Processing Chains in System of Systems

Tarmo Ploom
Integration Architecture
Credit Suisse AG
Zürich, Switzerland
tarmo.ploom@credit-suisse.com

Iain Last
Workflow Competence Center
Credit Suisse AG
Lausanne, Switzerland
iain.last@credit-suisse.com

Axel Glaser
Architecture IT Core
Systems
PostFinance AG
Bern, Switzerland
axel.glaser@postfinance.ch

Stefan Scheit
BPM & Integration Center of Excellence
Telstra Corporation
Melbourne, Australia
stefan.scheit@team.telstra.com

Abstract—SOA has been seen as one of the main approaches for managing “system of systems” (SoS) i.e. large scale IT landscapes. The extension of SOA to include the concept of process orientation could be the next evolutionary step for SOA. Before starting to define reference architectures for process-oriented SOA in SoS it would be advantageous to learn more about the current state of process handling in a SoS. The objective of the current paper is to describe the different approaches to implementing processes in a SoS. The term processing chain is introduced as an abstract representation of processes in SoS. A classification of processing chains is defined and an approximate distribution of different “processing chains” is presented in a case study based on the analyzed SoS. Also, the paper addresses the question which processing chain type would be appropriate in a certain context based on the underlying requirements.

Keywords: SOA; processing chain; SoS; workflow automation; workflow classification; human workflow; case management workflow; ad-hoc workflow; choreography; process orchestration; federated processing chain; orchestrated processing chain; stovepipe processing; enterprise information systems

I. INTRODUCTION

Current enterprise IT landscapes that form the backbone of the global economy are becoming larger and larger and moving slowly towards a system of systems (SoS).

A SoS is characterized by operational independence of its elements, managerial independence of those elements, evolutionary development, emergent behavior and geographic distribution of the elements [1].

This paper uses the IT landscape of Credit Suisse as an example of a SoS that corresponds to the above properties of a SoS. The evolutionary development and emergent properties of the Credit Suisse IT landscape have been described in [2],[3].

The main question posed about a SoS is, of course, how can these highly complex systems be managed?

Service-oriented architecture (SOA) has been proposed as the main approach for managing the complexity of a SoS [4]. The application of SOA for the management of SoS has been well described [5].

Most of the existing SoS are neither component, nor service oriented, and there is substantial work required to migrate monolithic legacy systems towards SOA based systems [6], [7]. Migration methodologies like SMART [8] have been proposed for this purpose.

But how could the manageability and flexibility of SoS be increased even further?

The focus of SOA is on service invocations and these are essentially one or two way interactions between components. Patterns for such interactions have been well described [9].

SOA does not however focus that much on cases where component interaction involves numerous service invocations and the resulting interaction patterns and types. Processing in SoS usually requires interactions of numerous SOA components and involves correspondingly more than two steps. In order to handle such processing interactions in SoS, it would be necessary to extend SOA in SoS with concepts around process orientation.

A combination of service and process orientation concepts could be the next evolutionary step in SOA based SoS. The architectural approach arising from this combination is called, in this paper, process-oriented SOA. Advantages of the process-oriented SOA would be increased flexibility, transparency, and maintainability:

- Flexibility is increased as processes can be simply composed from existing SOA services.
- Increased transparency arises as the processes are visible. Process flows become explicitly defined and are not hidden in the implementation details.
- Maintainability is increased as dependencies in the code of SoS become transparent and traceable.

Process-oriented SOA would simplify management of SoS and transform the art of managing SoS more to an engineering discipline with conscious and analytical decisions. This would also result in supporting the capability to manage even larger scale SoS.

Process-oriented SOA is a vision. Before starting to define roadmaps towards this vision of process-oriented SOA it would be advantageous to learn more about the current state of SoS.

How are processes implemented in current SoS? Which different types of processing chains exist in SoS? What different approaches exist to implementing processes in SoS? Why are there different approaches for implementing processes in existing SoS?

This paper aims at answering these questions by analyzing the underlying problems from different angles.

Firstly, we define the term processing chain as an abstraction of all process types. Then a classification of different processing chain types is proposed.
A theoretical framework is defined to explain scenarios when specific types of processing chains are more appropriate and sustainable.

