Using a CASE Tool in Developing an Information System for Slovak Telecom

Mária Bieliková, Miroslav Galbavý, Ivan Kapustík, Ľudovít Molnár, Pavol Návrat
Slovak Technical University, Dept. of Computer Science and Engineering
Ilkovičova 3, 812 19 Bratislava, Slovakia

1. Introduction

As a part of a strategic transformation of Slovak Telecom (ST), a major enterprise with several thousands of employees, dozens of organizatorial units with various degrees of autonomy which are geographically distributed across the whole country, it has become inevitable to direct substantial efforts also to completely rebuild its information system. The need is justified by the existing state in gathering, processing and using informations in ST. Besides that, the strategic transformation requires changes towards greater efficiency of the overall company's performance.

In a company like ST, there have been already various partial information processing systems in use. However, they are devoted to special particular tasks with little or no interconnection. Moreover, they function at various structural levels of the company's organization which further complicates the situation. Last but not least, the existing systems work on various hardware/software platforms.

The above described situation is in fact quite common for many large enterprises in Slovakia. Therefore, we shall not devote ourselves to any too detailed elaboration of the particular details specific for ST. Rather, we shall concentrate on more general issues of a methodology for developing such information systems, and a role of a Computer Aided Software Engineering (CASE) tool in supporting it [11, 5,8].

Usually, the aim of such an endeavour is to build an integrated information system for the whole company. Of course, before starting the actual development process, system requirements must be specified very carefully. The development process itself starts with analysis of the current state in information processing within the company. Having in mind the size of such a company, this alone is a task which requires many tens, or rather hundreds of man/months of work. The subsequent phase of coarse and later finer design of the system require similar efforts. To communicate within the project, to document the ongoing work, and to produce the required documentation which contains several hundreds pages would not be possible without a computerized support of a CASE tool. Moreover, the company wishes to utilize the system's models developed during these phases for its own purposes.

The rest of the paper is organized as follows. We shall briefly describe the techniques that we have used in the project. We shall show how these techniques are supported by the CASE tool *Systems Engineer* [10]. In particular, we explain how the technique of functional modelling by data flow diagrams was used in analysing the current physical model of the system, which was based on document flow diagrams. We report on some of our experiences with the Systems Engineer tool and conclude with some comments that generalize our experience.

2. Techniques for system's analysis and design

System analysis and design is a time consuming, expensive but critical phase in software (and system) development [6]. In the Slovak Telecom project, basically, a variant of the Structured system analysis and design methodology [4] was adopted. For both analysis and design, functional and data models of the system were developed. The development of the model goes through the following stages (see Figure 1):

- physical current system model
- logical current system model
- logical required system model
- physical required system model.

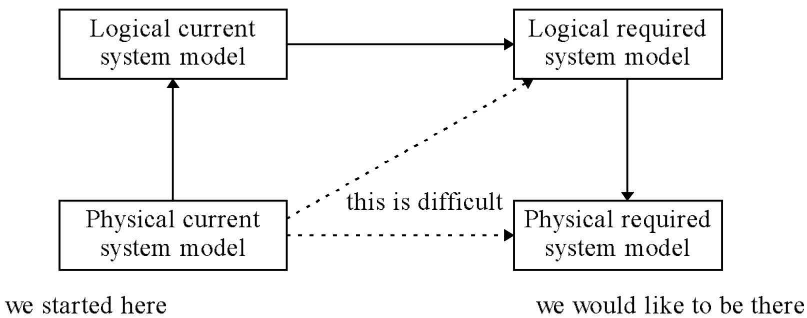


Figure 1: System model development stages

The path via the 'logical' (i.e., more abstract) level is inevitable due to the big complexity of the system to be modelled. To overcome this difficulty the problem should be solved incrementally through all four stages of system description.

Very often it is desirable to produce alternative logical required system descriptions. These can show for example differences in covering existing activities as modelled in current system model. Each of these logical alternatives may then lead to several physical descriptions.

The current physical model which we have developed for Slovak Telecom consists of:

- document flows diagrams
- statements of problems and/or requirements
- description of the hardware and software currently in use
- description of topologies of the computer networks.

In the current logical model we have used these techniques:

- data flows diagrams
- data model diagrams.

The required logical model consists of:

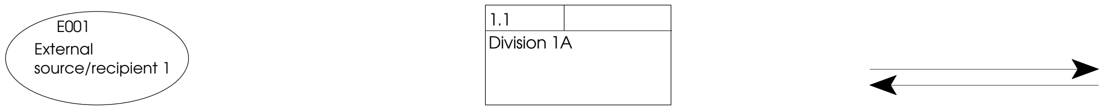
- data flows diagrams
- data model diagrams
- description of the proposed solutions to problems/requirements.

