Threats to model driven development - what happened to eighteen companies twenty years later

Stein Erik Ellevseth

Ellevseth Modell utvikling, Oslo, Norway

steinee@eunet.no

Abstract. Introducing new methods and tools that support higher abstraction levels in software development was expected to improve profitability, lower development cost, improve quality, shorten time to market and increase controllability. Several examples from a Norwegian project proved these expectations achievable in many cases. Assessing a large industrial initiative that supported embedded systems development through 8 years in 18 companies, formed the bases for this claim. This initiative was regarded as a great success among the participants and in academia twenty years ago. It gave the community object-orientation in SDL (Specification and Description Language, Z.100), MSC (Message Sequence Charts, Z.120) and later UML (Unified Modeling Language from OMG, Object Management Group). Twenty years later, not all of the participating companies did not harvest well from this investment in new technology. The methodologies and tools are somewhat still in use, proving that the project in many cases contributed to sustainable software development and the companies had economic benefits. Another positive result is that the people who worked in these companies matured the industry, improved methodologies, tool and standards. An important question is to what degree the methodology contributed to improved profitability in the companies. This is answered by measuring the maturity level in different disciplines. Another important questions investigated in this study is why the methodology is not being used on new products if profitable, or if unprofitable, why it continues to be used.

Keywords: Model driven development, SDL, MSC, OMT, UML, process improvement, maturity model, sustainable software

1 Introduction

This SISU initiative, (Support Integrated Systems Development), was a project supported by Norwegian Research Council from 1989 – 1996. This paper describes an assessment of SISU results twenty years later. The author assessed what happened with the companies, their products and the people involved. The evaluation of the Norwegian industrial companies was planned in three stages. First, to find the various companies` contact persons to get feedback. Second, to establish the maturity level five and twenty years after, or at the time the company closed their activities in Norway. Third and finally, to get the detailed story from the companies, assessing their profitability, efficiency, the quality in their products and the people, and document this in a report. Stages 2 and 3 are ongoing and this paper is an early summary of the assessment results.
Which results from SISU that are still in use and what happened with the tools and methodology when the product was phased out is ongoing research. Did the technology fade out with the products? Were the values and best practices in the development of these product worth taking care of could be proof of the sustainability of SISU tools and methodology introduced twenty years ago.

In late 1980´s we experienced that products were released with a lot of defects, high quality costs, low controllability and cost overruns. The methodologies in use were incomplete causing different practices in the organizations and uncertain approaches. They were dependent on individuals, the tool support was poor and the productivity low. This was the situation that initiated the SISU project.

1.1 Main Objectives

The main objectives of the SISU project were to «Improve the profitability in development and maintenance of embedded software through co-operation between companies". To achieve better profitability the SISU project was focusing on improving the companies capabilities to achieve precise quality, shorten the time to market, lower cost and high controllability. Based on this main objective, operational goals were defined for each company.

The operational goals were defined in five maturity levels inspired by CMM, the Capability Maturity Model in six disciplines : a) System Description, b) Verification & Validation, c) Transformation, d) Product Management, e) Reuse and f) Process. The criteria for maturity levels for each of the disciplines are explained in details in Appendix B [Chapter 5]

1.2 Main Results from the SISU project

During the SISU project the companies experienced improved cost-benefit, methodologies, advanced standards, better tool support and a successful forum network, the SISU Forum.
The SISU project was regarded a great success among the participants and in academia, and also gave the community object-orientation in SDL, MSC and later UML enhancements.

1.2.1 Cost-benefit improvement

A cost-benefit analysis [10.] was done prior to the project start to focus on the cost benefits by participating in the SISU project. It was anticipated that if the company reached its operation goals at the end of the project and kept it for five years it would give them cost reduction of 33%. Interest rate was set to 10% at that time and with this prerequisite it was estimated that SISU would give a cost savings of 300 million NOK. With an investment of 48 million NOK, SISU was a project with very good profitability.

The eighteen companies involved were in total lifted more than eighty levels during the eight years duration of the project and this was recognized as a great success. These levels are defined in the SISU Maturity model.

No assessment of the maturity level in the companies was performed after 5 years in 2000 to verify the cost-benefit analysis prior to the project. Now, twenty years later, the author performed a new assessment to evaluate the current or last maturity level for the companies. Twenty years later many companies have stepped back several maturity levels. They are at a higher maturity level than in 1989 when the project started, but lower when the project ended in 1996. This is also the case with the estimate of the maturity level in 2000. One company (company 12) which invested 4 million NOK in was estimated to have saved 24 million NOK participating in the SISU project. Twenty years later it is difficult to assess the economy of the company and to what degree SISU contributed to its finances.

