.

DOI: 10.12948/issn14531305/22.3.2018.07

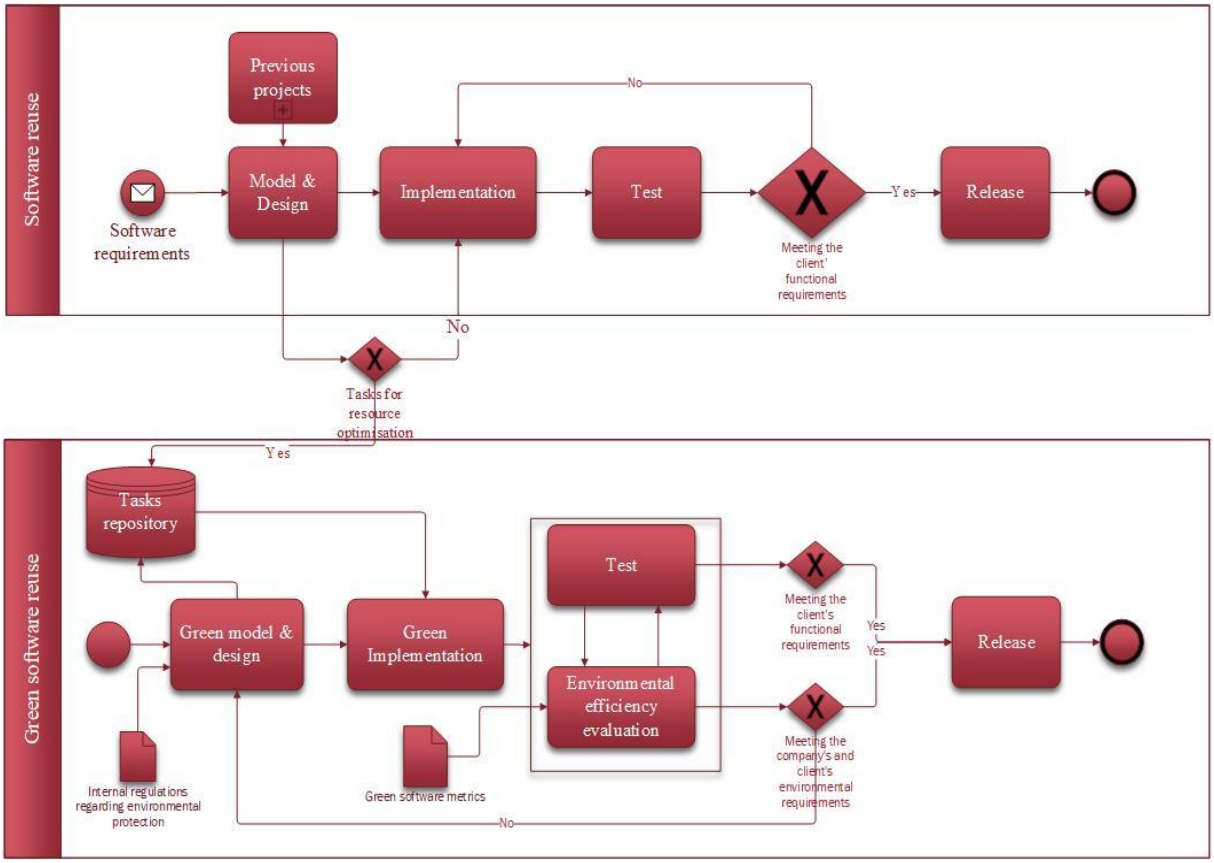


**Fig. 3.** Green software reuse life cycle

Starting from existing software components and based on the premise that the client, the developer or both are interested in developing environment-friendly systems, resource optimization specifications will be included in the task repository. These specifications are requirements that will need to be developed in a similar manner to functional requirements and will be tested in accordance with ICT companies' and clients' strategies regarding environmental protection.

Minimizing negative environmental impacts should be an indicator of software quality. To validate a favourable influence on the environment or to reduce negative environmental effects, the following metrics could be used: energy consumption, performance (response time), computing resource utilization (hard disk, storage, memory and I/O operations) and pollution or $CO_2$ generated by software usage or development [43]. In practice, it is recommended to use hybrid metrics to objectively and realistically assess the effects of developing new components versus software reuse, as well as the correlation between them. For example, in most cases, energy consumption increases the volume of $CO_2$ in the context of significant energy use from non-green sources.

## 5 Conclusions

Software reuse has a wide variety of benefits for software companies and their clients. This practice saves costs and time, increases the quality of the application and improves maintenance. In order to increase the efficiency of ICT, it is recommended to reuse only the software that positively influences the environment or has a minimum negative impact. Reusing a component is not a guarantee for environmental benefits. This paper presents some ecological aspects of software reuse and the possibility of their integration into the software development life cycle. The integration of environmental concerns in software development is not new, but it has not been studied in the case of

DOI: 10.12948/issn14531305/22.3.2018.07

software reuse. In the medium and long term it will bring not only ecological benefits, but also financial and technical benefits. In future work, the ecological dimension of software reuse will be used to develop a framework for software reuse processes.

