DE GRUYTER OPEN

STUDIES IN LOGIC, GRAMMAR AND RHETORIC 51 (64) 2017

DOI: 10.1515/slgr-2017-0032

Simplified Graphical Domain-Specific Languages as Communication Tools in the Process of Developing Mobile Systems for Reporting Life-Threatening Situations – the Perspective of Technical Persons

Kamil Żyła[1]

[1] Institute of Computer Science, Lublin University of Technology, Poland

Abstract. Reporting systems based on mobile technologies and feedback from regular citizens are becoming increasingly popular, especially as far as protection of environmental and cultural heritage is concerned. Reporting life-threatening situations, such as sudden natural disasters or traffic accidents, belongs to the same class of problems and could be aided by IT systems of a similar architecture. Designing and developing systems for reporting life-threatening situations is not a trivial task, requiring close cooperation between software developers and experts in different domains, who could possibly find industrially recognized languages and notations difficult. Thus, the question is whether using simplified graphical domain-specific languages (SGDSLs) could help in creating a common communication platform. It has been revealed that domain experts have a preference for such languages as they offer good learnability, readability and ability to focus on the idea of application. The perspective of developers (technical persons) is introduced on the basis of feedback obtained from 84 students of Computer Science at the Lublin University of Technology, who attended comprehensive workshops followed by an anonymous survey. All participants received theoretical and practical training in modeling mobile software using the same set of languages as domain experts. An analysis of the results revealed that opinions expressed by technical and nontechnical persons concerning SGDSLs oriented on defining a flow of actions is consistent. Most respondents claimed that such languages might be valuable as tools for creating a common communication platform.

Introduction

Due to the rapid pace of life today as well as the technical and organizational progress of modern societies, the risk of getting injured or facing a life-threatening situation is greater. Another factor are natural and human-caused industrial disasters, resulting in pollution, which are frequently transborder and affect large populations. Facing such challenges requires not only

ISBN 978–83–7431–536–4 ISSN 0860-150X

policy making and analysis (Sauer et al., 2012, 2013a, 2015), but also help from mobile technologies and regular people (social reporting). Crowe (2012) gives insight into social media usage in case of mass-scale life-threatening events. He focuses mainly on mainstream media such as Facebook, Twitter, or YouTube, among others; nevertheless, this could be seen as a general overview of the available solutions. In addition, Sauer et al. (2013b), in Chapter 10 of the paper, discuss the role of social reporting systems as an aid in solving environmental problems on the example of Visegrad countries.

Social systems for reporting life-threatening situations could be seen as a great opportunity for the improvement of organization of the process of rescuing people, by introducing access to a fast flow of rich information, including GPS coordinates and multimedia. Emergency services (police, ambulance and firefighting services), hospitals and regular citizens are among the beneficiaries. The technology and services are of great help as well as the relatively cheap and universal access to mobile Internet. Moreover, Forrester (Taylor, 2015) predicted that by the end of 2016, 4.8 billion people globally will be using a mobile phone, whereas 46% of the population will be using a smartphone.

The characteristics of contemporary systems for reporting life-threatening situations were discussed in greater detail in Żyła (2015) and Sauer et al. (2013b). Nevertheless, it is worth reminding that they should: distribute reports originating from heterogeneous data sources to proper emergency services on the basis of their responsibilities; provide mechanisms ensuring accuracy and reliability of data; genuinely, as opposed to theoretically, aid processing and aggregation of data. Introducing high technology for such purposes has numerous advantages (Huang et al., 2010; Jaeger et al., 2007; Kristensen et al., 2006; Namahoot & Brückner, 2015; Ziniewicz et al., 2011), although barriers might be faced (Pędziński et al., 2013) as well. In the author's opinion, the biggest challenges are ensuring interoperability of reporting system and making them compatible with procedures and policies that determine the actions of emergency services. Moreover, aspects of usability (Borys & Plechwska-Wójcik, 2013) and promotion (marketing) have to be kept in mind as well.

