MIXED Review

Company	DANS
Partners	Software Competence Center Hagenberg GmbH
Project	MIXED Review
Deliverable	D1.0
Document Name	MixedReview.doc
Prepared by	Josef Pichler
Version	1.0

Kplus
Kompetenzzentren-Programm
Funded by the Austrian government,
the State of Upper Austria, the Johannes Kepler University

Software Competence Center Hagenberg GmbH, Softwarepark 21, 4232 Hagenberg, Austria
Phone +43 7236 3343 800, Fax +43 7236 3343 888, office@scch.at, www.scch.at
FN 184145b, Landesgericht Linz, UID ATU 48056909

Contents

1 Introduction

DANS (Data Archiving and Networked Services) initiated the MIXED project about migration to intermediate XML for electronic data. The MIXED project addresses the problem of digital preservation of documents, spreadsheets, and databases.

After project start in January 2007 and the definition of the overall project goals, the project team has defined the software architecture to address project goals.

The purpose of this document is to provide early feedback on the MIXED architecture in the face of project goals of MIXED, mainly long-term use of the framework and interoperation with other frameworks and repositories.

1.1 Subject of Review

The review is based on the provided architecture document [1], the White Paper [2] available on the project home page[1] as well as communication with MIXED team. The MIXED architecture is organized in three levels (see [1]):

- Enterprise Architecture
- Information Architecture
- Technical Architecture

This review document concentrates on the *enterprise architecture*. The *information architecture* (mainly, the M-XML format) is currently under development at DANS and, hence, only some general aspects are subject of this review. The *technical architecture* is not covered by the actual architecture document. However, identified standards and technologies intended for MIXED (see [2]) are analyzed.

The review provided in this document is intended for the architect and software developers of the MIXED project team.

1.2 Contact Information

SCCH

Wolfgang Beer
+43 7236 3343 858
wolfgang.beer@scch.at

DANS

Dirk Roorda
Dirk.Roorda@dans.knaw.nl

Kris L.Klykens
Kris.Klykens@dans.knaw.nl

1.3 Structure of this Document

Chapter 2 gives a short summary of review results. Chapter 3 contains the review results in more detail.

[1] Project home page: http://mixed.dans.knaw.nl

2 Summary

This section summarizes the review results. First, results are organized into a SWAT matrix. Second, certain questions from the MIXED team are answered according to the review results.

2.1 Strengths and Weaknesses

Due to the premature project phase, no weaknesses can be recognized at this moment.

Strengths	Weaknesses
■ *Service Oriented Architecture* SOA facilitates both implementation of MIXED components as well as integration of certain components or entire MIXED system with other frameworks and products. ■ *Plug-in concept* Plug-in concepts are emerging in various domains leading to long-term and scalable architectures.	

2.2 Opportunities and Threats

Opportunities	Threats
■ *OAIS reference model* Following the OAIS reference model provides common understanding of use cases, functionality, and concepts required for integration of MIXED with other frameworks and products. ■ *Intermediate file format* The migration strategy based on M-XML is a novel concept with the potential for a long-term framework. ■ *Business process engine* Using a process engine that executes business logic implemented in a higher language like BPEL has the potential to accelerate implementation and maintenance of business logic.	■ *Early selection of technologies*. An early and uncritical selection of technologies like BPEL and ESB may lead to a technical architecture that does not meet requirements and integration use cases. ■ *Non-standardized file format* The definition of a new, non-standardized file format originates a strong dependency between archived data (in M-XML) and the MIXED system.

2.3 Questions and Answers

Besides general review results, following questions (defined by DANS, see Appendix A: Email 26.6.2007) are answered.