In the project, the required physical model has not been completed yet so we shall not discuss it.

3. How Systems Engineer supports the techniques

With regards to today's software tools we can observe a growth of client/server solutions with code generators [7]. Systems Engineer can be considered as such a tool. Systems Engineer is a complex client/server CASE tool running under MS Windows [10]. It allows collaboration of a team of software engineers. It offers various supporting techniques which address the full software lifecycle, from planning and design to code generation, reengineering and maintenance. Some of them are complementary, so it is possible to make a choice.

The document flows can be easily modelled by the standard technique of data flow diagrams. Following the Gane-Sarson [3] notation, there are symbols for processes, data flows, and external source/recipients (see Figure 2) which are applicable when producing document flow diagrams.



a/ External source/recipient b/Process c/Data Flow

Figure 2: Gane-Sarson data flow diagram notation

Symbols for processes are used to denote organizatorial units. Instead of decomposing processes, diagram decomposition denotes next lower level of the organization hierarchy of the particular unit.

For example, in Figure 3 there is a diagram "0" which shows three flows of documents between external sources/recipients *E001*, *E002*, and *E003* and the unit *System Overview*, in particular the sub-unit *Department1* within it. Moreover, there is a document flow *flow12* between *Department1* and *Department2* within the unit *System Overview*.

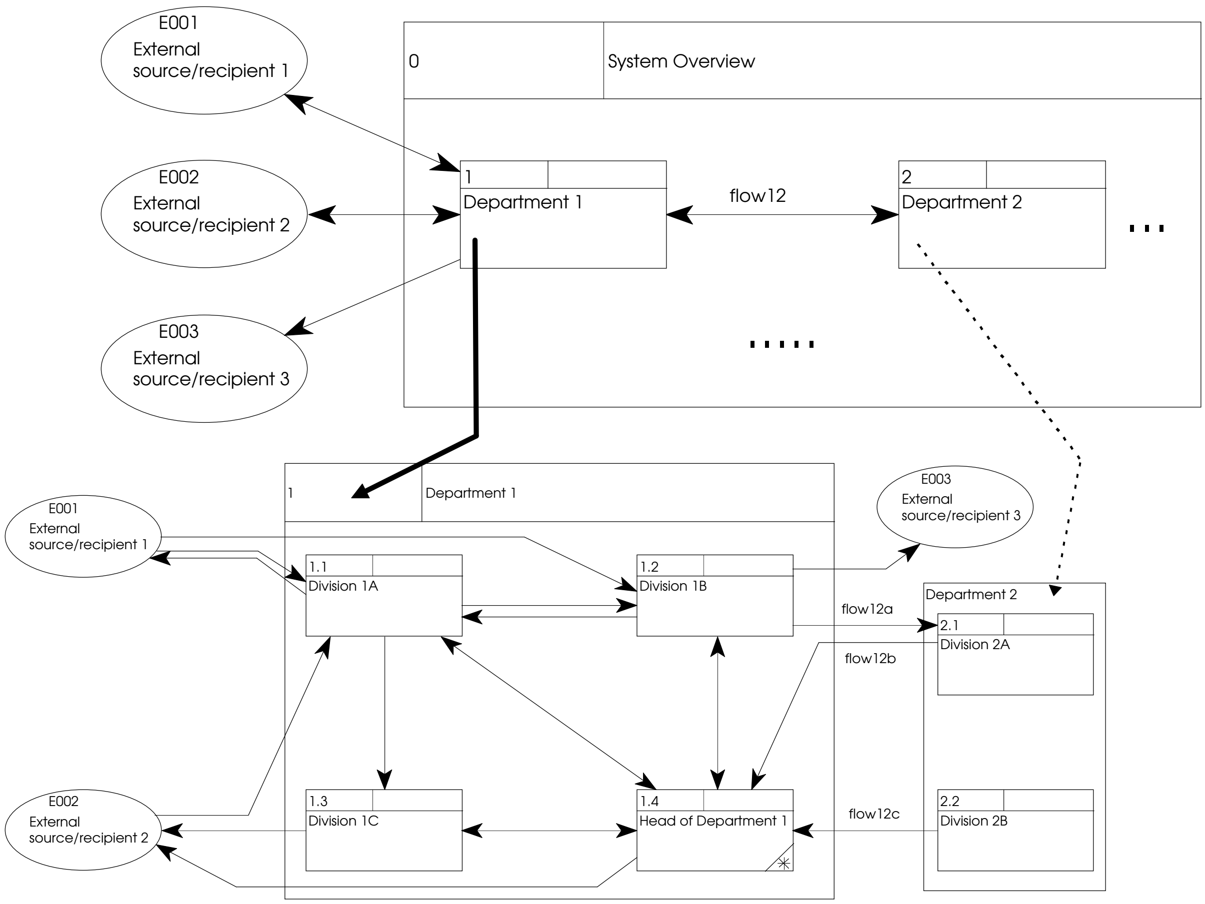


Figure 3: Document flow diagram example

At the next lower level, there are decompositions of both the sub-units *Department1* and *Deparment2*. In Figure 3, there is depicted the decomposition of *Department1*. Note that this diagram must maintain the same set of external sources/recipients as was the set of those which exchange documents in the next higher level diagram with *Department1*.