1.2.2 Standards

There were a common agreement that tools should be improved through standardization, which tool vendors would have to comply with.

During the project, SDL92, also called object-oriented SDL was finalized in 1993. This was the last real SDL (Z.100) recommendation from ITU. In 1996 there were some minor clean-ups. In SDL 2000 we got agents, mix of blocks and processes, interfaces, composite states, ASN.1 support, object-orientated data, signal types and exception handling. UML2 was also influenced by the SISU projects results when

SDL and MSC, Message Sequence Chart Z.120, were introduced in 2003 and with the introduction of composite diagrams.

The standardization was financed by the SISU project, based on another Nordic initiative, the Mjølner project. Several of the participants later took part in the improvements of UML 2, with the super structure and sequence diagrams. All based on similar constructs in SDL (ITU recommendation Z.100) and Message Sequence Charts (ITU recommendation Z.120). Later SDL 2000 was financed by the Norwegian Research Council and Ericsson, which also took part in the OMG standardization of UML.

A UML profile was later created that was executable with support tool for debugging and analysis, as well as an improved editor for sequence diagrams.

1.2.3 Tools

Object-orientation in SDL´92 was slowly introduced in the SDL tools. There were some uncertainty about how useful the o-o concepts were from the tool vendors, but they finally implemented SDL´92. The most used SDL tools used were SDT from Telelogic and ObjectGEODE from Verilog.

Important products/tools introduced to SISU companies were an automatic code generator from SDL and an SDL runtime system, which still is in use in many of the products developed during SISU. The code generator *ProgGen* and the SDL run time system *TST (Telox SDL tool)* are still in use in many companies inside and outside SISU. ProgGen was developed by Sintef Delab and maintained by Sintef for awhile. TST [9.], a run time system for SDL systems was developed thirty years ago and is still used in many embedded products. Code running on top of it could be manually written, but also ProgGen code skeletons were made for running on top of TST. It was delivered with source code and is now maintained separately by the companies still using it.

A tool that did not reach the SISU projects was DaSOM (Computer Aided SDL-Oriented Methodology) developed by Sintef Delab (former Elab) and CapGemini (former Computas) and was a forerunner to the SDL tools to come later. It was a system development tool which especially gave computer support to specification and design, but also provides support for implementation, production and maintenance of software systems in the communication- and process control areas. The tool was based on a software engineering methodology called SOM (SDL-Oriented Methodology). SOM was based on ten years of practical experience in the 70s, while DASOM has been developed during early 80s. Its development lost the competition against the major tool vendors and was discontinued mid 80s.

OOram ”The Object Oriented Role Analysis and Modeling (OOram) is a method, based on the concept of role, for performing object-oriented modeling. [1]. Originally

(1989) called Object Oriented Role Analysis, Synthesis and Structuring (OORASS), the method focused on describing patterns of interaction without connecting the interaction to particular objects/instances. OOram was originally developed by Trygve Reenskaug (1996), a professor at the University of Oslo and the founder of the Norwegian IT company Taskon. The use of "roles" in OOram is similar in application to that of Agent-oriented programming."

Other results to mention are JavaFrame, ServiceFrame and ActorFrame which was developed by research companies and industrialized.

A new tool Reactive Blocks [12.], using UML activity diagrams implemented with SDL semantics is showing positive results. At Sintef, a new tool ThingsML [11.], was developed to simplify application development for Internet Of Things.

1.2.4 Networking

An important result was the networking among the engineers. Even though there were companies competing with each other, it did not affect the communication between the engineers. Their goals in SISU were the same; to have betters tools and methodology.

SISU Forum was a yearly conference for the SISU participants and was a great contributor to the success of SISU.

The participation in SISU raised the awareness on software quality and the way of working to achieve that. In the companies there were people being trained into advanced ways of developing software, a knowledge the participants brought with them when joining a new company. Thus the program had a positive effect on the entire industry. In addition there have been several side-effects in methodologies, tool and standards.

1.2.5 SISU follow-up plans

After SISU ended in 1996, follow-up activities were expected and a yearly SISU Forum was agreed to be planned. After time, maybe due to the .com decline and telecom «ice age» the momentum was lost. It appeared that SISU and TiME methodology did not have the momentum needed for the existing and new companies to stay connected.

