

Process Model of DSM Solution Development and Evolution for Small and Medium-Sized Software Companies¹

Dmitrij Koznov

Software Engineering Department
Saint-Petersburg State University
Saint Petersburg, Russia
e-mail: dkoznov@gmail.com

Domain-Specific Modeling (DSM) approach is developing in connection with problems of UML practical usage. This approach is intended for faster development of new visual languages, graphics editors, and tools of code generation, oriented for various problem domains. Commonly, DSM approach is applied within the bounds of a single software company for development of product lines, large products, etc. Development and support of the DSM solution turn out to be company's inner project, and the company itself appears to be the customer. Such situation discloses a mass of problems. Moreover, the company, commonly does not specialize in development and adoption of visual modeling solutions. Additionally, for small and medium-size companies, there are other problems: they are capable to assign only a relatively small budget for a DSM project, companies lack experienced developers for DSM project participation, have troubles with supply of stable maintenance process of a DSM solution, etc. So, an effective process has to be established in such company, if it is desired to design, implement, and use a DSM solution. In this paper advantages and problems of DSM projects at small and medium-sized companies are considered, a new process model based on MSF for DSM solution development and evolution is presented. The model includes flexible requirement management, a pilot project, provides iterative development, and further maintenance and support of the DSM solution.

Keyword – visual modeling, model-based development, DSM, process model, MSF.

I. INTRODUCTION

Domain-Specific Modeling (DSM) is an approach to software development that raises the level of abstraction beyond coding during development process by specifying programs directly using domain concepts. Final products can then be generated from these high-level specifications, and this became possible because the modeling language

and generator only need to fit the requirements of one domain, often in only one company [1, 2].

The given approach is widespread in the industry due to the following reasons: difficulties in industrial use of UML modeling language and UML tools [3]; product-line development, which requires companies to create internal unique development infrastructures [4]; increasing maturity of modeling metatools, such as Eclipse Modeling Project and, in particular, GMF [5], Microsoft Visio and DSL Tools [6], MetaEdit+ [7].

Following [2, 9, 10], we consider DSM approach in the context of a company. It should be mentioned, that in various companies worldwide a considerable quantity of DSM-solutions had been created and are applied at the moment. However, being a significant advantage, these solutions also cause problems to the companies. For the most part, DSM solutions had been developing and still are developed ad hoc, inspired by the enthusiasm of individual developers, who most often concentrate on technical aspects, leaving out management and maintenance. Fairly often, at the beginning, no one expects this solution to function for years, so it evolves spontaneously and grows gradually. DSM project generally is created internally, which brings further difficulties: absence of an external customer, "crawling" project scope, obstacles of maintenance, configuration management problems, etc. Practically, this situation resembles general software development process problems of the past, putting a question of adequate DSM process, which would have provided proper final product quality, predictability, and control over development process.

Apart from larger organizations, small and medium-sized ones have significant potential for DSM-solutions adoption. In general, developers and managers in such companies have a more flexible mindset, and they are ready for innovations. Small companies are also interested in increasing efficiency but can't afford staff increase or other significant investments. Successful DSM-solutions may facilitate resolution of this problem. However, small companies also have a number of peculiarities and restrictions: immature software development process,

¹ This work is partially supported by RFBR grant 11-01-00622-a.

occupancy and lack of qualified personnel, modest budget that could be assigned for a DSM project.

Apart from the DSM solution development tools, there are a lot of additional advanced techniques. For instance, in [15] light-weight approach for deployment of DSM languages in order reduce initial costs of DSM solution development. [16] suggested aspect-oriented approach to specify DSM language and eliminate efforts to implement of code generators. [17] proposes a mapping model, which allows to establish complex mappings between elements of abstract and concrete DSM language syntax. However, these and similar works are mostly concentrated on technical aspects of the DSM solution development. [8] is dedicated to investigation of various problems occurring in DSM solutions development with emphasis on technical characteristics and does not consider process difficulties per se. Only [2] considers in detail process and management issues and presents description of the DSM process. But, it only provides discrete techniques and recommendations without strict step-by-step model. Moreover, the study does not mention specific DSM approach for small and middle-sized software companies.

