Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 143

An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development

Mykola Tkachuk[1], Iryna Martinkus[2] , Rustam Gamzayev[3], and Andrii Tkachuk[4]

Abstract: A domain-driven design (DDD) as a modern approach to improvement of software development quality, especially to support code reuse is considered. To emphasize DDD advantages a 3-level design scheme is proposed which is similar to well-proved 3-level vision about data representation in database development. According to this metaphor the main attention is paid to the phases of logical domain specific modeling (DSM) and to the physical modeling both, with usage of 2 alternative DSM-methods with appropriate CASE-tools: JODA- and ODM approaches respectively. To evaluate their impact on generated code reusability (CR) the software complexity metrics are chosen, and the analytic hierarchy process (AHP) is used to make a final decision about the relationship between CR and DSM.

Keywords: software quality, domain-driven design, domain model method, code reusability, complexity metrics, analytic hierarchy process.

1 Introduction: Research Actuality and Aims

Modern software development is a complex interdisciplinary process, which consists of several interconnected phases, which are quite expensive for large software system projects. In general all traditional and modern software methodologies are supposed to decrease these project costs taking into account some functional and non-functional requirements (or quality attributes) to be met in a target system (see e.g. [So11]). One of the most effective ways to resolve this problem is a reusing of different project solutions (assets): domain knowledge, requirements specifications, software architectures, and finally programming code. This approach is the basis of advanced concepts of software engineering as the creation of software products lines and software factories [GS04], as well as methods of software variability management [CBK13].

Nowadays a Domain-Driven Design (DDD) in considered as a recognized methodology to build a complex software in different application areas with respect to this important challenge: to provide a high level of assets reusability in a given project [Ev03], [AF07], [Re13]. Although main essential advantages and some limitations of DDD are already

[1] NTU "KhPI", SEMIT Dept, Frunze str., 21, 61002 Kharkiv, Ukraine, tka@kpi.kharkov.ua
[2] NTU "KhPI", SEMIT Dept, Frunze str., 21, 61002 Kharkiv, Ukraine, imartinkus@gmail.com
[3] NTU "KhPI", SEMIT Dept, Frunze str., 21, 61002 Kharkiv, Ukraine, rustam.gamzayev@gmail.com
[3] NTU "KhPI", SEMIT Dept, Frunze str., 21, 61002 Kharkiv, Ukraine, tkachuk.andrey.polt@gmail.com

discussed intensively in many recent publications, from our point of view the positive core of DDD-methodology can be emphasized once again, if we draw an analogy between DDD-approach to software applications and well-known 3-level vision about data representation in database development [BCN92] (see Fig. 1).

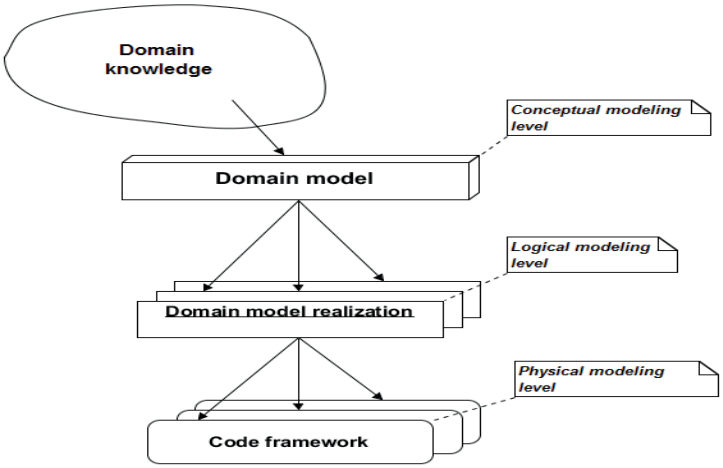

Fig. 1: 3-level scheme in DDD approach

It is to note that according to this vision about DDD-approach for the one and the same domain model (at the conceptual modeling level) a lot of different its realizations (at the logical modeling level) can be constructed, and for each of them an appropriated code framework might be generated finally (at the physical modeling level) using some CASE-tools.

