DE GRUYTER OPEN

STUDIES IN LOGIC, GRAMMAR AND RHETORIC 43 (56) 2015

DOI: 10.1515/slgr-2015-0048

# Perspectives of Simplified Graphical Domain-Specific Languages as Communication Tools in Developing Mobile Systems for Reporting Life-Threatening Situations

**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 constantly gaining popularity, especially when it comes to environmental and cultural heritage protection. Reporting life-threatening situations, such as sudden natural disasters or traffic accidents, constitutes the same class of problems and could be aided by IT systems of similar architecture. Emergency services also could benefit from such kinds of solutions, e.g., by getting an accurate location of the place where help is needed as well as fast exchange of information. Designing and developing systems for reporting life-threatening situations is not a trivial task, requiring tight cooperation among software developers and experts of different domains, who most likely would have trouble with industrially recognized languages and notations. Thus, the question is whether using simplified graphical domain-specific languages (SGDSLs) could help in creating a common communication platform. In order to answer that question, short workshops were conducted at the University of Economics in Prague, Czech Republic and in Lublin, Poland. They involved people trained during their studies to work in administration and emergency services, among others, as policy makers, medicine specialists, environmentalists, or managers. All participants received brief training in graphical modeling languages in the context of modeling mobile applications and were asked to voluntarily fill in an anonymous survey in order to check their knowledge about and attitude toward the presented technologies. Most respondents claimed that SGDSLs oriented on defining a flow of actions might be valuable as tools for creating a common communication platform.

## Introduction

Due to heavy traffic, frequent natural disasters, and terror attacks, the risk of serious injuries and accidents is very high. Constantly improving accessibility of technically and functionally advanced mobile devices as well as decreasing prices of access to the mobile Internet (Sauer et al., 2013) create a good environment for introducing systems for reporting life-threatening

ISBN 978–83–7431–464–0 ISSN 0860-150X

situations. Their usage presents a great opportunity to improve organization of the process of rescuing people, by introducing access to the fast flow of rich information, including GPS coordinates and multimedia. Emergency services (police, ambulance and firefighting services), hospitals and regular citizens could be mentioned as beneficiaries. Despite possible barriers (Pędziński et al., 2013), benefits of using advanced technology for such purposes have already been noticed (Huang et al., 2010; Jaeger et al., 2007; Kristensen et al., 2006; Namahoot et al., 2015; Surowik, 2009; Ziniewicz et al., 2011).

Contemporary systems for reporting life-threatening situations should make use of modern technology, especially mobile devices. Technical possibilities significantly influence the complexity of such systems that are currently expected to:

1. Allow reporting of all kinds of life-threatening situations requiring intervention of specialized emergency services – Reports should be distributed to proper services, based on their responsibilities and distance from the place of accident. Moreover, the system should be able to gather data from heterogeneous sources such as automatic systems embedded in cars, as well as from mobile applications installed on smartphones.

2. Aid processing and aggregation of the reports by emergency services – The system should allow many parties (police, hospitals, blood banks, local authorities, etc.) to participate in the whole process of rescuing a victim, including his/her identification, if possible. Moreover, the reporting system should be able to cooperate and exchange data with IT systems that are already in use, e.g., with medical databases or car navigation systems displaying the current situation on the roads.

3. Provide mechanisms ensuring accuracy and reliability of data, even suggesting the scope of intervention – Reliability of data is particularly important as it will be aggregated from many sources that could provide different data about the same accident, e.g., because of human factors like stress, inability to pinpoint the location, inability to determine the scale of the accident, or intentional fake reports. Moreover, the shape of the user interface and overall functionality should allow emergency services fast and simple processing of reports.

Such a high level of complexity requires competences that are usually beyond the capabilities of software developers. They know how to program IT systems, but they most likely do not have knowledge about: legal regulations concerning medical care and processing medical data; internal procedures and organization of police, ambulance or firefighting services; regulations concerning data exchange among the aforementioned services. People

who have knowledge concerning one or many of the aforementioned aspects (topics, domains) are called domain experts and they are an invaluable and necessary source of information during the software development process. One of the goals of software engineering is to create a communication platform that will allow for an efficient and systematized exchange of ideas among domain experts and software developers.