In this paper we consider characteristics and problems of DSM solutions development and evolution for small and medium-sized companies. After that, basing on our experience of various DSM solutions development [11, 12, 13], and several consulting projects, we suggest a DSM solution process model. The model is based on MSF [14]. We have chosen MSF due to its orientation on development of solution for individual customer instead of abstract software. The model includes flexible requirement management, pilot project, provides iterative development, and further maintenance and support of DSM solution. The model allows avoiding numerous specific difficulties, making the process manageable and predictable. Practices from [2] (especially promotional ones) and various existing design techniques, like [16, 17], may be used in the context of the suggested process model.

II. SMALL AND MEDIUM-SIZED SOFTWARE COMPANIES

According to [20], the general problem, hindering adoption of advanced software engineering techniques in small companies, is that their main business-goal is survival, which significantly narrows technological horizons of those organizations and deters elaboration of continuous force motivating any systematical development method. Commonly, the staff consists of a handful of enthusiasts, who are overloaded with day-to-day work. In medium-sized companies the situation is usually better, the enterprises are more resistant and possess extra liberty in technological aspect. They already have a management hierarchy, career prospects, etc. However, it is hard to draw a line between small and medium-sized companies, and therefore, we shall consider them together.

It should be noted that small and medium-sized companies have difficulties in introduction of significant

investments for DSM projects. Such companies regularly lack adequate funds, and are also very dependent on the market: for example, radical shift of priorities is possible at obtainment of a long-term and profitable contract.

We shall also note that small companies frequently lack well-defined development processes. Commonly, there dominates a so-called “heroic” chaos: everyone does everything; successful project completion depends on pure luck; extra efforts from the team members are demanded (overnight work, no weekends). So, it is important that DSM solution does not substitute processes improvement activities and establishment of essential practices like testing, requirement and configuration management, etc. (according to [22] necessity in process improvement usually arises when amount of company staff reaches 12–15 people). These substitutions we have observed at the DSM project described in [12].

As pointed out in [21], typical scenario of a small company existence is development of a specific idea, embodied in one product, which is being modified for various customers. Using an efficient DSM solution is often very perspective in that case, if it allows specifying application’s architecture compactly and demonstrably in diagrams and provided codegeneration of target applications. The DSM approach permits broadening and alteration of original DSM language, code generators, etc., embracing more and more diverse specific features of the domain. Such process, for example, took place in the DSM project described in [11]. Practically, the DSM solution appears as a kernel of the product line infrastructure in this situation. At that disclosed a prospect of radical productivity increasing without involving new staff, which is a very inviting perspective for small companies, which often have difficulties with adding new members to the team (lack of financing, a formed group of enthusiasts resisting to accept newcomers, etc.).

We shall also note, that at small and medium-sized companies personnel’s innovation potential is quite high, people are ready for changes on a much greater scale than staff of larger companies, which, according to [2], is a sufficient positive factor for DSM solutions promotion.

III. DSM- PROCESS PROBLEMS IN SMALL AND MEDIUM-SIZED COMPANIES

The process problems determined during DSM solution development in small and medium-sized companies are presented below.

A. *Difficulties in solution requirements collection*

Models, which users will create by the means of a DSM solution, need to be understandable and easily developed. At [8] problem of assuming that everyone understands the language as its creator have been stated. Creators of DSM-solutions frequently possess above-average qualification and easily employ very abstract models. So modeling languages and tools they created are often inconvenient for ordinary

developers, located in the next room or even at the next desk. In order to overcome this hindrance, pilot projects and iterative development are required.

B. *Who is the customer of a DSM-solution?*

Universally, customer is paying for software and aware of how it is to be used. There is a huge number of approaches for developer/customer cooperation, decreasing risk of creating a needless system. Consequently, development in its majority is controlled by the customer. In case of internal projects, it is often not easy to determine a solution customer. The users are obvious, but who is the customer? In these circumstances we stumble on inexistence of a responsible person and presence of numerous project stakeholders, who have different expectations from the DSM solution. Often those projects provoke “games of politics”, which consequently lead to hard difficulties in development and promotion of DSM solution.

