# Formalization of the software module development using matrix forms

Igor V. Kovalev [1,2,3,4,5], Vasiliy V. Losev [1], Mikhail V. Saramud [1,2], Petr A. Kuznetsov [1] and Alexandra S. Lifar [1]

[1] *Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, 660037, Russian Federation*
[2] *Siberian federal university, 79 Svobodny pr., Krasnoyarsk, 660041, Russian Federation*
[3] *Federal State Budgetary Educational Institution of Higher Education "Krasnoyarsk State Agrarian University", 90, Mira Av., Krasnoyarsk, 660049, Russian Federation*
[4] *Krasnoyarsk Science and Technology City Hall of the Russian Union of Scientific and Engineering Associations, 61, Uritskogo street, Krasnoyarsk, 660049, Russian Federation*
[5] *China Aviation Industry General Aircraft Zhejiang Institute Co., Ltd, China*

**Abstract**

The article considers the formalized approach to develop a multi-user document exchange system based on a matrix form of data representation. The multi-stage development of modular software by several participants is presented as basic process. The algorithmic implementation of information flows in the context of software development is presented. The document life cycle model is described using the UML diagrams. The considered algorithm contains a hierarchical document flow structure with the subordination of some documents to others. According document flow the participant interaction model of software development process is considered. The model of administrative document flow includes the following roles (owners): the product development manager; the customer representative; the developer; the tester. As a result, the presented matrices fully describe the graph of the considered document flow process. The formalized approach based on the matrix form of representation can be used as input data for multi-user systems' algorithmization in solving modular software design problems.

**Keywords**

Matrix form, document, module

## 1. Introduction

Nowadays, many software design methods (software) are developed and unified approaches for different categories are established [1]. One of the features of modular design is the possibility of using formalized approaches. Let us consider a similar approach the matrix form of representation of multi-user systems [2], such as document exchange. The representation of information flows passing through the document management system (DMS) is formalized in the form of a visual graph model [3].

## 2. The matrix form of data presentation

As a basic, consider the process of software module developing with the following tasks: module developing, its testing and accepting as an element of common software package [4]. Moreover,

III International Workshop on Modeling, Information Processing and Computing (MIP: Computing-2021), May 28, 2021, Krasnoyarsk, Russia
EMAIL: kovalev.fsu@mail.ru (Igor Kovalev); basilos@mail.ru (Vasiliy Losev); msaramud@gmail.com (Mikhail Saramud); forubox@yandex.ru (Petr Kuznetsov); alifar15@mail.ru (Alexandra Lifar)
ORCID: 0000-0003-2128-6661 (Igor Kovalev); 0000-0002-1996-2889 (Vasiliy Losev); 0000-0003-0344-9842 (Mikhail Saramud); 0000-0003-3950-8760 (Petr Kuznetsov); 0000-0002-4158-1513 (Alexandra Lifar)

checking the accepted task adequacy and verification is carried out as well as the developed software module is archived at the final stage of development.

Supposed the forms $F_1$...$F_8$ are documents used in simulated process: F1 – development assignment form; $F_2$ - form of the task approval document; $F_3$ - assignment to the developer; $F_4$ - development report; $F_5$ - test report; $F_6$ - notice of test results; $F_7$ - notice of development results (claim letter); $F_8$ - acceptance certificate.

Supposed the actions $A_1$...$A_4$ are procedures performed on the documents for changing states in the simulated process: $A_{1.1...1.7}$ – document creation; $A_2$ - document approval; $A_{3.1...3.2}$ – document sending; $A_4$ –archiving.

Supposed the performers $P_1$…$P_4$ are performers of action in simulated process: $P_1$ – product development manager; $P_2$ - customer representative; $P_3$ – developer; $P_4$ - software tester (Figure 1:).

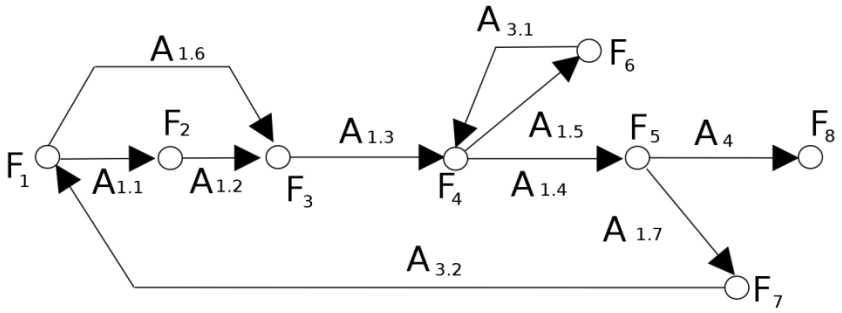


**Figure 1:** The graphic of software development process

In the framework of specified paths there are three document flow scenarios. According to three document flow scenarios let us build a document flow matrixes. Performers in different scenarios are highlighted differently. At the first step the product development manager issues a development task and the customer's representative approves it.