- Does the current plan support the goals of MIXED?
  At this moment, there is no obvious reason that could harm this project.
- Is it generic enough to support a long-term use (also taking into account a landscape where multiple frameworks work together)?
  A service-oriented architecture based on standards like SOAP and following a reference model (OAIS) facilitates both long-term use as well as integration with other frameworks.
- Should we consider a different approach instead of ESB?
  Some concepts and common functionality provided by ESB products are required anyway for the MIXED system (service container, SOAP bindings, etc.). The usage of a business process engine (for BPEL) may have potential to implement the business logic based on available service.
- Is the concept of loose binding between the different components as laid out the best approach or should we consider tight binding for some specific components?
  MIXED components are well decoupled by means of services. Loose coupling between file detection, and service providers is a prerequisite for long-term use of the framework and integration with other frameworks. The coupling between a plug-in container and both service providers could be decreased by using an internally used service interface not published to the ESB.
- Are the options chosen in the architecture favorable or are there other design patterns or practices that would allow us to build a better platform?
  Service-oriented architecture is favorable in general. Plug-in model for converters is favorable with regard to scalability and integration of external plug-ins.
- Do we need a distinction between internal and external plug-ins? What is the best way to manage this complexity?
  No distinction between internal and external plug-ins. One plug-in model shall be used that is aware also of some certain properties of external plug-ins like availability, access right, and latency.

3 Architecture Review

3.1 Enterprise Architecture

Figure 1 gives a schematic overview of components and there relationship that constitute the MIXED architecture. MIXED follows a service-oriented architecture where following components are available as first-class services:

- Orchestration. The orchestration represents the business logic within MIXED. The business logic will be composed with respect to available plug-ins and data-format of given document (ingest) or expected document (dissemination). The business logic implemented in this component (file detection and file conversion) is published to the outside world.
- File Detection. This component provides detection of a given file (document) and delivers the classification including file type, version, vendor, etc. necessary to select appropriate converters available as plug-ins.
- Plug-In Service Provider. This component is the access point to plug-ins available in the system locally (internal plug-ins) and externally (external plug-ins). The difference between internal and external plug-ins is transparent to clients (other components like orchestration).
- Plug-In Assembly Provider. This component provides metadata about available plug-ins to other services like orchestration. Metadata includes information about the file format and about the quality of service (QoS) like availability and access rights (especially for external plug-ins).

Furthermore, infrastructure services for logging, monitoring, management, configuration, and security are available to administrate components within the MIXED architecture.

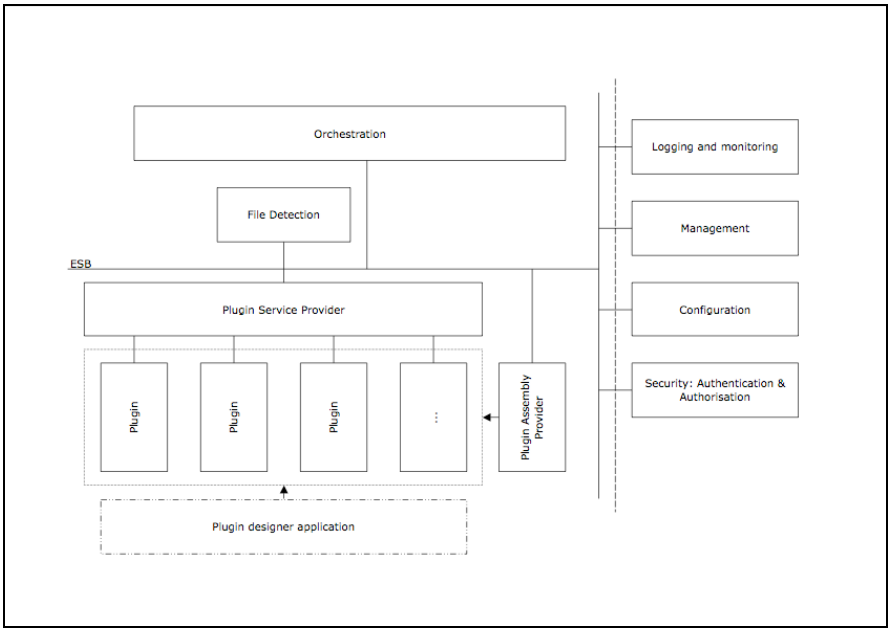

Figure 1: Schematic Overview (see [1])

3.1.1 Service Oriented Architecture

Service Oriented Architecture (SOA) represents a popular architectural paradigm for applications, with Web Services as probably the most visible way of achieving an SOA. Most properties of a service-oriented architecture directly meet requirements of MIXED.

