7 COTS benefits versus custom-made

Commercial Off-the-Shelf (COTS) software solutions have become commonplace in many domains, including the military, because they can provide standardised functionality with more responsiveness, a shorter time-to-market and at lower costs than custom made solutions. In one domain, however, that of certifiable safety-critical applications, COTS software has not been adopted. One particular type of certifiable safety-critical domain, the civil air transport industry, is under pressure to reduce cost and time-to-market whilst simultaneously increasing safety. Therefore, the use of COTS software, rather than exclusive reliance on custom-made software, would appear to be a solution worthy of investigation. This study examines the certifiability of COTS software, its technical feasibility in this environment, and the ability to achieve the expected responsiveness, time-to-market, and cost benefits. A detailed evaluation of COTS software and domain-specific certification requirements is used to demonstrate that the certification of COTS-based systems is possible. A prototype COTS-based system (built upon a number of COTS components) is created to illustrate the technical feasibility of such a system in the civil air transport domain. Expected benefits from COTS solutions are evaluated both by examining process artefacts from the development of the COTS-based system and by comparing this development process with the domain’s traditional custom-development process.

7.1 Introduction

In the general domain, commercial off-the-shelf (COTS) based software solutions are commonplace for their timely and affordable provision of standardised functionality. Although definitions of COTS differ, the following, generally accepted, wording highlights the difference with custom-made software. The salient characteristics of COTS software are: COTS exists a priori from a commercial vendor, COTS is available to the general public and COTS can be bought (or leased or licensed) (Oberndorf, 1998). In 1994 the US Department of Defense started the
transition from custom made software to software developed using COTS components, based on the observation that it had historically fielded technology that was more than 10 years old, while new technology emerges every 18 to 24 months (Oberndorf, 1998). The Department of Defense’s current policy (DOD 2003) is to acquire products that satisfy user needs with a measurable improvement to mission capability, in a timely manner at a reasonable price. This is known as the cheaper, faster, better paradigm. COTS software has been successfully used in many systems for the general domain and, subsequent to the transition, for the military domain as well.

The contribution of this paper is to assess the feasibility of COTS software in another highly regulated domain, that of certifiable safety critical domains and to assess its relative advantages with respect to the current custom made approach. Due to their justifiable strict safety requirements, safety critical domains have mandated certification of software. With increasing safety criticality levels, more and more stringent requirements are enforced. Due to the very low failure rates aspired to, certification of compliance of software with safety and reliability requirements (DO-178B, 1992), puts requirements on the software processes, as compliance cannot be observed directly. As a result the domain is sticking to custom made software, produced using well understood certification compliant processes. To examine the feasibility and expected benefits of a COTS-based approach with respect to the traditional custom made approach, this paper will study one particular certifiable safety-critical domain, civil air transport.

The air transport domain is chosen as a typical example of a certifiable safety critical and demanding domain relying on proprietary, custom made solutions. The air transport domain has a good safety record and its certification processes are properly documented. To provide an indication of the domain size, a cursory overview of the number of organisations for some major actors is provided. Actors include airlines, airports, air traffic management, ground handling, meteorological offices, aircraft maintenance, etc. The International Air Transport Association already represents 265 airlines (IATA, 2005) with dozens of airlines remaining not affiliated. The Airport Council International bundles 567 organisations from 175 countries that are operating over 1540 commercial airports (ACI, 2005). The current world-wide commercial fleet consists of 34 576 aircraft (International Civil Aviation Organisation ICAO, 2003). Air traffic management is organised nationally, with separate organisations and systems for civil and military traffic.
In such a highly regulated domain, voluntary changes involving multiple actors take long negotiation times and are rare. Usually changes are mandated by national regulators after long multinational negotiations and subsequently allow for transition periods of up to a decade. Even developing a new aircraft type or a new ground-based air traffic management centre takes a decade. This slow change is a result of both aircraft and ground-based centres having an economic life of several decades. This is similar to the military situation, which prompted their COTS initiative (Oberndorf, 1998). If it is also true in this domain that, as according to the (Standish, 2004) report, 70% of the application code is in fact standard infrastructure for which COTS software or software components could be used, COTS could be relevant for the domain. In addition, results for the air transport domain could be relevant for similar safety critical domains such as the chemical, rail transport and medical domains due to commonality in the principles of safety certification (Kesseler, 2004a).

The civil air transport industry is concentrated in North America and Europe. Both regions have defined visions for the domain to remain viable and economically attractive for the next 15 years. The US vision (Walker et al., 2002) includes requirements for a 50% cost reduction and a drastic time-to-market reduction from the current decade to years to months or even weeks, with a simultaneous five-fold increase in flight safety. Similarly the European vision (Argüeles et al., 2001) mentions a 30% cost reduction and a halving of the time-to-market with a similar five-fold reduction of accidents per flight, as a prerequisite for a viable European air transport system. As a COTS-based approach intends to deliver on the affordability, responsiveness and time-to-market criteria, in this domain a need and a means to solve that need are simultaneously recognised; in other words, a situation of push pull exists.

Based on the discussion above, the derived research questions addressed in this paper are:

- Is COTS software feasible from a safety certification point of view?
- Is COTS software technically feasible in this certifiable safe domain?
- Does a COTS-based approach provide expected time-to-market, responsiveness and affordability benefits when compared to the traditional custom-made development approach in this domain?
7.2 Background

In the traditional custom based approach, within the budget constraints, all user requirements can be accommodated by a well understood development process from architectural design via detailed design to implementation, testing, verification and ending with validation. For a COTS-based approach the high level user requirements will remain the same, but a process is needed to match those requirements to the available COTS solutions and select the best fit. Custom-made extensions are then needed to fulfil those user requirements not satisfied by the selected COTS software but deemed indispensable by the users to satisfy their objectives. Such an approach needs flexibility from the users with respect to their (detailed) requirements while keeping their objectives in mind, a spiral or iterative development process with short term achievable targets and continuous user involvement. The architecture needs to be flexible to accommodate changes for the next iteration and due to changes in the COTS software (Albert, Brownsword, 2002).

An important benefit of COTS is the reduced time-to-market. This has been observed by many, including (Oberndorf, 1998) for the military domain and by (Mili et al., 2000), (McKinney, 1999) (Schmidt, 2002) for the class of safety critical distributed real time systems in general. Air transport domain systems fit the latter class.