Follow-up projects were proposed to take care of the SISU results. TiME needed further development, but first of all it needed to be applied in an organization. Several SISU companies were interested, and a project was started with TiMEaxe, TiME for AXE main public switches. Sintef maintained the most important results from the SISU project, web pages, ftp server and the email lists. They also further developed the SISU/TiME methodology by commercializing the electronic text book, ELB, sup-

ported evaluations, piloting and introduction of TiME. There were plans to establish companies to commercialize TiME, but there was no confidence that the methodology was good enough to be self-contained. One reason may have been that there had been no full scale trial or that TiME was not mature enough for a company to carry. A Norwegian industry association, ITUF, which initiated the SISU project, established a group to continue the network that could to bring together the SISU participants through thematic meetings. Follow-up of the yearly SISU Forum was also planned. Work with the ITU standardization continued, including important changes planned for SDL96, was put on hold because the tool vendors were hesitant in keeping up with SDL-92. SDL-96 therefore contained minor updates of SDL-92 and aimed for major updates in SDL-2000.

1.2.6 SOON and TIMe methodology

A textbook "Engineering Real Time Systems" [4.] was published in 1993 and an electronic book TIMe, The Integrated Methodology [8.] was finalized in 1996. Both were a direct result of the SISU project. The textbook has now sold 5000 copies in 2016.

TiME, The Integrated Methodology, was a continuation of the SOON, SISU object-oriented notation, methodology documented in the textbook "Engineering Real Time Systems". It was maintained by Sintef until 2005, and a version was developed for Ericsson AXE development process, called TiMEAXE. Unfortunately this was never put into real projects due priority changes at Ericsson early 2000. TiME, was also taught at HiO, NTNU and other education institutions in Norway. SDL is taught in Norway as well as universities around the world, but mostly in Europe. Many of the SISU results are also referenced in [7.] a thematic number of Telektronik 4 of 2000, Languages for Telecommunication Applications.

2 The SISU Assessment approach

Each company was assessed using the criteria of the SISU maturity model. The maturity level of the company was assessed in 1992 at the beginning of the SISU II project. Then the company defined its operational goals for each selected discipline by the increase in maturity. At the end of the project in 1996 or when the operational goals were met, a re-assessment was performed stating the new maturity level.

2.1 The SISU Maturity model

The SISU maturity model for measuring success was very effective in understanding how to identify the improvements the companies wanted and needed. Operational goals were defined that eased measuring the success of the project. These operational

goals were the 6 characteristics : a) System Description, b) Verification & Validation, c) Transformation, d) Product Management, e) Reuse and f) Process. Each discipline has 5 maturity levels, inspired by The Capability Maturity Model, CMM[3.], from SEI, Software Engineering Institute at Carnegie Mellon. Each maturity level has well defined criteria for when the level could be accepted as reached. These criteria are software best practices. These were also a starting point for which actions were needed to reach the next level.

	a) System Description	**b) Verification & Validation**	**c) Transformation**	**d) Product Management**	**e) Reuse**	**f) Process (CMM)**
1	Realization-oriented	Test-oriented	Compiled	Initial	Initial	Initial
2	Partial design-oriented	Inspection	Partial generated	Version-oriented	Product- oriented	Repeatable
3	Design-oriented	Animated	Generated	Product-oriented	Domain- oriented	Defined
4	Product- oriented	Analyzed	System generated	Integrated	Integrated	Managed
5	Required	Synthesized	Design transformed	Required	Global	Optimized

Table 1: Maturity levels

2.2 SISU company assessment

All SISU companies were assessed at start of the SISU II (1992) project to establish the maturity level as a basis for identifying what improvements are necessary to reach a higher level in the selected disciplines. The criteria of the maturity levels are a selection of best practices to reach that level. This approach is a tool for initiating improvements with best practices that can contribute to better profitability through lower development cost, precise quality, shorter time to market and high controllability.

With this maturity model it is possible to measure and compare the maturity level twenty years later because the best practices defined are still relevant for software development. Software quality through fewer defects always influence profitability. However, it is not the only factor that contributes to economic success. There are also factors like technical issues beyond methodologies, market situation, strategic decisions, etc.

3 What happened with the companies ?

In the duration of the SISU II project, the companies in total were lifted more than 80 levels during the four years from 1992 to 1996. This was recognized as a great success. Of the eighteen companies, ten were lifted at least one level.

Five years later there should have been a re-assessment according to the cost-benefit analysis, but it did not take place.

Twenty years later the author wanted to know what happened by assessing how they are performing today, how the investments in training, new methodologies and tools were maintained during the years after the project ended in 1996. The status in the participating companies does not reflect the same success as twenty years ago. Of the eighteen companies, ten still exist, three have ceased to exist, three have merged, and two have moved their activity to another country. Many products do not exist anymore, but in the ten remaining companies there are positive results too, as there are products still in service with the same technology.