Overcoming these challenges is a non-trivial interdisciplinary task that requires close cooperation between experts from many domains, such as medical doctors, firemen, policemen, policy makers, clerks, or developers of IT systems, among others. There is a high probability that such experts (apart from developers of IT systems) possess no or highly limited IT skills, concerning not only programming but also designing software with the use

of industrial tools (languages and editors). Despite this, they are usually a precious source of information; hence, involving them in the process of creating systems for reporting life-threatening situations is highly important. This fits in with the trend of "democratization of software development" (Kawasaki, 2016; Bau et al., 2017), which postulates, among others, involving nontechnical persons as domain experts or software creators. In the case of the aforementioned reporting systems, due to their complexity and interoperability, the latter option is more feasible.

To be more precise, a nontechnical person (sometimes called a person with limited technical skills) can be further defined as a person unable (or encountering significant difficulties) to create software utilizing classical methods and programming languages. Such difficulties might result from personal intellectual deficiencies (problems with learning the subject in question) or lack of proper education. In contrast, a technical person can be defined, for the purpose of this paper, as a person with skills and experience sufficient for developing software utilizing classical methods and programming languages.

During this decade, there has been a time of dynamic evolution of model-driven engineering (MDE), where models are the primary artifacts during the process of software development. Brambilla et al. (2012), as well as Stahl and Volter (2006), provide a fair introduction to the issue. Information concerning its adoption by the IT industry can be found in the following studies (Davies et al., 2014; Hutchinson et al., 2014; Schlieter et al., 2015; Whittle et al., 2013, 2014). Nevertheless, research into the utilization of MDE concepts by the industry is still at an unsatisfactory level. A literature analysis and the author's experience gained at the Lublin University of Technology (Żyła & Kęsik, 2012) show that MDE tools might play an important role for people without programming skills.

One of the key MDE concepts are graphical modeling languages. They could be used to create platform-independent models at different levels of abstraction, which could help to involve nontechnical persons into the process of software development. In a study by Żyła (2015), graphical modeling languages available for the mobile domain were divided into 3 groups: G1 – languages oriented on event-based programming (represented by App Inventor), G2 – languages oriented on defining low-level interactions among objects (represented by UML – Unified Modeling Language), G3 – languages oriented on defining the flow of actions (represented by AergiaML). The focus is on G3 (also called SGDSLs), as one of the goals is to check whether such languages are suitable for nontechnical users. For more details (such

as groups characteristics, justification for the selection of groups representatives) please refer to Żyła (2015). The following chapter summarizes the research conducted among nontechnical persons, based on the division presented above. Finally, the general purpose of this paper is to expand on the aforementioned research study by including the perspective of technical persons.

SGDSLs – Nontechnical Persons' Perspective

The purpose of the research was to gain insight as to whether the use of SGDSLs could help in creating a common communication platform during the development of large mobile reporting systems. The assumptions were made after conducting a series of workshops (67 attendees) and carrying out surveys (29 answers) designed for nontechnical persons who have never participated in a process of creating software nor learned how to do it. Details can be found in papers Żyła (2015, 2017). In general, respondents gave positive feedback on SGDSLs, which encourages further research into their utilization in the process of developing complex reporting systems. Surprisingly enough, respondents gave low rates to UML, an industrial standard.

Respondents indicated SGDSLs, represented by AergiaML, as the easiest to read and understand. According to the Wilcoxon signed-rank test, there was a statistically significant difference between UML (G2) or App Inventor (G1) and AergiaML, in favor of the latter. UML and App Inventor received similar scores – no statistically significant difference was revealed. The same results were obtained in the areas of ease of learning and use. Calculations were performed with the use of R environment. In another part of the survey, respondents were asked to recognize the functionality depicted by the models. AergiaML received the highest percentage of correct responses – a score 13% higher than App Inventor and 50% higher than UML. In each of the cases, the level of detail in the models was sufficient to generate a piece of a working application.