Given the often significant development times of air transport applications and the subsequent decades of use, it is important to accommodate, in a timely fashion, the inevitable changing requirements unknown during system development, i.e. to improve responsiveness. Responsiveness can to some degree be attributed to the use of COTS solutions (Oberndorf, 1998). Ironically the low responsiveness caused by safety certification requirements can jeopardise safety when affordable, but uncertifiable COTS software which could reduce risks or even prevent some types of accident may not be used.

Increasing the affordability of solutions is another well documented benefit of using standardised COTS software, as mentioned by (Oberndorf, 1998), (Mili et al., 2000), (McKinney, 1999) and (Schmidt, 2002). Also this expected benefit could improve safety as customers are more likely to purchase aircraft with non-mandatory, advanced systems which do improve safety if the additional cost is low. For the
same reason retrofitting existing aircraft with new capabilities is also more likely with reduced costs.

Currently, deployment of new releases of certified safety critical software is slow, as it has to be done by certified personnel during scheduled maintenance of aircraft or ground based systems at significant cost and lost revenue. As a result sometimes software containing additional features is never deployed in some aircraft. Using Java or similar COTS Internet technologies would allow downloading of already certified software while the aircraft is stationary during the airport turn around process. Using more recent and more homogeneous software by the aircraft fleet would also benefit the safety of the air transport system.

When combining several functions on one (multiplied) hardware unit, services tend to be of different software criticality levels. It is too expensive to certify each service at the most stringent safety level, if at all possible. Airborne software safety certification standards allow software of different criticality levels to co exist on one hardware unit, provided special partitioning software separates these functions from each other. Of course such partitioning software has to be certified to the highest criticality level of all functions concerned. For custom-made software this partitioning has been done for some time (Kesseler, 2000). Current commercial certifiable software development of operating system kernels and Java kernels (Hunt, 2005) intends to provide such partitioning.

According to Schmidt (2002) the pay offs of COTS include that it moves standardisation up several levels of abstraction for safety critical real time embedded systems. Another way of phrasing this is that COTS increases independence from lower level evolution, like quickly evolving communication services. Consequently COTS allows the development cost of low level technology to be spread over more users (McKinney, 1999).

Another expected COTS benefit is support of innovation. Currently the domain relies on proprietary solutions based on proprietary architecture, referred to as vendor lock in. By opting for an open architecture it is expected that new suppliers can provide new services or improvements on existing services (Kowalski, Karcher, 1994).

The current lack of COTS components for certifiable safety critical software in the air transport domain is caused by a number of factors. The highly regulated but fragmented national certification processes impede re-use of COTS. To improve on
this the European Union has embarked on a single certification for all its currently 27 member states. Complex and protracted certification processes imply long product realisation times, typically around one decade, followed by very long utilisation times, typically decades. These realities favour organisations with a significant corporate history and the ability to wait long for their return on investment. Such organisations tend to be more conservative towards process innovations, with respect to new entrants who cannot afford the long payback period (Voordijk, Meijboom, 2005). Also software certification is based on very domain specific procedures, which requires intimate knowledge of the domain standards and processes used, which again favours using proven technology, proven processes and established providers. Once the first party has successfully completed certification of an innovation, it is proven technology and is rapidly adopted in the domain.

Although COTS software is not common in this domain, re use of proprietary solutions is. Solutions for one aircraft programme or ground based system are often adapted for re use in subsequent programmes. Also software used for one aircraft is often re used in upgraded aircraft models of the same type. One major aircraft manufacturer even makes commonality of features between its aircraft types a selling feature. As the software manufacturer owns the software and all certification artefacts, re use reduces costs. However re use results in a vendor lock in with reduced competition and less innovation, which is quite different from the potential for increased competition and increased innovation with a COTS-based approach.

By their nature aircraft move around, so a network of a ground based system with the aircraft in its vicinity will be constantly changing. Also connection to fast moving aircraft might be temporarily lost. This is similar to for instance mobile phone users, who move between base stations. For various similar situations in the general domain COTS–based systems are available which provide a self forming self healing capability. Consequently COTS-based solutions are expected to be able to provide this important capability for the air transport domain as well.

To summarise the COTS-based software approach is expected to provide the following technical benefits as addressed in the second research question:

- Simultaneous support of services with multiple safety criticality levels;
- Increased independence from evolution of lower level services;
- Support of innovation;
The third research question covers the following benefits expected from the COTS-based software approach:

- Reduced time-to-market;
- Improved responsiveness;
- Increased affordability.

### 7.3 Research approach

To address the first research question on COTS software certification potential, a study was performed on the relevant certification documents with respect to the use of COTS software. As air transport software relates to an airborne segment, a satellite segment (for positioning services and optionally for communication services) and a ground segment (for air traffic management), all relevant standards (DO-178B, 1992), (DRD-920, 1999), (EGNOS, 1999), (ESARR 4, 2002), (DO-278, 2002), (EUROCONTROL, 2002), and (FAA AC120-76A, 2003) have been studied for how their process requirements address COTS software for their various safety criticality levels. To check the correct understanding of the documents a software specialist of a national airborne certifying authority was interviewed, confirming the presented results. For the ground segment documents, the author participated in consultation meetings between the European rule making authority and stakeholders from academia, research institutes and industry. The presented results reflect the documented rules and their current application.

The second part of this research relates to the second research question on the technical feasibility of COTS software for this domain. Consequently this research question is addressed by the realisation of a certifiable COTS-based demonstration system. For the demonstration, a system is selected which addresses recognised domain needs and where a COTS-based approach is expected to contribute to the system objectives. The technical objectives are described in the results section below.
The remaining research question relates to expected improvements of the COTS-based approach with respect to the current custom-made approach, so the COTS-based case is compared to a carefully selected custom-made case. The custom-made case was recently and successfully completed before being used in this research to prevent this research from influencing the custom-made case data. None of the people involved in the custom-made case were involved in the COTS-based case with the exception of the author in a consulting role for the entire duration of both cases. Only one organisation contributed to both cases. A section below justifies why the two cases are comparable. Subsequently some key results are provided.

Grounded theory (Orlikowski, 1993), (Glaser, Strauss, 1967) is used as it is inductive and allows for iterations between data and conclusions. For the result to have relevance beyond the originating case, at least two cases are needed. Indeed others like (Orlikowksi 1993) have found that grounded theory can be used to extract meaning from samples as small as two like in the current research. Grounded theory has been used previously for context-based process-oriented descriptions of information systems development (Goulielmos, 2004), the subject of the present research.