The companies are in 2016 at a higher maturity level than in 1989 when the SISU I project started, but in average below the level when the SISU II project ended in 1996. 10 of the companies have products that were developed during SISU project and are still in service. Some products have gone through changes and do not use SDL or code generation anymore. 4 or 5 products are still using SDL, and SDL code generation (3 with TST [9.] and 2 with ProgGen [ref [5.]). Many companies and products have gone through changes, as well as the people who participated. For many participants, the SISU methodology was important for them, as it has influenced their way of working and they still use parts of it. Others have adhered to the way they work in their company without influencing.

The companies joined SISU for different reasons, mainly in order to improve the company skills of developing software that could solve their problems. These problems could be related to costs reduction, quality improvements, shortened time to market or better predictability of their projects.

This setback is not very encouraging since it was expected that the success would encourage the companies to continue utilizing the methodology and continuously improve their maturity.

If the SISU companies reflects the situation for embedded software companies in 2016, it is discouraging because products are still being released with too many defects, incomplete methodologies, person dependency and diversity in tools in use. Additionally, not many modeling tools are in use anymore. Education is much better, but appears to be on advanced programming levels.

Did we promise too much 20 years ago? The history of AI, Artificial Intelligence, promised much, but disappointed due to lack of computing power and limitations. After the "AI Winter" caused AI to become an inflamed word and therefore choking the cash flow. Later they called it expert systems and today it is called Machine Learning. Is Model Driven Engineering (MDE) a similar story?

Details of the participating companies are described in Appendix A [Ch 4. The enumeration is according to the Table 2: Maturity assessment 1992 - 1996 - 2016]

Company	System Description			Verification & Valid			Transformations			Product Manageme			Reuse			Prosess			Change	Change	Change
	1992	1996	2016	1992	1996	2016	1992	1996	2016	1992	1996	2016	1992	1996	2016	1992	1996	2016	'92- '96	'96-'16	'92-'16
1	3	4	2	2	3	2	3	3	2	2	3	3	2	2	2	3	3	4	3	-3	0
2																					
3																					
4	1	2	2	1	2	2	1	2	2	2	3	3	2	2	2	2	3	3	5	0	5
5	1	2	2	1	2	2	1	2	2	1	2	3	2	3	3	2	3	3	6	1	7
6	1	3	3	1	2	2	1	3	3	1	3	3	2	3	3	2	3	3	9	0	9
7	2	2	1	1	2	2	2	2	1	2	3	3	2	3	2	2	3	3	4	-3	1
8	2	3	2	1	2	2	1	3	3	2	2	2	1	1	1	2	3	3	5	-1	4
9																					
10	2	3	3	2	3	3	1	3	3	3	3	3	1	2	3	3	4	4	6	1	7
11	2	3	1	1	2	2	1	1	1	2	2	3	1	2	1	2	3	2	4	-3	1
12	1	2	1	1	2	2	1	2	1	1	3	3	1	3	3	1	3	3	9	-2	7
13	1	2	1	1	2	2	1	3	2	1	2	3	1	2	2	2	3	2	7	-2	5
14	3	4	2	2	3	2	1	3	2	2	3	3	1	1	2	3	4	2	6	-5	1
15	2	3	1	3	3	2	3	3	1	2	3	3	1	2	2	2	4	3	5	-6	-1
16	2	3	2				2	3	1										2	-3	-1
17																					
18	2	3	3	1	3	3	1	3	3	1	2	2	1	1	1	1	2	2	7	0	7
19	1	3	1	1	2	1	1	2	1	2	3	2	1	2	1	2	3	2	7	-7	0
																			85	-33	52
Average	1,7	2,8	1,8	1,4	2,4	2,1	1,4	2,5	1,9	1,7	2,6	2,8	1,4	2,1	2,0	2,1	3,1	2,8			

Table 2: Maturity assessment 1992 - 1996 - 2016

3.1 Findings

During the interviews, many observations were made. Here is a summary of the observations :

- Methodology enthusiast and improvement agent were missing, moved or lost interest.
- Missing industrial association network (i.e. SISU Forum)
- Different view of the benefits of a higher abstraction level in the companies
  - Resistance to changes
  - Real programmers: source code rules, was and still is an «obstacle»
- Tool vendors were slow in applying new constructs from the SDL standard
- Missing target support in the language and the tools, to faster adapt to new hardware.
- Technical issues beyond methodology, hardware issues, and software issues (embedded challenges with ISRs, etc.)
- Using SDL as a programming language
- Slow tool support. PC version came too late and was not bug-free

