D5.2.1 Process Forecasting and Simulation (Prototype I)

Authors: UVI

Delivery Date:	Due Date:	Dissemination Level:
2013-03-15	2013-02-28	PP

This deliverable provides a description of the first prototype implementation of task T5.2 Process Forecasting and Simulation. As stated in the Description of Work (DOW), this deliverable is a prototype (software) deliverable. As such, this document is reduced in length and its only purpose is to briefly describe the prototype functionality as well as to give installation instructions and usage clarifications. This document will be shipped together with the software itself.

Document History	
Draft Version	V0.1, UVI, 13 March, 2013 V0.3, UVI, 15 March, 2013 V1.0, INESC, 15th March, 2013
Contributions	UVI - Juergen Mangler - Tobias Hildebrandt
Internal Review 1	Filipe Ferreira, INESC
Internal Review 2	Dieter Schuller, TUDA
Final Version	March 15th, 2013

Table of Contents

List of Figures

Executive Summary

Forecasting & Simulation of the process comprises the execution of the processes in forecast mode, monitoring the forecast / simulated results and displaying them to the user. For this, it takes information from different sources into account and integrates it. It allows a step-by-step execution of the manufacturing process. After the execution of each manufacturing step, the Adaptive Process Execution component will decide about the execution of the next step based on (recognised) unexpected events such as delays in the manufacturing processes, strikes, delays in the transportation, etc. and on monitored results from the monitoring component.

Prototype D5.2.1 represents the main outcome of task T5.2 Process Forecast and Simulation of the first 18 months. It delivers the first limited functionality implementation of the software components that will allow ADVENTURE to simulate and forecast processes.

As stated in the Description of Work (DOW), this deliverable is of a prototype (software) nature. Hence, the purpose of this document is to briefly describe and visualise the prototype functionality, as well as to give installation instructions and usage guidance.

1 Introduction

ADVENTURE – ADaptive Virtual ENterprise manufacTURing Environment – is a project funded in the Seventh Framework Programme by the European Commission. ADVENTURE creates a framework that enhances the collaboration between suppliers, manufacturers and customers for industrial products and services.

1.1 ADVENTURE Project Aims

The framework proposed by ADVENTURE provides mechanisms and tools that facilitate the creation and operation of manufacturing processes in a modular way. ADVENTURE combines the power of individual factories to achieve complex manufacturing processes. It provides tools for partner-finding, process creation, process optimization, information exchange as well as real-time monitoring combined with the tracking of goods and linking them to Cloud services.

There have already been several research projects that address the combination of different independent manufacturers to so-called virtual factories. Most of these research projects focus primarily on the business-side in general and on aspects like partner-finding and factory-building processes in special. However no proven tools or technologies exist in the market that provide the creation of virtual factories applying end-to-end integrated Information and Communication Technology (ICT). ADVENTURE is aiming to provide such tools and processes that will help to facilitate information exchange between factories and move beyond the boundaries of the individual enterprises involved. The collaborative manufacturing process will be optimised by enabling the integration of factory selection, forecasting, monitoring, and collaboration during runtime.

ADVENTURE builds on concepts and methods of Service-oriented Computing and benefits from the advancements in this field. The monitoring and governance of the collaborative processes will be supported by technologies from the Internet of Things such as wireless sensors. Existing tools and services that can be integrated will be considered during the development of the platform for ADVENTURE.

The increased degree of flexibility provided through ADVENTURE will benefit SMEs especially as it helps them to react quickly to changes and to participate in larger, cross-organizational manufacturing processes. Furthermore, ADVENTURE will help manufacturers in assessing the environmental friendliness of actual manufacturing processes and resulting products and services. Other objectives of ADVENTURE include research in areas such as service-based manufacturing processes, adaptive process management, process compliance, and end-to-end-integration of ICT solutions.

1.2 Deliverable Purpose, Scope and Context

The purpose of this deliverable is to accompany the prototype implementation of task 5.2 Process Forecasting and Simulation. As such, its main purpose is to briefly clarify

the scope of the prototype and to present the download and installation instructions of the software. As the main focus of the task is the development of the software itself, this accompanying document targets more on providing a short summary of the main functionalities and serving as user guide for the current status of the development.