Finally, a series of questions concerning the perspectives of SGDSLs was asked. The responses were as follows:

1. 83% of respondents were willing to use AergiaML-like languages in the future. 7% were unsure.
2. 79% of respondents stated that AergiaML-like languages allow them to focus on the idea of application and they are not distracted by too many technology-specific details. 14% were unsure.

3. 52% of respondents stated that AergiaML-like languages could help them to communicate with software developers successfully. 17% were unsure.
4. 57% of respondents stated that they will be able to learn AergiaML in a degree that allows to create applications fulfilling their everyday needs. 21% were unsure.

The high rate of unsure persons (question 3 and 4) might be the result of no prior experience with software development.

Aim of the Study

An analysis of scientific literature and the author's own research revealed that nontechnical persons might have problems with using the modeling languages that are industrial standards, such as the commonly used UML and BPMN. It was also revealed that they might be dealing well with simplified graphical modeling languages oriented on defining the flow of actions between components performing complex activities. According to Żyła (2015), nontechnical persons preferred such languages as they offer good learnability, readability and can be used as communication tools that allow to focus on the idea of application. Nevertheless, for a fuller picture, feedback from technical persons should also be presented.

The research question is similar as in the case of nontechnical persons and the research described in Żyła (2015), i.e. whether the use of SGDSLs could help in creating a common communication platform during the development of large mobile reporting systems. Although this paper presents the perspective of technical persons, it also attempts to find out whether statements given by both the groups in question are consistent. With this in mind, the following research hypotheses were formulated:

H1. Models in SGDSLs could be easier to read and understand for technical people than in the cases of industrially recognized solutions.
H2. SGDSLs could help developers to communicate with domain experts during the process of development of mobile reporting systems.
H3. Technical persons are willing to use SGDSLs.

A positive verification of H1–H3 makes SGDSLs worth considering as part of the common communication platform during the development of large mobile reporting systems, as they are acceptable, readable and learnable both by technical and nontechnical persons.

Materials and Methods

In order to verify the formulated research hypotheses, workshops on modeling software for mobile devices were conducted at the Lublin University of Technology. They were targeted at technical persons, mostly students of the Computer Science course. One of the goals was to involve persons with appropriate predispositions, such as analytical thinking and experience in software development. After the workshop, all the participants were asked to fill in an anonymous survey.

A single workshop lasted about 20 hours, with each person participating in one workshop only. The main covered topics included: notions related to model-driven engineering, the role of models in software development, communication with domain experts, graphical modeling of mobile systems (available tools and languages). Due to time constraints, as learning languages and making models is a time-consuming process, three graphical modeling languages were chosen to represent language groups G1–G3: App Inventor, UML and AergiaML (rationale can be found in Żyła (2015)). Their characteristics are representative for the groups, thus assumptions concerning the whole group (the way/paradigm of modeling) are made based on feedback concerning the chosen language. After introducing the theory of the chosen languages, respondents modeled few real-life mobile applications utilizing each of the modeling languages.

The anonymous survey was divided into 3 parts:

1. Personal background – information regarding studies and employment, and skills in mobile software development.
2. Tasks – focused on a comparison of the chosen languages.
   - T1: Mark 1 to 6 the usefulness of App Inventor in a process of specifying functional requirements of software.
   - T2: Mark 1 to 6 the usefulness of UML in a process of specifying functional requirements of software.
   - T3: Mark 1 to 6 the usefulness of AergiaML in a process of specifying functional requirements of software.
   - T4: Mark 1 to 6 how easy it is for you to read and understand models in App Inventor.
   - T5: Mark 1 to 6 how easy it is for you to read and understand models in Unified Modeling Language (UML).
   - T6: Mark 1 to 6 how easy it is for you to read and understand models in Aergia Modeling Language (AergiaML).
3. Questions – focused on collecting opinions about SGDSLs.

