Communication Gaps in a Tender Process

The supplier's understanding of the requirements

Søren Lauesen, slauesen@itu.dk
IT University of Copenhagen, Rued Langgaardsvej 7, DK-2300 Copenhagen S
Phone: +45 7218 5153, Fax: +45 7218 5001

Jens Peder Vium, iqm@JPVium.dk
Skindbjergvej 5, DK-9520 Skørping

Abstract. Large IT systems are often acquired in a tender process where the customer states the system requirements, a number of suppliers submit their proposals, and the customer selects one of them. Usually the supplier uses an existing system as basis for his proposal. He adapts it more or less to the customer's requirements.

This paper is a study of a specific tender process. The customer was a Danish municipality that supplied electrical power, water, gas, garbage collection, etc. for around 100,000 households. The customer wanted a new system for meter inspection, invoicing, planning the meter inspector's routes, etc.

We have studied the requirements, how they were perceived by the suppliers, and how they were intended by the customer. The main findings are that the parties didn't understand each other, although the suppliers sometimes pretended that they did so. One consequence was that the business goals pursued by the customer were not properly achieved. Among the causes of this were an excessive democratic elicitation process and an inadequate use of requirements techniques, particularly use cases. There were also issues that existing requirements techniques couldn't deal with, for instance integration with future systems.

Keywords: COTS, tender process, business goals, software requirements, integration requirements, use cases.

1. Introduction

Large IT systems are often acquired in a tender process where the customer states the system requirements, a number of suppliers submit their proposals, and the customer selects one of them. Usually the supplier uses a COTS system as basis for his proposal. He adapts it more or less to the customer's requirements. (COTS means Commercial Off The Shelf - a system that is not made for a single customer, but for a market.)

Why buy COTS instead of making something yourself? Because it is much cheaper! You get vastly more functionality with much higher quality than what you could develop yourself. The disadvantage is that your organization has to adapt to the new system, rather than the opposite. Typical figures are that 98% of the delivered functionality is in the COTS part, while 50% of the price covers tailor-made parts and data conversion.

This paper reports a study of a specific tender process. The customer was a Danish municipality that supplied electrical power, water, gas, district heating and garbage collection for around 100,000 households. The customer wanted a new system for meter inspection, invoicing, planning the meter inspector's routes, etc. One of the expected benefits was handling all the supply types through the same system rather than as separate applications.

Five suppliers sent a proposal. The price for the selected system over a five-year period was around 3 million Euro. The price for the most expensive proposal was around 9 million Euro. (These prices were the customer's calculation of the entire cost, including software, hardware, operating costs, and maintenance. Parts of these costs were not covered by the proposal, for instance hardware to some extent.)

Public organizations in the EU have to follow strict rules for such acquisition processes. The most common approach for software acquisition is *Restricted Tendering*, where a number of suppliers are prequalified based on general criteria such as their financial health and their experience in the application domain. Next they receive the tender documents, including the requirements specification, and write a proposal. This approach gives little opportunity for dialog between the parties, and much less negotiation. In the present case study the customer chose another approach: the *Negotiated Procedure*. It is rather similar to restricted tendering, but allows a dialog between the customer and each vendor after the first proposal. After the dialogue, the supplier is allowed to send a revised proposal.

In both cases the customer's demands must be expressed in a requirements specification, but it is hard to write one without being either too specific or too vague. If requirements are too specific, a good supplier may be rejected because his system doesn't meet the details the customer asks for, although it meets the customer's real demands. If requirements are too vague, the supplier doesn't know what he is supposed to deliver and cannot calculate a price. As a result the customer cannot compare the proposals in a fair manner. Furthermore, the requirements are not verifiable at the time of delivery, so the project may end up in costly conflicts.

How do the parties experience such a tender process? Do they understand each other? What works well and what causes problems? How does the customer select a supplier and how does the supplier perceive the process?

The research literature on COTS acquisitions recommends an iterative approach where the customer studies the various products on the market and gradually defines his requirements and narrows down the choices (e.g. Maiden & Ncube, 1998; Albert & Brownsword, 2002; Gorton & Liu, 2002). Unfortunately this approach doesn't work in the public tender case because products change all the time and are too complex to study. The supplier may also decide to expand his system in the direction wanted by the customer. Furthermore, the selection process must be open and defined by the requirements. Existing work has looked at the selection process from the customer's viewpoint. We have seen no work discuss the tender process from the supplier's perspective.

The investigation behind this paper was made by the authors as a pure research project. Neither customer, nor suppliers have contributed financially to it.

2. The acquisition process

March 1st, 2002, the customer sent the tender documents, including the requirements, to the five suppliers that had been prequalified, A, B, C, D and E.

March 27, 2002, they submitted their proposal. Until March 11 they could ask questions in writing. Their questions and the replies were made available to all the suppliers. However, the suppliers found it impossible to use this possibility within the time limits.

Until April 24, the customer conducted presentation meetings with the suppliers, discussed prices, and asked for revised proposals.

May 1st, the customer had chosen supplier A and they signed the contract. Here are the main points in the decision:

Supplier A and B had roughly the same price while C, D and E were about 100 to 200% more expensive. In order to compare prices, the customer had included various additional costs such as the necessary hardware. Even the cheapest proposals were more expensive than anticipated.

During the presentation of the proposals, the most expensive supplier (E) made a very positive impression and convinced the customer that E's proposal gave significant business advantages - exceeding the stated requirements. The proposal was based on the SAP COTS product, which the customer had considered in another case four years ago. At that time SAP had not looked attractive (expensive and very complex), but the situation had clearly changed. The customer considered the advantages - among others the cost savings - but concluded that they couldn't justify the higher price. E was already supplier of a similar system in the capital, but the population in the customer's area was much smaller and made the IT costs per capita much higher. However, E had calculated this and estimated that the cost per capita would still be acceptable.

The other proposals didn't provide visible advantages compared with the two cheapest, so the customer had to choose between supplier A and B. The customer wrote an internal report with a detailed comparison of the two proposals. They were compared against the seven stated evaluation criteria and against the extent to which they met the customer's business objectives. Supplier A was number one on all points except references to customers of the proposed system (the proposed system was under development). For this reason the choice was very robust against alternative weights of the criteria.

Against supplier B was the facts that (1) the solution was operational for only one kind of commodity (electrical power); (2) it was developed in another country while B was only a vendor. B's greatest advantage was that they already had an operational solution for the new legislation, while A proposed a temporary solution until the final solution was operational.

