Distr.
GENERAL

WP.9
13 April 2010

ENGLISH ONLY

UNITED NATIONS ECONOMIC COMMISSION FOR EUROPE (UNECE)
CONFERENCE OF EUROPEAN STATISTICIANS

EUROPEAN COMMISSION
STATISTICAL OFFICE OF THE EUROPEAN UNION (EUROSTAT)

ORGANISATION FOR ECONOMIC COOPERATION AND DEVELOPMENT (OECD)
STATISTICS DIRECTORATE

Meeting on the Management of Statistical Information Systems (MSIS 2010)
(Daejeon, Republic of Korea, 26-29 April 2010)

Topic (i): Developing common high-level architectures

An event-driven architecture for data collection

Prepared by Jakob Engdahl, Statistic Sweden, Sweden

I. Introduction

1. During the most recent years, Statistic Sweden has realized that a substantial part of the development budget has gone to the data collection and data processing process and its underlying IT systems. Focus has therefore been to modernize these processes and IT systems. To get a coordinated view of the development in these processes, the development has been gathered under the name Triton.

2. This document describes the architectural forms that Statistic Sweden has considered reasonable alternatives for the Triton development. The architectural decisions that have been made to support the vision of orchestrating service oriented functions are described as well as our experiences with the event-driven service oriented architecture that has been created. Towards the end of the document, thoughts and ideas about possible international collaboration are described.

II. Background

A. Process orientation

3. Today, all statistical subject matters have their own IT system to support most functions in the statistical production. This has spread the logic for a specific process to a lot of different IT systems and new requirements for a specific process are complex to implement since all IT systems have to be updated. Example of new requirements can be that a new ISO standard should be implemented.

4. Statistic Sweden has had the ambition to adopt a process oriented work form since a couple of years back. Therefore, new IT systems should be created to support a specific function or activity instead of a specific statistical subject matter. When an IT system is created for a certain activity, the same logic should be removed from the statistical subject matter specific IT systems.

B. Vision for service-oriented architecture

5. Statistic Sweden has had a vision to adopt service-oriented architecture (SOA) and also provide the subject matters with the possibility to choose from centrally developed services and functions to construct their own production system. Their own IT system will then be the combination of selected services and functions.

III. Requirements

6. Within Statistic Sweden, the situation is similar to other national statistical offices with a lot of subject matter specific IT systems in "stovepipe form". An exception within Statistic Sweden is that part of the data collection and dissemination process has had a more process-oriented form. An experience from this is that it is quite possible to centrally develop IT systems that several subject matters can use. Another experience from this is the problem with data adapters to and from the centrally developed IT systems. When a subject matter should use a centrally developed IT system for a specific activity in the process, a new adapter has to be created to transfer information from the subject matters IT system to the centrally developed IT system. This part of the production process has proven costly and require a lot of IT resources. Since the adapters are created uniquely for each subject matter and activity, the adapters often create dependence to a specific IT staff member.

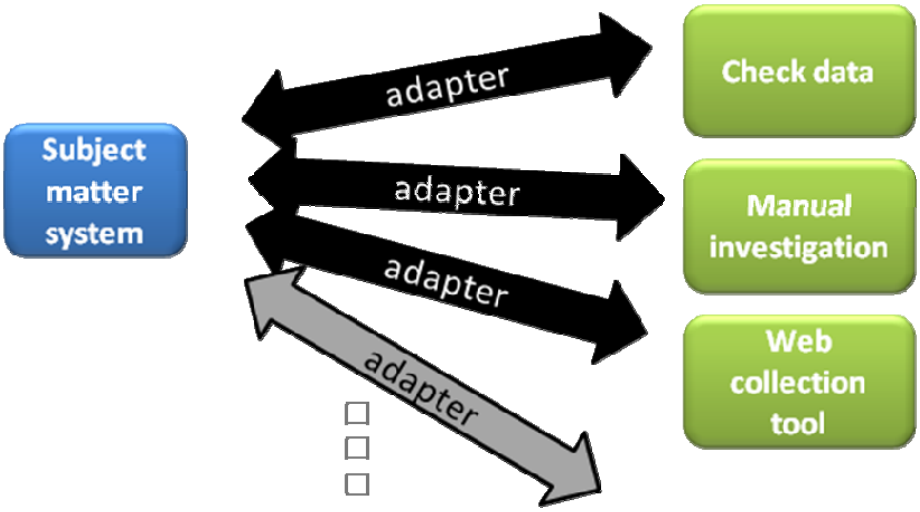

Figure 1: "Adapters between subject matter specific system and data collection IT systems"

7. During the development of Triton, both the vision of a service oriented architecture and our experience with the problems with data adapters were combined.