C. *Requirements management problems*

At the beginning of the development process of DSM solution there is a lot of “great” enthusiastic plans, and as stated in [2], sometimes, team and management “loses their heads”: DSM project stakeholders’ sense of reality becomes feeble: users desire unrealizable, developers, opportunistically dream to develop the “right” system architecture, management dreams to apply a DSM solution everywhere. Therefore, it is considered necessary to originate an answer to a question “what do we create?” And this answer should not be significantly changed despite requirements refinement, and other circumstances. The situation, where we create a system for ourselves, is regularly a cause of confusion itself.

D. *Resource management problems*

Commonly, budgets of given internal projects at small and medium-sized software companies are fairly modest – these institutions can’t spend much on inner goals. There is also a lack of competent staff to participate in those undertakings: personnel qualification for such projects implementation must be rather high, however, these personnel are already engaged in production process. At [2] is stated that companies find a way out and hire students and summer interns to development, which has negative impact on a DSM project as these people lack sufficient background in problem domain. Additionally, at small and medium-sized companies (that especially concerns small companies) routine business-priorities often demand staff to divert their attention away from a DSM project. Project financing is frequently suspended, if a company has financial problems: DSM projects’ budgets are the first to be cut. It is also difficult for companies to organize balanced background support activity: mistakes correction, implementation of new features upon users’ requests, etc. Company management often wants to seal the matter of a DSM solution – it has

been developed, it is working, therefore, no additional funding is needed.

E. *Configuration management problems*

Common problems of configuration management, which are described, for instance, at [18], are typical for DSM-projects: lost system source codes, problems with distribution, etc.

F. *Legacy-tendencies*

Created and implemented DSM solutions have a tendency to turn into the legacy systems, which is caused by the lack of proper maintenance, and obsolescence of technologies DSM solution we created upon. Porting a DSM-solution to a new technology often turns out to be an outrageously expensive process. However, for maintenance of a legacy DSM solution staff acquainted with older off-market technologies is required. There are additional difficulties with integration of such solutions with process life cycle support tools.

So, at development and evolution of DSM solutions in small and medium-sized companies as sufficient assistance stands out preliminary designed of development process. This process might be designed, for example, after the management was persuaded to sponsor the project (it is better to design the process before starting). This process should be grounded upon known software development methodologies, adapted for conditions of the given task. As the basic methodology we select MSF.

IV. MICROSOFT SOLUTION FRAMEWORK

MSF (Microsoft Solution Framework) is a software development methodology, created by the Microsoft Company, and is designed for creation of a solution, which would have coordinated delivery of the elements needed (such as technologies, documentation, training, and support) for successful response to a unique customer’s business problem [14]. Special emphasis at the MSF is put on solution integration into the customer’s business, together with its handover to the client’s maintenance and support team. After this submission the development process is considered completed, support is not included at the MSF.

MSF consists of the following parts: process model, team model, project management model, risk management model, readiness management². MSF process model includes both spiral and waterfall models. Following the spiral model there are frequent editions release, iterative functional gain, and gradual development of project documentation. Waterfall model introduces phases and milestones, setting progressive project development pace from the beginning to its

² MSF 4.0 and above also include templates and guidelines for TFS [23], but the paper isn’t focusing on these features.

conclusion. MSF divides development process into the sequence of phases following one another.

- The *envisioning phase* addresses one of the most fundamental requirements for project success—unification of the project team behind a common vision. The team must have a clear vision of what it wants to accomplish for the customer and be able to state it in terms that will motivate the entire team and the customer. Envisioning, by creating a high-level view of the project’s goals and constraints, can serve as an early form of planning; it sets the stage for the more formal planning process that will take place during the project’s planning phase.
- The *planning phase* is when the bulk of the planning for the project is completed. During this phase the team prepares the functional specification, works through the design process, and prepares work plans, cost estimates, and schedules for the various deliverables.
- During the *developing phase* the team accomplishes most of the building of solution components (documentation as well as code). However, some development work may continue at the stabilization phase as well in response to testing.
- The *stabilization phase* conducts testing of a solution whose features are complete. Testing during this phase emphasizes usage and operation under realistic environmental conditions. The team focuses on resolving and triaging (prioritizing) bugs and preparing the solution for release. The stabilizing phase culminates in the release readiness milestone. Once reviewed and approved, the solution is ready for full deployment at live production environment.
- During the *deployment phase*, the team deploys core technology and site components, stabilizes the deployment, transitions the project to operations and support, and obtains final customer approval of the project.