In many ways the decision was easy to make due to fortunate circumstances that allowed a gradual narrow down of the winner. Three proposals were vastly more expensive and didn't provide significant advantages. They were discarded. Among the remaining two, one scored higher on nearly all points. In other projects the choice is much harder because one proposal scores high on one factor, while others score high on other factors. The choice then depends on the weighting of the factors. Ncube and Dean (2002) review solutions to this.

3. Course of the project after the contract

January, 2003, the most critical part of the system - support for the open power market - was delivered on schedule as a temporary solution that integrated with the customer's old billing system. The rest of the system was scheduled for delivery June 1st.

The supplier's analysis report was scheduled for the end of February. It was delivered on schedule, and consisted of all the requirements and use cases with explanation of the detailed technical solutions for each requirement and each use-case step. Particularly for the integration requirements, the description was very detailed and included the protocol and the fields involved. It was an excellent technical document that ensured traceability from requirements to technical design.

However, the customer rejected it. It was too technical. He had expected user-oriented descriptions, such as screen shots, to see how his tasks would be supported. Two weeks later the supplier delivered such a report, and the customer accepted it.

In spite of the detailed analysis, integration of the new system with existing systems caused many problems, particularly relating to the mobile equipment. The rest of development proceeded as planned. The customer had dropped a few requirements and added some others in agreement with the supplier.

The system was not yet a true COTS product, but one being developed. The supplier had an old COTS system that was being completely reengineered. The old system platform was home-grown, the new was an extension of an existing ERP system (Axapta developed by Navision, now Microsoft Business Solutions). The supplier had estimated the reengineering effort based on his experience with the old platform. It turned out that he had been much too pessimistic. The Axapta development tools greatly speeded up the work.

Shortly before the planned delivery, June 1st, the customer's auditors decided to delay the deployment three months in order to check the data conversion thoroughly. They had heard other service providers run into financial disasters because billing went wrong for thousands of customers after similar data conversions. These three months also gave the parties space for preparing the deployment better.

During September the system was gradually deployed. There were minor problems, of course, but nothing serious. However, the mobile solution and the self-service internet site were still under development and test. Early October 2003 the message was: *It will be ready in a month*. Early March 2004 the message was: *The mobile solution is now used in practice, but concurrently with the manual method. Cut over in a month. And we are surprised at the number of issues we took for granted, for instance password problems for the mobile, need for manual configuration of 30 mobiles whenever something is to be changed, etc*.

4. Research method

The idea to make the investigation is due to Jens-Peder Vium. He knew the customer and saw the project as an opportunity to investigate the supplier's point-of-view, and also as an opportunity to study how the relatively new use case approach (Tasks & Support, Lauesen, 2002 and 2003) worked in this kind of project. The customer soon accepted the idea.

May 1, 2002, the customer had signed the contract with the winner. The planned delivery of the system was June 1, 2003.

October 25, 2002, the customer and we sent a joint letter to all the suppliers asking for permission to study the proposals. It went surprisingly smoothly. We promised of course to treat everything confidentially until all parties had agreed otherwise. The suppliers didn't even know whom the other suppliers were. At this point in time, we didn't know whether they were willing to reveal their identity in the final report.

November 19, 2002, all parties had accepted. They could see the benefit of knowing what went on behind the scene, and were willing to contribute.

December 28, 2002, the customer gave us a copy of the tender documents and all the proposals. Copying had been delayed because the customer moved to new premises.

February 6, 2003, we met with the customer's leader of the project to get his comments on the tender documents, the proposals, the selection process, etc. Lauesen had skimmed the five proposals (each around 400 pages) and studied selected parts in detail. At this point in time he knew who had been selected, but not why.

March 6, 2003, the customer had accepted our summary of the meeting, including the justification for choosing the winner. We had also got acceptance of an anonymous version of the summary where the supplier names were not mentioned. We could show this version to the suppliers.

From March 13 to June 2 the authors interviewed all the suppliers, wrote a summary and had it accepted by the interviewees. All interviews were conducted in the same way:

We met with two to three staff members who had been directly involved in writing the proposal and later presenting it at a meeting with the customer. These staff members were senior people with much experience in proposal writing. First we asked in a semi-structured fashion about how they had experienced the tender process. Next we gave them the three page summary of our meeting with the customer and allowed them ample time to read it (around 15 minutes). We then discussed the summary and any new issues that arose. Finally Lauesen explained his own subjective assessment of strengths and weaknesses in the proposal.

All suppliers expressed that the meeting had been very rewarding, that the customer's point-of-view had been surprising and fair, and that they looked forward to seeing the final report. A few days later they got a summary of the meeting for acceptance. They all had minor corrections, but were otherwise satisfied.

As we went through the interviews, the first three suppliers didn't mind that their name was revealed. However, we gradually judged that the results were so sensitive that they had to be anonymous. At the end, all suppliers got an anonymous summary of the part relating to themselves, and were asked to accept it for publication. Also here there were minor corrections.

July 22, 2003, we had acceptance from everybody. The summaries for each supplier are available in Lauesen & Vium (2003). We extracted the key problems from the raw interviews and the confirmed summaries, and then wrote the final report. August 2003 we reviewed it with the customer and made additional changes. To confirm the findings, we transformed the list of problems to a questionnaire form where the supplier could confirm or reject the problems on a 5-point Lickert scale.

September 19, 2003, we sent the final report and the questionnaire to the people we had interviewed. We got positive responses everywhere, but only two suppliers took the time to fill in the questionnaire (supplier B and C). The low reply rate was no surprise since 17 months had passed since they participated in the tender process.

However, the two replies confirmed the problems. Supplier C also added a new twist to problem I (that the suppliers wasted money on a proposal that was too costly to win). C couldn't have saved the money up front because their price and the benefit to the customer were revealed during the very work with the proposal.

5. Observations and problems

Below we have arranged the problems according to their main cause.

1. Some problems were caused by requirements being too close to the present way of doing things (too close to "as-is"). This gave suppliers little chance to show that their solution allowed a more efficient way of doing things.
2. Some problems were caused by requirements being too vague or too open. This made it hard for the supplier to suggest a solution, particularly if he didn't understand the customer's real needs. (It also made it harder to validate the requirements, of course.)
3. Some problems were caused by business goals and critical issues not being explained. This made it hard for suppliers to explain how their system could support the goals.
4. Techniques for dealing with the above issues are known, but the customer didn't master them sufficiently. However, other issues were much harder and there were no available requirements techniques to deal with them. It is no wonder that the customer couldn't come up with adequate requirements in these areas. One example is requirements for integrating the new system with future systems.
5. Finally, some problems were not related to the requirements as such, but to the elicitation and acquisition *process*. One example is an excessive user involvement that made it hard to see the common patterns in the tasks to be supported.