During the last decade, there was a time of dynamic evolution of model-driven engineering (MDE), where models are primary artifacts during the process of software development. One of the MDE concepts is domain-specific languages (DSLs). They could be used to create platform-independent models that could be later transformed into running applications by code generation. Translation to platform-specific runnable code requires a separate code generator for each platform. One of the goals of MDE is to facilitate communication with domain experts by using higher abstraction levels – providing the ability to focus on an idea, not on implementation details (Brambilla et al., 2012; Kęsik et al., 2011). Despite some doubts concerning claimed benefits of using MDE tools (Hutchinson et al., 2011a) and the small number of fully-mature tools, experience gained at the Lublin University of Technology (Kęsik et al., 2014; Żyła, 2013; Żyła et al., 2012) and the case of App Inventor (Wolber, 2011) show that MDE tools might play an important role for people without programming skills.

Research about utilization of MDE concepts by the industry is still not at a satisfactory level. In articles written by Hutchinson et al. (2011a) and Mohagheghi et al. (2008), the authors claim that the amount of experimental data, especially quantitative data, is not sufficient. Many studies are focused on the code-generation or technical aspects of adoption of MDE techniques by software companies, although they do not sufficiently address the ability of domain experts to use different solutions (Davies et al., 2014; Fan et al., 2015; Hutchinson et al., 2011b, 2014; Schlieter et al., 2015; Whittle et al., 2013, 2014). The same applies to using SGDSLs as communication tools in the process of developing mobile reporting systems, which makes the research presented in this paper valuable.

Due to the specific character of the mentioned reporting systems as well as research participants, only graphical languages that are intended for modeling mobile systems are within the scope of this study. The domain of mobile systems was chosen because of the need for high mobility of systems for reporting life-threatening situations, which affects specificity of devices and methods of gathering data, and the popularity of mobile technologies. Graphical languages were chosen due to the research participants' lack of IT background. As practice shows, there is more than high risk

that participants would have refused to take part in the research, stating that textual languages seemed to be too difficult for them, had graphical languages not been chosen.

For the purpose of the study presented in this paper, three subgroups were identified from the chosen group of graphical languages (LG stands for Language Group):

LG1: Graphical domain-specific languages oriented on event-based programming (model looks similar to code)
- require programming skills
- operate on the level of instructions
- simplified syntax – some irrelevant technical details are hidden;

LG2: General purpose graphical modeling languages oriented on defining low-level interactions among objects
- require object-oriented programming skills
- operate on the level of objects and invocations of methods;

LG3: Simplified graphical domain-specific languages oriented on defining flow of actions (sometimes also called SGDSLs)
- based on high-level components performing complex actions (activities)
- operate on the level of flow of complex actions.

In the case of LG3, the idea is as follows: language is based on components that perform complex activities such as: saving something to a database, sending a message, getting coordinates, etc. The role of a designer is to show how those components interact by connecting them in chains of complex actions, executed in a particular order, and to show how values/parameters flow between components.

## Aim of the Study

Developing large reporting systems, such as the mentioned system for reporting life-threatening situations, requires cooperation among software developers and many experts on different domains, who provide necessary/valuable knowledge about the system functionality, procedures, policies and integration with already existing systems used by emergency services. For the purpose of successful communication, a common platform is needed. Unfortunately, domain experts usually do not have the necessary IT background, which results in trouble with understanding and using industrially recognized languages and notations (solutions) in general. Moreover, quite often, parties communicate in a hurry due to short deadlines. Thus,

it happens that domain experts need to understand diagrams presented by a software developer despite having no previous knowledge/training regarding the used technology. Very often, the first look at the solution decides whether a person will give up, stating that something is so difficult that there is no point in trying to understand it at all.

Simplified graphical domain-specific languages (SGDSLs) oriented on defining a flow of actions might be the solution to at least some of the abovementioned problems. Thus, the question is whether usage of SGDSLs could help in creating a common communication platform during development of large mobile reporting systems.

With this research question in mind, the following research hypotheses were formulated (all of them concern domain experts without an IT background):

H1. Models in industrially recognized solutions could be hard to read and understand for domain experts.
H2. Domain experts could incorrectly read or understand models in industrially recognized solutions due to their complexity.
H3. Models in SGDSLs could be easier to read and understand for domain experts than in cases of industrially recognized solutions.
H4. Domain experts are capable of reading and understanding models in SGDSLs more correctly than in cases of industrially recognized solutions.
H5. SGDSLs could help domain experts to communicate with developers during the process of developing mobile reporting systems.
H6. Domain experts are willing to use SGDSLs.