Taking into account the scheme given in Fig. 1 and understanding the fact that there are a lot of different domain-specific modeling (DSM) methods and appropriate CASE-tools, the goal of this paper is to analyze some key capabilities and features of different DSM methods and tools. In this way we are going to propose an integrated approach to their evaluation with respect to prediction of source code reusability and, finally, to improvement of maintainability in a target software system.

2 Domain-specific Modeling and its Impact on Code Reusability. Related Work

Even a briefly overview of recent publications dedicated to code reusability (CR) issues shows that finding a relationship between CR and different impact factors in software development is not a straight forward process. E.g. in [An13] some common reuse design principles are considered, including technological, project's managerial, and even human factors (like experience of developers in different programming languages, etc.).

In [Ta14] the results of the empirical study on CR-trends in open-source software are presented, and a comprehensive collection of reusability metrics are proposed to define the most significant CR-factors. In [Na16] the research is focused on the constructing relationship between extent of CR and values of software complexity (SC) metrics in object-oriented programming. Especially, the following common SC-metrics are used for this purpose

- Depth of Inheritance Tree (DIT)
- Reponses for a Class (RFC)
- Number of Children (NOC)
- Coupling between Object Classes (CBO).
- Weighted Methods per Class (WMC)

Based on experimental studies a set of empirical hypotheses (EH) had been proposed regarding the impact of SC-metrics on CR, namely [Na16]:

- EH1: Better CR can be obtained by moderate value of DIT in every class.
- EH 2: The complexity in code design and CR decreases for maximized values of RFC.
- EH 3: CR degrades for increasing values of NOC related to given class.
- EH 4: An increasing of CBO values does not have much impact over CR
- EH 5: CR declines for increasing values of WMC in every class.

It is necessary to note that besides such empirical assertions like (EH1)-(EH5) concerning relationships between single SC-metrics and an extent of CR, in the most part of publications dedicated to these issues there are no more or less motivated suggestions about the quantitative estimation of correlation between SC metrics and extent of CR. Moreover even in the recent studies mentioned above the correlation between domain modeling methods and complexity of generated code is not analyzed. Hence, it is crucial to identify this correlation because in this way we can make decision about the choosing of an appropriate DSM methods and tools in order to reduce in this way the target implementation costs in DDD-oriented software projects.

There is another important aspect of this investigation. According to [IS16], the 6 main software quality attributes exist, namely: *Functionality, Reliability, Usability, Efficiency, Maintainability and Portability*, and each of them have some sub-characteristics. In case of such quality attribute as *Maintainability* the appropriate list of characteristics includes: *analyzability, changeability, stability,* and *testability*, where *changeability* characterizes an amount of efforts needed to change a software system to be maintained. From the other hand, in some publications (see e.g. in [AM13]), a correlation between changeability and software complexity metrics is emphasized, especially the CBO and WMC metrics are inversely proportional to the level of changeability in the given software system. From this point of view it can be concluded, that the complexity estimation of code generated basing on DSM finally allows us to make conclusion about such important software quality attribute as its *Maintainability*.

3 Some Domain Modeling Methods and Tools: Short Overwie

3.1 Domain Analysis and DSM methods

During last 10-15 years a lot of different domain analysis (DA) and DSM methods [Fe99], [KT08] were developed. Despite of their differences from the implementation point of view, the most suitable way to classify DA and DSM methods is consider them by type of phases / artifacts to be reused in a software development process. Based on this suggestion the following list of these methods should be considered:

1. DA&DSM methods for software product reuse;
2. DA&DSM methods for software process reuse;
3. DA&DSM methods for software technology reuse;
4. DA&DSM methods for software experience reuse.

Taking into account our main research goal: to identify a relationship between different DSM methods and code reusability (CR) in target application, we further consider more detailed the methods from the group (1). Therefore 2 such methods for software product reuse, namely JODA and ODM [Fe99], were chosen and they are presented briefly below.