Thirdly, the distribution of different types of processing chains is evaluated in our case study SoS – the Credit Suisse IT landscape.

Finally, the question of what the analysis of the current state mean for a broader SoS context is discussed. As a part of that, a research agenda for moving towards process-oriented SOA is proposed.

II. PROCESSING CHAIN CLASSIFICATION

A. Introduction

In this section the central term of this paper processing chain is defined. It is followed by a description of the main dimensions contributing to the classification approach presented to differentiate the various occurrences of such processing chains.

Serving as important input, there has been substantial research about common patterns in workflows, involving workflow control patterns, workflow exception patterns, workflow data patterns, and workflow resource patterns [10]. There have, however, only been minor attempts to define the next granular level for these “atomic” workflow patterns, i.e. what are the emerging types of workflows?

In contrast, the “atomic” interactions between services have been classified in great detail [9]. Nonetheless there has not been much discussion as to what happens when these “atomic” service interaction patterns are combined into complex service invocation flows as they usually occur as part of end-to-end workflows. Similarly, patterns for message flows have been defined [11], however the classification of sequences of message flows into different types has not been that much in focus.

Historically, workflows have been classified as production, administrative, ad hoc, and collaborative workflows [12]. This classification has its origins in the specialized workflow automation for particular application domains. For example, production workflows arose from streamlining heavily repetitive sequences of steps, whereas administrative workflows originated from office automation products like Lotus Notes or groupware which were developed to support more collaborative workflows [13].

In [14] an approach for a classification of individual workflows is presented that is decoupled from the context of the workflows as in [12]. Instead, it is focused on the internal factors of a workflow which correlate with the degree of automation as a representation of their maturity (i.e. structure and completeness) [15].

In this paper, the underlying scope is significantly extended to include a general classification of process types in the wider SoS context. This is derived from concentrating mainly on the internal factors of the processes themselves. These factors are defined in the next section.

B. Processing chain

The term “processing chain” is defined as a sequence of automated and/or manual activities (or steps) which are carried out by a system or human actor. It is used as an abstraction of commonly occurring communication flows in a SoS context in which complex processing is distributed among components (or systems).

Processing chains can be characterized according to the following criteria:

- Manual or automated execution
- Vertical (top-down invocation) versus horizontal (sequence of) communication flows
- Definition of the communication flow at design-time or runtime
- Duration of the communication flow from start to completion
- Distribution of the flow definition among components (or systems)

By viewing a SoS as a socio-technical system which involves human and system actors, all activities as part of the processing chain can be carried out by either human or system actors. Both actors are usually required even for scenarios of complete automation of all activities, because a certain part of activities – the exception flow – remains manual. It either cannot be completely automated to cover all possible outcomes or doing so is not efficient.

The vertical versus horizontal dimension characterizes the abstraction level of activities in the flow of the processing chain. A horizontal processing chain represents interlinked or sequenced activities in a flow which are on the same abstraction level, for example, a flow which orchestrates service invocations of the same granularity. In a vertical processing chain each subsequent activity is one abstraction level deeper as in a recursive function call. This could be a composite service which initially invokes atomic data services which in turn invoke further utility services (authorization, logging, etc.).

In classical SOA, vertical or horizontal processing chains would be handled as a “black box”; the flow or orchestration of SOA services is “encapsulated” by the service. In a small scale system in which all flows are encapsulated such an approach works.

However, in a SoS comprising of different SOA components (or rather SOA domains) the previously “encapsulated” service invocations along the flow of the processing chain become visible as the public component services are invoked. As a result, cross SOA component (or cross SOA domain) processing chains in a SoS cannot be handled just as “black boxes” that encapsulate the flow of processing chains but as “white boxes” that has an observable flow of SOA service invocations.

Such a “white box” approach for cross SOA component (or cross SOA domain) processing would correspond to the next level of SOA – “process-oriented SOA” – in which processing chains embedded in the SOA services become transparent and visible. The process-oriented SOA would consist of two main abstraction layers – a process layer and a SOA layer, whereby the process layer invokes services from the SOA layer.

The dimension measuring the proportion of design time to runtime flow definition distinguishes between processing
chains whose definition is determined ad-hoc during runtime or statically during design time.

The process duration dimension brings the discussion of time into the classification. There are long running processing chains like macro-flows or batch flows and short running processing chains like micro-flows.