The entire MIXED system can be viewed as service that publishes business logic provided by the orchestration component to the outside world. The provided service may be used directly by users (through a simple client UI) or may be integrated with other services and products within the domain of archival information and digital preservation. A lot of research projects emerged in these domains that cover certain aspects like repository systems. As consequence, integration of MIXED with other research projects and products on service-level appears correct and important.

The implementation of MIXED components as services (like file detection) facilitates weak coupling of individual components. Furthermore, published services are a prerequisite to use a business process engine to implement the business logic of MIXED.

As consequence, SOA is the state-of-art architecture for MIXED.

3.1.2 Enterprise Service Bus

The concept Enterprise Service Bus (ESB) emerged as the next generation platform for Enterprise Application Integration (EAI). ESB platforms typically follow a service-oriented architecture (SOA). For instance, platforms like *Open ESB* and *JBoss ESB*, which implement the Java Business Integration (JBI) standard, are based on a service-oriented architecture. However, this is not a prerequisite.

Table 1 lists key concepts of ESB and JBI and its relevance to the MIXED project.

ESB/JBI Concept	**Relevance to MIXED**
Service Oriented Architecture[2]	Strong
Communication Standard (e.g. SOAP)	Strong
Normalized Message Router	Weak
Business Process Engine (e.g. BPEL Orchestration)	Weak
Security	Strong
Management	Strong

Table 1: ESB/JBI concepts and their relevant to MIXED project

[2] Enterprise Service Bus does not enfore SOA but most ESB products follow SOA.

SOA along with communication standards are key concepts of most ESB products and, hence, strong related to MIXED as already mentioned in section 3.1.1. However, the focus lies on SOA and communication standards and not on ESB per se. As already mentioned, ESB does not enforce SOA.

At this stage, Normalized Message Router (NMR) provided by ESB is not essential for the MIXED architecture. As pointed out by Kris Klykens [4], "*...NMR is not a key requirement but can be interesting to use...*".

A Business Process Engine (e.g. BPEL engine) is not a key requirement of MIXED. However, it is a chance to get experience with it within this domain which is arguable within a research project. See section 3.1.3 for more on this topic.

Security and management are key requirements within MIXED, hence, an ESB providing a sound infrastructure ensures maximal reusability of such service components. Again, the focus lies on a sound infrastructure and not on ESB per se.

As consequence, an ESB product may be a good starting point for MIXED. EBS provides a sound platform for a service-oriented architecture including service infrastructure, communication standards, and service components like security and management. The use of a business process engine may have the potential to meet project goals. As MIXED is an agile project, a continuous architecture review concerning a business process engine is recommended.

Several JBI platforms exist today. The project *Open ESB* implements an Enterprise Service Bus runtime using Java Business Integration as the foundation. Open ESB includes various components for bindings, orchestration and collaboration. The *JBoss ESB* (formerly Rosetta from Aviva Canada) provides a mature ESB based on the JBoss Enterprise Middleware Suite (JEMS) technologies.

3.1.3 Orchestration

In context of MIXED, orchestration corresponds to the business logic of MIXED. The business logic comprises implementation for following functions (see [1]):

- file detection
- convert to
- convert from

The business logic will be implemented by means of available services like file detection and available plug-ins, the actual converters. Typical ESB products include a business process engine to execute business logic implemented in a corresponding high-level language like BPEL. This mechanism is especially appropriate when business logic is composed out from available services.

A business process engine may accelerate implementation of business logic by means of BPEL which is much higher abstraction compared to implementation in an object-oriented language. In context of a research project, it seems to be justified to gain experience with BPEL within the domain of MIXED. As business execution engines come with typical ESB products, there is no additional dependency to technology. Furthermore, the decision to realize components as services (as required for the successful usage of

BPEL) does not depend on the usage of BPEL. Hence, this component can easily be replaced by using another approach. This goes conformant to Kris Klykens [4] "*... BPEL is ... not a standard we are bound to infinity. If it would seem necessary, it is still possible to use an alternative...*".

3.1.4 File Detection

This component provides detection of a given file (document) and delivers the classification including file type, version, vendor, etc. necessary to select appropriate converters available as plug-ins.

The component is realized as a self-contained service available internally within MIXED and published to externally products. This service has potential to be reused from other products like repositories uncoupled from the rest of MIXED.