V. HOW WE CHANGE MSF PROCESS MODEL

We have performed the following changes over the MSF process model adjusting it to our needs.

- Initial planning has been simplified: requirements to a DSM solution are hardly determined straightway. Apart from business solutions, level of indeterminacy in this situation is much higher, due to its novelty, amplitude of involved parties, and other risks mentioned earlier. Hence, requirement management and planning are spread over the entire process. Besides, design is being modified, which is a part of a planning phase. On one hand, using such technologies as MetaEdit+, GMF and Microsoft DSL Tools considerably simplify development of the DSM solution architecture. On the other hand, an important design artifact is introduced: the DSM language specification. There are a number of design techniques for this – see [2, 16, 17].

- Process has been made more iterative: after each iteration we evaluate the results received and determine requirements for the next iteration. We also refine requirements of the entire DSM solution.
- Absence of emphasis on testing, documentation and training. On one hand, we save project’s budget and human resources, which is crucial for small and medium-sized companies. On the other hand, users are involved into development process and they also are, for the most part, testers of the solution. The latter is not overly human, but possible in small and medium-sized companies.
- Maintenance and evolution have been included into the process model. In our case is no solution handed over to the customer’s team: DSM solution is maintained by the modified developers team, shifted to the maintenance mode, which is significantly less expensive. Consequently, the entire process is split into *creation* of the solution (figure 1), and its further *maintenance and support* (figure 2). We also introduce *modernization* as reengineering of a DSM-solution. Most common example of modernization is porting of a DSM solution to another implementation technology, IDE, etc. However, this activity must be performed outside of the maintenance and support regime, since it usually takes considerable resources. It is better to organize a special new project for that.

We do not use MSF process iterations and milestones, as in our case they are notably different. Other MSF models we also do not apply. It should be noted, that the DSM solution development process is a lot more «fuzzy», and apart from the original MSF-process, has more variations.

VI. DSM SOLUTION DEVELOPEMENT MODEL

The DSM solution development model is shown at figure 1. As separate boxes inside the phases we have allocated activities, which vary from the ones accepted at the MSF. Elements of standard MSF actions, which we have reused, are omitted at figure 1.

A. *Envisioning phase*

Outlining of the project scope carries crucial importance due to variability of the DSM solution requirements. DSM solution stakeholders must clearly state the answer to a question – what will we do? If project scope is not thoroughly elaborated, than it will often be changed during project course. Numerous and radical scope changes, without proper support, lead to creation of some demo-functionality instead of a good tool. If project scope is altered, then this should be negotiated with all the parties involved, documented, and all of the other necessary actions taken. At this project phase, for scope outlining «proof of concept» technique may be engaged, which was described in [2].

It is necessary to estimate the budget of the DSM project, including costs for maintenance. Numerous DSM projects suffer for the reason that no one ever estimated the budget

carefully beforehand, yet, at later stages it turns out that companies lack funds sufficient for project completion.

At this stage it is also essential to settle on the DSM solution customer. This person is responsible for project financing and sustenance of open benevolent relationship between users and developers. She should also be granted of adequate administrative authority. As a rule, the customer has a casting vote at formation of requirements for the subsequent iteration. It is for the better, if the customer has personal interest in the project, e.g. motivated to use project's success as a career opportunity.

A team of the DSM project should also be created at this stage. It should be considered, that at small and medium-sized companies project members, as a rule, cannot participate in a DSM project in full-time. Commonly, they stay at their work places at different parts of the company office and partially participate in company business projects. DSM project leader should evaluate each potential project member to understand clearly if she really has an opportunity to participate in the project.