Method JODA (Joint integrated avionics Object oriented Domain Analysis) uses object-oriented approach to cover the domain analysis phase, and it includes the following processes:

- Domain data preparation: identification and gathering of appropriate data sources, references and software artifacts which are relevant for a given domain.
- Domain scope definition: elaboration of diagrams for higher-level entities, identify of generalization-specialization, aggregation and other relations within domain, build a domain glossary.
- Domain modeling: identification, definition and modeling several domain scenarios in order to group domain-specific objects and activities to represent them in next domain engineering process.

Method ODM (Organizational Domain Modeling) supports systematically mapping of domain-specific artifacts into reusable assets that can be reused in future software development activities. This approach includes the following phases:

- Plan domain engineering: this one is focused on understanding of stakeholders and defining of domain analysis scope.
- Domain modeling: it concerns collecting and documenting the domain-specific information resources which are relevant for future reusing.

- Domain assets base: the final phase of ODM method that supposes defining the project scope, creating (choosing) system architectures and implementing of physical asset base for the given domain.

In order to support all main phases / activities in any DA&DSM method the appropriate CASE-tool has to be used, and a short overview of them is given in the next paragraph.

3.2 Domain Modeling CASE-tools

Generally, visual modeling tools in software engineering have evolved a lot in recent years. One of the new trends in this domain is the transition from unified modeling environments like UML or SysML [Om10], to some domain-specific modeling (DSM) languages and tools, e.g. WebML, SoaML, and some others [Re13]. These DSM - approaches allow developers to design and to analyze software in terms of target problem domain, and finally to generate source code in different programming languages based on high-level requirements specifications.

It is to mention that exiting CASE-tools for DSM are quite varied in their capabilities, e.g. such wide-used CASE-tools such as: Eclipse Modeling Framework, Rational Rose, FeatureIDE,Visual Paradigm, Actifsource and others [Re13]. To compare them it is necessary to choose a set of criteria, and obviously there are a lot of different ways to define such criteria configurations. Generic enough, and in the same time, a practice-oriented one is the following list of criterion: a possibility to generate code by domain model, a possibility to build model by code, and last but not least: a necessity to have a mandatory license. Taking into account these criteria, exactly Actifsource and Eclipse Modeling Framework (EMF) were chosen for our future research. Both of these CASE-tools are license-free and they support JODA and ODM methods correspondingly.

4 An Integrated Evaluation Approach and Case Study Results

Based on already mentioned points concerning some relationships existing between extent of code reusability (CR) and values of software complexity (SC) metrics we purpose to evaluate the selected DMS methods: JODA and ODM with respect to the CR of the generated programming code. Moreover this approach has to produce an integrated estimation values for extent of CR in target DSM – based applications taking into account an impact of the weighted SC-metrics collection (see in Section 2). To perform this evaluation approach we propose the information technology which is presented in Fig. 2 using IDEF0 notation [ID16]. It consists of 3 fictional Blocks: “A1: DM construction and code generation”, “A2: Code analysis”, “A3: OOP Code Reusability estimation”. Block A1 operates with User stories, obtained from Domain expert and the result of Block A1 is a Domain model and generated source code. After this operation it is possible to perform Code analysis (see Block A2) in order to calculate OOP code complexity metrics (SC-metrics). Finally, Block A3 performs OOP Code reusability estimation and calculates the integrated value of CR-extent.