3.1.5 Plug-Ins

A prerequisite of MIXED is extensibility concerning new file-formats. It must be possible to add new converters and to remove obsolete converters over time without impact on other component implementations within the MIXED system.

The MIXED architecture follows a plug-in model to implement, deploy, and execute file converters. This approach facilitates the development and management of individual converters. Plug-ins must be self-contained in the sense that adding a new plug-in is sufficient without further configuration tasks. For this, a plug-in must contain sufficient metadata about expected or resulting file format, vendor information, etc..

As plug-ins are locally or externally, the plug-in model must be aware of further metadata concerning quality properties of plug-ins that are based on external implementations. For example, an external plug-in may be available only for a certain number of tasks a day or only to registered users. Latency of an external implementation may be part of this metadata that is considered by the business logic to favor internal plug-ins over external one.

The plug-in model and the plug-in execution environment shall enable to run multiple instances of the same plug-in of different versions. This is a common requirement of plug-in based software systems which also holds in the domain of MIXED.

3.1.6 Plug-In Providers

On top of the plug-in model, two services provide access to plug-in functionality and to plug-in metadata. The separation into two services keeps service interface small and decouples interfaces for distinct purposes.

The connection and communication between service providers and plug-ins shall be based on an internal interface that avoids duplication of some common logic used for both service providers. This internal interface need not be published as service to the ESB.

3.2 Information Architecture

The information architecture is based on an intermediate format, the M-XML. In general, the usage of an intermediate format keeps number of required converters small, both for ingest and dissemination.

The definition of a new non-standard format originates a high dependency between documents stored in the M-XML format and the MIXED system. An open format, based on XML including self-describing element names can reduce this dependency even for a non-standardized format.

An important point for the definition of M-XML is that improvements and enhancements are considered from the very beginning even it is expected that the M-XML format will change less frequently.

3.3 Technical Architecture

At this stage, only a few assumptions concerning technologies to be used are considered. In this section, we briefly analyze standards and technologies intended to be used for MIXED as mentioned in the White Paper [2].

- XML and Unicode. XML is a de facto standard to store or exchange hierarchical data. Today, libraries for parsing, generating, and transforming XML documents exist for a plethora of programming languages. Furthermore, tools to specify an XML dialect in form of a *document type definition* (DTD) or *schema* and to validation XML documents against a specification are available, both for free and commercial.
- OAIS. The Reference Model for an Open Archival Information System (OAIS) is widely accepted by other resource projects.
- Java. Java is one of the most common object-oriented programming languages. Compilers and virtual machines exist for a lot of computer architectures and operating systems. Libraries and platforms for almost any technology like XML, Web Service, ESB, etc. exists for the Java programming language. Moreover, both free and commercial tools for development, code analysis, documentation and deployment of Java software exist today.
- Spring. Spring is a lightweight container, providing centralized, automated configuration and wiring of application objects. The container is capable of assembling a complex system from a set of loosely coupled components. However, Spring does not provide a mechanism to provide declarative metadata about component as used for MIXED plug-ins.
- ESB. A lot of ESB platforms exist for the Java programming language both free available like Open ESB and JBoss ESB as well as commercial products. Most products implement the Java Business Integration (JBI) standard, which reduces dependency on one certain product. JBI products provide sound platform for implementation, configuration, and management of MIXED components even if it turns out that BPEL and NMR are not usable within this project.

- SOAP. SOAP is a lightweight protocol to exchange XML-based message over a network. SOAP has the recommendation of W3C resulting in a de facto standard for communication of Web Services.
- BPEL. A lot of process engines exist for BPEL mostly part of an ESB/JBI product. Furthermore, tools exist today to develop and maintain business processes within this language.

4 References

[1] Kris Klykens: MIXED Architecture, Version 0.2, 21.6.2007.

[2] White Paper, http://mixed.dans.knaw.nl/node/114, 2007.

[3] Blue Book, Reference Model for an Open Archival Information System (OAIS), Issue 1, January 2002.

[4] Kris Klykens: Email communication from 3.7.2007.