The problems were published in some earlier papers (e.g. Lauesen & Vium, 2003) where they were arranged in another sequence and structured slightly differently. To avoid confusion, the problems are numbered alphabetically below, while they were numbered numerically in the earlier papers.

5.1. Requirements too close to "as-is"

In this section we look at problems caused by requirements being too close to the present way of doing things.

Problem A. Use cases too detailed

The functional requirements were to a large extent based on use cases. The use cases described the customer's essential work processes without describing a specific solution (Tasks & Support, Lauesen, 2002, 2003). The suppliers found most of the use cases quite successful, but for some use cases they wondered why the customer wanted to carry out the tasks in this particular fashion. These use cases were too detailed and corresponded too closely to the existing work procedures. They didn't allow the more efficient procedures that a new system could support. (See the discussion in section 6.3).

Problem B. Specification too long

The requirements specification was a 130-page document. It contained 119 requirements expressed in the traditional fashion (required features and product behavior) in addition to 33 use cases that had to be supported. Traditional requirements specifications written by large consultancy companies are usually of similar lengths, but without use cases. Instead they have a lot of traditional requirements - typically around 1000 in a project like this.

Although the requirements specification wasn't longer than in similar cases, the suppliers found that there were too many requirements and too many use cases. It should be possible to express the real demands much shorter and in a more general way. (See the discussion in section 6.3).

5.2. Requirements too open or too vague

Most of the requirements were quite open in the sense that they left it to the supplier to specify an adequate solution. In general the suppliers handled these requirements well because their COTS product included all the necessary features. For instance they had adequate features for finding a consumer in the database, although the requirements said nothing about it.

In many cases, however, the openness caused problems. We will look at them in this section.

Problem C. The customer assumed that the supplier knew

The system covered many application areas. Some suppliers had experience with some areas, others with other areas. The customer assumed that all suppliers knew all areas, and he believed the prequalification had ensured it.

However, suppliers were not sure what many of the requirements meant, particularly in areas where they had little expertise. Furthermore, the suppliers couldn't see the customer's real needs from the specification (section 6 shows several examples.)

Problem D. Taking the financial risk of vague requirements

The suppliers dealt with the uncertainties in one of three ways: (1) They replied that a more detailed analysis was necessary in specific areas, and they couldn't quote a price until it had been carried out. (2) They replied that they couldn't describe the *solution* until they had carried out a closer analysis, but the solution was included in the quoted price. (3) They described the proposed solution, which was covered by the quoted price. Two suppliers used reply-type one to a large extent, with the consequence that the customer found it hard to compare prices. Two suppliers used reply-type three to a large extent, which gave the customer the problem of assessing whether the described solution met his real needs. The selected supplier used mainly reply-type two, and in this way showed empathy by taking over the financial risk of the uncertainty.

In general, customers prefer a supplier that takes the financial risk of vague requirements, but most suppliers will not take this risk.

Problem E. Telling the customer that a requirement is wrong

What should a supplier do when he can see that the customer's requirements are obscure - or that the customer has misunderstood something. Pretend that everything is fine or correct the customer?

Some suppliers were reluctant to point out weaknesses in the customer's specification and explain how they perceived his real needs. They wouldn't like offending the customer. The result was that the customer perceived the reply as obscure. Other suppliers succeeded well in showing that they understood the customer's demands, and correcting his perception of how to meet the demands. All in a supportive and constructive fashion, of course.

Problem F. High risk when no supplier has a solution at present

For the open requirements, the supplier had to describe his solution in each particular area. This may be a good approach if the supplier actually has a solution.

However, for some of the open requirements (for instance integration with mobile equipment), none of the suppliers had a solution. The 26 days available for writing a proposal didn't allow them to invent one. As a result, the customer got only statements of intent and strategy, but no specific

solutions that he could compare. At the time of signing the contract, there were no verifiable requirements in the area. Mobile equipment turned out to be the most troublesome area during deployment of the winning system, and there was no basis for deciding when the system was good enough (see sections 3 and 6.2).

5.3. Goals and present problems hidden

In this section we look at requirements issues related to business goals and problems in the present way of doing things.

Problem G. Impossible to see how business goals were to be met

The suppliers tried to help the customer meet his business goals. Although the goals were listed in the tender documents, the suppliers didn't understand them and couldn't see how the customer expected to reach the goals. The only clearly visible business goal was that the new legislation about the deregulated power market had to be supported. However, the requirements in this area were completely open. The customer simply asked the supplier to describe his solution in this area.

The customer expected that the suppliers knew about the goals already, but largely they didn't. (See the discussion in section 6.4).

It is remarkable that a Danish Government investigation of public IT projects mentions lack of verifiable business goals as a major problem (Bonnerup, 2001). Unfortunately the report doesn't mention that the business goals should be reflected clearly in the requirements - or how to do it. The report only talks about roles and responsibilities - particular for management. Our own experience is that business goals and their relation to requirements is a common problem found in most IT projects. Apparently it doesn't suffice to tell management that they are responsible - management and IT people don't seem to know how to deal with the responsibility. (Don't believe that an IT tracing tool would help. It would only increase the barrier. This kind of tracing can easily be done manually, once you understand the principle.)

Gotel & Finkelstein (1994) give an overview of the requirements traceability problem, and observe that the most important problem is pre-requirements tracing, for instance tracing between business goals and requirements. Feblowitz & Greenspan (1998) show a systematic way to trace business goals to requirements.

Problem H. Impossible to see the problems to cure

Some of the requirements intended to remove a problem in the way things were done today. However, the requirements didn't mention the problem, and as a result some suppliers missed the whole point.

As an example, the customer mentioned that billing should be per customer, but didn't mention that at present billing was per point of consumption, even when the customer had several points of consumption. Understanding this problem would have helped the supplier understand the complexity of the data conversion involved. During system deployment, this conversion issue caused serious problems.

Problem I. What price range did the customer expect?