**Table 1**

The document flow matrix. Step 1

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|---|---|---|---|---|---|---|---|---|
| $A_{1.1...1.7}$ | $P_1$ | | | | | | | |
| $A_2$ | | $P_2$ | | | | | | |
| $A_{3.1...3.2}$ | | | | | | | | |
| $A_4$ | | | | | | | | |

The product development manager issues a development order to the developer. Further, the developer issues a report based on his work results.

**Table 2**

The document flow matrix. Step 2

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|---|---|---|---|---|---|---|---|---|
| $A_{1.1...1.7}$ | | | $P_1$ | $P_3$ | | | | |
| $A_2$ | | | | | | | | |
| $A_{3.1...3.2}$ | | | | | | | | |
| $A_4$ | | | | | | | | |

The developer sends a development report to the tester. Depending on the development results tester he issues the test report with possible notices and forward it.

**Table 3**
The document flow matrix. Step 3

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|---|---|---|---|---|---|---|---|---|
| $A_{1.1...1.7}$ | | | | | $P_4$ | $P_4$ | | |
| $A_2$ | | | | | | | | |
| $A_{3.1...3.2}$ | | | | | $P_4$ | $P_4$ | | |
| $A_4$ | | | | | | | | |

According to the first scenario, the report is approved by the product development manager. In the second scenario, the developer issues a new development report with adjustments (and the previous step is repeated). The second scenario is bold italic highlighted in the matrix.

**Table 4**
The document flow matrix. Step 4

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|---|---|---|---|---|---|---|---|---|
| $A_{1.1...1.7}$ | | | | $P_3$ | | | | |
| $A_2$ | | | | | $P_1$ | | | |
| $A_{3.1...3.2}$ | | | | $P_3$ | | | | |
| $A_4$ | | | | | | | | |

If the customer's requirements are satisfied, the customer accepts the development results and the product development manager archives its. If there are any notices (the third scenario begins) the customer forms claim letter and sends it to the product development manager.

**Table 5**
The document flow matrix. Step 5

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|---|---|---|---|---|---|---|---|---|
| $A_{1.1...1.7}$ | | | | | | | $\underline{P_2}$ | |
| $A_2$ | | | | | | | | $P_2$ |
| $A_{3.1...3.2}$ | | | | | | | $\underline{P_2}$ | |
| $A_4$ | | | | | | | | |

In the case of the first scenario, the project is archived. In the case of the third scenario, the product development manager produces an updated development task and sends it to the developer (then the third step is repeated). The third scenario is underscore highlighted.

**Table 6**
The document flow matrix. Step 6

| | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|---|---|---|---|---|---|---|---|---|
| $A_{1.1...1.7}$ | | | $\underline{P_1}$ | | | | | |
| $A_2$ | | | | | | | | |
| $A_{3.1...3.2}$ | | | $\underline{P_1}$ | | | | | |
| $A_4$ | | | | $P_1$ | | | | |

## 3. Document lifecycle model in document management system

Based on the obtained scenarios, let us consider an algorithmic implementation of the information flows in the context of software development. The algorithmic implementation is presented in the form of UML diagrams (Figure 2:).

According to this model, software development includes the following:

- Setting the task for software development. A task for software development is formed.