Data collection was based on analysis of software process artefacts. When unexpected findings emerged, additional data was collected to try to determine or eliminate a possible cause, compliant with grounded theory. To guide the data collection process, the Goal/Question/Metric method of (Basili et al., 1994) is used. In this method one (or several) questions are derived from a stated goal. To answer each question an observable metric is selected with key participants of the software process analysing the metrics results for relevance to the question and the goal. In case the metric is not felt to be appropriate for the goal, a new metric is selected. The key participants thus provide some verification of the selected metric and their interpretation. Apart from the researcher, the key participants were the local project manager, the system designer, and the main designer from another main developing partner. These participants of the COTS-based team were interviewed for their view on the collected data or emerging explanations and to correlate observed data with project events like reviews. The local participants were interviewed several times. The participant from the other partner was interviewed once with subsequent additional contact by phone and e-mail as required. For the custom-made case the same Goal/Question/Metric method has been applied, with as participants, apart from the researcher, the project manager, the systems designer and the test leader.
These custom-based case interviews and analysis were completed before the data was compared with the COTS-based case to guarantee independence. Only one organisation participated in both cases. The only person in common for both teams was the researcher in a consulting role.

To make a realistic comparison, the traditional custom-made case was selected based on the following similarities. It pertained to technical software for air transport, avionics. The custom-made case consisted of a team of two industrial partners plus one institute located in two countries. The COTS-based case consisted of three industrial partners plus two institutions from four countries carefully selected for their complementary contributions. Both teams hired third party personnel, which had to abide by the project procedures, making for a total of six organisations involved in both cases. Both cases used a similar one off multi partner co-operation which increases management cost with respect to established co-operations, but does not put one of the cases at an advantage.

In the custom-made case one partner learned how to make Level A (DO-178B, 1992) certifiable safety critical software based on previous experience with other safety critical software. The custom-made case used traditional Structured Analysis with Real Time extensions (Hatley, Pirbhai, 1988) followed by Yourdon structured design. In the COTS-based case the safety background was common, but Unified Modelling Language (UML) combined with Unified Software Development Process’ (USDP) incremental deliveries were new, although object oriented experience was available. The custom-made case was commercial with a strict line of command. The COTS-based case was realised by a consortium co-operation where each partner effectively has a veto right to block activities, while a mechanism for enforcing activities was lacking. In the commercial custom-made case it is clear all partners have an interest in successful completion of the work. In the COTS-based case each partner had to finance 50% of the effort from company resources, providing a similar incentive to succeed. The custom-made case had a well defined division of the work allowing most work to be performed by one partner on a single site. During the integration the parties involved were co-located on one site.

In both cases professional personnel was used, with a mix of mainly experienced personnel complemented by some novices. The physical team sizes are also comparable.
7.4 Results

7.4.1 Certification potential

This section addresses the first research question on the certification potential of COTS software for the various safety criticality levels of the respective certification documents. Domain specific certification is mandatory for software with a safety impact. All software in aircraft and increasingly also the supporting ground based software for air transport has to be assessed for its safety implications. In case aircraft use satellite positioning or communication services, this software, whether in orbit or ground based, needs to be certified as well, according to the airborne requirements. In order for COTS-based systems to be feasible, they need to be safe as well as certifiable, i.e. demonstrably safe.

For airborne software, regulations classify software into five categories with the consequences of failure ranging from level A (catastrophic failure, i.e. loss of life), level B hazardous (serious to potential fatal injuries), level C major (discomfort possibly leading to injuries), level D (minor) to level E (no impact). Airborne software regulations are also enforced for COTS software, necessitating fully compliant development processes. Re-use of previously certified software, including all certification process artefacts is allowed and commonly done. From a technical point of view, work in (the Open Group, 2003), co-inspired by our work, will lead to a specification and implementation of a real-time version of a Java kernel compliant with the most stringent level A, including certification process artefacts. The realised COTS-based system addressing the second research question is based on Java taking the certifiability restrictions into account. Commercial operating systems kernels which aim at certification to level A are becoming available. So on a technical level the choice of carefully selected COTS components or Java version does not impede achieving the required airborne safety certification.

The aeronautical telecommunication network, part of the developed prototype, is built to the airborne software certification level C with the option for additional certification to level B if required by a specific service. However all currently envisaged services only require D at most, suggesting software realisation processes compatible with level C could suffice. Such processes have been used.

Airborne regulations allow applications with different safety criticality levels to run on the same hardware, provided partitioning software strictly isolates the
applications from each other. The partitioning software itself is classified at the highest level of all applications it separates. The mentioned Java work, as well as several certifiable COTS kernels, provide such partitioning, substantiating one of the expected benefits of the COTS-based approach.

Ground based air traffic management certification is not yet mandated, and typically includes COTS software. Consequently ground-segment certification documents have to address COTS software. The US document is derived from the airborne certification document. The European document is independent, but does share one class (level B) with the airborne document. Under strict conditions both documents allow the use of service history for COTS software not developed in accordance with its software process requirements. COTS software with service history can be accepted up to airborne level C, with the option to negotiate the use of service history for level B. For the moment the latter seems less likely given the justifiable conservative attitude of the certifying authorities. System level mitigation is expected to obviate the need for the highest two classifications (level A and B), with the exception of satellite navigation services. However for our purposes COTS developed using non-compliant software realisation processes could be used, provided service history is obtained using compliant processes. By deploying such COTS software, certifiability need not be compromised.

One particular COTS-based application examined was an electronic flight bag, which is a COTS-based laptop like device in the cockpit and is a candidate for displaying the airborne part of the COTS-based solution. One of the main aircraft manufacturers has certified a commercial operating system kernel to level C for this purpose. So from a technical point of view providing safety and achieving safety certification of COTS-based systems seems feasible. Understandably updated software has to follow the full certification process, taxing the responsiveness requirement, one of the expected advantages of COTS usage.