The suppliers calculated the price for their proposal based on the rather strict rules they had. Some of them also estimated whether the customer was willing to pay this amount and what competitors would offer, but these estimates failed. The winner just stuck to his standard calculation - without considering what the customer was willing to pay, the value to the customer, or competitor's prices.

Several suppliers misjudged what the customer expected, and for this reason spent significant amounts on a proposal that couldn't win. The customer didn't adequately signal the expected level of ambition and price. These suppliers were able to estimate their costs very early, partly because the COTS license fee was a major factor and it was known in advance. Supplier C, however, later explained that they couldn't have saved the money up front because they couldn’t calculate their costs so early. They had to analyze the requirements carefully and imagine solutions, in order to find the price and the benefit to the customer. This required that they carried out most of the work needed anyway to write a reply.

5.4. Techniques lacking

In some areas there are no available requirement techniques. This project has two examples.

Problem J. How to specify integration with present and future systems?

The largest technical problems were related to the integration (interoperability) requirements. The system had to be integrated with around 20 other systems (*external systems*). Some of these were legacy systems, others being purchased at the same time, and some were just plans for the future. For a supplier who had not been closely involved in such an external system, it was obscure what the integration had to be used for in daily operation, and it was very costly to figure out what was technically feasible and what it would cost.

The integration requirements are hard to describe in such a way that they clearly specify the customer's needs - without prescribing a specific solution. (See the examples in section 6.2). Inspired by this tender project, we later came up with a way to deal with integration requirements in COTS tenders (Lauesen, 2004).

Problem K. Defining the project scope

Several suppliers explained that it would be an advantage if the customer replaced some of his old systems with the new one, for instance replaced his old payment management system with the equivalent one in the new system. The customer had considered these suggestions, but since the old system was integrated with entirely different systems, e.g. tax and social benefit systems, it would be an overwhelming task.

It is hard to make a "clean cut" in a confusion of existing, integrated systems. Among other things, it depends on what is available from the suppliers. For the same reason it is hard to decide what should and should not be included in the tender.

5.5. Inadequate acquisition process

Some problems were not related to the requirements as such, but to the elicitation and acquisition process. We look at them in this section.

Problem L. Excessive user participation

The business goal issues and the unsuitable use cases were closely related to the way requirements were elicited. The most successful projects we have seen, used a small team that wrote use cases and other requirements, had the various user groups review and revise them, and ensured that the business goals were reflected in the requirements. The team had a broad understanding of all the application domains as well as of requirements techniques. In the present project another approach was used.

The requirements were written in a "democratic process" where each department wrote their own requirements and tried to negotiate with other departments. This didn't leave space for designing new work procedures or figuring out how to meet business goals.

The Government report (Bonnerup, 2001) mentions this issue as "excessive user participation" that blocks looking at the large-scale work processes.

Problem M. Customer involvement during deployment

A successful project requires cooperation between customer and supplier. Many customers imagine that the supplier just "rolls in the system and plugs it into the power outlet".

Most of the suppliers believed that the customer had grossly underestimated the effort needed by his own staff during the project. They considered this the highest risk in the project. As an example, the winner had suggested to develop the customer-specific parts with Extreme Programming, where the customer participates all the way. The customer refused since he couldn't set off the necessary staff for this. Highly qualified staff was needed and "they don't grow on trees".

Problem N. Why we won or lost

Suppliers often don't understand why they lost and why the winner won. The result is that it is difficult for them to improve their approaches. In this project they imagined that the price was the main factor, that the customer chose an immature winner in spite of the high risk of doing so, etc.

However, the price is not the main factor. We often see a supplier win in spite of a high price, because he can convince the customer that he gets something that exceeds the expectations. In this project, the most expensive supplier convinced the customer that he could obtain large cost savings. The customer calculated the benefits carefully, but had to conclude that the price exceeded the pain threshold. Amazingly, the supplier was unaware of being so close at becoming the winner.

Problem O. Linear weighting of criteria

Some suppliers pointed out that the evaluation criteria in the tender documents were unsuitable. As an example, references from similar projects counted with only 5%. If an inexperienced supplier (without references) offered to meet all the requirements at a very low price, he would win the contract

although his chances of failing would be immense. (These suppliers thought that this was why the customer had chosen the supplier he did.)

The customer's reason for weighting the references so low was that the suppliers had been prequalified and their references found adequate. One may wonder why this factor was included in the final evaluation at all. Possibly the reason was that the customer felt that the prequalification was inadequate. The prequalification concerned the supplier's products in general, and you couldn't conclude anything about his abilities in the specific application area. Evaluating the references once more for the proposal, was possibly a compensation for this, but with the low weight it couldn't have much effect. This is just one example that a linear weighting of evaluation factors is unsuitable.

The weighting of the evaluation factors should not be linear. Some factors must have a minimum value. Below this, the proposal is unacceptable.

Problem P. Inadequate tender rules

The most common tender approach is *Restricted Tendering*, where a number of suppliers are prequalified. This approach doesn't allow dialog or negotiation.

Weaknesses in requirements usually cause huge problems in restricted tendering. In this project the customer had chosen the *Negotiated Procedure*. It allowed the parties to understand each other far better, and allowed the supplier to adjust his proposal after a first evaluation. Still it would be advantageous to everybody involved if the customer could discuss possibilities with the prequalified suppliers before writing the requirements. To give equal opportunities, however, the requirements must be stated in a completely solution-independent way. This is hard with traditional requirements, but in the present project, the customer actually knew how to do it.

Problem Q. Unrealistic deadlines

Each supplier spent around 500-1000 work hours on writing the proposal. The cost is huge, and furthermore many kinds of experience are involved. The tender documents allowed 26 days for writing the proposal, so it was a crash effort to do all the work in this short period. In this case, the reason for the short deadline was new legislation in addition to the fact that the customer had been forced to cancel the first version of the tender because two of the suppliers suddenly merged. EU rules allow a time limit of 26 days if notice has been given 52 days in advance. The normal limit is 40 days, but even this seems very demanding.

The Government report (Bonnerup, 2001) also says that the time limits are unrealistic, and adds that the 40 day time limit in practice is a maximum, partly because the customer spends too long time writing the requirements.

On the other hand, some suppliers said that they wouldn't like a much longer deadline since it would make them invest too much in the proposal.

6. Examples of requirements

Below we give an overview of the entire requirements specification. We give details for some of the requirements in order to illustrate the problems.