3.2 Discussion

Whether the companies have solved the problems which were the reason they invested in SISU methodology is still not clear for all companies. Some companies pulled out of the project when their goals were achieved, i.e. reaching their planned maturity level. Three companies withdrew at an early before any commitment and maturity level assessment was done. Three companies developing military equipment have managed to stay in business. Some of them even managed to increase their maturity level. Some of the explanations can be the missing network, the fall of the .com wave and the telecommunication «ice age» around 2000. Another explanation is that the higher abstraction tools were late in providing tool support at a higher abstraction level than traditional C/C++ when new hardware is introduced. It is important for a company to reach the market fast with a new product, and usually cost is not an issue at an earlier stage. Some positive results are that the people that worked in these companies has raised the maturity level in other companies. There have also been several side-effects in methodologies, tool support and standards. Higher abstraction do not fit for all type of applications, which maybe tool related, but also that it creates overhead due to the type of documentation provided vs. required. Smaller embedded systems are today running without any runtime systems and timing is driven by Interrupt Service Routines (ISR), an architecture which is not well supported by SDL.

Some skepticism has built up around MDD, Model Driven Driven Development, because developers have discovered its deficiencies. They now favors agile development. There are not too many good examples of enterprises that have achieved good results by investing in MDD (using SDL and UML). There are however anecdotes of poor results. Most companies are concerned about maintenance of existing systems, and most MDD methods and tools do not support this procedure well enough. Developers (and students) favors new programming languages rather than modeling. It seems that it is more important to be a good programmer than to model. Graphic modeling is perceived as providing poor productivity because modeling languages and tools are not good enough. They are not suited good enough for team development. It is code that sells, and debugging and testing happens at code level.

3.3 Conclusions

Important questions investigated in this study were : 1) If the methodology was good, why is it not still used on new products? 2) If the methodology was bad, why was it not replaced? These questions has not yet been fully investigated for several reasons. Mainly because it is a long time ago and some people that participated do not work in the company anymore. It therefore takes time to find out what really happened.

This study is a summary of useful results for standardization community, tool vendors, project leaders and quality managers. The SISU maturity model is useful for as-

sessing the companies capabilities of developing software and can be recommended for future use.

A comment from one of the participating companies is that TST is the best SDL runtime system available for embedded systems. So far they have not found others that can replace it, yet. Also here SDL has shown sustainable capabilities.

An important result is the awareness of state machine modeling, which has found a home in most of the companies. There were a lot of competence build-up during the SISU project, but this were not sufficiently taken care of later by the companies, industry associations and education institutions. The continuation and follow-up appear to have been run on CMM level 1, i.e. by the initiative of individuals, not by requirements, user needs, increased quality and time-to-market. The industry associations were not able to ensure that the investments done by public money were well spent in the long run. What do we see of technology in the companies today is varying.

A comment from SAM 2016 : "*UML has basically killed all other similar technologies on its way. 15 years later UML is still far from what other technologies such as SDL could do.*" Is this also a reason that maturity level has been reduced during 20 years?

3.4 Further work - "to win the future, you need to learn from the past"

Future questions to investigate are whether the methodology contributed to improved profitability, reduced lifecycle cost and increased lifetime of the companies and their products. The cost-benefit analysis of 1993 did not take into account how much of the sale in the company was affected by the SISU methodology. It is possible to find out how much sale the products developed with SISU methodology generated, but not how it would have been with traditional development. To do that, there is a need to compare with similar products using traditional technology. The author wants to know what are the long-term effects of implementing model-driven development and available results after twenty years use in a company. Larger corporate have initiatives inside their organization, but smaller companies could benefit from a local Norwegian network initiative. It would also be interesting to assess the companies that withdrew from the SISU project before it even started to compare their maturity level with SISU companies to check if they managed to raise their maturity level without SISU.

The author will assess the research institutes (Sintef, NR, Ife, etc.) about what happened with them, the contributing technology and the people. This is to assess the long-term effects of introducing model-driven development that can be observed after twenty years. Another follow-up is to follow the people that left for other companies and how the knowledge was spread to other companies and organizations.

The author propose to perform maturity level assessments every 4-5 years of the SISU companies, assessing related companies to SISU methodology (SDL and UML users) and companies where participants moved to, where SISU methodology can be of relevance.

The author will propose a reenactment of the SISU project, using improved SDL toolchain working in available modern (DevOps) development environments and implementing TiME in a lean and agile fashion supporting continuous design and delivery.