The “distributed” dimension of the flow definition shows how the flow definition in the processing chain is distributed across components in the SoS – it can be placed centrally in one component or distributed between different components.

Processing chains in which the definition of the flow is deployed into one component are defined as “orchestration processing chains”. Processing chains in which the definition of the flow is distributed in different components are called “federation processing chains”. This classification is outlined in Fig. 1.

C. Federated processing chains

A “federated processing chain” is defined as a processing chain whose definition is distributed and there is not a central conductor which executes the processing chain during runtime.

The following dimensions should be evaluated to classify federated processing chains:

- Explicit or implicit definition of processing chain: in case of an “explicit definition” the flow definition is extracted from the participating components. An “implicit definition” describes a definition which is hidden in the components of the processing chain.
- Runtime distribution of the explicit processing chain definition: the “explicit processing chain definition” can be available for the whole processing chain (global processing chain definition) or for local parts of the processing chain.
- Runtime management of processing chain flow state: the flow state of the processing chain can be managed in a distributed way or it can be in a single central location.

Two main types of a federated processing chain can be distinguished (Fig. 2):

- Globally federated processing chain
- Locally federated processing chain

A “globally federated processing chain” describes an explicit and globally defined processing chain that is distributed across multiple components in the runtime environment. In contrast, there is not a global definition for a “locally federated processing chain”. Instead, the implicit or local explicit processing chain definitions are distributed across multiple components in the runtime environment.

Globally federated processing chains can be decomposed into the following subtypes (Fig. 3):

- Choreography processing
- Document-oriented processing

“Choreography processing” means that definition of the global processing chain is explicitly defined and deployed by components which participate in the processing chain. The state of the flow in the processing chain is however not managed centrally. Each component involved in the choreography manages the flow state of the processing chain itself. Standard languages like BPMN choreography [16], Web Service Choreography Definition Language (WS-CDL) [17] or Let’s Dance [18] are proposed for service choreographies modeling.

Under this category, two further subtypes of “choreography processing” can be distinguished:

- Service choreography
- Human choreography

“Service choreography” means that a choreography definition language is used to determine the flow of all service invocations in the processing chain. There has been substantial research about service choreographies [19]. “Human choreography” is indicated by human actors
participating in the choreography processing in addition to system actors.

In the “document oriented processing” type the global processing chain is explicitly defined, however it is neither deployed nor is its state managed by specific runtime components. Instead the “processing flow definition”, its flow state, and flow data state is placed inside a document which is propagated from one component to the next.

“Locally federated processing chains” can be split into the following subtypes (Fig. 4):

- Split choreography
- Stovepipe processing

In “split choreographies” the local parts of the choreography are explicitly defined and distributed across multiple components. Also, there is not a global definition of the choreography deployed in the runtime environment. Nevertheless, the global choreography definition may exist during design time and can be split into multiple parts for deployment. This is the reason why this subtype is regarded as split choreography and represents a subtype of the choreography type.

In “split choreographies” two more subtypes can be distinguished: “local” and “inter-company”. For the case of “local split choreographies”, the parts of the choreography definition are distributed between a company’s internal components. In “inter-company split workflow” the distribution involves components of different companies.

“Stovepipe processing” sits next to the “split choreography” by the fact that there is not an explicit definition of the processing chain. The definition is rather embedded in the source of the components participating in the processing chain. Also, the processing chain flow state and data state are hidden as well as distributed across multiple components at runtime. Stovepipe processing can be broken down into the following subtypes:

- Horizontal stovepipe processing
- Vertical stovepipe processing
- Mixed stovepipe processing

In a “horizontal stovepipe processing chain” all activities have approximately the same abstraction level. Notably, it requires some effort in order to achieve a similar abstraction level in any of the invoked component services.

Figure 4. Classification of locally federated processing chain

Horizontal stovepipe processing chains can be built in either a synchronous or asynchronous fashion. Asynchronous horizontal stovepipe processing chains correspond to the Staged Event Driven Architecture (SEDA) [20] paradigm. In such a processing chain all participating components publish their output onto a message broker topic. All interested components subscribe to this topic and retrieve the corresponding messages as input. Any participating components decide whether to process a particular message, what to do with it and where to publish its output. Once processed, they publish their output messages onto the next topic, if required. The processing logic of the components is local and there is not a global flow definition. In case the local processing logic in components is defined using a business rules engines a very high level of agility can be achieved for horizontal stovepipe processing chains.