Positive verification of H1 and H2 creates the need to seek simpler methods of communication with domain experts. Positive verification of H3–H6 makes SGDSLs worth considering as a part of the common communication platform during development of large mobile reporting systems.

## Materials and Methods

In order to verify the formulated research hypotheses, short workshops on modeling software for mobile devices were conducted in May 2015 at the University of Economics in Prague. A group of 55 English-speaking students and graduates participated in them. Supplementary workshops (12 participants) were conducted in September 2015, in Lublin, in order to extend gathered data by way of feedback from a group of English-speaking medical students and graduates. After the workshop, they were all asked to voluntarily fill in an anonymous survey.

Participants did not have previous experience in developing software for mobile devices or an IT background (they were not studying computer science or in a related field; they were not trained in software engineering, modeling languages, programming languages, etc.). As mobile systems for reporting life-threatening situations were at the center of the research, among the workshop participants were people trained during their studies to work in administration and emergency services (parties involved the most in developing such systems) as policy makers, medical specialists, sociologists (e.g. domain of quality of life), managers, environmentalists, etc.

A single workshop lasted no more than a few hours, and each person could participate in only one workshop. The main workshop topics were: mobile systems for reporting life-threatening situations (what those systems are, why they are needed, how they work and how they should work) and graphical modeling of mobile reporting systems (why modeling could be useful, what tools are available and how to use them, how to read models, how to communicate using models). Participants were also working with exemplary models depicting functionality available for emergency services while processing the report. The sets of models were different for workshops in Poland and the Czech Republic.

The pilot studies conducted on another group of participants (the same profile as regular participants) revealed that: they could not handle too many technical details, they got tired quite fast (even when they were able to handle the content) when working with long technical documents, and they did not understand questions about the process of developing software that were too specific. As a result of participant and time limitations, some simplifications were necessary. The following are most important: that the survey is as short and simple as possible and that each category of modeling solutions is represented by one solution (that when a participant mentions the particular name of a technology, he/she has a whole category in mind). The following representatives of language categories were chosen:

LG1: App Inventor, as a tool well recognized by the community (Massachusetts Institute of Technology, AppInventor, 2015; Wolber, 2011).

LG2: Unified Modeling Language (UML), as an industrially recognized tool (Arisholm et al., 2006).

LG3: Aergia Modeling Language (AergiaML), as a representative of LG3 developed at Lublin University of Technology.

The anonymous survey was divided into 3 parts:

1. Personal background – age, country of origin, information regarding studies, experience in creating and modeling mobile applications.

2. Tasks – focused on comparison of solutions as well as on reading and understanding models (parts of models used in the survey are different from the models used during workshops).
   - T1: Mark from 1 to 6 how easy it is for you to read and understand models in Unified Modeling Language (UML).
   - T2: Mark from 1 to 6 how easy it is for you to read and understand models in App Inventor.
   - T3: Mark from 1 to 6 how easy it is for you to read and understand models in Aergia Modeling Language (AergiaML).
   - T4: What is depicted by the presented model made in UML?
   - T5: What is depicted by the presented model made in App Inventor?
   - T6: What is depicted by the presented model made in AergiaML?
3. Questions – focused on gathering opinions about SGDSLs.
   - Q1: Do you think that using AergiaML models would help you to communicate with software developers successfully?
   - Q2: Do you think that you could use AergiaML in the future?

Figures 1–3 present the set of models created for the purpose of the survey conducted in the Czech Republic. This set of models should be considered as an example to illustrate the level of complexity of models in the survey tasks.