4 Appendix A The companies summary

Still in business :

Company 1 is developing military and airport communication systems. Their activities in SISU II that was largely successful, achieving improvements in many areas, especially in the beginning. After SISU ended, improvement of methods and tools (also from SISU) competed with organizational changes and process-related activities. This resulted in varying commitment from management. Today UML to some extent is in use at a higher level, and SDL on existing products. No new product uses SISU methodology. The company started early with automatic code generation from SDL to CHILL using ProgGen. They developed a radio product during SISU which later was sold to company 10, which is still based on SDL. The SISU methodology is not very visible in the company anymore. Merged later with company14.

Company 2 is developing fire and marine safety systems. They started up in SISU with implementation techniques for transputer technology with a toolchain supporting design and code generation from SDL to Occam based on ProgGen. They withdrew early from the project and did not achieve any measurable improvements according to the SISU maturity model. The toolchain was transferred to company 13 which completed the toolchain in association with Sintef.

Company 5 is developing radar systems. They participated in SISU through an EU reuse project. They withdrew from SISU in 1993 due to high work pressure and that the other companies in SISU who also worked with reuse was so different that it was difficult to cooperate. Today it is unclear how much of SISU methodology is visible in the company.

Company 7 is developing airport communication systems. The quality of the delivered systems were significantly improved by participation in SISU in that they adopted SISU methodology in their business. Their toolchain was based on OOram and generated C code with ProgGen to run on TST. After SISU ended they continued to use SISU / SDL methodology and tools on existing products. They merged later with a company whose products were not subject to the use of SISU / SDL technology. Today OOram and ProgGen is no longer in use, and the code is maintained manually. TST is still used on previous OOram based products. An SDT tool chain was later used in a new product.

Company 10 is developing military communication systems. The SISU project was characterized as one of the most useful projects the company has participated in. After SISU ended they continued to use SISU / SDL methodology and tools. They took over a product from company 1 and replaced the CHILL (Z.200) code skeletons with their own C skeletons and the Chill Runtime System with TST. They used the same SDL developed during SISU on their new hardware. Today, SDT, ProgGen and TST are still in use.

Company 11 is developing systems for automation of the distribution and storage of oil products for the onshore industry. SISU contributed to better awareness regarding process and quality assurance, used more methodically. During SISU they developed reusable components designed in OMT and coded in Java. Ten years later when the software was converted to .net/C# they used UML for modeling by a consultancy company. Their reuse approach was not followed up. Today the SISU methodology and OMT/UML models are no longer visible in the company. No formal methods and tools are in use.

Company 12 is developing audio measuring equipment. The SISU project was very useful by being able to exchange experiences and gain inspiration from other similar environments. They started very early with PC based version of SDT and the code generated based on ProgGen. The code was generated to run on the TST (Telox SDL Tools). The system consisted of two controllers. A front-end where the SDL generated software ran and a signal processing unit which was written in assembly, later executing C code. After SISU ended, SDL and code generation was gradually abandoned due to poor tool support and diverse opinion on abstraction. Today state orientation and TST is still in use in the products developed during SISU. No new products is using SISU methodology.

Company 14 developed military communication equipment. The main objectives of SISU II was to improve the process of managing, analyzing and tracking customer requirements in the development process. The company had great benefits from using the SISU's textbook and SDL courses. SISU was a useful forum to exchange information, learn more about new opportunities with SDL and MSC, and not at least getting constructive help to improvements from SISU consultants. After SISU their implementation of the SISU methodology was not easily transferable to other areas, like military information systems. They later merged with company 1 and the products developed during SISU has been phased out.

Company 15 is developing intercom systems. The company achieved good results in SISU which inspired to improvements in the 6 disciplines of SISU II. In addition, they observed that product development with better methodology and tool support led to a good and stable product with fewer defects. After SISU ended they merged with another company with no products that was subject to the use of SISU methodology. The original product developed during SISU is still in use and is the only product

from SISU project with tool chain based on SDT, ProgGen and TST. It is not clear why the methodology was used on newer products.

Company 19 is developing door control and access systems. The company benefited from participation in SISU, especially in the development of a common quality system. After SISU ended, their administrative computer systems became more important than the reactive aspects of door lock and access control system. Today the SISU methodology is not visible in the company.

Consultancy companies, still in business:

Company 3 is an international consultancy company for information technology. They withdraw early when their project within ESA was cancelled. Before SISU started they developed a tool DaSOM that was were stopped mid/late 80s because it lost in competition with the major SDL tool vendors.

Company 8 is a consultancy company within telecommunication, electronics and information systems. They evaluated and adapted SISU methodology, SDL and tools for use in customer projects. When SISU results was not used at their customers they ended their use of the SISU methodology.