1.3 Document Status

The first prototypes of ADVENTURE software components have been developed in parallel to technical specification. This was planned in order to use these first prototypes to verify and to validate the requirements and the technical choices made, rather than to provide a complete functionality solution. Hence, the first prototype provides a limited set of basic features (even mock-ups), but no end-to-end complete functionality. The interactions between the different components are also not implemented yet, because as already noted the main objective of this prototype is to learn and adjust the interaction and calling points between the components, as well as provide valuable inputs to the technical specification about the state of the art, technical alternatives, test of tools, etc. This first prototype will evolve over the time, taking into consideration aspects like common graphical design for ADVENTURE user interfaces, embedding the tools into the Dashboard framework, interactions between the implemented software components, etc.

1.4 Target Audience

This document is listed in the DOW as PP, which means 'Restricted to other programme participants (including the Commission Services)' primarily since the audience of the document is largely internal and it is not a final version. It presents the first prototype of the software components of Process Forecasting and Simulation that is mainly targeted to the development team and to the user partners, rather than to external audience. The next version of this deliverable planned for Year 2 of the project with the complete functionality implemented will be made publicly available for external dissemination.

1.5 Abbreviations and General Terms

A definition of general, common terms and roles related to the realization of ADVENTURE as well as a list of abbreviations is available in the supplementary document "Supplement: Abbreviations and General Terms" which is provided in addition to this deliverable.
Further information can be found at: http://www.fp7-adventure.eu

1.6 Document Structure

This deliverable is broken down into the following sections:

- **Section 1**: Provides an overview of the entire document and the related pilot implementation, describing the objectives, constraints and status.
- **Section 2:** Describes the scope of the pilot implementation, its purpose and its functionality.
- **Section 3:** Introduces the information needed to deal with the pilot, in terms of technical and non-technical requirements, software to be installed, etc.
- **Section 4:** Describes the needed steps to locally install the pilot software, and how to build it from the source code.
- **Section 5**: Presents how to execute and use the prototype.
- **Section 6:** Depicts the current pilot limitations, and the expected improvements.
- **Section 7:** Describes the conclusions of the implementation of the first prototype.

2 Scope and Relationship

The Forecasting & Simulation (FS) component will be an independent service that relies on the Process Execution component and uses gateways (see Figure 1) to get temporal and execution information from partners without e.g. ordering something or invoking partner processes. Simulation data is always connected to particular versions of a process model and:

- Is intended to help ADVENTURE Brokers to decide for / against particular partners during design phase.
- Will serve as input for process optimization.

The idea of the FS component is that while an ADVENTURE Broker is using the Process Designer, he needs detailed information about how a certain partner / combination of partners may behave.