B. Planning phase

At this phase implementation technologies for DSM solution development are chosen. These technologies should be perspective from the viewpoint of solution maintenance and support, and also comply with IDEs and other development tools which are used in the company. Herewith basics of the DSM language specification are created. It would be appropriate to use simplified conceptual representations, which are acceptable for discussion with both users and customers.

C. Developing phase

In the frames of this phase, development iterations are performed. Iterative development of the DSM solution is one of the major means to mitigate requirement management risks. Iteration contains the following steps:

- Requirements refinement is carried out together with the users and customer basing on the previous phase results demonstration.
- Planning iteration is commonly held along with requirements refinement.
- Then development/modification of DSM language specification is executed.
- Following the above, generation/regeneration and additional coding of the DSM editor and other DSM tools, which part of DSM solution, are completed. Not all of the DSM solution features are set as DSM language specification properties for automatic generation. Partially, functionality is implemented manually. In order to integrate it into the generated code partial classes are deployed, for example, as in Microsoft DSL Tools, which allow inserting of an own pieces code at the points assigned by the Microsoft DSL Tools architecture. Such pieces of code are introduced into the DSM language specification and automatically inserted into generated code. GMF technology provides a different mechanism: the developer marks manually

changed methods in generated code with the special comments, and the methods are not affected at further regeneration. The first approach's disadvantage is that the Microsoft DSL Tools authors may have not foreseen all of the manual coding demands cases. The second approached is liberated of this defect, nevertheless, it is impossible to propagate changes from DSM language specification to marked methods.

- At last, demonstration of iteration results and feedback collection is performed. It is applied to eliminate risk of a needless functionality development, which would neither comply with users' interests nor management's expectations. Results of the first iterations are only demonstrated, the users perform no practical trials: the solution is still "raw" and users may receive negative impressions. At final iterations, solution developers may create some practical examples from the objective focus field together with the users. Notably, the examples should be proposed by the users themselves. Thoroughly instructed users may perform some operations independently, this issue is also important since they are able provide sufficient feedback. Commonly, up to this point the installation package is not yet ready, therefore, to free users from the deployment problems, all of the experiments should be held on users' computers, where the solution is installed by members of the DSM team. Close cooperation between project team and the solution end users is easily established in small and medium-sized companies due to their minor size.

For iterations organization and management Scrum techniques [19] may be used.

D. Stabilization phase

Main emphasis at this phase is put on the following. Commonly, the DSM solution possesses specific system and integration requirements: third side components, special versions of operation systems, standard technologies, protocols, interfaces, etc. So installation package provide ease way for DSM solution distribution and simplify updates (only uninstall old version and install the new one). Installation packages are especially important in case in if there are many users, and also during long-term solution evolution.

We have also placed a pilot project at this stage. The given activity is as well introduced at the MSF; however, we have highlighted it in our process model due to its major importance. Users try to implement the DSM solution to any non-key production project at last. At that, they frequently find numerous mistakes and are able to formulate the requirements for solution revision. Previous iterative development with demonstrations, stakeholders' discussions, and users testing ensure that no radical misunderstanding with solution redevelopment will not take place at this phase.

According to the pilot project results, improvement solution is held: correction of discovered errors and

inconveniences of user interface, extra functions development, etc.

E. Deployment phase

At this stage the solution is distributed on users' computers. If the additional system service requires, then the network administrators are provided with the necessary information. The user training is also held, if needed. However, we imply users' self-training since there are not too many people in small and medium-sized companies, and some of them were involved in solution development. Nonetheless, presentations, answering to questions and even lectures may be required.

At first, following the deployment, solution supervision is necessary: solution developers should be ready for errors of different kinds, which mainly are complicated and paralyze solution use. In those cases developers should operatively overcome the problem and distribute a new solution version. This activity should be properly organized to ensure good impressions from the solution. If the users would see the DSM team's sincere and willing help, then they will take easier a lot of bags and inconveniences. A solution's good quality costs a lot, and, in many cases, it cannot be achieved with DSM projects at small and medium sized companies. Also quick responds of DSM team are important in order to minimize damage to the production process. Time period when the whole DSM team receives salary for project participation is defined together with the customer and project leader. Following the conclusion upon the solution's quite steady the team is shrunk to the minimum necessary for solution maintenance and support. For small and medium-sized companies is not possible to keep the whole DSM team for a long time.