In “vertical stovepipe processing” chains the abstraction level of the service invocations decreases. Such a style is common in the classical SOA layering [21], e.g. a SOA task layer invokes the SOA entity layer, which in turns invokes the SOA utilities layer. This chaining is hidden in the service invocation graph.

The “mixed stovepipe” type is a style in which different abstraction levels vary over the entire processing chain. It commonly emerges as a result of agile methodologies or in case of less architecturally governed approaches.

“Mixed stovepipe” can be represented as a “rules based” system, in which every component participating in the processing chain has a number of rules that determine, based on incoming message and component state, the next activity to be executed.

The mixed stovepipe processing resembles an event based programming style in which any runtime events are used for managing the processing flow of the application. In a SoS it would correspond more to an event-driven architecture where components communicate with each other using event messages over a middleware component. In such an event-driven approach there is no global or central knowledge of:

- Processing chain definition
- Processing chain flow state
- Processing chain data state

The mixed stovepipe processing chain usually emerges due to a local and isolated engineering of the components until the processing chain starts seemingly to achieve the right outcome. Such a mixed stovepipe processing style lacks an end-to-end processing perspective as all contributors only optimize locally for an optimal result.

SOA based SoS following the mixed stovepipe processing approach show a clear structure, however their processing layer or dimension is not clearly defined. Judging from the process perspective, in such situations all services work well, but their aggregation is left to chance.

D. Orchestrated processing chains

“Orchestrated processing chains” have a central conductor holding the definition of the processing flow and also the processing flow state.

“Orchestrated processing chains” can be seen as synonymous to workflow definitions i.e. the automation of a business process, in whole or part, during which documents,
information or tasks are passed from one participant (human or technical actor) to another for action, according to a set of procedural rules [22].

Two subtypes of “orchestrated processing chains” can be identified (Fig. 5):
- **Ad-hoc workflow**
- **Predefined workflow**

“Ad-hoc workflow” is dynamic in nature and its final flow definition will be determined primarily during runtime. The predefined flow’s definition is static in a sense that it is completely defined during design time and cannot be changed during runtime.

Subtypes of “ad-hoc workflows” are (Fig. 6):
- **Collaborative workflow**
- **Case management workflow**

“Collaborative workflow” takes place between human actors by using systems like email or SharePoint. In such systems work is dispatched from one human actor to the next on an ad-hoc basis and the processing chain state is stored in a central location. “Collaborative workflow” does not have a design time representation as it also primarily determined during runtime. “Collaborative workflow” corresponds to level 1 process definitions in the process maturity model [15].

Subtypes of “processing orchestration” are (Fig. 8):
- **Process orchestration macro-flow**
- **Process orchestration micro-flow**

During the execution of a “process orchestration macro-flow” the processing chain flow state and data state are persisted between different service invocations. Conversely, the processing state in a “process orchestration micro-flow” is transient and is not persisted at all.

Main subtypes of predefined workflow are (Fig. 7):
- **Human workflow**
- **Process orchestration**
- **Document management workflow**
- **Batch flow**

“Human workflow” aims at automating well-defined processes with substantial human involvement. Human workflows are defined during the design phase of the workflow. The two subtypes of human workflow which can be distinguished are a human workflow “macro-flow” and a human workflow “micro-flow.” In “human workflow macro-flows” the state of the processing chain is persisted, whereas in “micro-flows” it is transient.

“Process orchestration”, also known as straight-through processing, categorizes workflows that are fully automated production processes. They are mostly standardized with the goal of supporting high volumes and low latency processing in business transactions.

“Document management workflow” focuses on dispatching and routing of non-structured data (documents) within the processing chain. The non-structured data is created or modified outside of the processing chain. “Document management workflow” takes place mainly between human actors and involves limited system actors, e.g. for archiving, signing or verification. “Document management workflow” is distinguished from other workflows in that its immutable workflow data state, the document, is not modified during processing but remains constant. Only the workflow flow state changes during processing.
The most popular approach for implementing process orchestration is through the use of “workflow engines” or “message flow engines”. The later cannot be used for transactional micro-flows.