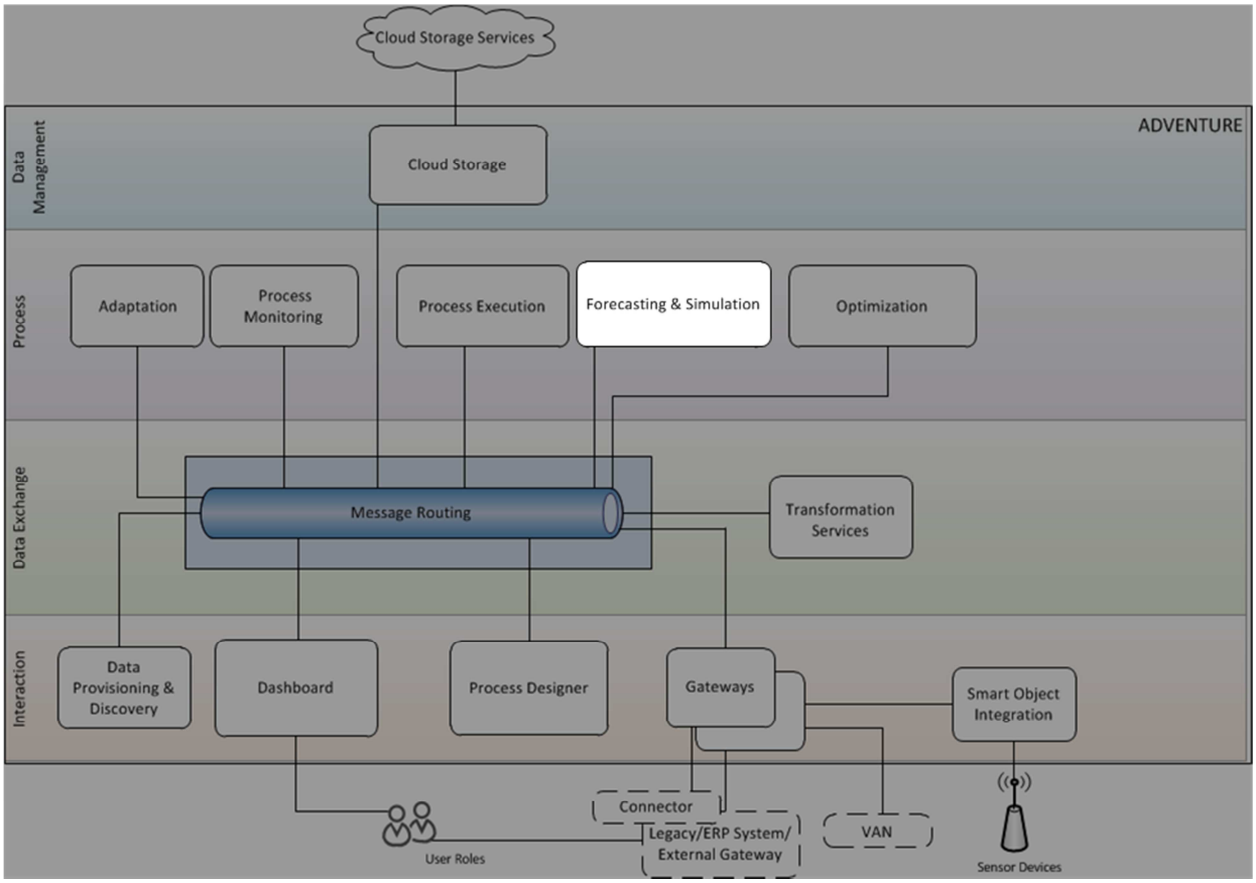

Figure 1: Overall architecture

The idea of the first prototype is to create the foundation for the simulation / adaptation workflow. Picture the following scenario:

- The ADVENTURE Broker creates a process model in the process designer (PD) component.
- The ADVENTURE Broker presses the 'SIMULATE' button in the process designer.
- The ADVENTURE Broker receives feedback about the duration and properties of certain activities contained in his process model.

The simulation itself consists of three parts:

1. Analyzing the process model to find all possible execution paths (typically called execution traces). Traces are strictly sequences. Because a process model can contain loops, decisions and parallel branches, not one trace is generated, but typically a list of traces.
2. For each trace (sequence) of steps, each step may have multiple partners assigned. Thus for each trace, and each occurring combination of partners, the simulation has to invoke the business intelligence logic of the partners through the forecasting abstraction provided by the gateway.
3. The collected data has to be: (a) saved into the cloud storage for consumption by the optimization component, and (b) compiled into a human readable report.

For the Prototype 5.2.1 we concentrated on 1. We only generate the traces, calling the actual forecast functionality of the partners through the gateways, and compiling the reports is realized in subsequent prototypes.

Traces are realized in three steps:

- The process model is executed through the process engine in simulation mode. The simulation mode, unlike execution, runs through all parts of the process once (i.e. loops, and all branches of decisions), and emits events while doing so.
- An independent simulation event monitor (which only listens to simulation events) collects all the events, and compiles a tailored[1] process structure tree.
- The process structure tree is then analysed to derive the traces.

[1] It's tailored, because we decided to support Multiple Instance Patterns (e.g. Dynamic Partial Join for Multiple Instances) supported through special constructs in the CPEE

2.1 Simulation Mode

In order to create a process structure tree[2] with the nodes described in the type attribute below, we require the process engine to implement a simulation mode, and while running in the simulation mode, emitting events of the following structure:

type	activity, choose, alternative, otherwise, parallel, parallel_branch, mutex, loop, as defined in [3]
nesting	none (for activity), :start or :end (for all others)
tid	trace id (unique element for every element)
parent	for all nested elements the tid of its parent
parameters	additional information; for loops the mode (:head or :foot), for choose the mode (:inclusive or :exclusive)

It is important to note that parallel_branches are special. Parallel might have loops and all kinds of other elements inside, that are always executed first – it is logic to decide how many parallel branches are to be opened. Thus parallel branches may have parent elements that are not the parent, but can be logically considered to be children of parallel for execution. The parent information delivered for the simulation can be taken to decide if branches are created in loops or decided upon in decisions (choose).

Similar to parallel, there can also be other logic inside choose elements, that is not 'alternative' or 'otherwise' in order to create a dynamic number of decision branches.