6.1. Requirements in traditional form

In the specification, requirements in traditional form are numbered from G1 to G119. They are consistently stated in four columns. Example:

ID	Requirement	Suggested solution	Code
G12	The system shall provide an electronic signature for internal controls and acceptance procedures		

Column two contains the customer's requirement. In column three the supplier writes his proposed solution or refers to an appendix. In column 4 he shows with a code (1 to 4) whether the solution is part of a COTS system (1); a customer-specific addition covered by the maintenance contract (2); a customer-specific modification outside usual maintenance (3); not offered (4).

Broad requirements. Requirements G1-G57 are about meeting laws and rules, security, integrity, usability, report generator, etc. Examples:

G2: [The system shall fulfill] the annual accounts regulation.

G12: The system shall provide an electronic signature for internal controls and acceptance procedures.
G22: The system shall be easily accessible to users who don't use it on a regular basis.

Comments: Requirements such as these are frequently used in business applications, and most of them don't cause problems, although some suppliers are reluctant to take responsibility for laws and rules, particularly future ones. The requirements are understandable to both parties and reasonably verifiable. However, G22 (usability) is problematic. How could we verify it? Most suppliers reply something like "our system is Microsoft-based and thus easy to use". This doesn't say much, of course. We all know Microsoft-based systems that look nice but are hard to learn. There are more precise ways of specifying usability without specifying the user interface (e.g. Lauesen 2002). In this project, the chosen system appeared to be sufficiently easy to use, so the vague requirement had no impact.

Data requirements. Requirements G58-G63 specify the data to be filed. As an example the specification shows a simplified data model (E/R model) followed by this requirement:
G58: The system shall support the described entities and relationships for contract, consumer/customer, point of consumption, . . .

Comment: The data description contained little about the fields in the entities. However, the data conversion requirements (G97-G104) referred to an appendix that gave all the details of the data in the existing system. This should allow the supplier to check that his system included all that was needed. In general, data descriptions are good requirements. Actually they should have been elaborated to better explain what the data was used for. Many of the use cases could have been vastly simplified in this way, as explained in section 6.3.

(Requirements G64-G87 deal with integration with other systems. They are explained in section 6.2.)

Platform requirements, etc. Requirements G88-G96 specify the technical platform, documentation trail for customer calls, capacity, response time and availability. Example:
G89: Data shall be accessible in a relational database based on open standards. The access must be at no cost to the customer.

Comment: There are no big problems here. It is a good idea to write that access must be at no cost. The customer had experienced cases where he couldn't get direct access to his own data because the supplier claimed that the data model was his intellectual property.

Data conversion. Requirements G97-G104 specify data conversion. Example:
G98: The following is to be converted. Data about existing points of consumption (about 100,000). Includes master data about the points, connected meters, notes and technical installations. The data is described in detail in appendix X.

Comments: No problem. A large part of the proposed price was set off for data conversion.

Development strategy. Requirements G105-G109 deal with the development strategy. Example:
G106: It shall be described how the supplier actively will ensure that the system at all times, with the least possible delay, meets future laws, etc. - including changes in duty legislation.

Comments: These requirements cannot be verified at delivery time, but they are the customer's attempt to get an impression of the supplier's reliability in the future. The customer didn't really check that the current development process at proposal time matched the submitted plans. This might have been a good idea, however.

(Requirements G110-G112 deal with integration with mobile equipment. They are explained in section 6.2.)

Open power market. Requirement G113 dealt with the open power market. It starts with a general description of the political situation in this area. It points out that legislation is not yet in place although the open market must exist at the end of the year. Next follows the requirement:
G113: The supplier shall provide a coherent description of how the solution supports the open power market.

Comment: This requirement is similar to the one for mobile equipment, but for the power market, several of the suppliers already had a solution or had planned one in detail.

G114-G119: Optional deliverables. Not discussed here.

6.2. Integration requirements

Requirements G64-G87 and G110-G112 specify integration with around 20 other systems. Here are three examples.

Document management. The specification contains a general description of the plans in the municipality for introducing electronic case and document management, followed by this requirement:

G64: The system shall support and be open for two-way integration with the municipality's electronic case and document management system.

Comment: It is not clear what the supplier should reply to G64. If he already has a solution in this area, he might describe it. Otherwise he needs additional information about the customer's situation in order to come up with a solution.

Civil registration. The specification describes how the Decentralized Civil Registration system (DCR) is used today by the payment systems. It is explained that customer addresses are not stored in the payment systems, but are retrieved directly from DCR in a batch-wise fashion. After this explanation follows this requirement:

G73: Integration must be made with the DCR system, cf. the above description.

Comment: For G73 there are two problems: (1) knowing the technical interface to DCR (which was not described at all in the tender documents); (2) understanding the business processes that used the integration, in order to judge whether a replicated database was needed, a batch-transfer for each customer lookup was adequate, etc. (See Guo, 2002, for an overview of the possible integration techniques).

Mobile data processing. The specification mentions the work situations where users might use mobile equipment, for instance meter reading and repair. It is pointed out that the customer hasn't acquired mobile equipment but expects the supplier to state the requirements to the future equipment. Next follows the requirements G110-G112, like this:

G110: The supplier shall provide a coherent description of how the solution supports the potential for using mobile equipment. It must be stated what can be provided at delivery time and . . .

G111: The supplier shall specify requirements to necessary/relevant equipment including interfaces and . . .

Comment: Here the customer asks the supplier to provide the complete business concept. It might be a good idea if the supplier had such a concept, but in this case the suppliers had nothing like it. Furthermore it was obscure which business processes were involved (apart from a few headings) and how mobile equipment could improve them. Requirement G111 is a good way to have the supplier specify what is to be bought. However, in this case it caused many problems because none of the suppliers had planned a solution and none had time to come up with one.

The mobile data processing was much delayed, and the parties gradually realized that there were many issues in the area that they had not imagined, for instance how to update software versions on the mobile, passwords, etc. Furthermore there were no requirements that could help decide when the solution was good enough. In this area the project proceeded much like traditional projects without requirements.

At our meetings with the customer, he confirmed that the integration issues were the largest problems. It was often obscure what the suppliers proposed and how far they had a solution already (problem E). Furthermore many of the systems to be integrated were not fully acquired because they were part of concurrent tender processes in other municipal areas. An alternative to the troublesome integration would be to contract with a consortium that took care of the entire integration. The customer had rejected this possibility because, as he said, “there would be only one supplier to choose between".