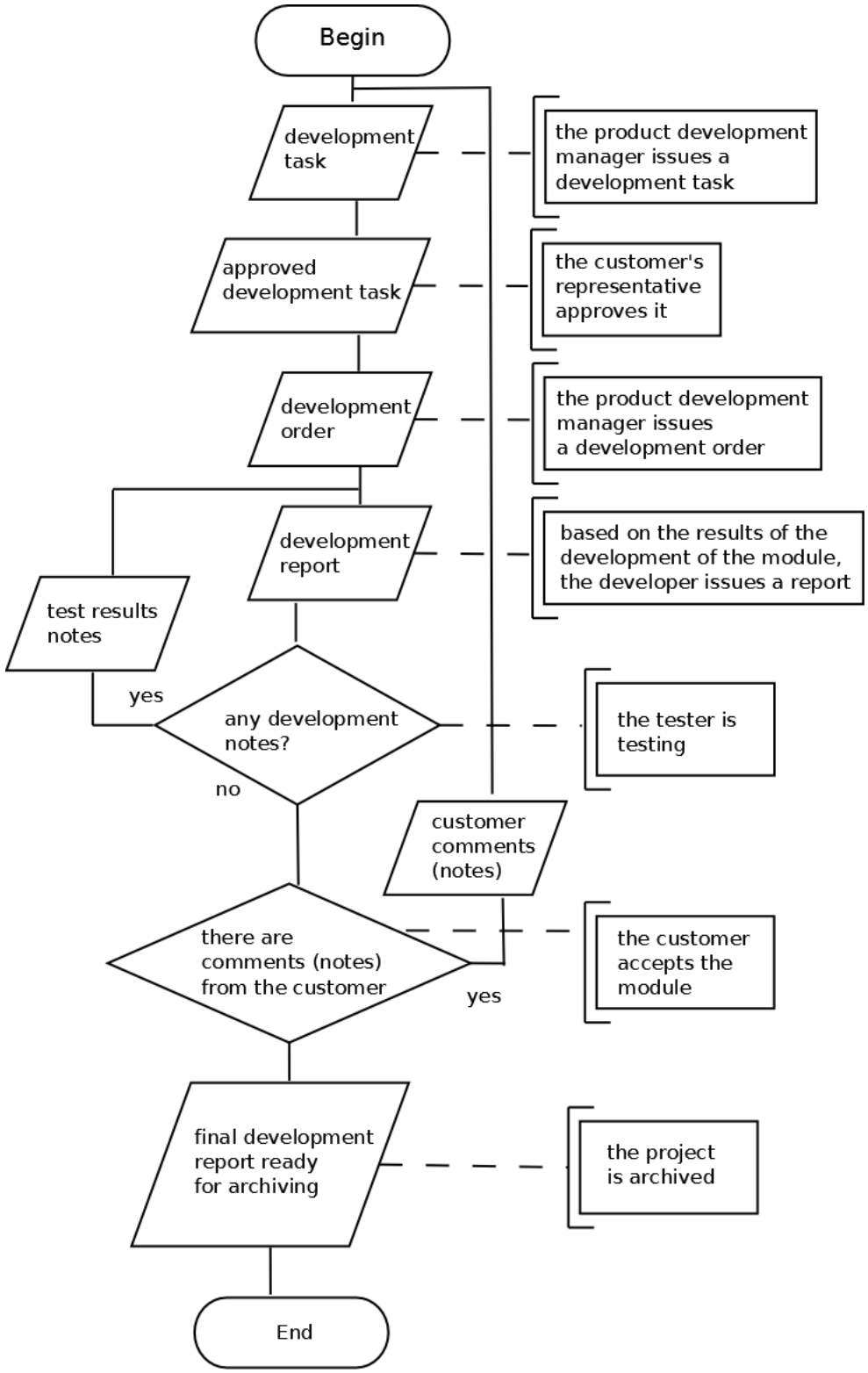


**Figure 2:** The document lifecycle model in the document management system in the form of a UML diagram

- The development mark is entered into the project documentation registration log. The service note for the specific module development is drawn up.
- After drawing up the development assignment it is agreed with the customer.

- The development task is sent for correction in case the customer has any corrections.
- After (possible) adjustment, the task is passed to the developer to create a software module.
- The developer creates the module algorithms and issues the development report based on work results.
- The source modules text is added to the set of documents.
- The developed software module is sent for testing and corresponding record is made in the documentation/
- If the test is successful, the module is sent to the customer verification. If not, it is submitted to the developer for correction.
- The final development report is generated in a form suitable for the archiving in case the customer has no comments. If there are any comments the task is sent for re-approval with the product development manager.

The considered algorithm contains a hierarchical document flow structure with the subordination of some documents to others [5]. Documents based on orders, administrative notes or instructions retain their link. Also, this algorithm assumes the new document release notification as well as the ability to track the development process (journal) for persons involved in the development process. In general, the layout is consistent with the ideas of building a document management system.

According document flow the participant interaction model of software development process is considered (Figure 3:).

The model of administrative document flow includes the following roles (owners): the product development manager; customer representative; developer; tester.