**References**

[1] D. R. Cooper and T. G. Gutowski, "The Environmental Impacts of Reuse. A Review," Journal of Industrial Ecology, vol. 21, no. 1, pp. 38-56, 2017.

[2] S. Necula, V. Păvăloaia, C. Strîmbei and O. Dospinescu, "Enhancement of E-Commerce Websites with Semantic Web Technologies," Sustainability, vol. 10, no. 6, pp. 1-15, 2018.

[3] D. Popescul and M. Georgescu, "Social Networks Security in Universities: Challenges and Solutions," Annals of the Alexandru Ioan Cuza University-Economics, vol. 62, no. s1, pp. 53-63, 2015.

[4] Y. Verma and R. Nandakumar, "Development of software asset management system to facilitate software reuse," in Proc. of. , International Conference on Software Engineering and Mobile Application Modelling and Development (ICSEMA 2012), Chennai, 2012.

[5] Y. Kim and E. A. Stohr, "Software reuse: survey and research directions," J. Manag. Inform. Syst., vol. 14, no. 4, pp. 113-147, 1998.

[6] C. W. Krueger, "Software reuse," ACM Computing Surveys (CSUR), vol. 24, no. 2, pp. 131-183, 1992.

[7] T. Xin and L. Yang, "A framework of software reusing engineering management," in Proc. 15th International Conference on Software Engineering Research, Management and Applications (SERA), London, 2017.

[8] M. Irshad, K. Petersen and S. Poulding, "A systematic literature review of software requirements reuse approaches," Information and Software Technology, vol. 93, pp. 223-245, 2018.

[9] V. Garcia, E. de Almeida and S. de Lemos Meira, "A Reference Model for Software Reuse Adoption in Companies," in Proc. of the Doctoral Symposium at the 11th International Conference on Software Reuse (ICSR'2009), Falls Church, 2009.

[10] J. Sametinger, Software engineering with reusable components, Berlin: Springer-Verlag Berlin Heidelberg, 1997.

[11] E. Constantinou and I. Stamelos, "Architectural Stability and Evolution Measurement for Software Reuse," in Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, 2015.

[12] W. Jie, T. Pei, S. Wen-qing and X. Yan, "The Generation of Software Reliability Test Cases Based on Software Reuse," in Proc. of 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, 2017.

[13] M. Harsu, "A survey on domain engineering," Institute of Software Systems, Tampere, 2002.

[14] M. Dabhade, S. Suryawanshi and R. Manjula, "A Systematic Review of Software Reuse using Domain Engineering Paradigms," in Proc. of International Conference on Green Engineering and Technologies (IC-GET), Coimbatore, 2016.

[15] Y. Xu, J. Ramanathan, R. Ramnath, N. Singh and S. Deshpande, "Reuse by placement: a paradigm for cross-domain software reuse with high level of granularity," in Top Productivity through Software Reuse. ICSR 2011. Lecture Notes in Computer Science, vol. 6727, S. K., Ed., Berlin, Springer, 2011, p. 69–77.

[16] S. Mingay, "Green IT: The New Industry Shock Wave," 2007. [Online]. Available: http://www.ictliteracy.info/rf.pdf/Gartner_on_Green_IT.pdf. [Accessed 20 April 2018].

[17] C. Calero and M. Piattini, "Introduction to green in software engineering," in Green in Software Engineering, C. Calero and M. Piattini, Eds., Cham, Springer, 2015, pp. 3-27.

[18] M. A. Beghoura, A. Boubetra and A. Boukerram, "Green software requirements and measurement: random decision forests-based software energy consump-

DOI: 10.12948/issn14531305/22.3.2018.07

tion profiling," Requirements Engineering, vol. 22, no. 1, pp. 27-40, 2015..

[19] B. Schmidt, "Sustainability Knowledge about Software Parts in Software Engineering Processes," in Proc. of 4th International Conference on ICT for Sustainability, Amsterdam, 2016.

[20] S. Murugesan, "Harnessing Green IT: Principles and Practices," IEEE IT Professional, vol. 10, no. 1, p. 24–33, 2008.

[21] G. Vickery, "Smarter and Greener? Information Technology and the Environment: Positive or negative impacts?," October 2012. [Online]. Available: https://www.iisd.org/pdf/2012/com_icts_vickery.pdf. [Accessed 10 June 2018].

[22] L. Ardito, G. Procaccianti, M. Torchiano and A. Vetro, "Understanding green software development: A conceptual framework," IT professional, vol. 17, no. 1, pp. 44-50, 2015.

[23] M. Dick, S. Naumann and N. Kuhn, "A model and selected instances of green and sustainable software," in What Kind of Information Society? Governance, Virtuality, Surveillance, Sustainability, Resilience, Berlin, Springer, 2010, pp. 248-259.