In order to run a process in simulation mode the process model needs to have the following properties (need either to be modified, or the engine has to support a special execution mode):

- Every loop is executed once: loop decision has to fire once.
- Every exclusive decision has to be converted to an inclusive decision: every branch is executed once.

2.2 Trace Generation for examples:

In the following section, a few exemplary BPMN elements that contain typical process elements will be presented together with the traces and their probabilities.

Traces always have the following form:

[2] A. Polyvyanyy, J. Vanhatalo, and H. Völzer, “Simplified computation and generalization of the refined process structure tree,” *Web Services and Formal Methods*, pp. 25–41, 2011.

[3] J. Mangler, G. Stuermer, and E. Schikuta, “Cloud Process Execution Engine-Evaluation of the Core Concepts,” *Arxiv preprint arXiv:1003.3330*, 2010.

- *Element, Element, Element (Probability) [COMMENT]*

Detailed information concerning how the probabilities are calculated can be found in the next sub chapter.

Sequence:

This model (Figure 2) contains two activities (“A” and “B”) that are executed sequentially.

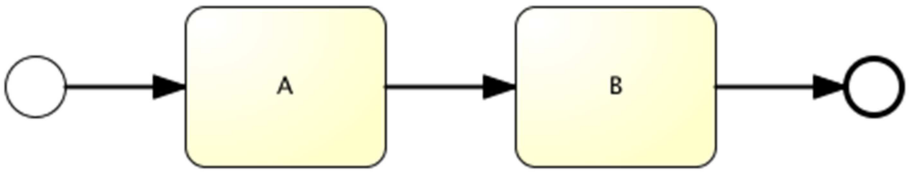

Figure 2: Sequence

Possible traces and probabilities:

- A,B (1)

Arbitrary cycles:

This model (Figure 3) generates a trace that in all cases starts with activity “A”, followed by 1 to n frequencies of “B”, and that ends with “C”.

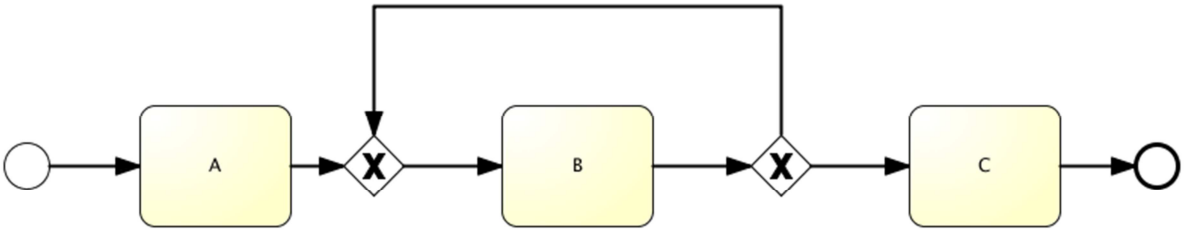

Figure 3: Arbitrary Cycles

Possible traces and probabilities:

- A,B,C (1*0.66*1)
- A,B,B,C (1*0.83*1)
- A,B,B,B,C (1*1*1)
- A,B,B,B,B,C (1*0.83*1)
- A,B,B,B,B,B,C (1*0.66*1)

Note that “A” and “C” are scored as sequence activities, whereas the loop activity “B” is scored according to the loop algorithm described below.

Exclusive Choice:

This model (Figure 4) generates a trace that only consists of one activity, either “A”, “B” or “C” (while “C” is the default path).