8. The reuse of metadata has so far been relatively low and has resulted in a heavy documentation burden and lower quality within the production system. A clear requirement when building a new data collection system is that the IT systems being build should be metadata driven and information entered in a previous IT system should be reusable in later processes.

9. "Big bang implementation" there all functions has to exist before any product can be implemented should be avoided. Implementation should instead be possible both product-wise and process-wise.

10. To build an effective data collection process, Statistic Sweden is implementing selective editing so that manual editing only has to be done on a minimal number of objects. Therefore a logical start was to integrate the new IT system for selecting suspicious objects; Selekt, with the new IT system for manually investigating the suspicious objects; EDIT. Data delivery from our previously existing web based data collection tool was also integrated.

III. Development method

11. A large part of our development budget is going to the data collection process and since this project is intended to address this problem, it has also had the support from the leaders within Statistic Sweden. This has also been crucial for the success of the project.

12. Development has been performed in several areas, sometimes parallel and sometimes sequential. At some points, parts of the work had to halt to await input from other areas. The work in total has not had a waterfall approach but has instead had a more iterative and in some ways agile form.

A. Roadmap

13. The statistical subject matters were divided into different categories depending on which functions they use in their statistical production. Together with information about how complex their IT systems were, a roadmap was created to support the development of the Triton platform. This roadmap was then used to prioritize the implementation and integration of different functions in Triton.

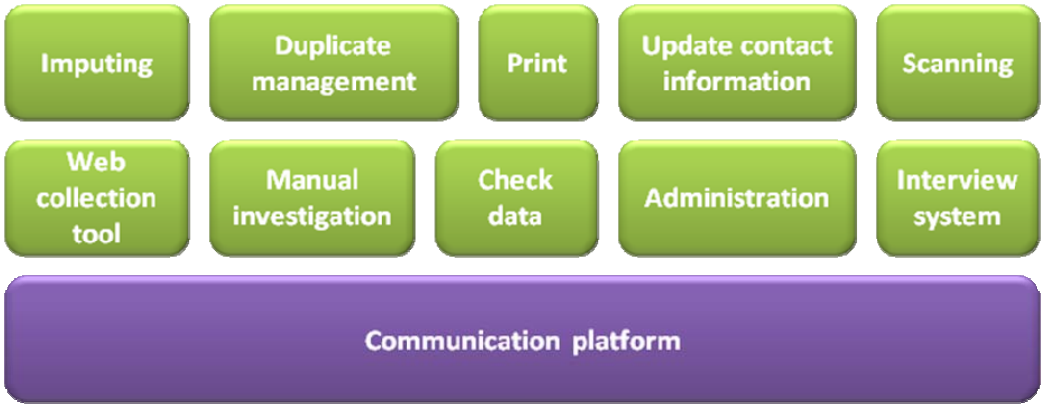

Figure 2: "Some of the functions in the roadmap"

B. Analyzing the processes

14. A more detailed model of the activities included in the processes was created to get a better view of the requirements. Information about the necessary data and metadata needed to execute the activities was added to the description of each activity. Chronological descriptions and information about the roles responsible for execution was also added.

C. Information models

15. To support development of IT systems for activities in the process, information models was created that described central business objects in the production processes. Since several systems should be integrated, the requirement to achieve a common view of the business objects was prioritized.

Figure 3: "Input and output of an activity"

16. The information models were then combined with the process models to describe which business objects was used and created in each activity.

D. Avoiding “boiling the ocean”

17. An experience learned both from our own previous work and the experience of other national statistical offices was that “boiling the ocean” isn’t a successful approach. To avoid getting stuck in the analysis phase, the group defined the scope of the platform with regard to the statistical process. This also simplified the work with the roadmap.