The subtypes of “process orchestration macro-flow” are run-book automation and business transactions.

“Run-book automation”, also called “system management orchestration” is process orchestration in the infrastructure services context.

“Business transactions” are multi-transaction (i.e. long-running transaction) service based workflows which have to provide transactional guarantees (atomicity, consistency, isolation, durability). Business transactions must keep the process and service level eventually in sync, especially in error situations. Business transactions have to provide full coverage and automation of exception handling, including compensation (backward recovery), and be able to make any guarantees about transactional safety and processing efficiency.

“Batch flow” is centrally managed by a scheduler or a batch flow controller that coordinates the execution of batch jobs (activities). Batch flow generally focuses on processing large volumes of data.

![Figure 8. Classification of process orchestration](image)

### E. Combinations of processing chain types

In real life it is common that different processing chain types can be combined to form combined processing chains for complex scenarios.

A special type of combined processing chains is end-to-end processing. These processing chains cross the entire application landscape during execution and, as such, represent a cross-section of the SoS. The also combine fragmented processing chains, or combinations thereof, together into a cohesive whole.

### F. Summary of processing chain types

A summary of orchestration processing chain types and their main features is shown in Tab. I.

#### III. THEORETICAL FRAMEWORK

In the previous section we saw that there are 17 different types of processing chains in the SoS scope based on the following criteria:

- Manual or automated execution

<table>
<thead>
<tr>
<th>Process Flow Type</th>
<th>Workflow Model</th>
<th>Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case Management</td>
<td>Semi-structured, flow partially defined during runtime and design time</td>
<td>Dynamic, information and communication intensive, individualized</td>
</tr>
<tr>
<td>Collaborative workflow</td>
<td>Unstructured flow, defined during runtime</td>
<td>Collaborative, each workflow instance is unique</td>
</tr>
<tr>
<td>Human Workflow</td>
<td>Process model structured and predefined (design-time)</td>
<td>Human involvement, content can differ</td>
</tr>
<tr>
<td>Document Management Workflow</td>
<td>Process model structured and predefined (design-time)</td>
<td>Dispatching of documents, workflow data state is not modified</td>
</tr>
<tr>
<td>Process Orchestration</td>
<td>Process model structured and predefined (design-time)</td>
<td>Fully automated, transactional</td>
</tr>
<tr>
<td>Batch flow</td>
<td>Predefined batch flow model</td>
<td>Processing of high volume data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing Chain Type</th>
<th>Workflow Model</th>
<th>Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service choreography</td>
<td>Global flow is predefined during design time</td>
<td>The same global flow definition is distributed between system participants</td>
</tr>
<tr>
<td>Human choreography</td>
<td>Global flow is predefined during design time</td>
<td>The same global flow definition is distributed between system participants</td>
</tr>
<tr>
<td>Document oriented processing</td>
<td>Global flow is predefined during design time</td>
<td>Flow, flow state and data state are transferred in message between participants</td>
</tr>
<tr>
<td>Local split choreography</td>
<td>Global flow is predefined during design time</td>
<td>Global flow is split and parts of it distributed between participants</td>
</tr>
<tr>
<td>Intercompany split choreography</td>
<td>Global flow is predefined during design time</td>
<td>Global flow is split and parts of it distributed between participating companies</td>
</tr>
<tr>
<td>Stovepipe processing</td>
<td>No process model, flow is determined during runtime</td>
<td>Flow is hidden in the invocation chain of component services</td>
</tr>
</tbody>
</table>
The different maturity levels are:

- Organizations or departments within an organization describes different levels of integration between different engineering factors.
- Architecture but also a combination of organizational and not only do engineering factors determine the resulting law resemble a socio technical perspective of SoS in which balanced.
- Communication paths within the organization are more orchestration style. In a collaborative SoS the communication pattern and as a result more emphasis on the particular system. In a directed SoS there is a more central the communication paths of the organization building the states that the underlying architecture of a system resembles a tendency is more towards the federation style of processing chains than in collaborative SoS, whereby the processing chains develop towards higher process maturity levels they process description. It should be noted that as processing moves more towards processing chain types with a defined process orchestration, document management workflow, process orchestration or document management workflow.