[24] G. Brundtland, "Report of the World Commission on Environment and Development: our common future," Oxford University Press, Oxford, 1987.

[25] P. Lago, "Software dedicated exclusively to environmental protection and monitoring," 15 November 2015. [Online]. Available: https://www.slideshare.net/patricia_lago/towards-software-sustainability-assessment. [Accessed 10 April 2018].

[26] N. Bachour and L. Chasteen, "Optimizing the value of green it projects within organizations," in Proc. of Green Technologies Conference, Grapevine, 2010.

[27] C. Calero and M. Piattini, "Puzzling out Software Sustainability," Sustainable Computing: Informatics and Systems, vol. 16, pp. 117-124, 2017.

[28] K. Sierszecki, T. Mikkonen, M. Steffens, T. Fogdal and J. Savolainen, "Green software: Greening what and how much?," IEEE software, vol. 31, no. 3, pp. 64-68, 2014.

[29] M. Dick, J. Drangmeister, E. Kern and S. Naumann, "Green software engineering with agile methods," in Proc. of 2nd International Workshop on Green and Sustainable Software (GREENS), San Francisco, 2013.

[30] P. Lago, "Software with a sustainable intent," 2016. [Online]. Available: http://www.informatics-europe.org/images/ECSS/ECSS2016/Slides/ECSS2016_Lago.pdf. [Accessed 20 May 2018].

[31] M. Sojer and J. Henkel, "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments," Journal of the Association for Information Systems, vol. 11, no. 12, pp. 868-901, 2010.

[32] E. Kern, L. Hilty, A. Guldner, Y. Maksimov, A. Filler, J. Gröger and S. Naumann, "Sustainable software products—Towards assessment criteria for resource and energy efficiency," Future Generation Computer Systems, vol. 86, pp. 199-210, 2018.

[33] T. Johann, M. Dick, E. Kern and S. Naumann, "Sustainable development, sustainable software, and sustainable software engineering: an integrated approach," in International Symposium on Humanities, Science & Engineering Research (SHUSER), Kuala Lumpur, 2011.

[34] S. Naumann, M. Dick, E. Kern and T. Johann, "The greensoft model: A reference model for green and sustainable software and its engineering," Sustainable Computing: Informatics and Systems, vol. 1, no. 4, pp. 294-304, 2011.

[35] E. Kern, M. Dick, S. Naumann and A. Filler, "Labelling sustainable software products and websites: ideas, approaches, and challenges," in Proceedings of EnviroInfo and ICT for Sustainability, Copenhagen, 2015.

[36] E. Kern, "Green Computing, Green Software, and Its Characteristics: Awareness, Rating, Challenges," in From Science to Society, Cham, Springer, 2018,

DOI: 10.12948/issn14531305/22.3.2018.07

pp. 263-273.
[37] C. Pang, A. Hindle, B. Adams and A. Hassan, "What do programmers know about software energy consumption?," IEEE Software, vol. 33, no. 3, pp. 83-89, 2016.
[38] I. Manotas, C. Bird, R. Zhang, S. D., C. Jaspan, C. Sadowski, L. Pollock and J. Clause, "An empirical study of practitioners' perspectives on green software engineering," in Proc. of the 38th international conference on software engineering, 237–248, 2016.
[39] J. Taina, "Good, bad, and beautiful software - In Search of Green Software Quality Factors," CEPIS UPGRADE, vol. 2011, no. 4, pp. 22-27, 2011.
[40] C. Calero, M. Moraga and M. Bertoa, "Towards a Software Product Sustainability Model," Denver, 2013.
[41] ISO/IEC, "25010, Systems and software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Software product quality and system quality in use models," International Organization for Standardization, Geneva, 2010.
[42] C. Atkinson, T. Schulze and S. Klingert, "Facilitating greener it through green specifications," IEEE software, vol. 31, no. 3, pp. 56-63, 2014.
[43] P. Bozzelli, Q. Gu and P. Lago, "A systematic literature review on green software metrics," 2013. [Online]. Available: http://www.sis.uta.fi/~pt/TIEA5_Thesis_Course/Session_10_2013_02_18/SLR_GreenMetrics.pdf. [Accessed 21 May 2018].

**Laura-Diana V. Radu** (b. September 26, 1978) received his BSc in Accounting and Information Systems (2001), M.Sc. in Business Information Systems (2003), and PhD in Accounting (2006) from "Alexandru Ioan Cuza" University of Iasi. Since 2009, she has been a researcher with the Research Department, Faculty of Economics and Business Administration, Alexandru Ioan Cuza University of Iasi. She is the author of one book, 10 chapters and more than 66 articles. Her research interests include accounting information systems, green information and communication technology, green information systems, smart cities and agile project management.

DOI: 10.12948/issn14531305/22.3.2018.07