Appendix A: Email Communication

On 26-06-2007 10:56, "Pichler Josef" <Josef.Pichler@scch.at> wrote:

> Dear Kris Klykens
>
> Thank you for the architecture document. We have already read the architecture document and other information on the project page including the white paper and the OAIS document.
>
> Nevertheless, we have some questions about both the review process and the provided information.
>
> First, are there any concrete questions that should be answered as part of this review? This would help us to investigate most interesting points.
>
> Second, some questions arose at our last review meeting.
>
> 1. What do you mean by ESB? Do you intend to reuse an ESB product or is it just a name for your own implementation (following the idea of enterprise service bus)?
>
> 2. How can BLOB data stored in M-XML? Inline using a encoding like base64 or external (outside an M-XML document)?
>
> 3. How can Image data (for instance, png format) stored in M-XML? Inline or outside the document? Btw, on page 7 you refer to an M-XML standard available as separate document. Can you provide us this document?
>
> 4. Do you intend to use an XML database system to store M-XML documents?
>
> 5. What about the converter from M-XML back to an older version? For instance, how can you convert new concepts introduced in 2050 back to the format defined in 2020?
>
> Many thanks,
> Josef Pichler

On 26-06-2007 15:47, "Klykens Kris" <Kris.Klykens@dans.knaw.nl> wrote:

> Dear Josef,
>
> Many thanks for the insights on your thoughts.
>
> Let me first try to give you a clear answer on the questions posed in your mail:
> - ESB: at this stage in the project, we are looking at the JBI-standard and products/packages supporting this standard
> - M-XML is currently in –early- development. I have started to find a consensus within DANS about specific guidelines a while ago, but not much of the standard is actually defined. More specific, as I have several tasks within MIXED, at the moment my first priority is to move further into a proposal/draft on M-XML. This also means that your question 2 and 3 cannot be answered at this stage.
> - question 4: MIXED must be seen as a conversion tool, which does not take any responsibility on maintaining or archiving the data captured. That task lies within the responsibility of a repository (which is EASY if we speak about the DANS-specific repository).
> - question 5: new concepts cannot be projected towards previous versions of the M-XML standard. This means that backward compatibility is supported, but limited in its functionalities.
>
> The main goal for MIXED is to be used by a repository, but we can already at this stage see different uses. In that regard, it might be favourable to take into account that the first initiative for collaboration has started (with DExT, a UK-project with a similar goal), where not only locally developed plugins can be used by the framework, but also a plugin developed and maintained by an external party (and vice versa, plugins developed by DANS could be used by external 'MIXED'-like frameworks).
>
> The questions which should be answered in this review are following:
> - does the current plan supports the goals of MIXED, and is it generic enough to support a long-term use (also taking into account a landscape where multiple frameworks work together). Should we consider a different approach instead of ESB. Is the concept of loose binding between the different components as laid out the best approach or should we consider tight binding for some specific components ?
> - are the options chosen in the architecture favourable or are there other design patterns or practises which would allow us to build a better platform

On one specific question I'd like to hear a second opinion: do we need a distinction between internal and external plugins. What is the best way to manage this complexity ?

Please do not hesitate to contact me again if anything would be unclear or further questions are raised.

Kind regards,
Kris L. Klykens

On 28-06-2007 09:53, "Pichler Josef" <Josef.Pichler@scch.at> wrote:

Dear Kris Klykens

thank you for your informative answers to my questions. I have some further questions about plug-ins and the responsibility of plugin service provider and plugin assembly provider.

1. Plug-Ins

In the architecture document, you write that a plug-in is the actual component to convert dataset-file to M-XML and vice versa.

Does this mean, that every plug-in must be able to transform to/from M-XML?

Maybe it should be possible to provide other plug-ins that convert between two dataset-file as in the following example.
Assume we need to convert formats RTF and DOC to M-XML and converters from RTF to ODF and DOC to ODF still exist. When it would be sufficient to provide a new plug-in that converts from ODF to M-XML and the actual transformation will be performed by means of two plug-ins

a) RTF -> ODF
b) ODF -> M-XML

This would reduce the number of converters that generate M-XML assuming that converters exists that convert actual document formats like RFT, DOC, etc. to one "standard" like ODF.

2. Plugin Service Provider

The plugin service provider is the container for plugins and provides both programming model and runtime environment to load/activate plug-ins. What is the interface published by this service to other components (lookup of available plugins etc.)?