A directed SoS is centrally managed during its long term operation so that it continues to fulfill its purpose. Collaborative systems are distinct from directed systems in that the central management organization does not have coercive power to run the system. The individual systems must, more or less, voluntarily collaborate to fulfill the agreed upon central purposes. As virtual systems lack a central management authority, they lack a centrally agreed upon purpose for the SoS.

We make the hypothesis that in a directed SoS there is a greater tendency towards the orchestration style in processing chains than in collaborative SoS, whereby the tendency is more towards the federation style of processing chains. This is in agreement with Conway’s law which states that the underlying architecture of a system resembles the communication paths of the organization building the particular system. In a directed SoS there is a more central communication pattern and as a result more emphasis on the orchestration style. In a collaborative SoS the communication paths within the organization are more balanced.

Explanations for processing chain based on Conway’s law resemble a socio technical perspective of SoS in which not only do engineering factors determine the resulting architecture but also a combination of organizational and engineering factors.

The organizational interoperability maturity model describes different levels of integration between different organizations or departments within an organization. The different maturity levels are:

- Level 0 - Independent
- Level 1 - Ad hoc
- Level 2 - Collaborative
- Level 3 - Integrated
- Level 4 - Unified

On the independent and ad hoc levels of the organizational maturity model there is minimal interaction between different organizations or organizational units. In these cases collaborative workflow types are sufficient as there are not clearly defined processes available. Independent and ad hoc level organizations processing chains would correspond to Level 1 processes in process maturity model.

On the collaborative level of the organizational maturity model initial collaboration frameworks for interoperability between different organizations are in place. Case management workflow, which represents partially defined and undefined workflows, would be appropriate processing chains for these types of organizations.

Integrated and unified level organizations have specific mechanisms for organizational interoperability in the form of predefined processes. Such processing chains would correspond, at minimum, to level 2 in the process maturity model and can be represented as human workflow, process orchestration or document management workflow.

As interoperability maturity in the host organization increases the processes’ maturity level rises in parallel and the processing chain types evolve more from more federated towards orchestrated types.

The process maturity model describes different levels of process maturity based on internal and external characteristics:

- Level 1 - Initial
- Level 2 - Managed
- Level 3 - Standardized
- Level 4 - Predictable
- Level 5 - Optimized

Can the different process types be mapped to the process maturity model? Stovepipe processing, collaborative workflow and case management workflow would correspond to Level 1 in process maturity model as these processing chain types lack a process definition. Human workflow, process orchestration, document management workflow, choreography, split choreography and document oriented processing would correspond to level 2 or higher in process maturity model.

As the maturity of processing chains rise their type moves more towards processing chain types with a defined process description. It should be noted that as processing chains develop towards higher process maturity levels they can be federated or orchestrated.

The next factor that can significantly influence processing chain types are non-functional requirements such as:

- Volume of the data processed
- Throughput
- Agility of the processing chain
In high volume data processing, batch oriented processing would be preferred as the most efficient way of process large amounts of data. However, as batch jobs run on prescheduled timeframes the duration of participating processing chains is increased.

With the use of process orchestration workflow engines the throughput is limited by the underlying databases. When thousands of service process orchestration process chain instances per second are required to be executed the federated style processing chains, like service choreography or document oriented processing, would be more appropriate if the organization wants to achieve a centrally defined process definition. An alternative to higher throughput systems is also the “do nothing” approach and let the mixed stovepipe processing chains emerge.

If there are high processing chain flexibility requirements, then collaborative workflow and business rules based horizontal stovepipe system would be most appropriate. In these cases the processing chain change cycle time would be minimal.

In summary, the optimal processing chain in SoS is to be determined by many different factors starting from the type of host organization and ending with non-functional requirements. These factors can reduce the choice of processing chain down to few appropriate types.

IV. RESULTS

This section initially provides a short overview of the Credit Suisse IT landscape followed by the measurements of different processing chain types and instances. Only business driven processing chains are analyzed in this section. Processing chains related to the management of infrastructure and IT are disregarded. Thus run-book automation processing chains are not analyzed in this section.

