EVENT-DRIVEN APPROACH TO THE CONSTRUCTION OF INFORMATION SYSTEMS

Damian Ivanov

dWare Ltd., dware@dware.bg

SUMMARY

Various statistical data shows that about 60% of the developed information systems end without achieving their set goals. Our analysis shows that the problems lies neither in the stages of development of the information systems, nor in the methodology.

The problems lies in the vision /absence of a suitable abstraction/ and adequate tooling to materialize the examined processes into an information system. The approach and tooling developed by us allow modelling of the business reality by means of 'dEvents'. As a result, we obtain simple and understandable notations, convenient to maintain and develop, and the relevant operating web-based information systems, based on a relational Database Management System.

KEY WORDS

Event, Object, Space, Attribute, Event-driven model, Methodology for Information System development, Physical layer, Logical layer, Presentation layer, Notation

I. INTRODUCTION

Our attention is focused on the construction /examination, design, elaboration, implementation, maintenance and development/ of information systems /IS/.

We will analyse the data about the software projects, showing that approximately 60% of them are not successful.

By tracing the life cycle of a project, we will try to outline the major issues underlying the failures.

We will propose a new philosophy and tooling pretending to eliminate the established reasons underlying the failure. Finally, we will present the benefits of the application of the new philosophy.

II.LIFE CYCLE OF THE SOFTWARE PROJECT, METHODOLOGIES AND PROBLEM ANALYSIS

Are there any problems that require analysing the construction of an information system?

It turns out that the processes concerning the construction of an information system constitute a complex, long-term and expensive undertaking, yet without a guaranteed result.

Some facts:

1. ROGER SESSIONS [1] assesses that the costs of unsuccessful projects on a **WORLDWIDE SCALE** for 2009 are USD 6.2 trillion.
2. Dr John McManus and Dr Trevor Wood-Harper [2] estimate that the costs of the unsuccessful projects in **EU** for 2004 are EUR 142 billion and only **one of every eight projects** has been accepted as fully successful according to a study performed in the period 1998-2005 and covering 214 projects in 10 areas /defence, healthcare, education, agriculture, trade, production, finance, construction, transport and logistics/.
3. The Standish Group [3] analyzed the successfulness of the projects for 2009 and published the following figures:
   - Successful projects (achieved goals, kept deadlines); 38%
   - Unsuccessful projects (goals not achieved): 33%
   - Suspended projects:

29%

Similar findings are valid for France, Switzerland, USA and the whole world. The figures and percentages will for sure be different in the various studies and for the different countries, but this is no important.

What matters is the ratio between successful and unsuccessful projects and the constancy of this ratio over the years.

Those figures demonstrate that the stake is great and it makes sense to seek for the reasons for the failure and most of all to find a solution.

Can we outline the problem more clearly?

The project life cycle analysis demonstrates that irrespective of the viewpoints, a life cycle incorporates the following stages:

- definition of goals,
- study,
- design,
- elaboration,
- implementation,
- maintenance and
- development.

The above-mentioned study by Dr John McManus and Dr Trevor Wood-Harper analyses the stage at which the studied 214 projects are suspended or delayed.

№1

SUSPENDED AND DELAYED PROJECTS

lifecycle stage	Number of projects cancelled	Number of projects completed	Number of projects overrun (schedule and/or cost)
Feasibility	None	214	None
Requirements analysis	3	211	None

Design	28	183	32
Code	15	168	57
Testing	4	164	57
Implementation	1	163	69
Handover	None	163	69

Out of the studied 214 projects, 28 were cancelled and 32 were delayed at the design stage! We can already note the fact that 15 projects were suspended and 57 were delayed or made more expensive at the programming stage.

What is important for the design stage?

New and new design methodologies have been developed since the 1950s. There are tens of methodologies.

'Agile' methodologies have also been appearing and 55 of them have been described.

What does this say, without even analysing any of them?

It says only one thing. There is a problem. None of them complies with the requirements regarding the construction of various information systems.

Why do agile methodologies appear?

It is well-known [4] that 17 people of the practice define the following conclusions:

- People and communication are superior to *processes and instruments*.
- Operating software is superior to *detailed documentation*.
- The cooperation with the customer is superior to *negotiations during contract execution*.
- Addressing changes is superior to *following the plan*.

In fact, the *right elements* underlie the various methodologies until now, which leads to:

- Failure to achieve the goals
- Time delay

- Increased costs

Is this true?

If we examine more carefully any of the methodologies, we will find several simple truths:

- All methodologies /new and old/ solve the same problem
- However each one of them gives priority to one or another aspect of the study, design, implementation in order to solve various problems.

Thus, the main problem is the time necessary to accomplish the project, since the increased costs and failure to achieve the goals are a function of time.

Taking into account that the scope of activities related to study, elaboration and implementation of a project, irrespective of the selected methodology, is the same, if the projects had been accomplished within the expected deadlines, most of the methodologies would have been obsolete.

Let's have a look at the above study, where we can see the number of projects cancelled or delayed at the programming stage – 15 and 57 respectively.

In other words, all those methodologies have problems as early as this stage – the programming.

This is an indication that:

- We will obligatorily have problems with the developed system due to the specific aspects of the programming itself /definition, elaboration, testing, integration, documenting…/
- Absence of vision, adequate abstraction of the essence of an information system as a reality model, which always requires additional development of new and new programmes.