Q1: Do you think that AergiaML allows you to focus on the idea of application and you are not distracted by too many technology-specific details?
Q2: Do you think that you could use AergiaML in the future?
Q3: Do you think that using AergiaML would help nontechnical persons (domain experts) to communicate with software developers successfully?
Q4: Do you think that AergiaML could be used by nontechnical persons?

Results obtained in question Q1 and tasks T4–T6 verify hypothesis H1. Hypothesis H2 is verified by answers to questions Q3, Q4, and marks from tasks T1–T3. Finally, hypothesis H3 is verified by answers to question Q2 and tasks T1–T6.

Results and Discussion

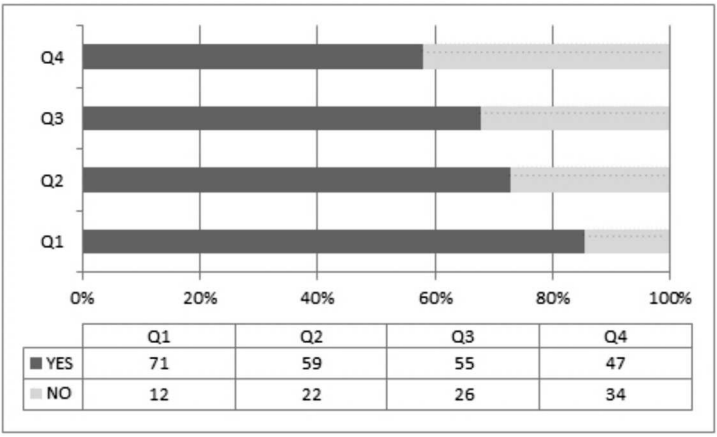
The workshops gathered 84 students of the Computer Science course with experience in software development. 42 (50%) of those were already employed by the IT industry. All of them filled in the anonymous survey.

The respondents, in their answers to the posed questions, generally gave positive feedback concerning the role of SGDSLs as communication tools in the process of software development. The following results were obtained (Figure 1):

Q1: 71 (86%) of respondents stated that AergiaML-like languages allow them to focus on the idea of application and they are not distracted by too many technology-specific details.
Q2: 59 (73%) of respondents were willing to use AergiaML-like languages in the future.
Q3: 55 (68%) of respondents stated that AergiaML-like languages could help them to communicate with nontechnical persons (domain experts) successfully.
Q4: 47 (58%) of respondents stated that AergiaML-like languages could be used by nontechnical persons.

The results prove that technical persons are willing to use SGDLs (hypothesis H3). Moreover, they are able to learn, accept and use languages of this kind as communication tools and perceive it as useful both for themselves and domain experts.

In order to check the statistical significance of the differences in marks given in tasks T1–T6, the Wilcoxon signed-rank test was used. Each time,



	Q1	Q2	Q3	Q4
YES	71	59	55	47
NO	12	22	26	34

Figure 1. Answers to survey questions Q1–Q4

due to multiple testing, the standard significance level of 5% was corrected, using Bonferroni, to 0.016. Calculations were performed using R environment. To improve evaluation, the IQR and median of the marks given are compiled in Table 1.

Table 1. The IQR and median of marks given during tasks T1–T6

	T1 (App Inv.)	**T2 (UML)**	**T3 (AergiaML)**	**T4 (App Inv.)**	**T5 (UML)**	**T6 (AergiaML)**
IQR	2,75	2	2	2	2	2
Median	3	4	4	4	4	4

The respondents, by solving tasks T4–T6, indicated readability and understandability of models in SGDSLs oriented on defining the flow of actions (G3, represented by AergiaML) to be as good as in the case of languages oriented on event-based programming (G1, represented by App Inventor). Marks given to UML, representing G2 (languages oriented on defining low-level interactions between objects), were significantly lower. Moreover, 86% of respondents stated that AergiaML-like languages allow them to focus on the idea of application. This proves that SGDSLs could be easier to read and understand for technical people than industrially recognized solutions (hypothesis H1), represented by UML.