The Credit Suisse IT landscape has emerged over a period of thirty to forty years and over this time programming style and paradigms have evolved. At the beginning, processing chains in Credit Suisse IT landscape were mainly of a batch or mixed stovepipe type. Over time, collaborative workflows, case management and human workflows were added. Currently there is a greater tendency towards document processing, macro-flow and micro-flow style processing chains. There are also the first attempts at inter-company split choreography. Service choreography is not being actively pursued yet.

The Credit Suisse IT SoS exhibits the characteristics of a directed system. Most of the IT landscape is centrally managed and enforced by a central authority but there are also parts of the Credit Suisse IT landscape where the central authority is relatively weak and collaboration between system components is voluntary.

Results presented in this section (Tab. III) were collected from the infrastructure in following ways:

- Number of overall processing chain types and part of collaborative workflows is based on Credit Suisse internal process repository.
- Number of process orchestration types, human workflow types and case management types were directly measured in corresponding flow execution engines.
- Number of process orchestration instances, human workflow instances and case management instances were directly measured in corresponding flow execution engines.
- Numbers of batch control flows are based on Credit Suisse batch control system.

With respect to processing chain types, the main measured indicators show that:

- There are over thirty thousand executable processing chain types at the executable level.
- There are around four hundred and eighty macro-flow process orchestration types, of which one hundred are executed in a workflow engine, three hundred and sixty in a message flow engine and the remaining twenty in a file flow engine.
- There are four more main batch control flows which coordinate ca. fifty thousand batch flow jobs.
- Choreography is not used.

Regarding process instance numbers the main measured indicators show that:

- The number of end-to-end business cases is estimated to be around forty-five million per day.
- The number of process orchestration macro-flows automated by workflow engines is ca. two hundred thousand per day.
- The number of process orchestration macro-flows automated by message brokers is ca. one hundred million per day.
- There are ca. two thousand document oriented processing chain instances per day.

<table>
<thead>
<tr>
<th>Processing Chain Type</th>
<th>Number of types</th>
<th>Number of instances per day</th>
</tr>
</thead>
<tbody>
<tr>
<td>All types</td>
<td>Ca 30,000</td>
<td>45 million</td>
</tr>
<tr>
<td>Federated processing chain</td>
<td>Ca 8,800</td>
<td>n.a.</td>
</tr>
<tr>
<td>Orchestrated processing chain</td>
<td>Ca 21,200</td>
<td>Ca 100 million</td>
</tr>
<tr>
<td>Ad-hoc workflow</td>
<td>Ca 20,500</td>
<td>n.a.</td>
</tr>
<tr>
<td>Collaborative workflow</td>
<td>Ca 20,000</td>
<td>n.a.</td>
</tr>
<tr>
<td>Case Management</td>
<td>500</td>
<td>Ca 8,000</td>
</tr>
<tr>
<td>Predefined workflow</td>
<td>764</td>
<td>100 million</td>
</tr>
<tr>
<td>Human Workflow</td>
<td>280</td>
<td>50,000</td>
</tr>
<tr>
<td>Process Orchestration</td>
<td>480</td>
<td>100 million</td>
</tr>
<tr>
<td>Batch flow</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Document oriented processing</td>
<td>6</td>
<td>2,000</td>
</tr>
</tbody>
</table>
V. DISCUSSION

From the results section it can be seen that there are ca. 10,000 automated processing chains in the case study SoS (collaborative workflows are not included as they are fully human centric). From these automated processing chains approximately 8,800 are federated and 1,200 are centrally orchestrated. This means that the overall IT landscape is heavily weighted in favor of federated processing chains.

Also, automated processing chains are implemented with a strong preference towards stovepipe based processing while the use of horizontal and vertical stovepipe processing is relatively limited. The majority of automated federated processing chains are implemented as the mixed processing chains type.

How can such a preference towards federated and mixed stovepipe based processing chains be explained in a directed SoS? It appears to be contradiction, provided that one would expect that directed SoS would mainly make use of centrally orchestrated processing chains as the directed SoS is centrally steered and governed.

The following primary reasons are proposed:

- Evolutionary perspective
- Host organization perspective
- Nonfunctional criteria

For example, in the case study SoS batch processing was initially federated and there was not a central scheduler for batch processing. However, such design did not fulfill the time requirements (all processing to be done in less than 24 hours). As a result, a central batch flow engine integrating batch processing into a cohesive flow of batch services (or activities) was introduced. Over time a federated processing chain was transformed to centrally orchestrated processing chain.