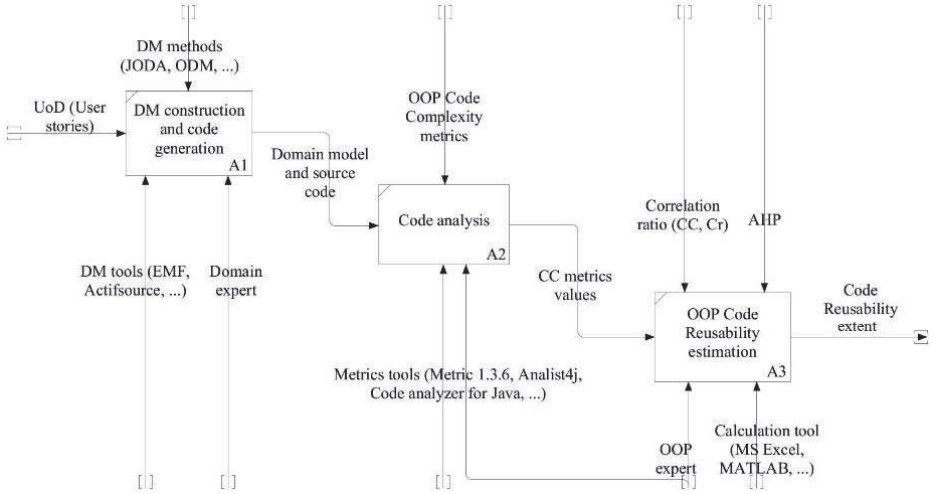

Fig. 2: Information technology to support the evaluation approach

It is necessary to note in our approach to estimate DSM methods we consider the data modeling level only, because at the initial stage of software development, where these methods are used, in fact for the most part of domain entities their methods and interfaces cannot be defined as usually.

According to the common DDD-framework shown in Fig. 1, to apply any DSM method first it is necessary to obtain initial data about target problem domain. One of such way can be representation of problem domain by a collection of user stories, especially if such agile methodologies as Scrum or XP (Extreme Programming) are used in project developer teams [Am16]. This is a high-level definition of system requirements, containing just enough information so that the developers can produce a previous estimate of the effort to implement them. As the case-study the user stories for the simple domain "Students data management in the educational process" ware formulated, and they are shown in Table 1.

Iteration	User Stories
1	Maintain student's personal information Maintain student's contact information (address)
2	Define main features of student Identify of student's residence
3	Maintain student's information about his educational career Definition of student's charge

Tab. 1: User stories description

The next step to be done in the proposed evaluation approach is their implementation with usage of 2 chosen DSM methods: JODA and ODM.

4.1 ODM and JODA Implementation

To apply ODM-method *Eclipse Modeling Framework* was used [EMF16]. First we build the logical domain model (see Fig. 3), which represents the initial user stories given in Tab.1.

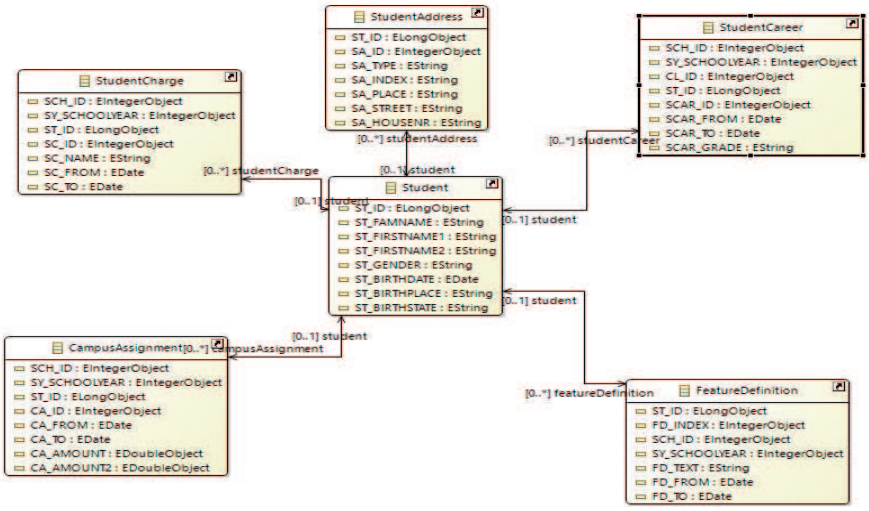

Fig. 3: The domain model in ODM / EMF notation

The next step is to generate Java-source code by this domain model (see Fig. 4).