The Wilcoxon signed-rank test shows that there is no statistically significant difference between ease of reading and understanding of models in either AergiaML or App Inventor (two-tailed test p-value of 0.810; the hypothesis being that the score for AergiaML would be different than for App

Inventor). However, there is a statistically significant difference between AergiaML or App Inventor and UML, in favor of AergiaML and App Inventor (p-values of < 0.001 and < 0.001, respectively; the hypothesis was that the score for AergiaML or App Inventor would be higher than for UML).

It might be surprising that models in App Inventor were indicated as more readable and understandable (despite the presentation issues described in Smutny (2011), Kowalczyk et al. (2016)) than those in UML. It might be the case that the respondents were extensively trained in programming in the course of their academic studies. Additionally, App Inventor provides simplified syntax oriented on the mobile domain, which is another advantage. The key disadvantage of UML, on the other hand, is its being general purpose, which makes models more complicated, especially when ensuring the same level of functional details as in the case of other chosen languages.

The respondents, by solving tasks T1–T3, indicated the usefulness of SGDSLs oriented on defining a flow of actions (G3, represented by AergiaML) in a process of specifying functional requirements of software to be as good as in the case of languages oriented on defining low-level interactions between objects (G2, represented by UML). Marks given to App Inventor, representing G1 (languages oriented on event-based programming), were significantly lower. Moreover, 68% of respondents stated that they could use AergiaML for communication purposes, whereas 58% were convinced that nontechnical persons (domain experts) would be able to handle such languages (as confirmed by feedback received from nontechnical persons). This proves that SGDSLs could help developers to communicate with domain experts during the process of development of mobile software (hypothesis H2).

The Wilcoxon signed-rank test shows no statistically significant difference between usefulness, in specifying functional requirements, of UML and AergiaML (two-tailed test p-value of 0.867; the hypothesis being that the score for AergiaML would be different than for UML). However, there is a statistically significant difference between AergiaML or UML and App Inventor, in favor of AergiaML and UML (p-values of < 0.001 and 0.002, respectively; the hypothesis was that the scores for AergiaML or UML would be higher than for App Inventor).

In this case, UML received significantly better scores than App Inventor. The reason might be that respondents, due to their knowledge/experience regarding the organization of software development, perceived UML as a proven industrial standard (rightly so) of software requirements specification. On the other hand, App Inventor, despite its obvious disadvantages, is perceived as a tool that could be used for educational purposes or designing personal applications, but not in commercial projects.

Conclusions

The goal of this paper is to add the perspective of technical persons (developers) to the discussion on whether using graphical domain-specific languages oriented on defining a simplified flow of actions (SGDSLs) could help in creating a common communication platform during the development of mobile reporting systems involving many parties, including police, ambulance and firefighting services, and regular citizens. The perspective of nontechnical persons (domain experts) has already been introduced in Żyła (2015, 2017). They declared that: models in SGDSLs are easy to read and understand; SGDSLs are easy to learn and use; SGDSLs might help in creating a common communication platform during the development of large mobile reporting systems; they are willing to use SGDSLs in the future. Finally, the largest number of correct answers regarding the functionality depicted by the models was recorded for SGDSLs.

In order to introduce the perspective of technical persons, three hypotheses (concerning the ability to read and understand, the ability to facilitate communication with nontechnical persons and the willingness to use) were formulated. They were verified during a research study (workshops consisting of a theoretical and a practical part) conducted with 84 students of the Computer Science course with experience in software development. All of them filled in an anonymous survey. An analysis of the submitted surveys confirmed the hypotheses and revealed that technical persons: indicated SGDSLs as useful in the process of specifying functional requirements of software; indicated models in SGDSLs as easy to read and understand. Most of them also declared that such languages allow to focus on the idea of application and help to communicate with nontechnical persons who should be able to use them.