VII. DSM-SOLUTION MAINTENANCE AND SUPPORT MODEL

In the course of the DSM-solution use, key success factor is the persistently maintenance DSM solution team. The DSM solution should always "live" – errors correction, new functions realization, and, most importantly. This is extremely important for small and medium-sized companies due to budget and staff constrains. For instance, at the DSM project, described in [12], no special attention to this issue was drawn and, as years passed, the DSM solution grew into a legacy system. It was not only upgraded, moreover, the simplest errors were fixed in years. The maintenance team may consist of one person only – the main creator, who obtains additional resources, if required.

The scheme of maintenance and support model is presented in figure 2.

The first step of maintenance and support is establishment of maintenance infrastructure, which includes version control and building management, in case it has not been established earlier. The infrastructure also has to contain update establishing and feedback collecting procedures. It may last for years; therefore, for safe vitality process we need to launch a thoroughly established order.

Further DSM solution life cycle is organized as subsequent editions releases. This approach benefits in

testing and new version distribution, including additional actions such as migration from user models old formats to new ones, when DSM language is being changed.

Feedback collection means collection of errors and users suggestions for DSM solution improvement. Bag tracking systems fit these purposes.

Planning of the new version is necessary when there had been quite many requests or a rather complicated function to be implemented.

Development of planned version, migration of old models (if needed), and distribution of new version to all users follow. Regarding the latter, it should be noted, that it is better if everyone uses the same solution version.

Successful DSM solutions function for years, occasionally – for decades, which prove that solutions stale. For instance, new programming languages come and generators are to be reimplemented. This modernization can take a lot of resources and cannot be performed in the context of maintenance and support. Consequently, modernization should be organized as a new project. Attempts to modernization in the background regularly fail and lead to legacy-tendencies consolidation. At that, modernization project does not contain the risks, the original DSM-solution creation does, and consequently, it should be developed with standard development approaches which used in the company.

VIII. CASE-STUDIES

Ideas of the model presented came to the author during participation in a RTST DSM-project [12, 24]. This project was aimed to create a solution for telecommunication systems product line. A middle-sized Russian company has been developing and maintaining product line of telecommunication systems since 1984 up to now. More than 20 different systems were created. RTST was developed to support this product line. In a project framework we have used visual language SDL [25] as event-oriented skeleton for modeling telecommunication algorithms. Content of the SDL graphic symbols was filled with target language constructions. As a target language Algol 68 was employed. Since the generated system code was restricted of manual modification, developers were able to work with models only. A lot of additional information was attached to the models to make generation of the completed target system code possible. Employment of visual modeling in this project allowed solving a problem of communication between telecommunication engineers and software developers. The architecture of the system was strictly formalized, which simplified production of new product line members. RTST has been created by 7 developers in 4 years. In different time periods from 10 to 20 developers used the solution. RTST is using to support deployed systems now. This project faced problems of continuous support, inner customer, trying to solve project management problems by the means of a DSM solution, and a legacy problem.

The first version of the process model presented in the paper was created and successfully adapted in the context of development and employment of a REAL-IT DSM-solution [11] in 2002 – 2006. This solution was developed in a small

company that was implementing information systems for Saint-Petersburg State University. REAL-IT is oriented on development of data intensive systems with database business logic. These systems are not aimed to support complicated business functionality. The structure of the system's user actions is typical and very simple. Therefore, the corresponding part of user interface also has a typical structure. There are a large number of such systems or those with corresponding subsystems. The code size for such applications is exceptionally large and their development process is an unwanted routine. REAL-IT allowed modeling a scheme of database using UML class diagrams. By the means of dialog-based tools, grounding on fixed types of window forms, system developers created a UML-model of user interface application (class diagrams). Using interface and database scheme specifications (both in class diagrams), special generators produced a target source code. REAL-IT was integrated with UML CASE-tool (IBM Rational Rose) and MS Visual Studio/Visual Basic. REAL-IT has been developed by 2 employees in 1,5 years, and after that 1 person maintained it. There were 7–12 users of the solution at the same time. Totally more than 50 employees used the solution. The main system, developed by means of the REAL-IT, has been deployed by 30 different customers with changes, which were applied by the means of a DSM solution. Up to now REAL-IT is used by a team of 2–3 developers in order to support the systems deployed. REAL-IT has significantly improved the process model and allowed to stabilize it.