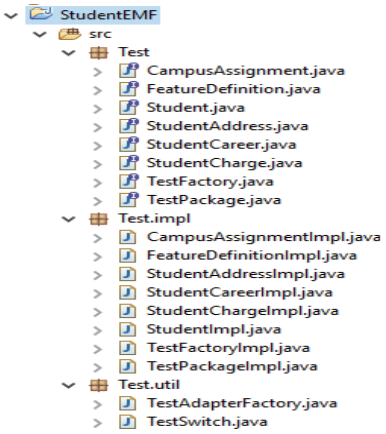

Fig. 4: The package tree generated in EMF CASE-tool

All generated in EMF Java-classes are divided into three packages: *Test* (it includes interfaces), *Test.impl* (contains classes, interfaces, implementation), *Test.util* (utility

classes). The fragment of generated code from one package only is shown below.

```
        /* <!-- begin-user-doc -->
        * <!-- end-user-doc -->
        * @generated
        */
       public Long getST_ID() {
             return sT_ID;
       }
       /**
        * <!-- begin-user-doc -->
        * <!-- end-user-doc -->
        * @generated
        */
  public void setST_ID(Long newST_ID) {
     Long oldST_ID = sT_ID;
     sT_ID = newST_ID;
     if (eNotificationRequired())
      eNotify(new ENotificationImpl(this,
             Notification.SET,
             TestPackage.STUDENT__ST_ID,
             oldST_ID,
             sT_ID));
      }
```

For JODA-implementation the Actifsource CASE-tool was used [Ac16], and the elaborated domain model is shown in Fig.5.

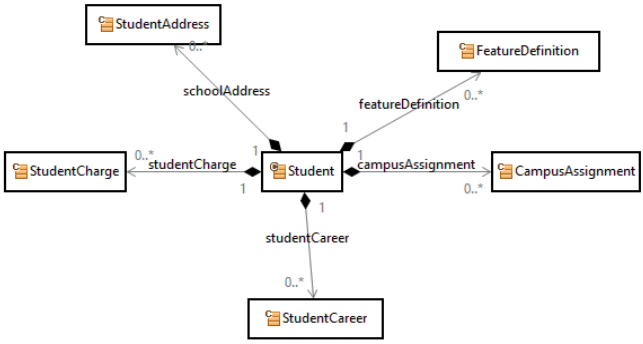

Fig. 5: The domain model in JODA notation

Similar to the previous approach the next step supposes a code generation by this model, and the generated project's package tree is shown in Fig. 6.

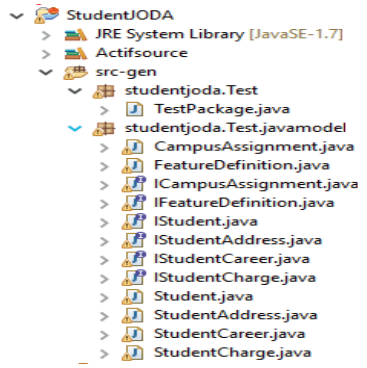

Fig. 6: The package tree generated in Actifsource CASE - tool

The fragment of generated Java-code from one package is shown below.

```
@Override
public java.lang.String selectST_ID() {
   return_getSingleAttribute(java.lang.String.class,
studentjoda.Test.TestPackage.Student_ST_aE_ID);
  }

  public void setST_ID(java.lang.String sT_ID) {

_setSingleAttribute(studentjoda.Test.TestPackage.Student_ST_
aE_ID, sT_ID);
  }
```

The Actifsource-tool creates for the same domain model 2 packages, namely *studentjoda.Test* (contains additional resources) and *studentjoda.Test.javamodel* (includs implementation of classes and interfaces).