Combining all this information allows answering the first research question. For the envisaged safety criticality levels carefully selected COTS software can be certifiable. For systems relying on COTS software developed using non compliant software realisation processes, certifiability to lower criticality levels implies complying with strict requirements on the service history.
7.4.2 Certifiable COTS-based demonstration system: concept and realisation

The domain specific objective for the COTS-based system is to support co-operation between various organisationally independent actors, exploiting existing information technologies. Such cooperation is expected to provide major benefits to the domain, as worded in the European Union’s air transport Vision 2020 (Argüelles et al., 2001) which is agreed by all concerned. Figure 7-1 depicts the solution concept and its realisation for the selected COTS-based system. It consists of a federated architecture and two demonstration services. The demonstration services are chosen for their capability to provide short-term benefits for the domain and their capability to evolve after the initial release, i.e. demonstrate responsiveness.

Due to their demonstration nature, no expensive safety assessment and classification of these services have been performed. However relevant domain specific software certification concerns have been taken into account during their realisation. The first demonstration service, meteo(rological), provides pilots with in flight weather updates. Actual weather information allows pilots to optimise their flight route while being airborne. This application demonstrates tangible benefits, like shorter flights and fuel reduction, currently unavailable to pilots and airlines, and has a lower safety criticality level, as long as severe weather is avoided. The second service, Traffic Information Services (TIS), provides the pilot with enhanced situational awareness through in flight airport information like current landing conditions, especially during landing, the most critical and busy flight phase. Loss of situational awareness is a major cause of human error. The largest contributing factor to accidents is human error, with estimates of up to 80% involvement (NASA, 2004). By reducing the probability of such errors, TIS enhances safety, so this service has a higher safety criticality level. Together these two services demonstrate the simultaneous support of services with different safety criticality levels, one of the expected COTS benefits.

For implementation the certifiable subset of Java was chosen. For COTS components produced using software processes compatible with airborne certification, the aeronautical telecommunication network was chosen. It demonstrates increased independence from the evolution of lower level services, one of the COTS advantages. OpenWings addresses the US military requirement for interoperability between all their systems. In the civil domain OpenWings aims for similar interoperability. OpenWings is Java based and facilitates integration of
proprietary networks like the aeronautical telecommunication network. OpenWings became available just after the case study started, and it was decided to select it because of the self forming self healing properties, thereby addressing the domain specific COTS advantage. It also allowed to obtain experience with COTS developed using non-compliant development processes.

Figure 7-1. Architecture overview on concept level and in COTS-based prototype.

Figure 7-1 shows the solution concept and how the COTS-based prototype builds upon various COTS components, which in turn support various hardware platforms. The application services, which enable the actual co-operation between the actors, will run on top of the federated architecture. For the domain a special purpose data link, called Aeronautical Telecommunication Network, has been developed which has been certified to a safety level sufficient for the intended services. The increased independence from the evolution of lower level services, demonstrates another expected COTS benefit.

7.4.3 Technical feasibility

This section discusses the results obtained while realising the COTS-based system to arrive at a judgement on the technical feasibility of using COTS software components in a safety critical domain.

The realised prototype consists of three parts, the federated architecture (see Figure 7-1) and two application services (meteo and Traffic Information Services or TIS) to illustrate the capabilities of the architecture. Table 7-1 summarises which choices have been made in the COTS-based system to obtain the expected benefits, elaborated above.
Table 7-1. Summary of how selected COTS are expected to provide benefits.

<table>
<thead>
<tr>
<th>Expected benefits</th>
<th>Selected COTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain related technical benefits (second research question)</td>
<td></td>
</tr>
<tr>
<td>Reduced deployment time</td>
<td>Java</td>
</tr>
<tr>
<td>Simultaneous support of multiple safety levels</td>
<td>Certifiable Java kernel</td>
</tr>
<tr>
<td>Provision of self-forming self-healing capability</td>
<td>OpenWings</td>
</tr>
<tr>
<td>Increased independence from evolution of lower level services</td>
<td>OpenWings for communication technology independence (excluding domain specific protocols)</td>
</tr>
<tr>
<td></td>
<td>Aeronautical Telecommunication Network (for domain specific communication protocols)</td>
</tr>
<tr>
<td>Support of innovation</td>
<td>Open architecture + COTS</td>
</tr>
<tr>
<td>COTS paradigm related benefits (third research question)</td>
<td></td>
</tr>
<tr>
<td>Reduced time-to-market</td>
<td>COTS + iterative development (USDP based)</td>
</tr>
<tr>
<td>Improved responsiveness</td>
<td>COTS + iterative development</td>
</tr>
<tr>
<td>Increased affordability</td>
<td>COTS, open architecture</td>
</tr>
</tbody>
</table>

Most non-safety related objectives pertain to COTS providing such functions for a larger market. Several of these benefits, like benefiting from quickly changing communication technology, are too expensive for the domain to realise using the custom-made approach. This illustrates the “better” part of the cheaper, faster, better paradigm.

Table 7-2 provides some characteristic data on the COTS-based realisation. To demonstrate feasibility of incremental deliveries i.e. responsiveness, the derived detailed requirements were prioritised as build one, scheduled for delivery at month 16, build two, scheduled for delivery at activity completion in month 23 and build future for those outside the current prototype’s scope. Many costly requirements had to be postponed to the future build. The first row of Table 7-2 provides the number
of requirements, i.e. requirements identified as such in the current documentation. The second row lists the number of identifiable requirements based on current wording. The requirements could easily have been phrased in smaller, separately identifiable units to ease incremental implementation and testing, thereby enhancing responsiveness. For example a single requirement is currently worded as “A service shall be able to define asynchronous events. An event may be a simple stimulus (1) or may be accompanied by data (2). A component shall be able to publish an asynchronous event (3). A component shall be able to subscribe (4) and unsubscribe (5) from asynchronous events.” The added numbers denote five identifiable requirements, providing the option for a first release not to implement an event accompanied by data or not to implement dynamical subscription to events.

Table 7-2. Overview COTS-based prototype realisation.