3. Plugin Assembly Provider

What is the difference between service provider and assembly provider? Both services rely on available plug-ins so i do not unterstand the separation of this providers. I guess service provider and assembly provider are two services published to other components of the same component.

4. External Plug-In

How external plug-ins are discovered and accessed by MIXED? I mean, is it necessary that an adapter plug-in must be available within the service provider that actually forwards request to the actual external plugin?

Many thanks,
Josef Pichler

On 28-06-2007 10:26, "Klykens Kris" <Kris.Klykens@dans.knaw.nl> wrote:

Dear Josef,

Below you'll find some answers on the questions. If it would be easier to have a conference meeting to discuss specific points, please advise.

1. This leads back to a discussion we had within DANS. There are several arguments which should be taken into account for setting up a system which would work with – as we call it – atomic plugins. The main argument not to implement this in a production environment is the threat to quality: it is practically possible to test each atomic plugin for its functions, but chaining specific plugins at runtime can easily change behaviour, and can/will lead towards unexpected results. This result would be unacceptable towards the clients of DANS and the goals defined in MIXED.
Naturally, this does not exclude re-using specific parts during development, but we would like to be consistent in the principle that a plugin has been fully tested, and we can guaranty specific functionalities and behaviour, and from that, specific guidelines for users.
2. The plugin service provider provides an easy manageable access to a specific plugin towards conversion, and nothing more (see 3.). The reason for this component is that if missing, every

plugin had to be connected to the Esb, and therefore would lead to a significant overhead towards development and system management of the production environment.
3. The plugin assembly provider collects the information from the available plugins (such as the vendor-specific format they support, the availability on the system) so that orchestration can select which plugin to use. The plugin assembly provider therefore takes the responsibility about the specificiations of plugins, whereas the service provider provides only an easy access.
4. Exact. In the most convenient circumstances, this adapter is only a proxy, whereas more complex transformations enable custom actions within this adapter.

Kind regards,
Kris L. Klykens

On 03-07-2007 13:36, "Pichler Josef" <Josef.Pichler@scch.at> wrote:

Dear Kris Klykens,

I have some more (final) questions.

1. Enterprise Service Bus

An ESB provides a lot of infrastructure that will be certainly required by MIXED. For example: container for web services, SOAP bindings, security, management. However, a key concept of ESB is messaging, refered to as Normalized Message Router (NMR). We cannot see that a NMR is a key requirement for MIXED but think that component can be integrated on service level (web services).

2. BPEL Engine

A second key concept of EBS are service engines like BPEL engines. Do you think that a busines process engine is a key requirement for MIXED? How many busines processes to you expect within MIXED and how often such processes have to be adapted?

From our point of view, busines processes will be rather simple (only a few processes that are rarely adapted) within MIXED so that the usage of a busines process engine might be an overhead.

3. Plugin Container

From the schematic overview in the architecture document, two components, (a) Plugin Service Provider and (b) Plugin Assembly Provider, are based on /conntect to plugins and on the certain plugin container. This connection is internal only and not published on the ESB, right?

4. Scalability

How many plugins (internal and external) do you expect (10, 100, 1000, ...)? Is it required to activate/deactivate plugins at runtime?

Many thanks,
Josef Pichler

On 03-07-2007 17:11, "Klykens Kris" <Kris.Klykens@dans.knaw.nl> wrote:

Dear Josef,

Please find below our answers to your questions.

1. NMR is not a key requirement but can be interesting to use, specially with heavy attachments (performance)

2. in my opinion, BPEL is not a key requirement for MIXED; it does principally settle the question about the specific plugin to use for a specific task. In that regard, MIXED is an R&D project and as purposes grow and change, these function may/will expand too (I do not adept fully the ‘rarely adapted’ in your remark). I do not feel that the overhead of BPEL is that much (a junior developer on our team managed to get processes deployed in only a couple of days), however I can see the origin – and have asked myself the same question. In that regard, BPEL is – like many others – not a standard we’re bound to infinity. If it would seem necessary, it is still possible to use an alternative.

3. yes

4. short-term: about 25; middle-long term: ? (also depending on the succes of our project)

I hope these answers are clear. If not, do no hesitate to contact me again.

Kind regards,
Kris L. Klykens