The main difference to operate with these facilities is the followings: in Actifsource-tool a source code is generated automatically in case of any changes acquired in the model. By turn, in EMF-tool to generate new code version we need to create a special model-generator utility (see in [EMF16] for more details).

4.2 AHP-based Calculation Scheme and Case Study Results

As already mentioned above, in [Na16] the results of experimental studies are presented, which show the relationship between values of single SC-metrics and the given extent of CR. The fragment of these numerical data is shown in Table 2, where column Value includes the values of SC-metrics, *AVG (Cr)* shows the average values of the CR-extent.

Since our goal is to calculate the integrated value of CR-extent with respect to all

available SC-metrics, we need to define an appropriate coefficient for each such metric. It can be done using the Analytic Hierarchy Process (AHP) method [Sa00].

Metric	Value	AVG (Cr)	Cr/Value
DIT	1	10.46	
	2	25.21	
	…….	…….	
	6	99.11	16,60
RFC	5	21.21	
	10	19.44	
	…….	……..	
	200	95.34	0,46
NOC	0	0.33	
	1	66.67	
	……..	……	
	6	258.33	43,30
CBO	1	22.22	
	3	21.11	
	……..	…….	
	24	91.73	3,79
WMC	3	40.38	
	5	30.00	
	……..	……	
	100	105.86	1,05

Tab. 2: CR estimation according to single SC metrics (see in [Na16])

Based on the experimental data presented in [Na16] we can calculate the relative "weight" of one “unit value” of each metric, i.e. the ratio (Cr/Value). It can be concluded, that e.g. the increasing of DIT-metric value to 1 makes CR- index higher, than increasing to 1 of CBO - metric, etc. In this way we are able to determine the relative importance of one metric to others. According to AHP –approach the following assessments scale is used: 1 – equal importance, 3 - moderate importance, 5 - a strong importance, 7 - very strong importance, 9 - extreme importance, and we get the pairwise comparisons for single CR-metrics shown in Table 3.

	WMC	RFC	DIT	NOC	CBO
WMC	3\3	3\1	3\7	3\9	3\5
RFC	1\3	1\1	1\7	1\9	1\5
DIT	7\3	7\1	7\7	7\9	7\5
NOC	9\3	9\1	9\7	9\9	9\5
CBO	5\3	5\1	5\7	5\9	5\5

Tab. 3: Pairwise comparisons

The final AHP estimation values and the weighted coefficients (K) for all CR-metrics are presented in Table 4.

	WMC	RFC	DIT	NOC	CBO	SumRow	K
WMC	1	3	0.42	0.33	0.6	5.35	0,1198
RFC	0.33	1	0.14	0.11	0.2	1.78	0,0398
DIT	2.33	7	1	0.78	1.4	12.51	0,2801
NOC	3	9	1.29	1	1.8	16.09	0,3603
CBO	1.67	5	0.71	0.55	1	8.93	0,2000
Total						44.66	1,0000

Tab. 4: Final AHP estimation values

Thus applying these weighted coefficients to corresponded SC-metrics, we obtain the following formula for integrated value of the CR-extent

$$CR_{extent} = 0{,}1198 * WMC + 0{,}0398 * RFC + 0{,}2801 * DIT + 0{,}3603 * NOC + 0{,}2000 * CBO \quad (1)$$

To prove experimentally this approach and to perform the appropriate calculation such tools as Analyst4j, Metrics 1.3.6 plugin for Eclipse, OOMeter, Eclipse Metrics Plug-in 3.4, and some others can be used [LLL08].These facilities allow not only to calculate average value of each SC-metrics (see Table 5), but also depict them graphically.