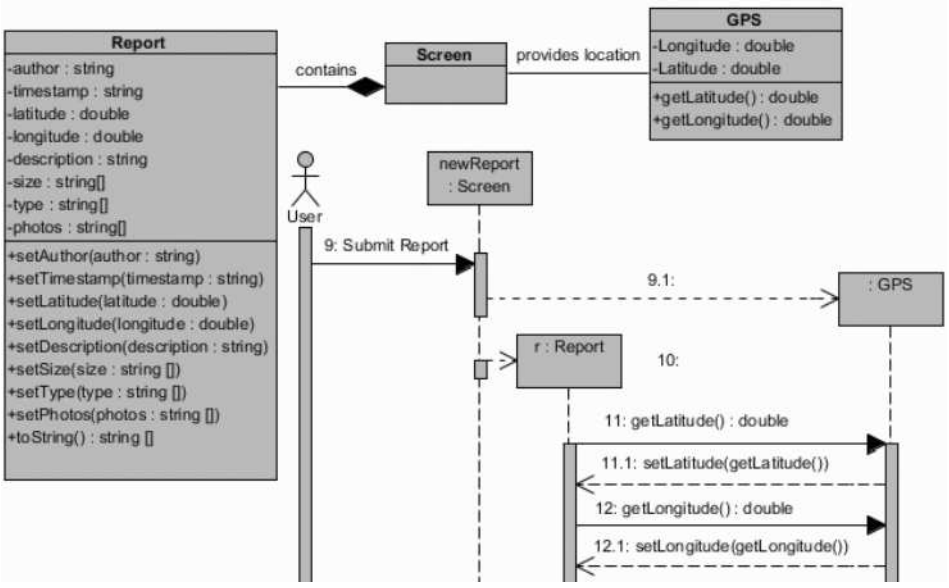


**Figure 1. Part of the model in UML from task T1**

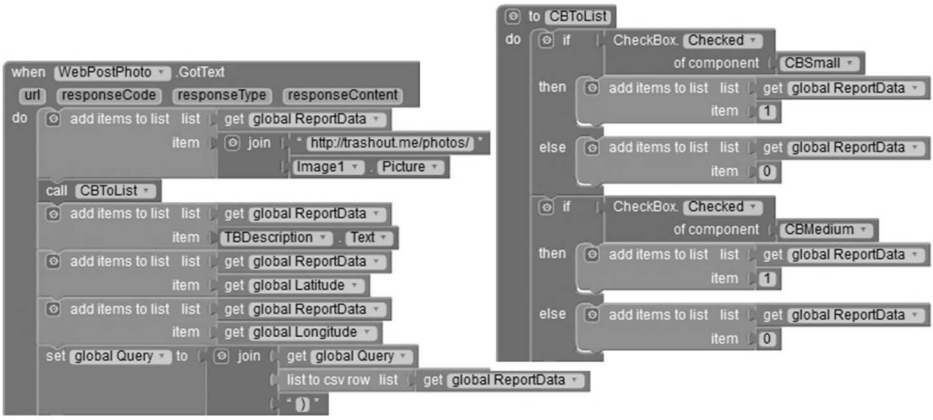


**Figure 2. Part of the model in App Inventor from task T2**

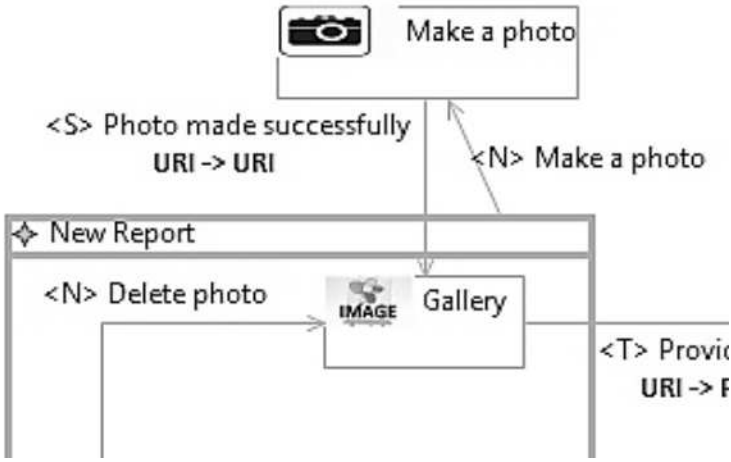


**Figure 3. Part of the model in AergiaML from task T3**

## Results and Discussion

Workshops were not included in the students' curriculum as a mandatory part of regular classes, thus they, as well as the survey, had to be voluntary. This significantly affected response rate – 67 persons attended the workshops but only 29 of them (including 7 medical specialists) submitted the survey. Taking any measures in order to forcefully improve the rate would be considered unethical.