E. Selecting architectural form

18. When viewing the components necessary to build IT systems their expected stability differs. The stability of each component in decreasing scale is business objects, business processes, architectural form and software. This means that the analysis work on the business objects and business processes will outlast the architectural form and software. The architectural form should therefore not be too influenced on our current set of software but should instead be created with the intent to support our business objects and business processes.

19. The vision for the Triton architecture is an architecture that:
- (a) supports each process and activity with the necessary data and metadata;
- (b) has a service-oriented form;
- (c) is metadata driven;
- (d) meets the quality requirements such as stability, security, traceability etc;
- (e) gives the subject matters the possibility of choosing relevant services and functions;
- (f) lessens the dependence between the process services;
- (g) helps create a overview of the progress of the collection process;
- (h) does not require unique adapters between each subject matter and function or service;

IV. Architectural choices

20. To support a process-oriented business, the focus shifted from *“where is the information stored”* to instead supporting processes with the needed information in the form of data and metadata.

21. On high level, four architecture forms were discussed and compared with regard to the requirements.

- (a) Traditional SOA with business object focus;
- (b) Traditional SOA with business process focus;
- (c) Event-driven SOA with choreography;
- (d) Event-driven SOA with orchestration;

22. Event-driven SOA is sometimes described as SOA 2.0 and targets some of the criticism regarding implementing traditional service oriented architecture. Service orientation describes a number of principals like *service encapsulation* and *service loose coupling* etc which gives the solution quality advantages but also leads to some overhead when it comes to the communication. This could reduce the performance and give the end user a slower system since service calls are performed in a synchronous form.
In event-driven SOA the communication form is reversed so that information is communicated as soon as it is created or updated instead of when it is needed. This means communication can be performed in an asynchronous way and information is transferred to all functions when no user waiting for service calls to complete. The overhead in the communication is still there but the end user will not notice any reduced performance since the communication is performed before the user tries to access the new information. In other words, the communication is no longer the bottleneck in execution time in an event-driven SOA architecture.

23. When we integrate systems we often have to weigh temporal quality, of the information, against performance. Any system that requests information from a master data storage system will always get a snapshot of how the information is at the time for the request call. Depending on how important it is that the gathered information is still correct and how often the information in the master data storage is updated, the temporal quality of the gathered information will decrease in different rate. To avoid using obsolete information the request has to be redone in an appropriate frequency. Like once a minute or once an hour etc. This will, however, create a lot of unnecessary service calls.

24. In an event-driven architecture this is avoided by instead transmitting the information to all functions once it has been modified or created. The temporal quality is higher since all functions knows that the information in their system is up to date and corresponds with the information is the master data system and the load on the infrastructure is lowered since unnecessary service calls is avoided.

A. Traditional SOA with business object focus

25. Traditional SOA is often based on encapsulating master data systems with service interfaces. Each function like *web data collection tool* and *manual investigation* then gather information by calling the services on the master data systems.

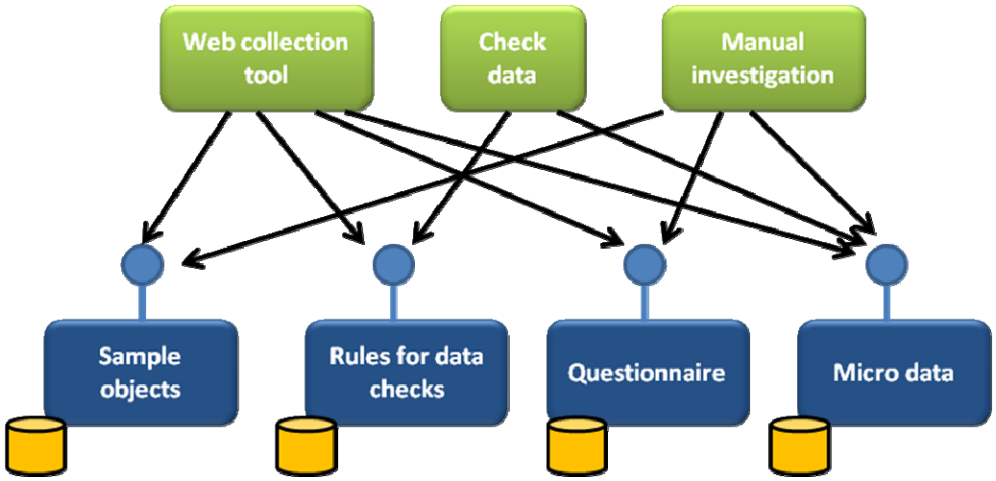

Figure 4: "Traditional SOA with business object focus"