From the host organization perspective, an organizations consisting of many centers of excellence with basic collaboration would result in a corresponding SoS based on Conway’s law [25]. This means that the resulting system components would have rules only for their host center of excellence. Processed work of one center of excellence is published into other centers of excellence, e.g. via “messages” and the receiving center would evaluate whether the message is relevant to them. As a result, federated processing chains emerge.

From the non-functional requirements perspective, processing based on workflow engines has limitations around throughput. Although these limitations can be circumvented by using message flow engines that offer less transactional control but better scalability, this means that high throughput requirements for processing chains could be implemented in an orchestrated fashion by relaxing transactional constraints. Also, high throughput issues could be solved by choreography approaches which offer significantly higher scalability compared to the orchestrated approach. However, it should be noted that choreography is still a relatively new approach for enterprise IT landscapes. In the future, it will hopefully be applied to more high throughput processing chain types in the SoS.

As stated previously, the Credit Suisse IT landscape is predominantly a directed SoS and it is difficult to generalize results to all different types of SoS. Nevertheless, by observing mainly federation type processing chains in the studied SoS, it can be extrapolated that the percentage of federated processing chain types would be even higher in a virtual or collaborative SoS.

In the case study it was observed that the main drawback of federated batch flows – high duration – forced the migration of federated batch flow towards centrally orchestrated batch flows. Could there be other requirements which could reduce mixed stovepipe processing in SoS?

What are the negative side effects of mixed stovepipe processing that could eventually lead to the migration of mixed stovepipe processing towards choreography or central orchestration?

The side-effects of mixed stovepipe processing are:

- Highly manual effort in exception handling: it is common to have higher manual effort in new processing chain types. At least, if the flow definition is known then it can be optimized over time and manual efforts in processing can be reduced.
- Consistency of processing chain state: if the state the overall processing chain is distributed it may show inconsistencies. A large effort on the operations side may be required to resolve such inconsistencies.
- High change effort in engineering: as the processing chain flow is unknown it is difficult to change the flow of the processing chain. Changes in many different system components may be necessary, but it is not known in advance which system components are involved.

It is likely that additional costs are incurred in case initial implicitly defined mixed stovepipe style processing chains should be transformed towards more explicitly defined choreographies or orchestration at some stage.

Stovepipe processing is negative as such. For example, if there is requirement for very high agility in processing chains then the horizontal stovepipe processing chain using a business rules engine in system components would be most appropriate.

In the results section it was also shown that there is a large amount of fragmentation of existing processing chains or missing end-to-end processing chains. This is best illustrated by the fact that while ca. forty-five million business cases are processed per day, the number of identified processing chain instances is over one-hundred million per day. This means that local processing chains are not integrated into global, end-to-end, processing chains.

Side effects of such a local processing chain approach opposed to a global end-to-end processing are:

- Only local optimization of processing chains is feasible instead of a global optimization.
- The boundary controls in the SoS require special attention. It may not be known which incoming requests into a SoS result in outgoing responses and
which outgoing response was initiating by an incoming request [27].

- Auditing of end-to-end processing becomes a challenge and massive log-file consolidation systems have to be built to get some audit information about end-to-end processing.

In summary, in the studied, directed SoS, a strong tendency towards federated processing chains was observed and there is almost no evidence of explicitly defined end-to-end processing chains.

A. Further Work

The current paper is just an initial analysis of processing chains in SoS. This paper raises many questions:

- What does process-oriented SOA mean in SoS considering different types of processing chains? For which processing chains types is the concept of process-oriented SOA applicable?
- How can mixed stovepipe processing type processing chains be transformed into explicitly described processing chains (choreography or orchestration) in SoS?
- How can the transformation towards explicitly defined end-to-end processing chains based process-oriented SOA be achieved?
- What are the reference architectures for different types of processing chains in SoS?

The intention is to address this in future publications.

ACKNOWLEDGMENT

We would like to thank Claus Hagen, Markus Kradolfer for reviewing this paper and providing valuable feedback.

Most of all we would like to express our appreciation for the continuous support of our Ex-Chief Architect Stephan Murer and Chief Architect Stephan Hug in driving forward process-oriented SOA concept.

REFERENCES