<table>
<thead>
<tr>
<th></th>
<th>Federated architecture</th>
<th>Meteo service</th>
<th>TIS service</th>
</tr>
</thead>
<tbody>
<tr>
<td># of requirements build 1/2/future</td>
<td>22/7/7</td>
<td>54/14/-</td>
<td>10/13/8</td>
</tr>
<tr>
<td># of identified requirements</td>
<td>66</td>
<td>118</td>
<td>108</td>
</tr>
<tr>
<td># of pages requirements documentation</td>
<td>90</td>
<td>85</td>
<td>97</td>
</tr>
<tr>
<td># of review comments (high/medium/low)</td>
<td>23/29/1</td>
<td>12/-/-</td>
<td>20/-/-</td>
</tr>
<tr>
<td>Analysis (man month)</td>
<td>27</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>Implementation (person month)</td>
<td>101</td>
<td>41</td>
<td>39</td>
</tr>
<tr>
<td>K Lines of Code (KLoC)</td>
<td>32.2</td>
<td>17.6</td>
<td>18.9</td>
</tr>
<tr>
<td># of classes</td>
<td>106</td>
<td>65</td>
<td>75</td>
</tr>
</tbody>
</table>

As integration with the selected COTS (OpenWings) took an unexpected amount of effort, only partial unit testing has been performed. The planned integration testing and verification testing had to be omitted. The COTS integration effort also caused the shift of build one to the end of month 29. Build two could only be completed by reducing functionality. At the time of realisation OpenWings was not sufficiently stable making it hard to predict when it would work or whether to abandon it and change to more stable but less powerful COTS alternatives combined with more custom-made extensions.
To answer the second research question, the prototype was successful in demonstrating that the COTS-based approach is feasible to satisfy the technical requirements, but at significantly higher costs, lower responsiveness and longer time-to-market than expected. The other expected benefits like reduced deployment time, simultaneous support of multiple safety criticality levels and (limited) self forming self healing network capabilities have been observed. The experience obtained, including the unanticipated downscaling, does not support the significant productivity increase reported for Java by (Quinn, Christiansen, 1998) and (Tyma, 1998). The team’s experience supports the contention of (Basili, Boehm, 2001), that integrating COTS software can be a high-risk activity.

In the prototype development the expected benefits from the COTS-based approach as summarised in the third research question could not be observed. For a possible explanation, taking software process context into account, the software realisation data were reassessed using grounded theory. The team suggested three possible causes which have been analysed below using the goal/question/metric method, as shall be discussed further.

A significant part of the federated architecture is interfacing code. According to (Basili, Boehm, 2001) such interfacing code takes three times the effort of developed code. However, as the productivity of the federated architecture task, with its labour intensive interfacing code, does not differ significantly from the productivity of the demonstration services tasks, the interfacing code can neither explain the disappointing productivity observed nor the resulting unexpected time-to-market.

The document volume combines the requirements and the UML use cases. The number of review comments per page of specification is low, indicating friendly reviewing. In comparison, (Gantner, Barth, 2003) found 1.1 comment per requirements page of which they rated 45% as critical. This should have helped and not hindered achieving the affordability and time-to-market objectives.

Project management including quality assurance accounted for an additional 15% of the total project budget. This figure reflects the reality of the maximum acceptable to the client (the European Union). The remaining project management effort is included in each partners’ budget, in our case an approximate 10%. Such figures are in line with our other international projects, so the amount of project management effort cannot explain the disappointing affordability results.
Re-assessing the software process data for some suggested causes does not provide an explanation for the observations. Consequently to address this remaining research question, the COTS-based case was compared with a carefully selected custom-made case.

### 7.5 Assessing COTS-based case through comparison with custom-made case

#### 7.5.1 Case characteristics

Table 7-3 summarizes some key characteristics of the COTS-based case and the custom-made case. Full details of the custom-made case are available in (Kesseler, 2000).

<table>
<thead>
<tr>
<th></th>
<th>COTS-based case</th>
<th>Custom-made case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Realisation time (calendar months)</td>
<td>30</td>
<td>26</td>
</tr>
<tr>
<td>Effort (person months)</td>
<td>280 (+37 certification study)</td>
<td>235</td>
</tr>
<tr>
<td># of organisations involved</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Software size (K Lines Of Code KLOC)</td>
<td>68.7</td>
<td>90 (30 +60 comment incl. detailed design)</td>
</tr>
<tr>
<td>Technical documentation (pages)</td>
<td>822</td>
<td>1 144</td>
</tr>
<tr>
<td>Software process documentation (pages)</td>
<td>6 690 (incl. 3 872 supporting technical doc)</td>
<td>696</td>
</tr>
</tbody>
</table>

Table 7-3 shows the cases to be comparable on several key characteristics. Similar methods were used to arrive at the initial effort estimates of both cases. Both cases used a similar one off multi partner co operation which does not put one of the cases at an advantage. Only the last metric in table 7-3 does show significant differences.
Generating such software process documentation requires significant effort as well as realisation time.

The COTS-based case was forced to omit the integration and verification tests, which reduces the technical document volume (Table 7-3 fifth line). The custom-made case did include this significant extra effort, needed to produce the required documentation containing certification evidence for the safety critical software. In the custom-made case, verification took around half the effort from requirements definition up to unit testing. Additionally all procedures had to be developed and documented rigorously. This consumed significant effort but was advantageous when third-party personnel had to be hired. An undocumented rule of thumb states level A software uses twice the effort of level B software, which in turn requires twice the effort of level C code. This should have put the COTS-based case at an advantage. The similarity of these cases, as reflected in these metrics, suggests that the original objectives of the COTS-based case should have been achievable.

The sections below will address the part of the research question related to responsiveness. First the results are provided, followed by a section analysing them, with both sections taking the process context into account.

7.5.2 Responsiveness results

At activity start the objectives and the federated architecture concept were sufficiently mature. The details of the two demonstration services were to be determined, complying with the COTS-based approach. To address responsiveness the goal/question/metric method suggests analysing the change administration. The stability of the requirements and the design are reflected in the change administration, which is shown in Figure 7-2.
All changes relate to the specifications of the three components, none were submitted for other process artefacts, including software. As the completion of the first build kept shifting, most code was only submitted to the software repository shortly before activity completion. The change administration before that date was the responsibility of the developing partner and hence is invisible in the Figure 7-2. Consequently the impact of the unexpected COTS integration effort also remains invisible.

One team member submitted 30 changes to downscale his part of the Traffic Information Services. These changes explain the peaks at month 19 and month 24-25 and part of the peak at month 11-13. This preceded the client reviews in month 15 and 26. Another member submitted 10 changes to clarify and downscale his application, which explains the remaining part of the month 11-14 peak. The remaining two changes clarified some requirements. No changes relate to user feedback i.e. demonstrate responsiveness.