Not in business anymore:

Company 6 developed telecommunication products for consumer market. SISU was important for improvement work in the company. An important contribution was that they could demonstrate that the SISU methodology and SDL tools was suitable for small complex (embedded) micro-controllers with high real-time requirements and limited space requirements. Their SISU activity ended in 1995 when they had achieved their objectives. In 1997 they became a part of the mother company platforms. They developed mobile phones and platforms for mobile communication devices where the development environment after merging with mother company was based on traditional coding in C / C ++ and incremental development based on the waterfall method, later scrum. Development environment based on SISU methodology and SDL were eventually discontinued as the products were phased out.

Company 4 developed communication systems for financial market, police and air traffic control. SISU was a catalyst for improvement work in the company. They developed TiME based development processes which was very promising for further use of TiME. Unfortunately it was never put into practice. They adapted a toolchain based on SDL and ProgGen for PLEX and runtime system in use on their systems. After SISU ended, this activity was terminated in 1997 when the products developed in SISU was transferred to mother company abroad and the department was later closed down. The statues of these products in the mother company is unclear.

Company 9 withdraw early due to a reorganization that caused the department that participated in SISU to be closed down.

Company 17 developed a tool OOram which was state-of-the art at that time, but the tool lost in competition with the major tool vendors. The OOram tool and methodology was used in company 1 and 13 and piloted in company 7.

Company 18 was a consultancy company in advanced realtime and database systems for control-, communication and information systems. They developed and maintained TST, Telox SDL tool, which was used, and still is in use, by many SISU companies. The goal in SISU was to build up expertise in formal use of SDL-92 and build a complete tool chain for software development, business support and maintain results from SISU II. They used this to develop an alarm system for a customer using SDL-92 and generated code to C running on top of TST. After SISU the employees who worked with real-time development left the company and activity was discontinued around 2000.

Company 13 developed audio and telecommunication systems for the broadcasting domain. In the SISU project they developed a digital mixing system and needed formal methods due to its complexity. SISU was a great inspiration for improvements for the company. Particularly helpful was the sharing of experiences with other companies. SISU's standardization work gave smaller companies an opportunity to indirectly influence tool vendors. They introduced SDL/Occam tool chain for their communication module, generation code to Occam with ProgGen running on top of an Occam running framework. In the user console they used OOram (the Taskon tool) with automatic code generation to C++ with ProgGen running on top of QNX. After SISU ended came a turbulent time with technical problems, delays and several bankruptcies. Details on what really happened and why is still not clear. Several technical and management issues causing delays that can explain it somewhat. The company closed down in 2003 and their products was transferred abroad and maintained there for some time. The availability of this product today is uncertain.

Company 16 developed data storage systems. They were encouraged to invest in new methods and tools when the economy was very good. They were market leader in data storage and had the largest suppliers in their customer base. The company achieved its main goals of the SISU project and its planned improvements, which was 30% reduction in development time and that half of the developers were to have good SDL and SISU methodology knowledge. After SISU ended, different views on abstract models appeared in the development of their data storage devices. SDL was eventually abandoned and they rewrote the whole system to ISR based execution, round-robin scheduling and hand coding. Customer (IBM) requirements greatly influenced the development of quality, but not in areas where SISU methodology focused. After unsuccessful efforts on creating a new storage standard sales went down which caused cutbacks, and the company closed down in 2009. Some people from the company ended up in other SISU companies, working with SISU methodology, like in company 7. Another used SISU methodology in the development of a target laser designator, with state modeling, asynchronous message passing, runtime system, etc.

5 Appendix B Maturity levels explained

Each of the levels of the disciplines are defined with best practices which best characterizes each level.

a) System Description criteria :

- **Realization-oriented level (1)** The descriptions that are being maintained are realizations, i.e. software and hardware descriptions. Other descriptions that are available are incomplete and are not always maintained.
- **Partial design-oriented level (2)** Incomplete functional and non-functional requirements are more or less formal using SDL, UML or an appropriate modeling language. These are used as basis for implementation which is manual coding. The requirements are maintained during the development and the lifetime of the product.
- **Design-oriented level (3)** Complete functional, non-functional design and implementation design are created. The software is derived from these description based on defined rules, and is less significant as documentation.
- **Product-oriented level (4)** Emphasis on creating reusable building blocks and generic products. There is a well defined process on composing and configure product releases.
- **Required level (5)** Focusing on having a rigorous and complete description of market needs. There is an automatic transition from needs to functional design. The rest is automated based on a library of reusable components.

b) Verification & Validation criteria :