26. A problem with this type of system is that the execution time for a function now depends on the performance on all the information services which is called to gather the information needed to perform the execution of the function. If the function for presenting a questionnaire to a user becomes dependent on performance on the *questionnaire service*, *micro data service*, *data editing rules service* and the *contact information service*, the function will be fragile and slow. The sum of the execution time for all services will probably make the function unusable to the end user. Every service needed in the execution time will also become a single point of failure. If any one of these services isn't responding the function will render unusable.

27. In solutions like this, there is also often a need to add some form of permission and authorization service to decide what information the service caller can access. This in turn creates a hierarchy in the information services which will reduce the performance even more and make the total solution even more fragile.

28. This architecture form requires a lot of synchronous service calls and this means that the quality requirements are difficult to meet, and nor will the form of the architecture help create an overview of the progress of the data collection.

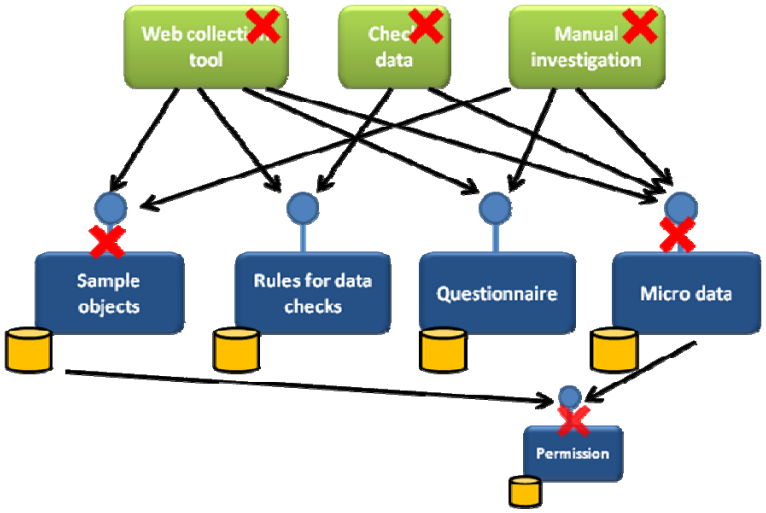

Figure 5:"Problem with hierarchy solutions"

B. Traditional SOA with business process focus

29. Another solution is that each function, like *check data* and *manual investigation*, provide services to receive all the information needed within the function. The subject matters will then prepare the execution of the function by providing the necessary information by calling these information services.

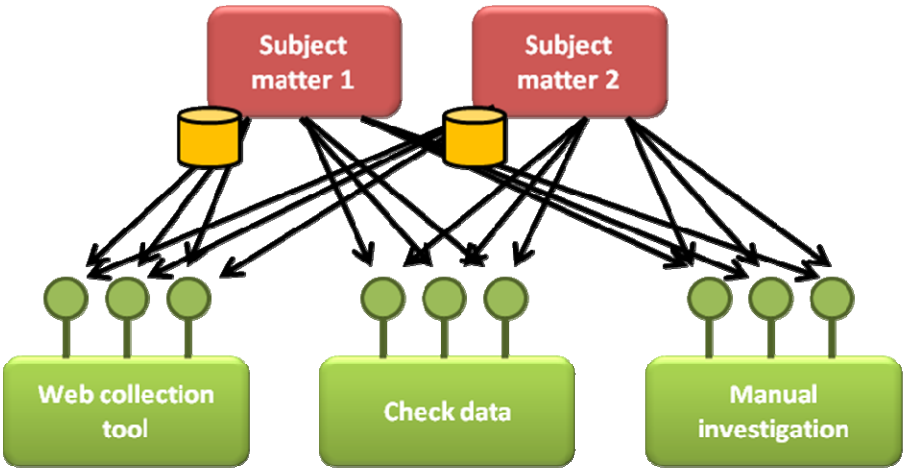

Figure 6: "Traditional SOA with business process focus"