A few papers report on experiences with integrating COTS products (Boehm & Abts, 1999; Balk & Kedia, 2000; Liu & Gorton, 2003). However, there is a long way to go from recognizing the integrator's problems to stating integration requirements in a COTS tender. We have seen no papers

describing how to deal with integration requirements in a tender process. Inspired by this and other acquisition projects, we have now developed some solutions (Lauesen, 2004).

6.3. Use cases (task descriptions)

The majority of the functional requirements are specified as use cases, numbered from 1 to 33. Use cases may be expressed in many ways. In this case, the customer was inspired by the version of use cases called *tasks and support*, successfully used in West Zealand county (Lauesen, 2003). These use cases describe a task to be carried out jointly by user and computer - without being explicit about who does what. The requirement is that the new system must support these tasks, and the supplier specifies how his solution will do this.

We first give an overview of the 33 task descriptions (use cases), then we show one of them in detail. We have experienced in other projects that task descriptions can give great advantages, but the first time a project team tries to use them without guidance, they don't succeed so well. This case is no exception.

Task descriptions:

1-3 Marketing, customer creation, creating a point of consumption.
4-5 Administration on behalf of other power suppliers, sales of Renewable Energy Certificates.
6-7 Maintenance of customer data, change of power supplier.
8 Maintenance of real-estate data based on electronic data transfer from the public registry.
9-14 Handling returned mail, address changes, customer calls, self service through the Web.
15-21 Receiving meters for stock, delivering meters from stock, mounting and dismounting meters, sample tests of meters, etc.
22-25 Customer-initiated meter reading, mailing self-service meter cards, reading meters at the property, recording and checking meter readings.
26-29 Invoicing, change of invoice agreement, payment handling, reminder procedure.
30 Design of invoice layout, sometimes customer-specific electronic layouts.
31 Transferring invoices to Social Security handling.
32-33 Optional deliverables. Not discussed here.

Example of a task description from the requirements specification

Use case 2	**Creating a customer relationship**		
Goal:	To state the conditions for the new customer relationship and activate the relationship.		
Frequency:	Each year 12,000 new customers are created in connection with moving and 2,500 for other reasons. After marketing campaigns . . . there will be a high number.		
Step		**Example of solution**	**Code**
1.	The customer is created in the system with all master data	If there already are data on the customer, they are re-used or updated	
2.	The customer is connected to one or more customer groups	Customer groups can be defined based on a set of customer properties, for instance type of housing, family type, or consumption pattern . . .	
3.	The customer relationship is composed of an arbitrary number of products and services		
4.	Invoice agreements are stipulated for each product or service, and recorded	. . .	
5.	The supply agreement is activated		
	Variants		
1.a.	Not all master data is available	. . .	
1.b.	The customer relationship is established as part of a group agreement		
1.c.	Customer concepts: The Garbage Collection Authority needs to distinguish between		

	owner and tenant relationships		
3.a.	A new product or service is defined and connected to the customer		
3.b.	The customer has accepted a proposal on which the customer relationship is based	. . .	
3.c.	Additional types of contracts: The Garbage Collection Authority works with additional types of contracts, for instance key contracts, contracts about container rental, and warranty declarations		
3.d.	For garbage collection services, removal day or removal frequency is defined		
3.e.	Garbage collection services are connected to the route plan		
4.a.	The invoice agreement is stated individually for each product or service in the relationship		
4.b.	The system must handle that each customer may have several garbage collection agreements that must be invoiced together		
	General requirements		
1	The system supports the use case and the variants		
2	It must be possible to make agreements with a group of customers	An agreement is made with an association that . . .	
3	It must be possible to complete the task without having all the data		
4	It must be possible at any time to follow up on order status, services, and supply contracts with the customer		
5	When a supply agreement is activated, a message must automatically be sent to the real-estate system, cf. the interface		
6	It must be possible to print a contract or terms of delivery from the system		

Comment: This use case (task description) illustrates the basic idea. The task is described in column two, first as the basic steps, next as the variants. The description should *not* define what the computer does and what the user does. In principle some steps could be fully manual, others partially manual, and some fully automatic. It all depends on what the supplier can provide.

In column three the customer has written some examples of what the system could do (for instance what the present system does). Above we have only shown a few of these examples. The supplier is expected to modify column three to show his proposed solution or refer to an appendix. In column 4 the supplier shows with a code (1 to 4) whether the solution is part of a COTS system (1); a customer-specific addition covered by the maintenance contract (2); a customer-specific modification outside usual maintenance (3); not offered (4).

At closer study, this task description (and most of the others) doesn't comply with the idea behind the technique:

Not a closed task. The use case doesn't correspond to a user task that runs from trigger to closure. What is the trigger? Does the customer call by phone? Is it a bunch of letters or notices of address change? Is it a reply to an earlier proposal (cf. variant 3.b). Apparently, the case may be parked temporarily, for instance because data is missing. What makes it continue? Should the system remind users when the case should be resumed? Since the task isn't clearly delimited, the supplier cannot see whether his system supports the task properly.

Duplicate task descriptions. Several use cases largely say the same, and they might be combined into one - with a few optional subtasks. As an example, use case 6 (maintenance of customer data) has the same parts as use case 2 above. Use case 12 (customer calls) should probably have the same

parts, but focus on complaint issues. Finally, use case 27 (change of invoice agreement) deals with a subset of things that are also in use cases 2, 6, and 12.

Probably these duplicated descriptions reflect the existing way the customer has organized himself, but it makes the supplier's work harder and prevents the customer from improving and rationalizing the customer support.

There should be only one use case, but described on a level where existing arbitrary details are hidden - the level that Constantine & Lockwood (2001) call *essential use cases*. The existing work tasks are then special ways of carrying out this essential use case.

Data requirements embedded in the use cases. A large part of the use case steps and the variants doesn't really explain about the task, but about data to be recorded. Examples are use case 2, steps 1 to 4 and variants 1.c, 3.c, and 3.d. In principle the same details should be written also for use cases 6 and 12. This is a major reason that the use cases have become so long and hard to understand.

The recommended technique is to write a separate data description and briefly refer to it from one or more of the steps. As an example, steps 1 to 4 and most of the variants could have been replaced with this single step:

Step 1. Record data about the customer and his service and invoice agreements (see the data description in appendix x).

The advantage is that data is described in one place only. This also encourages the analysts to describe data more carefully, including explaining where data comes from and what it is used for.