Respondents who submitted the survey originated from more than 12 different countries throughout the world, including the Czech Republic, USA, Republic of Korea, Ukraine, Poland and France. Twelve of them (44.4%) declared that they had either earned or were studying for a bachelor's degree, while 15 of them (55.6%) declared the master's degree as the degree they either held or were working toward. The age structure of the respondents is presented in Table 1. None of them had ever created mobile applications or modeled mobile applications using App Inventor, Unified Modeling Language, or any other solution.

**Table 1. Age structure of the respondents**

| Years old | 15–20 | 21–25 | 26–30 | 31–35 | 36–40 | 41–45 |
|---|---|---|---|---|---|---|
| Number of respondents | 5 | 10 | 9 | 0 | 2 | 1 |
| Percentage of respondents | 18.5% | 37% | 33.3% | 0% | 7.4% | 3.7% |

Respondents, by solving tasks T1–T3, indicated models in simplified graphical domain-specific languages oriented on defining a simplified flow of actions (LG3), represented by AergiaML, as the easiest to read and understand – the average mark being 4.82. General purpose graphical modeling languages oriented on defining low-level interactions among objects (LG2), represented by UML, received an average mark of 3.43. Lastly, languages oriented on programming the application (LG1), represented by App Inventor, received an average mark of 2.86. The distribution of marks is presented by Figures 4–6 (the higher the mark, the better).