30. This solution has an advantage over the previous architecture since all the information needed to perform the execution of the function is already in place when execution should be performed. The solution does however require the subject matters to keep all the needed information locally in their system. If the structure of a specific data or metadata needs to be updated, it is much more difficult to implement the update since a lot of subject matter systems have to be updated. This also makes it more difficult to move logic from the subject matter systems to new process oriented functions when these are developed.

31. Another problem with this solution is that it requires unique communication between all functions and the subject matter systems. This problem therefore means that the architecture form does not meet the requirements on the architecture.

C. Event-driven SOA with choreography

32. A way to establish an event-driven SOA is to let each function pass forward the created and modified information to the next function as soon as it is available. This would mean that the "*web collection*" function would send the collected answers to the "*Check data*" function. The"*Check data*"

function would then execute the function to check the data and then pass the information forward to the next function etc. The delivery of information is therefore performed before the execution and not during.

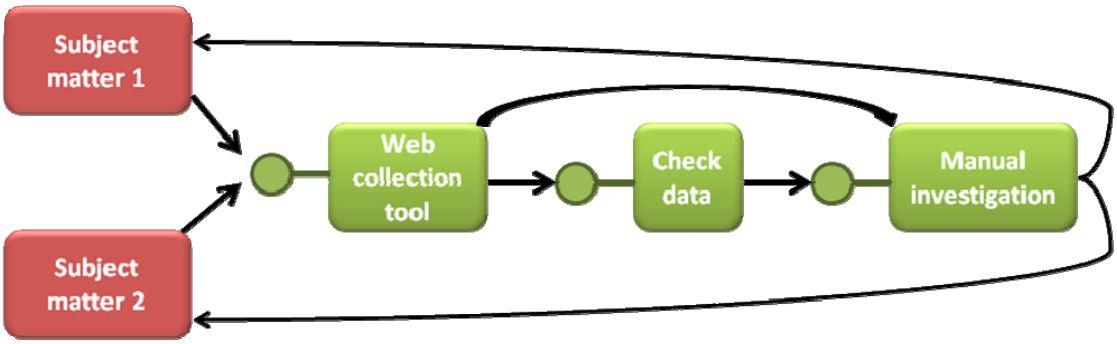

Figure 7: "Event-driven SOA with choreography"

33. This solution would meet most of the requirements on the architecture, since a lot of the quality requirements are met and it would be possible to move logic from the subject matter systems to the new functions. The solution also avoids the need to create unique communication adapters between the subject matter systems and the functions since most of the communication in this solution will take place between the functions.

34. Since each function communication with the next one in a predictable way, it would also be possible to monitor the communication and help create an overview of the progress of the data collection process.

35. The disadvantage of this solution is that it is difficult to reduce the dependency between the functions since they need to communicate with each other. This also means that the possibility for a subject matter to decide which function and services that is relevant is more difficult to implement. A way to create this functionality is to let each function receive configuration information which tells the function which the next function is that should be called. This would however create even bigger dependencies between the functions since each function would have to be able to communicate with a number of subsequent functions.

D. Event-driven SOA with orchestration

36. Another way to implement event-driven SOA is to separate the functions, the subject matter systems and the configuration describing which functions that should be used and the order. To communicate between the subject matter systems and to communicate between different functions, a communication platform is instead used. Each function knows only how to communicate with the communication platform and therefore the dependency between the functions is low. The same applies to the subject matter systems which communicates only with the communication platform and does not connect directly to any function.