The opinion of nontechnical persons is consistent with that of technical persons. Both parties are willing to accept and use SGDSLs as communication tools and perceive them as useful. Nontechnical users declared good learnability and usefulness of such languages, whereas technical users should be able to learn and use the languages easily and at a satisfactory level, due to their personal predispositions, education and experience. Technical persons, as a result of the knowledge they possess, sympathized more with classical methods and tools, although, seeing the potential of SGDSLs, they gave it high marks. On the other hand, nontechnical persons, due to their problems with classical methods and tools, were very enthusiastic towards them, thus the dominance of SGDSLs over other groups of languages was not statistically significant in a single instant only.

To summarize, the positive verification of H1–H3, the consistent opinions of technical and nontechnical users, as well as the advantages of model-driven development (Stahl & Volter, 2006; Żyła, 2013) make SGDSLs worth considering as optional tools to aid in the creation of a common communication platform between developers and domain experts involved in the process of development of complex systems, such as the reporting system for emergency services mentioned earlier in this paper.

REFERENCES

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable Programming: Blocks and Beyond. *Communications of the ACM*, *60*(6), 72–80. doi: 10.1145/3015455

Borys, M., & Plechawska-Wójcik, M. (2013). Usability and accessibility testing of mobile application interfaces. *Nierówności społeczne a wzrost gospodarczy*, *35*, 63–77.

Brambilla, M., Cabot, J., & Wimmer, M. (2012). *Model-driven software engineering in practice*. Morgan & Claypool Publishers.

Crowe, A. (2012). *Disasters 2.0. The application of social media systems for modern emergency management*. Boca Raton, FL: CRC Press.

Davies, J., Gibbons, J., Welch, J., & Crichton, E. (2014). Model-driven engineering of information systems: 10 years and 1000 versions. *Science of Computer Programming*, 89, 88–104.

Huang, C. M., Chan, E., & Hyder, A. A. (2010). Web 2.0 and internet social networking: a new tool for disaster management? – Lessons from Taiwan. *Medical Informatics and Decision Making*, 10:57. doi: 10.1186/1472-6947-10-57.

Hutchinson, J., Whittle, J., & Rouncefield, M. (2014). Model-driven engineering practices in industry: social, organizational and managerial factors that lead to success or failure. *Science of Computer Programming*, *89*, 144–161.

Jaeger, P. T., Shneiderman, B., Fleischmann, K. R., Preece, J., Qua, Y., & Wua, P. F. (2007). Community response grids: e-government, social networks, and effective emergency management. *Telecommunications Policy*, *31*, 592–604.

Kawasaki, B. (2016, May 20). App development should be democratized, *DZone / Mobile Zone*. Retrieved from https://dzone.com/articles/app-development-should-be-democratized

Kowalczyk, R., Turczyński, L., & Żyła, K. (2016). Comparison of App Inventor 2 and Java in creating personal applications for Android on example of a notepad. *Advances in Science and Technology Research Journal*, *10*(31), 247–254. doi: https://doi.org/10.12913/22998624/64058

Kristensen, M., Kyng, M., & Palen, L. (2006). Participatory design in emergency medical service: designing for future practice. In *CHI '06 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 22–27 April 2006 (pp. 161–170). New York, NY, USA: ACM. doi: 10.1145/1124772.1124798

Namahoot, C. S., & Brückner, M. (2015). SPEARS: Smart phone emergency and accident reporting system using social network service and Dijkstra's algorithm on Android. *Lecture Notes in Electrical Engineering*, *310*, 173–182.

Pędziński, B., Sowa, P., Pędziński, W., Krzyżak, M., Maślach, D., & Szpak, A. (2013). Information and communication technologies in primary healthcare – barriers and facilitators in the implementation process. *Studies in Logic, Grammar and Rhetoric. Logical, Statistical and Computer Methods in Medicine*, *35*(48), 179–189.