7.5.3 Responsiveness analysis

In the COTS-based case, breaking down the requirements into smaller ones, which as indicated in Table 7-2 could have been done easily, would have helped to realise incremental deliveries which improve responsiveness. This lesson was also learned
in the custom-made case (Kesseler, 2004b). The significant effort to get the requirements “right” before proceeding is an indication that the waterfall paradigm is retained. Such mixing is referred to as paradigm contamination and produces worse results than using either paradigm (Cloete, Gerber, 2004). Due to the lack of experience with the evolutionary approach, no partner read these signs, despite the small volume of changes.

The COTS-based case, using object oriented methods, aimed for incremental deliveries but achieved only two nearly coinciding deliveries. Ironically the custom-made case, using a waterfall model based on structured methods compatible with airborne certification, produced 15 deliveries of which two have been certified in a shorter realisation period. Due to the useful intermediate deliveries the custom-made case received significant user feedback resulting in a huge requirements evolution with around 500 changes on 200 requirements. The COTS-based case only obtained user feedback when it was nearing completion, at which time valuable requirements evolution could not be accommodated anymore. The custom-made case confirms that the incremental deliveries should have been achievable in a safety critical environment. The limited requirements evolution of the COTS-based case also cannot explain the time-to-market and affordability challenges that were encountered.

Grounded theory encourages iterations between data and analysis, which initiated the following experiment. To assess the feasibility of incremental deliveries, i.e. demonstrate responsiveness, in the COTS-based case, in month 26 a much smaller service was prototyped using 1.5 person month. This service mimics a taxiing application for use on an electronic flight bag. A COTS-based approach was used to implement the service. In the last few days, many small upgrades were implemented, including user comments from actors like pilots. This service was realised by a single partner, with people literally walking into each other’s office. The service generated ideas for three other electronic flight bag services, also unrelated to the services of the demonstration system, which were realised using a similar COTS-based approach. Their realisation took 3.5 person months. These four limited implementations demonstrate the feasibility of COTS-based incremental deliveries in a domain with safety concerns. Like in the custom-made case (Kesseler, 2000), incremental deliveries demonstrate their value in obtaining user feedback as was confirmed by users from all actors consulted.
To conclude the answering of the first part of the third research question, using a COTS-based and evolutionary approach does not necessarily lead to improved responsiveness, but a COTS-based approach can be compatible with improved responsiveness. It seems that for people with a safety critical background it is easier to accommodate user initiated changes in a system fully developed, tested and understood by the development team than to cope with requirements evolution in a COTS-based system. In the latter case available documentation does not always fully cover special cases, resulting in unexpected behaviour during integration. Integration challenges were less with the Aeronautical Telecommunication Network, which is developed and documented using certification compliant processes. Performing more experiments with the COTS software to determine its behaviour would have been beneficial, as was also concluded by (Albert, Brownsword 2002).

A salient difference between the COTS-based case and the custom-made case is the amount of software process artefacts produced. As producing such documentation consumes effort and delivery time, the document volume and its delivery over time will be analysed using the goal/question/metric method to determine their possible influence on affordability and time-to-market. Again a section on results is followed by a context based analysis.

### 7.5.4 Software process artefact results

Certifable safety-critical software processes produce much documentation containing software process artefacts. For all artefacts types the volume is measured in printable pages. The COTS-based case study comprised the following five main tasks. Management relates to consortium-level project management only. The architecture design task includes requirements capture, definition of the architecture and definition of the software process activities including verification. The federated architecture task comprises the detailed design, coding and (partial) unit testing of the COTS-based federated architecture. The demonstration services task comprises the realisation of both demonstration services, the meteo service and the more critical Traffic Information Services. The last task is the study on the airborne certification aspects of the new technologies used, including the supporting tool set. Figure 3 splits the documentation volume, in printable pages, for these main tasks. To extract more information from the documentation volume, the data is split into four classes. Technical deliverable is used for all officially delivered technical documentation. Supporting technical documentation comprises all technical
documentation which was not officially delivered like technical notes exchanged between partners. *Management documentation* groups all non-technical documentation including project management, progress reporting and quality assurance. Previous versions of any of these documents have been labelled *previous versions*. The effort figures in the next sections were derived from the financially audited project administration.

![Artfact volume](image)

**Figure 7-3.** Produced software process artefact volume in printable pages per task for the COTS-based case.

### 7.5.5 Software process artefact analysis

The most recent version of all artefacts comprised 64% of all documentation in the configuration-controlled archive. Previous versions of the same artefacts make up the other 36% of the total document volume. Second versions account for 20%, third versions for 8% and higher versions for the remaining part. So the production of multiple versions of the documentation cannot explain the significant amount of unaccounted effort as observed in the achieved productivity.

Consortium level management documentation amounts to 24 % of the total document volume (Figure 7-3) using 15% of the total effort. Management and quality assurance of the other tasks contributed 12% to the document volume consuming 10% of the effort. Together 36% of the document volume is management documentation using 25% of the effort. Compared with the custom-made case the amount of management effort is reasonable. Although the amount of management
documentation produced is significant, it can not explain the observed challenges in the COTS-based case.

The certification study produced 18% of the documentation (Figure 7-3) using 12% of the effort, in accordance with the original effort allocation. These study results were not intended to influence the first two deliveries, so the certification study activities can not explain the observed productivity challenges either.

Only 11% of all documentation is a technical deliverable, the volume of which is comparable to the custom-made case. The remaining 35% are supporting technical documents like white papers. Most technical exchange, i.e. the cognitive synchronisation of (Cherry, Robillard, 2004), is being performed by exchanging white papers supplemented by numerous e-mails. Only the final result became a technical deliverable. The supporting technical documentation volume is large, especially when compared to the custom-made case. However it can only explain part of the productivity difference at most, as the federated architecture and demonstration services tasks produced relatively small volumes of documentation. The architecture design task, which produced most technical documentation, was completed on time and within budget.

As document volume cannot explain the time-to-market and affordability observations, the document completion time is analysed for information. Figure 7-4 depicts the total document volume over time, as submitted to configuration control. The peak in the architecture-design task document volume in month three is partly due to UML course material. The peaks in month 10 and 13 are caused by internal and external submission deadlines prior to the first client review, with rework completed in month 16 and 17. The second client review caused the peak in month 26 and 27, which relates mainly to updates of the architecture documentation. Some dissemination activities combined with documentation completed for the final review are reflected in the deliveries in month 29 and 30. From month 20 the submitted document volume declined, with only a small volume submitted by the federated architecture and demonstration services tasks. The discussed results confirm that activities concentrated on solving the COTS integration challenges and not on producing documentation. This is in line with the opinion of the team’s key participants based on formal reporting as well as on posterior analysis of their personal activity records.
In summary, the COTS-based case study demonstrated the certification potential of carefully selected COTS for the safety criticality levels considered, both for COTS software produced with compliant processes as well as for COTS software produced using unknown processes when accompanied by a service history obtained using compliant processes. It did not demonstrate the expected responsiveness, affordability and time-to-market advantages. Comparing the COTS-based case with an appropriate custom-made case indicates that the original objectives are reasonable. A salient difference is the amount of software artefacts produced, but these are not likely to explain the different outcome. Using the iterative approach of grounded theory, the next sections re-assess the software process data for a context based explanation.