Let's have a look from another viewpoint.

Generally, an information system is a reflection /model/ of some aspects of the business reality in an organization. /Now we are disregarding the technical and communication aspects, human resources, software, which 'bear' and achieve the goals of the information system./

In order to reflect a reality, we need the appropriate **tools**. An artist uses paints and paintbrushes, a photograph – a camera, a writer – a pen, etc.

Is the reflection complete and accurate? Of course, the reflection is incomplete, however it achieves the set goals to some extent and does the expected work.

Let's have a closer look at the business reality.

From information point of view we have contracts, invoices and other documents, nomenclatures…

We could also have a look at the business reality from the point of view of:

- organisational structure,
- 'actors' /roles/ in business
- etc.

We have seen that the programming stage is reached as a result of the study and design.

The consumer software is most often presented as an aggregation of algorithms and programmes implementing them in order to achieve the goals of the information system.

The result of the design of an information system consists mainly of describing the business processes according to certain methodology or without a methodology /information flows, information carriers, algorithms for the processing of information, organizational structure, 'actors'…/ and necessitates the creation of the relevant programmes.

Sometimes the programmes turn into software implementing certain functionality, supplemented with options to determine the parameters for the purpose of more flexible compliance with the requirements of the specific business.

Determination of the parameters, but within the framework of the specific functionality! The new functionality requires new programmes or new software.

WHAT IS THE RESULT?

Usually the result is an information system that achieves the set goals to a different extent, however always with **restrictions** in terms of development towards inclusion of a new functionality from the whole business reality /as stated above/ and system support.

IS THERE AN ALTERNATIVE?

An alternative to what?

The business space is a whole. Examining it from various viewpoints, subsystems, tasks … is made for greater convenience. Mainly for convenience of management. But most of all, due to the absence of **philosophy, abstraction**, covering and reflecting adequately the set goals, in a simple and understandable language, and the **whole** business reality.

In other words, what we have seen in the previous paragraphs is the alternative to the impossibility to reflect the business reality in any other way.

IS THERE AN ALTERNATIVE TO THE EXISTING PRACTICE?

Yes, there is, and it comprises the creation of new philosophy, new abstraction of an information system, and may be a more general way to model the business reality, while trying to awaken a new and unknown sensibility to perception as a whole.

And apart from philosophy, we would also need tooling to do this.

III. PHILOSOPHY AND TOOLING OF dWareOS

Our solution is dWare OS. dWare OS /dOS/ is:

- **philosophy, abstraction,** trying to present in a simple and accurate way the business reality /within the framework of your needs, according to the degree of your knowledge/, and
- **tooling,** aimed to fix such description and enliven the described models.

The main idea is the finding that everything in the business reality can be presented as **dEvents**.

The elements of dWare OS and dEvent, dObject, dSpace.

A dEvent is described by:

- name
- **attributes**

dEvents are materialised in **dObjects**. All dEvents are at the ENTRANCE of all dObjects.

Every dObject generates a dEvent as an EXIT. The aggregate of dObjects, dEvents and their movement form **dSpace**.

We can differentiate between several **layers within a single dEvent**.

- logical/functional,
- presentation,
- physical.

Examples of dEvents include contracts, invoices, letters of acceptance and delivery, commodity receipts, inventories, petty cash credit orders, petty cash payment orders, advance reports. Examples of dObjects include accounting warehouse, marketing, but also register of invoices, register of contracts. An example of movement is the invoice, which will be located in the dRegister of invoices, in dAccounting, but also in dWarehouse, and dMarketing. It will also go to the entry point of dContracts, where it could be rejected depending on the specific business practice. A single dSpace may be the space of accounting, cash register, but all objects could also be in the same space, depending on the organization of work in the specific business practice.

Graphic presentation /Notation/

Business spaces are described by a set of graphic symbols, such as beginning, object, condition, entrance: on-screen, object, e-mail, file, service, exit to: file, e-mail, print… according to the methodology for construction of an information system, developed by us.

Managers

Managers are the programmes that ‘enliven’ the dSpaces.

What is the result of the implementation of an information system by this philosophy and tooling?

IV. CONCLUSION

The results are most of all in the overcoming of the main problems of the existing methodologies for the construction of information systems.

- Accurate and timely reflection of the customer’s needs
- Quick result
- Achievement of the goals
- Low price

It is worth noting that our abstraction does not reject or oppose none of the methodologies, approaches or practices for the construction of an information system, but only allows the opportunity to achieve the goals set for the information system.

The opportunities in a dSpace are a consequence of our abstraction:

- time ‘rewinding’
- changing the ‘rules of the game’ /functions/
- ‘expanding/shrinking’ the space itself with new objects/functionality.

Two more significant results are achieved alongside:

- Easy support
- Predictability and controllability.

And /almost/ no programming!

References

[1] http://www.zdnet.com/blog/projectfailures/worldwide-cost-of-it-failure-6-2-trillion/7627

[2] http://www.bcs.org/content/conwebdoc/19584

[3] http://www.journaldunet.com/solutions/expert/50420/projets-informatiques-sans-gouvernance-ni-conduite-du-changement---le-deraillement-assure.shtml

[4]

Http://bg.wikipedia.org/wiki/%D0%93%D1%8A%D0%B2%D0%BA%D0%B0%D0%B2%D0%B0_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F