Fuzzy distinction between business cases and user tasks. One reason that the use cases are vaguely defined is that the description mixes up two levels of use cases. The *high level* deals with pursuing a business case, for instance that the consumer gets a proposal, replies and has his agreements changed, later delivers missing data, and finally has the changes carried out. Different users may perform different parts of this high-level use case. Probably some pieces of use case 2 above are parts of a use case on this level.

The *low-level* use cases should be closed tasks carried out by a single user in a single session. It might for instance be a telephone call from a consumer. The use case terminates when the consumer hangs up. A later call or a letter from the consumer resumes the business case by means of another low-level use case. An essential use case in the sense above, is a low-level use case. It just happens to combine several low-level use cases into one.

One solution is to describe only the low-level use cases. These are the situations that the system must support well. The high-level use cases are only reflected in data descriptions where each business case is a separate object. The object may be in different states corresponding to how far the business case has progressed. Another solution is to describe high-level as well as low-level use cases, but clearly separated.

The standard literature in this area (e.g. Cockburn, 2000; Constantine & Lockwood, 2000; Lauesen, 2002) doesn't explain this subtle interplay between high and low-level use cases. Experience shows that even seasoned analysts find it hard to disentangle the issues.

Existing problems not mentioned. The chosen use case technique (tasks & support) recommends that column 2 mentions any problems in the way the task is carried out today. In these areas the customer wants the new system to exceed the present one. Some business goals correspond to removing some of the problems. In this project no problems are mentioned. This makes it harder for the supplier to show how he can help the customer reach his business goals (see the example in section 6.4). Showing how problems are removed would also make it easier to get user acceptance since improvements become visible.

Insufficient explanations. Many of the descriptions in use case 2 are insufficient unless the supplier already knows the customer's business processes in detail. One example is step 5 (The supply agreement is activated). What does this mean? If it is water supply, a fitter may have to go to the point of consumption to open valves or set up a meter. Should this be supported - and what kind of solution does the customer imagine? If it is garbage collection, the route plan must be changed (this *is* somehow mentioned in variant 3.e).

Amazingly, customer as well as suppliers stated nothing about how the system could support step 5. All suppliers simply said in the code column that they supported this step. During our interviews, it became clear that they had no idea what the customer expected them to do.

Traditional requirements disguised as use cases. The *General requirements* at the end of the use case are in many cases traditional functional requirements inserted into the use case. One example is general requirement 4 in use case 2 (It must be possible at any time to follow up on order status ...). This is a functional requirement that specifies something the system has to do. However, it is unclear when this will be done and in which user task (in spite of it being written in use case 2).

Minor problems. General requirement 1 (*The system supports the use case and the variants*) is pointless. Other parts of the requirements state that all use cases must be supported. In the proposal, several suppliers have painfully tried to write something meaningful in column three - for each of the 33 use cases.

Variant 4.a says exactly the same as step 4 itself - as far as we can tell. In the same way, general requirement 3 says the same as variant 1.a.

General requirement 2 (*agreements with a group of customers*) has an elaborated explanation of the task in column 3. This elaboration is important, but it belongs to column 2, not column 3, which is intended for proposed solutions.

Weaknesses in the use cases, such as those above, are a major cause of the problems that the suppliers had to fight. The customer believed that the use cases were clear and easy to reply to, but admitted that it had been a tough organizational job to write them. (Apparently the use cases have been written by independent user groups, and nobody seemed able to consolidate them in a coherent fashion.)

The customer had been inspired by the task & support descriptions pioneered by West Zealand County, who however had used a different method for elicitation. In West Zealand, a small task force of expert users with a broad knowledge of most of the user tasks, had started with a single high-level use case. Then they transformed it into a few essential use cases and wrote the first draft of them. Next they discussed and revised the essential use cases with each department. Many departments wanted particular solutions, and they were easy to accommodate as possible solutions in column three - not as requirements. The result was not only much fewer use cases, but also a much shorter process. The entire requirements process (elicitation plus writing) took just two weeks, rather than the 25 weeks it used to take with a traditional approach. All parties found the requirements much easier to understand and to compare against the proposals. As a result the decision process also went much faster than usual.

6.4. Relation to business goals

The requirements specification itself doesn't contain the business goals, but the tender conditions do. The top-level goals are defined as follows (slightly abbreviated):

a) Meet market and legal requirements as a consequence of the open (deregulated) power market.
b) Possibility for better customer service.
c) Possibility for simplified administration.
d) Possibility for using data for economic, strategic and technical analysis and simulation.

Goals on a lower level were expressed in this way (abbreviated):

1) Modular COTS-based system on a standard platform with potential for extension on demand.
2) Support the vision of the digital office and automated procedures.
3) Provide value adding to the customer through quality, security . . .
4) Support decision processes concerning economy, marketing, and expansion of the supply lines.

(plus reliability of operation and several other purposes)

Comments: Looking at the top-level goals, *legal requirements* (a) are reflected in requirement G113, where the supplier is asked to describe his solution in the area. The other top-level goals are only sporadically reflected in the requirements in a visible manner.

For instance it is hard to find requirements that reflect *better customer service* (b). Later the customer explained us where it was reflected - for instance in use case 2, *creating a customer relationship*:

It is obvious that we want to create a customer and make invoice agreements for each product. The old system doesn't have a customer concept, but only points of consumption. Further, invoicing is always on a three-month basis, paid on account.

None of the suppliers had noticed that these requirements were crucial for the business goals. Most of them had no idea how the old system worked. Had they known, they had probably also allocated more resources for the data conversion task, which is very complex when the old system has no customer entities. Specification-wise it might have been done by mentioning the problems in Use Case 2. Step 1 might for instance include this statement:

Problem: In the existing system there is no customer concept. Billing is done for each point of consumption even when the customer has many points of consumption.

As another example, which procedures are to be automated (sub-goal 2)? The requirement about mobile equipment mentions this goal, but leaves it to the supplier to suggest how the equipment might be utilized. Are other processes candidates for automation or rationalization? The customer's internal evaluation of the proposals shows that he primarily had imagined automation through self-service, but it is not visible in the requirements.

The tender documents don't explicitly use the word "rationalization" - a term that isn't popular in the democratic process. However, they hint at it through expressions such as *automation* and *simplified administration*. The customer claimed that he had considered the usual rationalization principles, for instance direct data capture, reducing the number of processing steps, decision support and problem resolution on the spot, but it had not been possible to find specific ways to do it. (This is virtually impossible in the democratic analysis process.)