7.6 **Context based re interpretation of results**

The federated-architecture requirements and the demonstration-services requirements were documented well before month 20. The requirements are considered sufficiently clear as integration activities caused no changes and no major effort consuming misunderstanding between partners have been encountered.
Consequently the reduced productivity cannot be ascribed to lack of documented objectives. With initial requirements documented around month 13, the initial delivery planned at two thirds of the project completion time should have been achievable. The analysis of the software process artefacts does not help to explain the observations with respect to the remaining two parts of the last research questions on time-to-market and affordability.

Re-assessing the software processes of the compared cases, two factors that were observed to be different were management style and team communication. Using the goal/question/metric method, the remainder of the paper analyses them for their possible relevance to these remaining research questions. Again first the results are presented, followed by an analyses section.

### 7.6.1 Management style results

Traditionally in multinational co-operations regular face to face meetings are held to align the partners’ views, to align the partners’ ambitions and to guide the activities. The custom-made case used this model. The COTS-based case planned 46 meetings for this purpose, mostly technical sub-meetings involving only a few partners with a few all partner meetings per year, attended by at least one person from each partner. Regular partner meetings present a significant management cost. At activity start the consortium leader made a deliberate, but implicit, choice for a distributed management style by reducing costly all partner meetings and even sub-meetings to red-flag events i.e. problem solving events. Instead, short weekly phone teleconferences (without supporting graphical or document sharing tools), e mail and electronic document exchange were used to achieve the objectives traditionally achieved by convening meetings. This seemed an innocuous innovation to increase project management efficiency. Such distributed management style works well for much Free/Libre/Open Source Software (FLOSS), which depends on volunteer contributions (Crowston, Scozzi, 2004). Using fully paid professionals should put us in a better position.
Figure 7-5 depicts the realised formal communication pattern. At the first client review a remedial action was prescribed as economic realities caused one partner to severely reduce its contribution. Even then no all-partner meeting was organised, but a number of additional teleconferences were held during the three-month resolution period. This lengthy resolution process taxed consortium co-operation. The remaining nine months were extended to 15 months, without budget increase. Around month 19 it became apparent that the first delivery, re-planned from month 15 to month 21, started shifting again due to the unanticipated and persistent COTS integration problems. At the same time teleconferences were reduced in frequency, due to some key participants not being able to attend. During the last six months all effort was focussed on producing a workable delivery. At the same time formal communication was reduced further (Figure 7-5). Interestingly it took a co-location working meeting at the end of month 29 to get the first delivery working. The co-location confirms the “lock them up together” organisational pattern described by (Coplien, 2001).
7.6.2 Management style analysis

This experience suggests that the adopted distributed management style was not advantageous for a team performing the rigorous software development processes required for certifiable safety critical software. Between organisations most individuals involved did not know each other prior to the start of the consortium, as in the custom-made case. Omitting the initial meetings violates the organisational pattern of “team building during a face to face meeting before working together” (Coplien, 2001). In the custom-made case such initial meetings were held.

The planned weekly teleconferences were envisaged to resolve technical issues but tended to focus on project management issues instead. As a result the teleconferences tended to become biweekly and less frequent after month 20 (Figure 7-5). For one sticky management issue a videoconference was held between two partners in month 23. Due to the additional face to face contact this proved to be efficient and cost effective. Still this affordable facility has not been used since. This success supports the view of (Ramesh, Dennis, 2002) that a virtual team requires information rich media like face-to-face meetings to achieve its goals. Other studies e.g. (Robillard, Robillard, 2000) indicate that 40% of software development is spent on ad hoc collaboration activities. According to (Cherry, Robillard, 2004) 57% of that time is spent on cognitive synchronisation, i.e. developers exchanging information to ensure they share the same understanding. The team felt the voice teleconferences did not support this efficiently.

Lack of effective project-level communication offers a potential explanation why in the COTS-based case deadlines kept changing without a solid consortium level management discussion to analyse the cause and re-assess effort-consuming requirements. Such effort escalation is not uncommon and should have been followed by de escalation in which the detailed requirements were unfrozen, redirected and commitment to the adapted requirements obtained (Pan et al., 2006). The latter, i.e. committing to the adapted requirements, is also referred to as re-freezing. Applying an evolutionary approach (Gilb, 2004) could have helped to reduce the requirements while preserving user value. In the custom-made case such unfreezing, redirection and re-freezing was successfully performed when requirements evolved beyond the initial hardware capabilities. Several face-to-face meetings were instrumental in this process. The lack of such communication in the COTS-based case due to the distributed management style is felt to have significantly contributed to the disappointing responsiveness. Consequently, it is
believed that an appropriate communicative management style would facilitate successful deployment of COTS. The data do not provide evidence that a more communicative management style would have eased the integration issues which caused the disappointing time-to-market and affordability results.

The last characteristic in which the COTS-based case and the custom-made case differ significantly is the amount of e-mail exchanged. This team communication is analysed to determine whether it contributed to the time-to-market and affordability observations.

### 7.6.3 Team communication results

For technical discussions nearly no meetings were organised (Figure 7-5) and only a limited amount of technical documentation was produced by the realisation tasks (Figures 7-3 and 7-4), so the numerous e-mails are analysed for a possible explanation of the observed challenges. A total of 7143 e-mails have been exchanged, 9.5 times as much as the 749 messages in the custom-made case. Figure 7-6 shows the distribution of the number of e-mails over time.

![Figure 7-6. Monthly distribution of the number of e-mails send per task for the COTS-based case.](image)

The peak in the number of management e-mails after the first review in month 15 reflects arranging a suitable remedial action. Management e-mails also rise from the second review onwards (month 26 to 30) to arrange a suitable activity completion. The architecture design activities are concentrated in the first year. The same holds
for the e-mails, as expected. After that the realisation activities (federated architecture, demonstration services) take over which is reflected in the monthly e-mail distribution. The e-mail peaks and valleys do not coincide with the submitted document volume in Figure 7-4, confirming they measure different processes.