On the other hand, the flow *flow12* is decomposed at the lower level. During analysis, it was possible to identify not only departments, but even divisions as their sub-units which exchange documents between *Department1* and *Department2*. There are three flows *flow12a*, *flow12b*, and *flow12c* with different origins and ends. Obviously, in order to remain consistent, *Division2A* and *Division2B* both must be sub-units of *Department2* (the dotted arrow from diagram "0" to diagram "1" in Figure 3). Alas, although checking this property could be done in principle with the information available to the tool, Systems Engineer does not perform such checks.

Very important part of the physical model of a system is problems and/or requirements specification. Systems Engineer supports recording of changing problems and/or requirements and possible solution as they evolve. Problems, requirements are associated with solutions which we identified in the required logical model building stage. Moreover they are associated with other entities such as processes, data flows, software, hardware, data entities, etc.

Description of the software and topologies of the computer networks currently in use we put into Systems Engineer data dictionary through 'General Forms' tool and 'General pictures' tool respectively. They are associated not only with each other but with other entities like entities in dataflow diagrams, data model diagrams, etc.

In data modelling, we used with advantage the possibility to split large models into smaller logical groupings known as data model subsets.

4. Conclusions

In the paper we have described some of the main points of our work on Slovak Telecom analysis and design project. As we mentioned in section 2, the project has not been completed yet. In spite of this, we have gathered a lot of experience with application of analysis and design methods during nearly the whole last year. We tailored the known methods of structured analysis. Moreover, we developed an alternative view of logical model which is based on object oriented approach, namely user object modelling [9].

We have described modification of a document flow diagram technique with the notation of data flows diagrams together with pointing to possible support of this technique.

We adopted in the project an approach which can be entitled as User-Centric Software Engineering [2]. The main idea is to encourage collaboration between users and developers to help design more efficient business processes and supporting software.

We find the CASE tool Systems Engineer very useful tool for this kind of projects. We have experience with other tools such as *StP* (Software through pictures) and *Excelerator/SSADM*. Excelerator was the first tool that we used in training our software engineering students, before we shifted recently to using Systems Engineer. For several years, our students were developing their course projects using Excelerator. We found very useful the possibility of data flows and data store decomposing which is not supported in Systems Engineer. StP is a tool that, contrary to the Systems Engineer, runs in a Unix based windows oriented environment on work stations.

Software engineering is still a developing field [1]. Producing software is closely tied to our understanding of the nature and value of different analysis, design methods and is very largerly based on the previous experience.

References

[1] Andriole S.J., Freeman P.A.: *Software Systems Engineering: the Case for a New Discipline*, in Software Engineering Journal, May 1993, pp.165-178

[2] deBellis M.: *User-Centric Software Engineering*, in IEEE Expert, February 1995, pp.34-41

[3] Gane C., Sarson T.: *Structured Systems Analysis*, Englewood Cliffs NJ: Prentice-Hall, 1979

[4] Hares J.S., *SSADM version 4: The Advanced Practicioner's Guide,* John Wiley & Sons, 1994

[5] Lehman M.M.: *Software Engineering, the Software Process and Their Support*, in Software Engineering Journal, September 1991, pp.243-258

[6] Mannion M., Keepence B.: *SMART Requirements*, in Software Engineering Notes, Vol.20, No.2, April 1995, pp.42-47

[7] Pyle D.T.: *Software Development Methods and Tools: Some Trends and Issues*, in Software Engineering Journal, March 1996, pp.99-103

[8] Robinson D.: *CASE Support For Large Systems*, in Proc. ESEC'91, Springer-Verlag, LNCS 550, 1991, pp.504-508

[9] Slovak Technical University, *Analysis and Design of Slovak Telecom IIS*, technical report, 1996

[10] Systems Engineer 6, *Concepts Guide*, Learmonth & Burchett Management Systems Plc, 1995

[11] Vessey I., Sravanapudi A.P.: *CASE Tools as Collaborative Support Technologies*, in Communications of the ACM, Vol.38, No.1, January 1995, pp.83-95