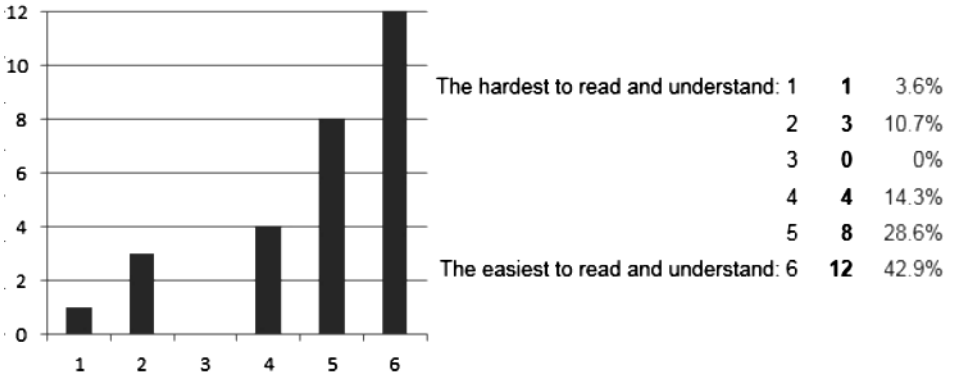


**Figure 4. Distribution of marks – AergiaML-like languages**

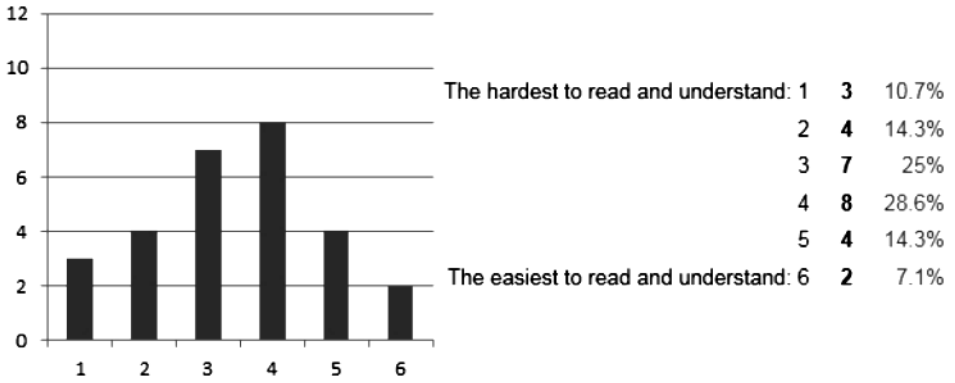


**Figure 5. Distribution of marks – UML-like languages**

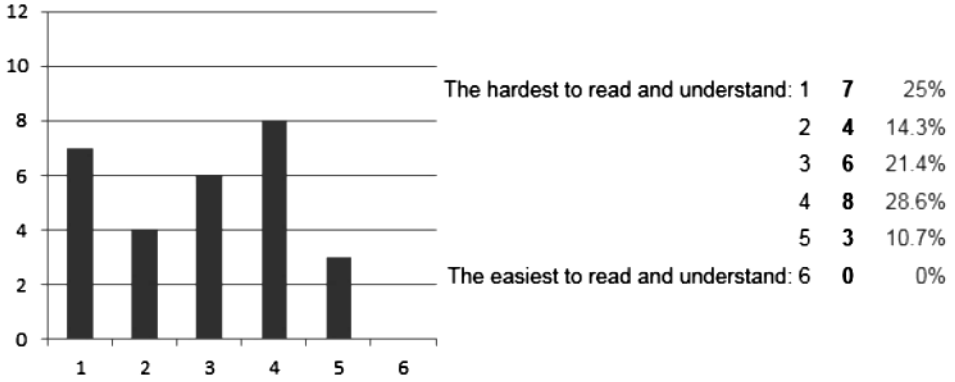


**Figure 6. Distribution of marks – App Inventor-like languages**

The Wilcoxon signed rank test shows that there is no statistically significant difference between ease of reading and understanding of models in UML and App Inventor (two-sided test p-value of 0.0965; the hypothesis was that the score of UML would be higher than App Inventor). However, there is a statistically significant difference between UML or App Inventor and AergiaML, in favor of the latter (p-value of 0.0029 and 0.0002; the hypothesis was that the score of AergiaML would be higher than any of the two other). Due to multiple testing, the standard significance level of 5% was corrected, using Bonferroni, to 0.016. Calculations were performed using R environment. This proved H1 and H3.

When it comes to reading and understanding exemplary models (tasks T4–T6), the results were as follows:

1. LG3, represented by AergiaML – 24 correct answers (85.7%).

2. LG1, represented by App Inventor – 21 correct answers (72.4%).
3. LG2, represented by UML – 10 correct answers (35.7%).

The McNemar test shows that the percentage of correct responses for App Inventor and AergiaML was statistically significantly greater than for UML (p-values of 0.0154 and 0.0003; the hypothesis was that the percentage of correct responses for App Inventor or AergiaML would be greater than for UML). However, the percentage of correct responses for App Inventor was not statistically significantly greater than for AergiaML (p-value of 0.1445; the hypothesis was that the percentage of correct responses for App Inventor would be greater than for AergiaML). Due to multiple testing, the standard significance level of 5% was corrected, using Bonferroni, to 0.016. Calculations were performed using R environment.

This especially proved H4, as well as H1–H3, and partially proved H5. What is significant and surprising is that an industrially recognized solution like UML had the lowest rate of correct answers (even App Inventor had a higher rate). This might have been caused by the complexity of UML, which requires knowledge regarding basics of object-oriented programming and two types of diagrams (sequence and class diagrams). App Inventor, in turn, offers the mixture of event and structural programming, which apparently is easier to grasp.

Respondents also answered questions Q1 and Q2, declaring ability and willingness to use AergiaML-like languages (LG2). This proved H5 and H6. The distribution of answers is presented in Figures 7 and 8. In the case of question Q1, 5 respondents (17.2%) chose the answer "I am not sure". Those respondents' uncertainty might have been the result of problems with imagining their role in a process of developing software, or the assumption that they would never participate in such a process (the same could be true in the case of respondents who chose "No" as an answer).


**Figure 7. Distribution of responses to question Q1, asking whether AergiaML-like languages could help the respondent to communicate with software developers successfully**


**Figure 8. Distribution of responses to question Q2, asking whether the respondent was willing to use AergiaML-like languages in the future**

Responses of medical specialists followed similar rules as those of other respondents. Exchange of thoughts during workshops in Lublin revealed that they would prefer languages that are easy to grasp and allow them to create small but complete models and focus on the idea of an application instead of implementation details. This was confirmed by the survey results – they gave higher marks to AergiaML (representing SGDSLs) and were able to recognize model functionality more correctly than in the case of other solutions.

## Conclusions

The goal of this paper is to discuss whether using graphical domain-specific languages oriented on defining a simplified flow of actions (SGDSLs) could help in creating a common communication platform during development of mobile reporting systems involving many parties, including police, ambulance and firefighting services, and regular citizens. Six hypotheses (concerning ability to read and understand models made in different kinds of modeling solutions, as well as their usefulness in creating a common communication platform) were formulated for that purpose and verified during research (workshops) conducted with 67 persons without any IT background – future domain experts able to work for the mentioned parties. Only 29 (including 7 medical specialists) of 67 persons submitted the survey, although this number reflects a reasonable response rate, as the workshops and the survey were voluntary.