Figure 7-7. E-mail distribution per sender/receiver per work package for the COTS-based case.

Figure 7-7 provides the distribution of the e-mails per task, further subdivided for sender and receiver. Only 8% of the e-mails relate to communication with parties outside the consortium, labelled “extern” in Figure 7-7. Virtually all this external e-mail is related to setting up user interviews and attending conferences, activities which stayed within their allocated budgets.

7.6.4 Team communication analysis

The e-mail distribution confirms the participants’ opinion that due to the distributed management style a lot of issues have been discussed using e-mail. The e-mail volume might suggest that e-mailing consumed a significant amount of time. Surprisingly, for the three tasks the researcher’s organisation participated in (management, architecture-design and study), the company internal e-mail volume exceeded the volume sent directly to partners. However as the architecture-design
task achieved its objectives, the observed company internal e-mail volume was considered effective and did not produce the challenges of the federated-architecture and demonstration-services tasks. The same holds for the management task. The e-mail volume confirms the importance of e-mail for team communication.

The study task, with two partners, used roughly two thirds of the effort of the architecture-design task with all five partners while involving a similar number of individuals. The e-mail volume exchanged in the study task is much smaller. This reflects the strict work division in the study, obtained during the face to face start up meeting, confirming the organisational pattern (Coplien, 2001). Even the study task generated a significant company internal e-mail volume, mostly to exchange ideas and for informal document reviews. For this the partners considered e-mail as an effective communication medium. Consequently the observed e-mail volume does not deviate a significant amount of effort from the main objective, i.e. this does not explain the challenges encountered in the architecture-design and demonstration-services tasks.

The e-mail volume in the federated-architecture and demonstration-services tasks visible at the consortium level is substantial compared to the other tasks. In the other tasks this accounted for a quarter of all e-mail. This suggests an even more substantial amount of additional person to person e-mail, which could not be detected. This e-mail volume seems a consequence of the chosen distributed management style. These e-mails indicate that traditional face to face meetings might be more efficient than inferred by their perceived lack of concrete results, possibly by performing cognitive synchronisation (Cherry, Robillard, 2004) as well as team building. The voice-only teleconferences tended to disappear when the pressure got higher (Figure 7-5), as all participants, including those at the critical path, needed to be available at the same time. As e-mail is asynchronous it is perceived as being more efficient. The distributed management style combined with the reduced voice teleconferences meant management was not effective, especially in the critical last period. This finding contrasts with the custom-made case where project control allowed accommodating the evolving user requirements reflected in many incremental deliveries. Whether more information rich media might be able to effectively support a distributed management style warrants more study.

The obtained experience suggests that when deadlines start slipping, management should start an open minded analysis (also referred to as blame free in air transport) of the situation, possible causes and potential remedial actions. With the benefit of
hindsight this could have been done within the available management budget. Changing the management paradigm needs more attention than just replacing face to face meetings with teleconferences and e-mail. Readily available research results on team management should and could have been taken advantage of when it became clear that the current style was not effective.

This case does not provide evidence that a more traditional management style could have solved the COTS integration challenge more efficiently resulting in a better affordability. The uncontrolled extension of the integration effort and time resulted in the uncontrolled reduction of integration tests and skipping of verification tests. Omitting these tests reduced feedback on the assumptions and technical requirements, important aspects of the evolutionary approach. The COTS-based case suggests that using COTS software in a safety critical environment does not necessarily lead to lower costs. Recently (Scott, Gollings, 2005) came to the same conclusion. The resulting amount of effort in our case taxed the time-to-market.

In summary, whereas this study has indicated that the deployment of COTS-based software in safety critical domains is possible from a technical point of view and can be certifiable, research has yet to demonstrate the expected time-to-market and costs benefits of such systems in this domain. The following section will discuss this in greater detail.

### 7.7 Conclusions

In the general domain, COTS-based software solutions provide benefits such as improved responsiveness, reduced time-to-market and lower costs with respect to custom-made software. In the certifiable safety critical domain COTS solutions have not been widely adopted. This study assesses COTS software for a specific safety critical domain, air transport.

The certifiability of COTS software is examined by studying the relevant air transport certification documents. COTS software can be certifiable by selecting COTS components developed using compliant software processes. Note this has to include obtaining access to the software process artefacts of the supplier. For systems relying on COTS components developed using non-compliant software realisation processes, certifiability to lower criticality levels implies complying with strict requirements on service history. Such safety criticality levels comprise by far the majority of the certifiable air transport software. Due to commonality in the
principles of safety certification, similar certifiability results could be expected in other certifiable safe domains like chemical, rail, automotive and medical. However, in each case careful study of the pertinent certification documents is needed for confirmation.

To obtain results with relevance to practise, the applicability of COTS software for this domain is studied by performing a case study realising a COTS-based prototype. The obtained results confirm the applicability of a COTS-based approach for this domain. The expected advantages on reduced deployment time of new software releases, simultaneous support of software classified at different safety criticality levels, self-forming self-healing capabilities and increasing the independence from the lower layer communication services have all been observed. In similar cases, where requirements in a certifiable safe domain can be satisfied by COTS, similar results might be expected. More empirical data will strengthen this conclusion.

The use of COTS software is justified by reduced time-to-market, increased responsiveness to evolving requirements and improved affordability. Unfortunately these could not be observed in the case study. To address this, the COTS-based case was compared with a carefully selected custom-made case.

The distributed management style lacking information rich communication turned out not to be conducive to obtaining responsiveness or effective project control. The latter, i.e. the level of project control achieved, also inhibited the time-to-market objective. The work presented does not allow conclusive statements on the cost of the COTS-based approach with respect to a custom-made approach for certifiable safe software. Additional work in a better controlled practical environment is recommended.

Based on the outcome of this study and others (Schmidt, 2002), (McKinney, 1999), (Alev, Finkelstein, 2002), (Albert, Brownsworth 2002), and (Carney et al., 2003), the management style should be compatible with the software development paradigm used. In particular, this study suggests not changing both at the same time, and that management style can significantly influence process outcome, possibly to an even greater degree then the software development paradigm used. This is a subject justifying additional research.