Matrices	EMF	Actifsource
WMC	55.00	15.54
RFC	36.70	16,53
DIT	2.9	1.32
NOC	0.44	0.46
CBO	7.80	6.95

Tab. 5: Results of the software metrics' calculation

These values of all CR-metrics for EMF and Actifsource implementations are shown

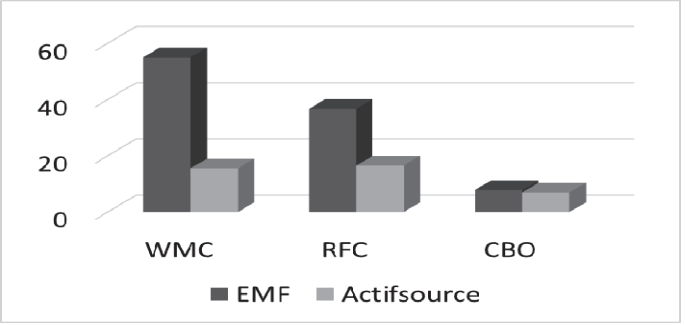

Fig. 7: Results comparition for WMC, RFC and CBO metrics

in Fig. 7 and Fig. 8 respectively.

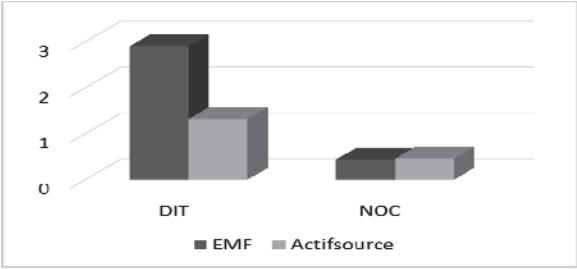

Fig. 8: Results comparition for DIT and NOC metrics

To calculate the integrated estimation value of CR-extent according to formula (1) the special software tool is developed, and the final result of this DSM methods evaluation is shown in Fig. 9.

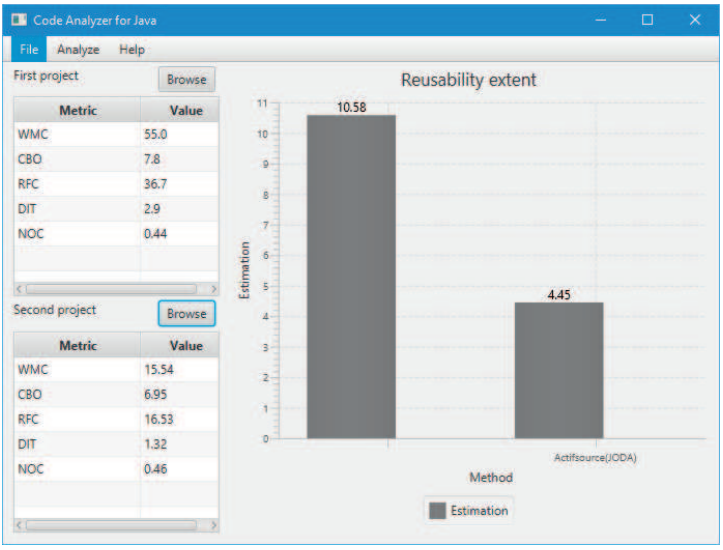

Fig. 9: The prototype main window

The final integrated estimation values for the CR-extent by usage of the chosen DSM-methods: JODA and ODM are the following

$$CR_{extent}(EMF) = 10{,}58 \tag{2}$$

$$CR_{extent}(\text{Actifsource}) = 4{,}45 \tag{3}$$

Therefore, according to the values in formulas (2) - (3) we can conclude that domain modeling by usage of ODM / EMF tools provides the essential higher extent of code reusability (CR) than by using of JODA method and Actifsource tools.

5 Conclusions and Future Work

In this paper we have considered some essential aspects of domain-driven design (DDD) methodology in modern software engineering. A special attention is paid to comparative analysis of different domain-specific modeling (DSM) methods, and to the appropriate CASE-tools for their implementation. As an important factor for their evaluation the extent of code reusability (CR) is chosen, and the relationship between extent of CR and values of object-oriented software complexity (SC) metrics is emphasized.