- **Test-oriented level (1)** Testing to verify the system is the main verification technique, and is a significant part of the development cost and resources. Some code reading and other techniques are used.
- **Inspected level (2)** Verification and validation is done by manual inspection of all descriptions, with emphasis on requirements and design specifications on regular basis. Code reading is done planned. Support for creating test scripts.
- **Animated level (3)** Functionality is validated through animation (simulation) of functional design. Transition from functional design to implementation is verified through inspection of the implementation design. Testing is performed to verify that the functionality is consistent and correct implemented, and that non-functional requirements are correct.
- **Analyzed level (4)** A formal static and dynamic analysis is performed.
- **Synthesized level (5)** Functionality is synthesized from requirements from environment, when functionality is analyzed and proven consistent. Automatic transformation to software through error free realization.

c) Transformation criteria :

- **Compilation level (1)** Transformation by compilation of programming language followed by linking and loading.
- **Partial program generation level (2)** Templates for software are generated automatically and is completed manually from functional design.
- **Program generation level (3)** Functional design is formal and complete that large part of the software can be generated without modifications.
- **System generation level (4)** Variations of code are generated from the same functional design. Support for composition and configuration is possible from component library.
- **Design transformation level (5)** Support of transformation from functional and non functional requirements.

d) Product Management criteria :

- **Initial (1)** Files are maintained by the individual programmer
- **Versions-oriented (2)** Defined routines for version control of official components using simple tools like sccs, pvcs, subversion, github or similar.
- **Product-oriented (3)** Routines for error reports and change management are in place. Full overview of deliverables which can be easily reconstructed.
- **Integrated (4)** All descriptions are versioned in a common and integrated manner. Support to generate and configure deliverables and executables from library.
- **Needs-oriented (5)** (Not defined)

e) Reuse criteria :

- **Initial level (1)** No systematic reuse, but some reuse of obvious component occur, e.g. Interface to common hardware components, operating systems, databases or user interface.
- **Product-oriented level (2)** Variants of products can be generated from a base of product specific components. The components are customized product architecture and are rarely used in other products. High degree of reuse, as a result of customization of customer needs.
- **Domain-oriented level (3)** Strategic decision on reuse. Increasing base of reusable components that are used in several products in an application domain. Standardized architecture are in use. Conscious efforts in creating reusable components are encouraged. Tool support for search for and insertion of new components.
- **Integrated level (4)** Reuse is fully integrated in process and organization. Reuse practice in all development phases. Reuse tools are fully integrated with development tools.
- **Global level (5)** Library is integrated in a network of reuse libraries. Libraries are shared between other companies. Contribution to standardization of architecture and components in relevant domains.

f) Process criteria (similar to CMM) :

- **Initial level (1)** No common methodology, ad-hoc and at times chaotic. No basis for systematic changes in workflow. Dependent on individuals and there are no documented procedures.

- **Repeatable level (2)** Basic procedures are established to monitor cost, progress and functionality. Experiences from earlier projects can be repeated in new projects.
- **Defined level (3)** Procedures are defined, standardized and integrated in a well defined process covering management and technical execution in the organization. All projects use a documented and approved version of the process for development, maintenance and quality assurance.
- **Managed level (4)** Quality of all parts, products and projects are continuously assessed. Measures of process and product quality are collected. Cost estimation are accurate.
- **Optimized level (5)** The process is continuously optimized through quantitative and qualitative feedback from the process and new ideas and techniques are systematically tried out.

References

1. Geir Melby, et.al. : SISU final report L-2301 (In Norwegian)(1996)
2. Geir Melby, et.al. : SISU plan (In Norwegian) (1993)
3. CMM, from SEI, Software Engineering Institute at Carnegie Mellon
4. Bræk, Rolv; Haugen, Øystein : Engineering Real Time Systems ISBN 0-13-034448-6, Prentice Hall (1993)
5. ProgGen[1], The Program Generator. 2000, Oct 12
6. Teletronik 4 2000[2], Languages in telecommunications
7. Teletronik 4 2000[3], Implementing from SDL
8. TiME[4], The Integrated Methodology (2016) (online version[5])
9. TST, Telox SDL Tools
10. T. Stålhane, A Cost-Benefit Analysis for the SISU II project, SINTEF DELAB. November 24, 1994.
11. ThingML[6], Tool chain to support Internet Of Things
12. Reactive Blocks, Tool from BitReactive[7]

[1] https://www.ercim.eu/publication/Ercim_News/enw36/nilsen.html

[2] https://www.telenor.com/innovation/telektronikk/archive/

[3] https://www.telenor.com/innovation/telektronikk/archive/

[4] https://www.researchgate.net/publication/250604578_TIMe_-_The_Integrated_Method

[5] http://www.sdl-forum.org/Tools/TIMe/HTML/elb/StartHere.htm

[6] http://thingml.org/

[7] http://www.bitreactive.com/