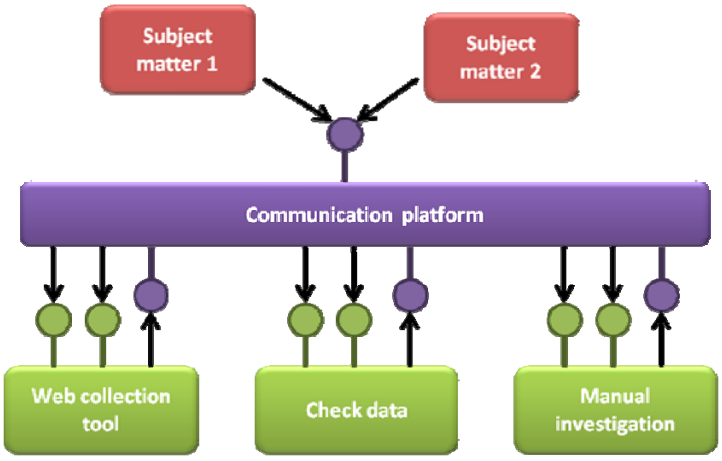

Figure 8: "Event-driven SOA with orchestration"

37. Each subject matter “orchestrates” how the information flows between the functions by configuring the communication platform. When information is received by the communication platform, the configuration, set by the subject matter, is read and the communication platform passes the information forward to the next function.

38. Since the communication in this solution is centralized it is also easier to implement functionality to give an overview of the progress of the data collection process. Energy to meet the quality requirements could also be focused on the communication platform so that requirements like security and traceability is met.

39. Since information is communicated in an asynchronous form in this solution it is also easier to meet performance requirements in the functions.

40. When the communication is centralized, the communication platform itself becomes a single point of failure. This does however not hinder the functions to continue execution if the communication platform goes down. It simply means that no new information is delivered to the functions until the communication platform is back up.

41. Since this architecture form offers the best solution to meet the requirements, the Triton platform was built based on *event-driven service oriented architecture.*

V. Implementation of event-driven architecture with orchestration

A. Information flows

42. When the communication platform receives some information and should pass it forward, it does so by the information flows that has been set up. These information flows takes viewpoint in how a specific business object should be transmitted. The design of these information flows is therefore a crucial component when describing the business processes and the requirements on the platform.

43. An exception there the information flows isn’t enough to support the functions is when an unexpected event occurs. An example of this is when a respondent calls and asks for information about the answers he or she has provided. In this case the system makes a search in the data storage system for micro data by calling a request-response-service.

B. Flexibility

44. When creating process oriented IT systems, the need to support a lot of subject matters has to be balanced against the risk of creating to complex tools. By developing the orchestration functionality, in the communication platform, in a way so that the subject matters can describe how exceptions in the information flows should be made, we can still provide flexibility in the solution without losing control. A subject matter could then decide if an existing tool meets the requirements or if a subject matter specific tool has to be created to support a specific activity or function. The information flows and communication platform can still be used to handle the communication to the subject matter specific tool. The subject matter could for example decide if the centrally developed tool for checking the answers; SELEKT is the correct tool the specific subject matter or if another tool has to be made for this function.

C. Master data and distributed systems

45. When information is distributed in several systems, a problem often arises when updating the information. All functions then have to get access to the updated information. The event-driven form of the solution simplifies this problem since any modified or created information is immediately distributed to all systems. This minimizes the risk of working with incorrect information in each system but makes it even

more important to keep track of which systems are allowed to modify and create the business objects. To avoid creating complex information flows, it is reasonable to try to limit the number of systems that are allowed to create and modify a specific business object.

D. Metadata driven platform

46. The form of the architecture does not itself make the solution metadata driven. To accomplish this, the perspective must be widened when developing and setting the input and output for each function. It is not enough to be able to handle data in the form of micro and macro data. The functions also need to handle other types of business objects. This simplifies the reuse of previously entered information and does not require that the same information has to be manually entered in all systems. This supports the vision for a metadata driven architecture.

47. Since several systems are integrated and are expected to communicate with each other, it becomes very important to have a common view of the business objects. A prerequisite is therefore that information about a business object entered in one system, can be interpreted by the other systems.