Based on this suggestion the integrated approach to evaluation of different DSM / CASE-tools configurations is proposed, which defines the final CR-extent taking into account weighted coefficients for single SC-metrics to be calculated using Analytic Hierarchy Process method. This approach is tested within the elaborated case study domain model, and it allows make decision about the usability of given DSM-methods and tools for reducing of implementation costs in DDD-oriented software projects based on predicted extent of code reusability. Finally, this result can also be considered as a way to improve such important quality characteristic as *Maintainability* for a target software application to be developed using an appropriate DSM-method.

In future we are going to construct the more sophisticated collection of the software complexity metrics, e.g. with metrics of relationships between packages, and to improve the prototype of our software tool to support the proposed evaluation approach.

References

[Ac16] Actifsource . http://www.actifsource.com, accessed on: 15.06.2016

[AF07] Abel, A.; Floyd, M.: Domain Driven Design Quickly. Lulu.com, 2007.

[Am16] Ambler, S.W.:Agile/Evolutionary Data Modeling: From Domain Modeling to Physical Modeling., http://agiledata.org/essays/agileDataModeling.html#DisasterStrikes, accessed on: 15.06.2016.

[AM13] Ayalew, Y., Mguni, K.: An Assessment of Changeability of Open Source Software, Computer and Information Science., Vol. 6, No. 3; 2013.

[An13] Anguswamy, R.: A Study of Factors Affecting the Design and Use of Reusable Components, Software Reuse Lab, Virginia Tech, 2013.

[Ba06] Balmelli, L. et.al: Model-driven Systems Development. IBM Systems Journal. Vol. 45. P. 569-585, 2006.

[BCN92] Batini, C., Ceri, S., Navathe, Sh.: Conceptual Database Design: An Entity-Relationship Approach. – Benjamin Publishing Company, 1992.

[CBK13] Capilla,R., Bosch, J,. Kang, K.,: Systems and Software Variability Management, Springer, 2013.

[EMF16] Eclipse Modeling Framework (EMF). https://eclipse.org/modeling/emf/, accessed on: 15.06.2016.

[Ev03] Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software 1st Edition. Prentice Hall, 2003.

[Fe99] Ferré, X.: An Evaluation of Domain Analysis Methods.,In 4th CAiSE.IFIP8.1 International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design, P.1-13, 1999.

[GS04] Greenfield, J.: Short K. Software Factories: Assembling Application with Patterns, Models, Frameworks and Tools, Wiley:-Indianapolis, 2004.

[ID16] Official Web-site of IDEF Family of Methods, http://www.idef.com, accessed on 15.06.2016

[IS16] ISO 9126 Software Quality Characteristics http://www.sqa.net/iso9126.html, accessed on: 15.05.2016.

[KT08] Kelly, S., Tolvanen, J.: Domain-Specific Modeling: Enabling Full Code Generation. Wiley Computer Society Press. 2008

[LLL08] Lincke, R., Lundberg, J., Löwe, W.: Comparing Software Metrics Tools, ISSTA '08 Proceedings of the 2008 international symposium on Software testing and analysis, P.131-142, 2008.

[Na16] Nandakumar, A.N.: Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design, International Journal of Advanced Computer Science and Applications, Vol. 7, No. 2, 2016.

[Om10] OMG Unified Modeling Language, Superstructure. Version 2.3. OMG, 2010.

[Re13] Reinhartz-Berger, I. et al., eds. Domain Engineering: Product Lines, Languages, and Conceptual Models. Heidelberg, Springer, 2013.

[Sa00] Saaty, T.,L.: Fundamentals of the Analytic Hierarchy Process. RWS Publications, 4922 Ellsworth Avenue, Pittsburgh, PA 15413, 2000.

[So11] Sommerville, I.: Software Engineering. Addison Wesley, 2011.

[Ta14] Taibi, F.: Empirical Analysis of the Reusability of Object-Oriented Program Code in Open-Source Software, International Journal of Computer, Electrical, Automation, Control and Information Engineering, Vol.8, No.1, 2014.