Analysis of the submitted surveys revealed that domain experts without any IT background might have serious problems with understanding and using industrially recognized solutions such as Unified Modeling Language (hypotheses H1 and H2 were confirmed). The reason could be the

level of complexity of such solutions and the need for initial knowledge of some aspects of software development, for example the basics of object-oriented programming in the case of UML. This is a good entry point to start a discussion regarding whether SGDSLs could improve something and what the perspectives are. Analysis of the survey results confirmed the rest of the hypotheses and revealed that:

1. Most respondents declared that models in SGDSLs were easy to read and understand.
2. Most respondents declared that they would be willing to use SGDSLs in the future.
3. Most respondents declared that SGDSLs might help in creating a common communication platform during development of large mobile reporting systems.
4. The largest number of correct answers, regarding functionality depicted by the models, was recorded for SGDSLs.

Another advantage of using domain-specific languages (DSLs) is fast prototyping. DSLs, in the most cases, could be used to create platform-independent models that could be transformed into running applications by a code generator. This presents a great opportunity to use prototyping during the communication process, as a domain expert would be able to see effects of requested changes or other arrangements. Drawings (models) suddenly changing into something real, in just a few mouse clicks, might also improve the ability of a domain expert to participate in a software development process.

To summarize, positive verification of H3–H6 makes SGDSLs worth considering as optional tools to aid in creation of a common communication platform among developers and domain experts involved in development of complex systems, such as the reporting system for emergency services mentioned in this paper. The research results are promising. Thus, further research in this field will be conducted.

## REFERENCES

Arisholm, E., Briand, L. C., Hove, S. E., & Labiche, Y. (2006). The impact of UML documentation on software maintenance: an experimental evaluation. *IEEE Transactions on Software Engineering*, 32(6), 365–381.

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

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.

Fan, Z., Qiu, X., Dai, W., Ge, Y., Liu, L., & Duan, Y. (2015). A meta-modeling framework for a specific social domain: public opinion event. *Computer Science and Information Systems*, 12(2), 743–763.

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., Rouncefield, M., & Whittle, J. (2011a). Model-driven engineering practices in industry. In *ICSE '11 Proceedings of the 33rd International Conference on Software Engineering*, 21–28 May 2011 (pp. 633–642). New York, NY, USA: ACM.

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.

Hutchinson, J., Whittle, J., Rouncefield, M., & Kristoffersen, S. (2011b). Empirical assessment of MDE in industry. In *ICSE '11 Proceedings of the 33rd International Conference on Software Engineering*, 21–28 May 2011 (pp. 471–480). New York, NY, USA: ACM.

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.

Kęsik, J., & Żyła, K. (2011). *Współczesne technologie informatyczne. Technologie MDE w projektowaniu aplikacji internetowych*. Lublin: Politechnika Lubelska.

Kęsik, J., Żyła, K., & Nowakowski K. (2014). Domain-specific languages as tools for teaching 3D graphics. In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development (MODELSWARD '2014)*, 7–9 January 2014 (pp. 498–503). Lisbon, Portugal: IEEE.

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

Massachusetts Institute of Technology, AppInventor. (2015). *App Inventor Tutorial* [Video]. Retrieved from http://appinventor.mit.edu

Mohagheghi, P., & Dehlen, V. (2008). Where is the proof? – A review of experiences from applying MDE in industry. *Lecture Notes in Computer Science*, 5095, 432–443.

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., Svihlova, D., Dvorak, A., Lisa, A., Bitta, J., Kęsik, J., Żyła, K., et al. (2013). *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 (WI '2015)*, 4–6 March 2015 (pp. 497–511). Osnabrück, Germany: Association for Information Systems.

Surowik, D. (2009). Preface: Temporal aspect in information systems. *Studies in Logic, Grammar and Rhetoric*, 17(30), 7–12.

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.

Wolber, D. (2011). App Inventor and real-world motivation. In *SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education*, 9–12 March 2011 (pp. 601–606). New York, NY, USA: ACM.

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., & 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.