E. Middleware

48. An event-driven SOA platform could be implemented completely from scratch. On the software market there is however a number of software providers that offer products that simplifies the development of an event-driven SOA-platform. Statistic Sweden has a software policy that means we should first search the software market, before developing something *"in house"*. Since the quality requirements on the communication platform will be high when it comes to security, performance, availability etc, investing in a middleware seemed reasonable. Statistic Sweden had also previously developed solutions based on Microsoft BizTalk which seemed suitable for implementing an event-driven architecture.

F. Overview of process and activity progress

49. Since the communication platform is responsible for transmitting information between the functions, it is also possible to add a monitoring capability within the platform to support the need for creating an overview of the progress of the process.

50. When information is sent between the functions, the information that the communication took place can be used as process information. This process information can then be used to create insight into the progress of the overall process. Example of information that can be gathered is *"number of respondents who answered the questionnaire using the web"*, and *"number of respondent answers that is queued to be manually investigated"*. By providing the production staff with process information, they will be able to make more informed decisions.

51. By adding the process information functionality to the communication platform the requirements regarding process information support in each function is lowered. All functions developed within the data collection and editing processes isn't required to add their own process information functionality unless very specific process information is required that only concerns a specific function.

G. Shell application

52. The requirement to create a common user experience between all functions sets new requirements when developing the graphical interfaces for each function. Work has been initiated to accomplish this by designing a *"concept design"* for all graphical interfaces in the Triton platform. This describes the guidelines for how the interfaces should be designed.

53. Together with this, a technical framework, called *"shell application"*, has been created to assist the development of the graphical interfaces. The *"shell application"* itself does not provide any specific

functionality to support a specific activity in the data collection or editing process but is instead a windows *"framework-application"*. Within this *"framework-application"* the graphical interfaces for several functions run. Even if the *"shell application"* is a windows application framework, the purpose of the system could be compared to a web browser. A web browser itself does not provide any news, movies, blogs etc but acts instead as a framework for visiting different types of websites. These websites can provide *"functionality"* in the form of news, movies etc. So instead of switching applications when the user needs to access a different functionality, the user switches to another *"tab"*.

54. The reason for creating this *"framework-application"* is to be able to keep the functions loosely coupled without requiring the user to have a lot of different applications running at the same time.

VI. Experiences so far

55. To create a communication platform means investing a lot of resources in quality- and architecture aspects. These are not always easy to demonstrate since a lot of this functionality is implemented *"under the hood"* and does not provide graphical interfaces. The consequence of this can be that the business feels the return on the investment is low. It is therefore crucial that a project like this has an informed internal customer who understands the value of investing in architecture and not only technology.

56. It is also important to begin implementation of subject matters early to validate the architecture and to give early return on investment. If no return on investment can be showed, it is important to allow taking a step back and rethink the ideas. If the development continues for too long before any implementation is done, there is also a risk that the project becomes *"to big to fail"*.

57. So far the biggest problem hasn't been creating technological solutions for the architecture, but has instead been more regarding business requirement analysis. The business is not accustomed to describing their requirements in a vocabulary that isn't subject matter specific

58. The architecture should not prohibit the use of clever workarounds but should instead be used to minimize the need for them. If a specific functionality requires a technical solution that isn't completely aligned with the architecture, it should still be considered an option.

59. It is important to find room for adopting architecture, and not simply encapsulate or convert existing systems in to services. The architecture form of Triton has made many of the design decisions simplified.

VII. Continued work

60. Within Statistic Sweden, the work continues to create support for the data collection and editing processes by expanding the Triton platform. The road map describes the work until spring of 2011, when most parts of the system is scheduled to be complete.

61. The interest of collaborating internationally when developing IT tools, to support the statistical production process is high. We see a big collaboration potential when developing functions to support a specific set of activities or processes in the generic statistical business process model (GSBPM).

62. The order that activities are performed and *"information flows"* in between might prove difficult to share. The form of the architecture does not however, limit the possibility for collaboration when it comes to building tools for a specific function or activity.

63. Just like the GSBPM describes the business processes, a generic statistical business information model (GSBIM) could be used to describe the information architecture. This would provide a more complete view of the requirements when collaborating internationally to create common IT tools for the statistical process. The information model is not dependent of the architecture or technology used for implementation so it should be compatible in any organisations that base their processes on the GSBPM.