Sauer, P., Fiala, P., & Dvorak, A. (2013a). Modelling of environmental risk management under information asymmetry. In J. Hrebicek, G. Schimak, M. Kubasek & A. E. Rizzoli (Eds.), *Environmental Software Systems. Fostering Information Sharing. ISESS 2013. IFIP Advances in Information and Communication Technology, vol 413* (pp. 391–402). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-642-41151-9_37

Sauer, P., Fiala, P., Dvorak, A., Kolinsky, O., Prasek, J., Ferbar, P., & Rederer, L. (2015). Improving quality of surface waters with coalition projects and environmental subsidy negotiation. *Polish Journal of Environmental Studies*, *24*(3), 1299–1307. doi: 10.15244/pjoes/27811

Sauer, P., Kreuz, J., Hadrabova, A., & Dvorak, A. (2012). Assessment of environmental policy implementation: two case studies from the Czech Republic. *Polish Journal of Environmental Studies*, *21*(5), 1382–1391.

Sauer, P., Svihlova, D., Dvorak, A., Lisa, A., Bitta, J., Kęsik, J., Żyła, K., et al. (2013b). *Visegrad countries: environmental problems and policies*. Prague, Czech Republic: CENIA.

Schlieter, H., Burwitz, M., Schonherr, O., & Benedict, M. (2015). Towards model driven architecture in health care information system development. In *Wirtschaftsinformatik Proceedings 2015*, 4–6 March 2015 (pp. 497–511). Association for Information Systems.

Smutny, P. (2011). Visual Programming for Smartphones. In *12th International Carpathian Control Conference (ICCC), Velke Karlovice, Czech Republic*, 25–28 May 2011 (pp. 358–361). IEEE.

Stahl, T., & Volter, M. (2006). *Model-driven software development*. West Sussex, England: John Wiley & Sons Ltd.

Taylor, H. (2015, November 9). How mobile will transform business in 2016: Forrester. *CNBC*. Retrieved from http://www.cnbc.com/2015/11/09/forrester-mobile-predictions-for-2016.html

Whittle, J., Hutchinson, J., & Rouncefield, M. (2014). The state of practice in model-driven engineering. *IEEE Software*, *31*(3), 79–85. doi: 10.1109/MS.2013.65

Whittle, J., Hutchinson, J., Rouncefield, M., Burden, H., & Heldal, R. (2013). Industrial adoption of model-driven engineering: are the tools really the problem? *Lecture Notes in Computer Science*, *8107*, 1–17.

Ziniewicz, P., Malinowski, P., Milewski, R., Mnich, Z. S., & Wołczyński S. (2011). Clinical department information system's internal structure. *Studies in Logic, Grammar and Rhetoric. Logical, Statistical and Computer Methods in Medicine*, *25*(38), 191–200.

Żyła, K. (2013). Economic aspects of user-oriented modeling for mobile devices. *Actual Problems of Economics*, *4*(142), 334–340.

Żyła, K. (2015). Perspectives of simplified graphical domain-specific languages as communication tools in developing mobile systems for reporting life-threatening situations. *Studies in Logic, Grammar and Rhetoric. Logical, Statistical and Computer Methods in Medicine*, *43*(56), 161–175. doi: https://doi.org/10.1515/slgr-2015-0048

Żyła, K. (2017). Simplified graphical domain-specific languages for the mobile domain – perspectives of learnability by nontechnical users. *Applied Computer Science*, *13*(3), 32–40.

Żyła, K., & Kęsik, J. (2012). Podsumowanie i kierunki badań nad MDE na Politechnice Lubelskiej. In M. Miłosz, & W. Wójcik (Eds.), *Kompetentny absolwent informatyki 2012* (pp. 135–152). Lublin: Polskie Towarzystwo Informatyczne.