It is noteworthy that no suppliers figured out what the business goals meant - apart from the legislation requirements. Unfortunately, the missing link between goals and requirements is a widespread phenomenon, particularly in public organizations, but also in industry. We have for instance heard other customers explain:

All this stuff with business goals is something we always have to write. We have a department that does it. Everybody knows that it doesn't mean anything, and those who write the requirements are in another department that doesn't care about such stuff.

7. Conclusion

After revealing all these problems, it may be surprising that the present acquisition project was rather successful compared to other acquisition projects that the authors have experienced. There is no doubt that the customer chose the right supplier on a solid basis. On the other hand, the customer could have met his business goals much better if he had avoided the problems explained in section 5. He could also have avoided the problems with endless development of some new product parts. Here is a summary of the problems.

Requirements too close to "as-is". Some problems related to requirements being too close to the existing way of doing things. This made it hard to see how a new system could improve the processes. The customer had chosen a requirements style that allowed a more future-oriented approach (Tasks & Support), but didn't master the technique sufficiently well. Another cause of the problem was an inadequate elicitation technique with excessive user involvement, which made it hard to imagine new or re-engineered work processes.

Traditional quality criteria for requirements, such as *completeness* and *non-ambiguity* (IEEE Std. 830) don't help in this area. Maybe they even encourage the analyst to do the opposite of what is necessary: not being too specific (but at the same time avoiding being too vague).

Requirements too vague. Some problems related to requirements being too vague or too open, thus leaving too much to the supplier. In some cases the supplier didn't understand what the customer wanted, in other cases the supplier understood the problem but had no solution and couldn't come up with one in the time frame allowed. The customer's requirements style could deal with this, but again he didn't master the technique sufficiently. The result was endless development in some areas where no solution existed at proposal time.

Quality criteria such as *completeness* and *non-ambiguity* would have helped pointing out these problems. Creativity techniques such as brainstorm and focus groups could also have helped the customer define requirements in these areas.

Goals and problems hidden. Some problems related to goals and present problems not being reflected in the requirements. This made it hard for the suppliers to see how they could help the customer meet his goals. Tracing techniques could have helped here, but the customer didn't try to use

them - or didn't know about them. (Tracing could easily have been made manually. There was no need for an IT tool.)

Quality criteria such as *traceability* might have pointed this out. However, the classical IEEE Std. 830 doesn't mention tracing from business goals to requirements, which in this case was the main problem.

Techniques lacking. For some problems there was no available techniques at that time. This was the case for requirements about integrating the new system with other systems - existing or future. The customer couldn't come up with a good solution, and research seemed necessary.

Inadequate process. Finally, the acquisition process caused many problems. Some of them related to excessive user involvement and insufficient use of creativity techniques, others to unrealistic deadlines. These factors are under the customer's control. Other problems relate to inadequate rules and limitations of the communication between the parties. These are given by European Union rules and are beyond the control of the customer.

So what should the customer have done? If he just wanted to be as good as his peers, there was no reason to do anything. However, he *would* like to do better. Would a consultant have helped? Unfortunately, the quality of the existing specification is comparable to what many IT consultants produce, so from the customer's point of view it was unnecessary to involve a consultant.

There are techniques to deal with many of the problems, but training is needed to master them - particular on-the-job training. Requirements specialists with theoretical background as well as thorough practical experience seem necessary. Unfortunately they don't grow on trees.

References

Albert, C. & Brownsword, L. (2002): Meeting the challenges of Commercial-Off-The-Shelf (COTS) products. In: J. Dean & A. Gravel (Eds.): ICCBSS 2002, LNCS 2255, pp. 10-20.

Balk, L.D. & Kedia, A. (2000): PPT: A COTS integration case study. Proceedings of the International Conference on Software Engineering, ICSE 2000, pp. 42-49.

Boehm, B. & Abts, C. (1999): COTS integration: Plug and Pray? IEEE Computer, January 1999, pp. 135-38.

Bonnerup, E. (2001): Experiences from government IT Projects - how to do it better. Teknologirådet, March 2001. (In Danish)

Cockburn, A. (2000): Writing effective use cases. Addison-Wesley.

Constantine, L. & Lockwood, L.A.D. (2001): Structure and Style in Use Cases for User Interface Design. In: Harmelen, M. V. (ed): Object modeling and user interface design, Addison-Wesley.

Feblowitz, M.D. & Greenspan, S.J. (1998): Scenario-based analysis of COTS acquisition impacts. Requirements Engineering (1998) 3: pp. 182-201.

Gotel, O.C.Z. & Finkelstein, A.C.W.: An analysis of the requirements traceability problem. Proceedings of the International Conference on Requirements Engineering, ICRE 1994, pp. 94-101.

Gorton, I. & Liu, A. (2002): Streamlining the acquisition process for large-scale COTS middleware components. In: J. Dean & A. Gravel (Eds.): ICCBSS 2002, LNCS 2255, pp. 122-131.

Guo, Jiang (2002): Interoperability technology assessment. Elsevier Science, Electronic notes vol. 65 No. 4.

IEEE Guide to Software Requirements specifications. ANSI/IEEE Std. 830-1993 or 1998.

Lauesen, S. (2002): Software requirements - styles and techniques. Addison-Wesley.

Lauesen, S. (2003): Task Descriptions as Functional Requirements. IEEE Software 2003, March/April, pp. 58-65.

Lauesen, S. & Vium, J.P. (2003): Experiences from an EU tender process - successes and failures. (Includes a detailed account for each supplier of his perception of the process). (In Danish)
www.itu.dk/people/slauesen/Papers/Leverandorvalg9.pdf

Lauesen, S. (2004): COTS tenders and integration requirements. Proceedings of the International Requirements Engineering Conference, RE 2004, pp. 166-175.

Liu, A. & Gorton, I. (2003): Accelerating COTS middleware acquisition: The i-Mate process. IEEE Software, March/April 2003, pp. 72-79.

Maiden, N.A. & Ncube, C. (1998): Acquiring COTS software selection requirements. IEEE Software, March/April, pp. 46-56.

Ncube, C. & Dean, J.C. (2002): The limitations of current decision-making techniques in the procurement of COTS software components. In: J. Dean & A. Gravel (Eds.): ICCBSS 2002, LNCS 2255, pp. 176-187.