The process model as a whole has been applied in the ViDIP DSM project [13]. This solution has been developed for a small company. The company needed some database of hardware elements, which were used for development of a software-hardware broadcast system product line. The company needed a tool to specify a particular configuration of instances of hardware elements for a specific system. We developed a XML database and dialog-based application to fill and modify the database. Basing on Microsoft Visio we have also developed a graphic editor for designing a hardware part of the system using existing hardware elements. The solution has been created by 3 developers in a half year, and now it is employed successfully by the company. The process model has allowed us to concentrate on the development process and user priorities, save time and efforts during users' needs refinement and changing.

IX. CONCLUSIONS

The approach we presented may be further formalized in the framework of complex automation development tools, such as Microsoft TFS especially for maintenance and support. In the latter case DSM solution configuration management, update, distribution, and feedback collecting procedures will be simplified. However, the automation level of the DSM development and evolution process required additional research. It is also possible to integrate various agile-techniques into the process model, and, in particular, as stated above, arrangement of iterations as Scrum sprints.

X. REFERENCES

- [1] J. Tolvanen, S. Kelly, "Defining Domain-Specific Modeling Languages to Automate Product Derivation: Collected Experiences", LNCS, vol. 3714, Springer Berlin / Heidelberg, 2005, pp. 198–209.
- [2] S. Kelly, J. Tolvanen, Domain-specific modeling: enabling full code generation. Wiley-IEEE Computer Society Press, 2008.
- [3] M. Dalgarno, M. Fowler, "UML vs. Domain-Specific Languages", Models & Tools, vol. 16(2), 2008, pp. 2–8.
- [4] J. Greenfield, K. Short, S. Cook, S. Kent. Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Wiley, 2004.
- [5] Eclipse Modeling Project, <http://www.eclipse.org/modeling/>
- [6] Microsoft DSL Tools, <http://msdn.microsoft.com/en-us/library/bb126235.aspx>
- [7] MetaCase MetaEdit+, <http://www.metacase.com/>
- [8] S. Kelly, R. Pohjonen, "Worst Practices for Domain-Specific Modeling", IEEE Software, vol. 26, n. 4, 2009, pp. 22–29.
- [9] J. Greenfield, K. Short, et al., Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. John Wiley & Sons, 2004.
- [10] K. Czarnecki, U. Eisenecker, Generative programming: Methods, Tools, and Applications. Addison-Wesley, 2000.
- [11] A. Ivanov, D. Koznov, "REAL-IT: Model-Based User Interface Development Environment", Proc. IEEE/NASA ISoLA 2005 Workshop, Verification, and Validation, Maryland, USA, pp. 31–41, 23-24 September 2005.
- [12] D. Koznov, M. Katrashov, G. Gagarsky, et al., "Round-trip engineering of reactive systems", Proc. ISoLA, 2004, pp. 343–347.
- [13] D. Koznov, A. Peregudov, D. Buagachenko, et al., "Visual environment for broad cast systems desing and development", System Programming J., vol. 2, Saint-Petersburg State University Publishing, 2006, pp. 142–168. (In Russian).
- [14] Microsoft Solutions Framework, White Paper, MSF Process Model, v. 3.1, 2002.
- [15] S. Robert, S. Gérard, F. Terrier, F. Lagarde, "A Lightweight Approach for Domain-Specific Modeling Languages Design", Proc. EUROMICRO-SEAA, 2009, pp. 155–161.
- [16] A. L. Santos, K. Koskimies, A. Lopes, "Automating the construction of domain-specific modeling languages for object-oriented frameworks", J. of Systems and Software (JSS), 83(7), 2010, pp.1078–1093.
- [17] I. Ráth, A. Ökrös, D. Varró, "Synchronization of abstract and concrete syntax in domain-specific modeling languages - By mapping models and live transformations", Proc. Software and System Modeling (SOSYM), 2010, pp. 453–471.
- [18] W. Humphrey, Managing the Software Process. Addison-Wesley, 1989.
- [19] K. Schwaber, Agile Project Management with Scrum, Microsoft Press, 2004.
- [20] M. Paulk, "Using the Software CMM in Small Organizations", Proc. 16th Annual Pacific_ Northwest Software Quality Conference, 1998, pp. 350–362.
- [21] D. Muthig, J. Bayer, "Helping Small and Medium-Sized Enterprises In Moving Towards Software Product Lines", Proc. Software Product Lines: Economics, Architectures, and Implications Workshop at 22nd International Conference on Software Engineering (ICSE), 2000, pp. 137–140.
- [22] R. V. Horvat, I. Rozman, J. Gyorkos, "Managing the Complexity of SPI in Small Companies", Software process improvement and practice, vol. 5, 2000, pp. 45–54.
- [23] E. Blankenship, M. Woodward, G. Holliday, B. Keller, Professional Team Foundation Server 2010. Wrox, 2011.