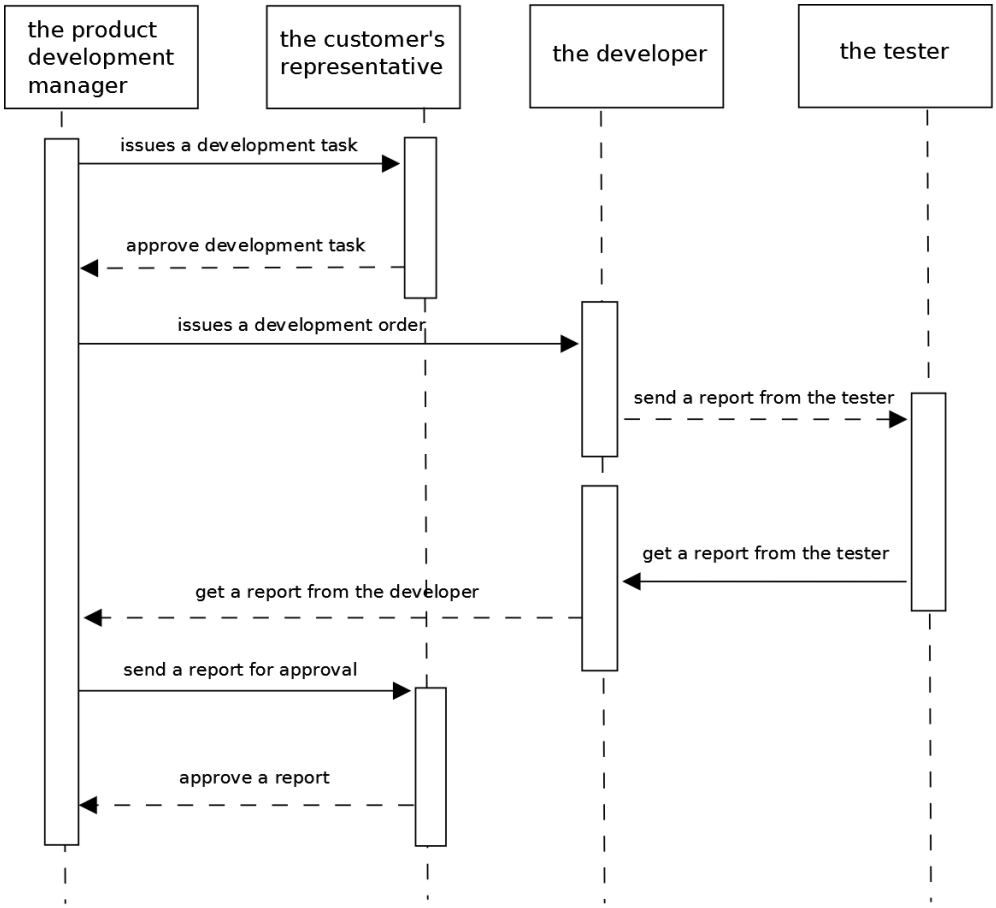


**Figure 3:** The participant interaction model in the software development process

The product development manager is responsible for the formation of main development assignment agreed with customer representative. Firstly, the product development manager carries out approval at the customer representative taking into account software comments and requirements. Responsible

performers are assigned for certain tasks. After agreement with the customer representative, the product development manager gives the development assignment to programmer.

During the development the process is controlled by organizing a time control system. Programmers return results of each software module in the form of development reports. After the modules are developed they are sent for testing.

Testing determines the module suitability for operation and the high-quality performance of all the required functions. Programmers responsible for development submit development reports for testing. The tester combines modules into single software package and tests software. If the test is successful and no customer comments the project modules are archived and the final project is put into operation by the customer.

This diagram describes the normal development scenario without comments. Scenarios with notice are described by a similar model. If the module does not pass the test, it is sent for revision [6].

The project is submitted for approving with the customer in case of successful test. If there are comments the product development manager issues a new development assignment. The final project's modules are archived and the final project is put into operation by the customer in case of comments absences.

Thus, the presented matrices fully describe the graph of the considered document flow process. The formalized approach based on the matrix form of representation can be used as input data for multi-user systems' algorithmization in solving modular software design problems.

## 4. References


[1] A. M. Langer, Guide to Software Development, Designing and Managing the Life Cycle (2012) 354. Springer London.

[2] B. Molnár, A. Benczúr, "Modeling Information Systems from the Viewpoint of Active Documents", Vietnam Journal of Computer Science 2(4) (2015) 229-241.

[3] M. Yu. Krukovskiy, Graph model of composite workflow, Mathematical Machines and Systems 1(120) (2005) 136.

[4] M. V. Poskonin, A. O. Kalinin, I. V. Kovalev, M. V. Saramud, Optimization of electronic document management systems by means of encoding and visualization of stored data in the integrated development environment, MATEC Web of Conferences 226 04021 (2018). DOI: 0.1051/matecconf/201822604021.

[5] I. Paramonova, "Electronic Document-Management Systems: A Classification and New Opportunities for A Scientific Technical Library", Scientific and Technical Information Processing 43(3) (2016) 136-143.

[6] I. V. Kovalev et al., J. Phys.: Conf. Ser. 1679 052037 (2020).