- [24] V. V. Parfenov, A. N. Terechov, "RTST: solution for real-time embedded system development". System Informatics, vol. 5, Russian Academic Press, Novosibirsk, 1997, pp. 228–256. (In Russian).
- [25] Specification and Description Language (SDL). ITU-T Recommendation Z.100. 2002.

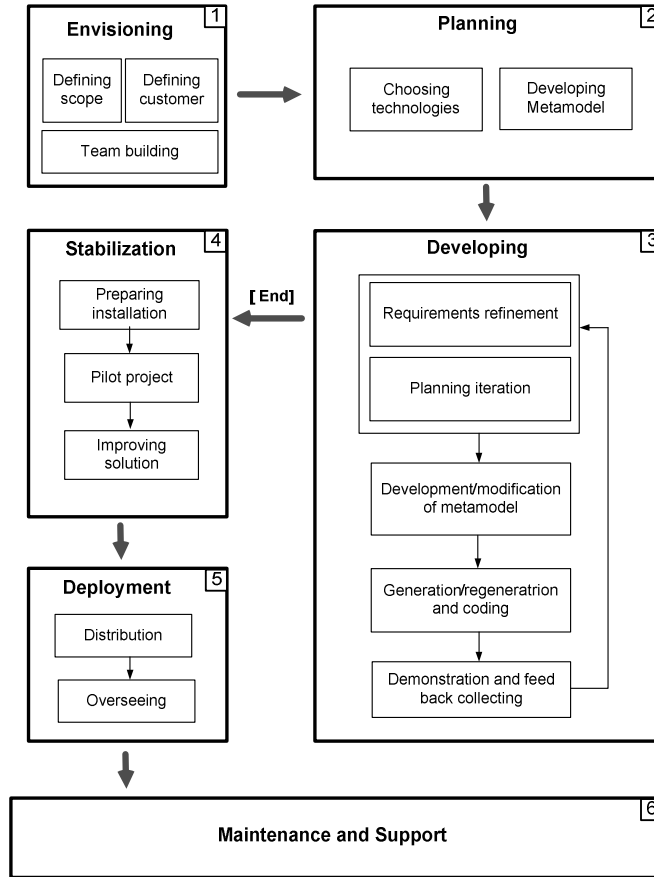


Figure 1. DSM solution development model.

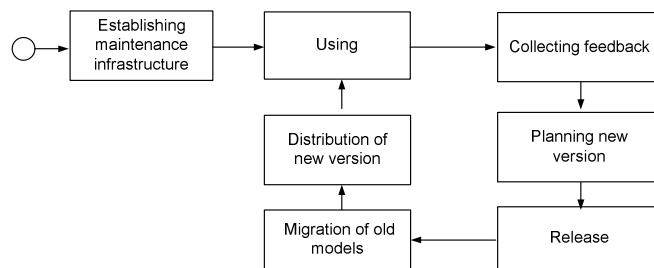


Figure 2. DSM solution maintenance and support model.