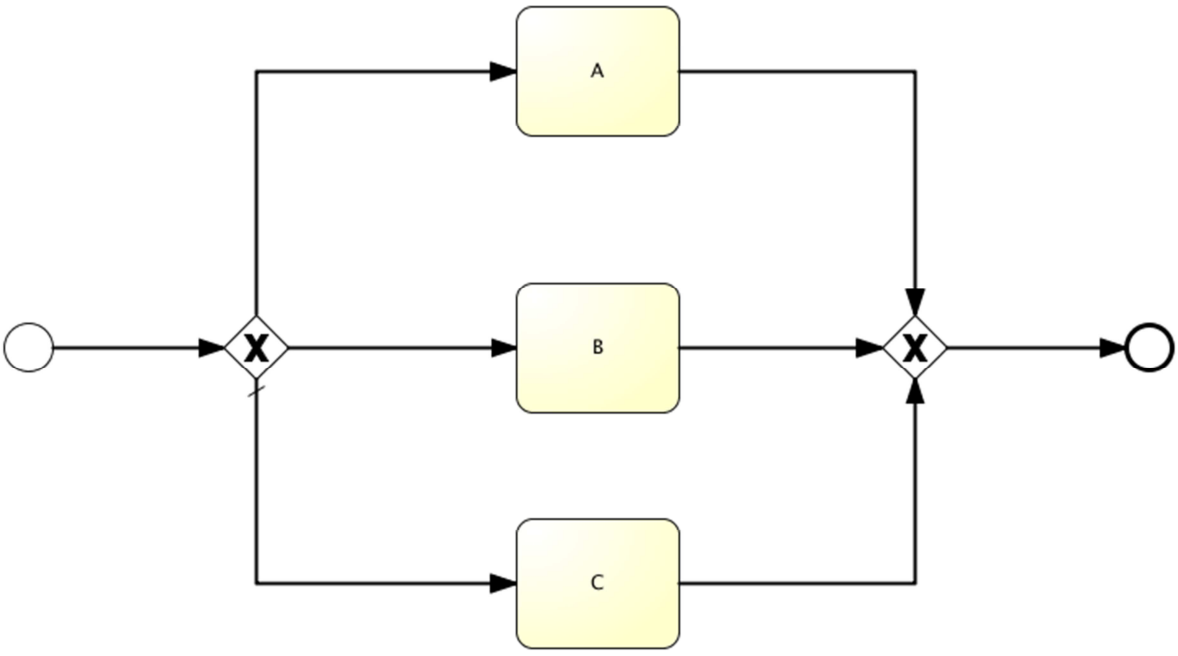

Figure 4: Exclusive Choice

Possible traces and probabilities:

- A (0.33)
- B (0.33)
- C (0.33)

Parallel branches:

This model (Figure 5) consists of three branches that contain two or three sequential activities (“A”, “B” and “C”), (“D” and “E”) and (“F” and “G”). All three branches are being executed in parallel.

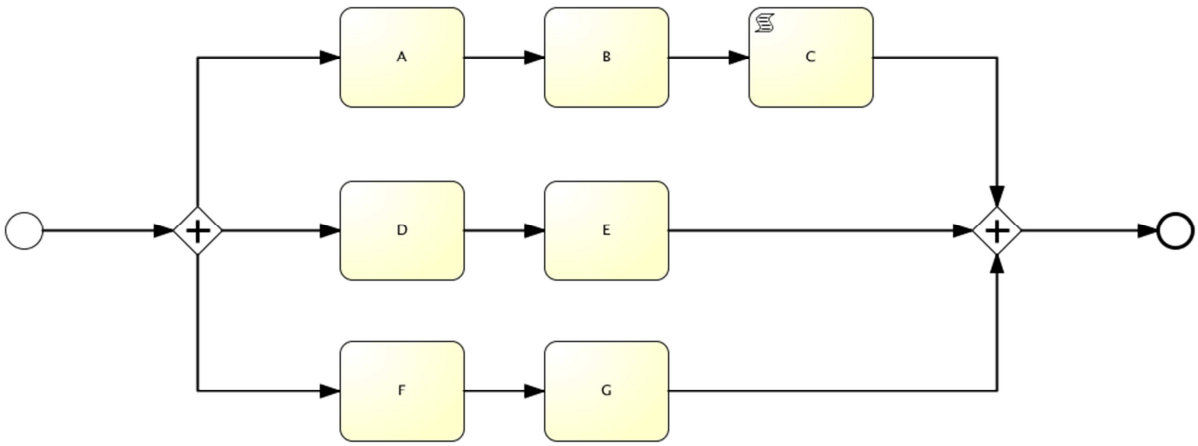

Figure 5: Parallel branches

Possible traces and probabilities:

- A,B,D,F,E,G (0.01) [2 changes from default trace]
- A,B,D,F,G,E (0.01) [2 changes from default trace]
- A,D,B,F,E,G (0.5) [1 change from default trace]
- A,D,B,F,G,E (0.5) [1 change from default trace]
- A,D,F,E,B,G (1.0) [standard case]
- A,D,F,B,E,G (1.0) [standard case]
- A,D,F,B,G,E (1.0) [standard case]

  ….

Please note that for this case “C” is not considered, as it is a Script Activity. It cannot be forecasted, as its local logic is only executed in the Smart Process Engine (SPE) itself. Please also note that for this case 36 default traces (out of 90 existing traces) exist. The concept of default traces will also be explained in the next sub chapter. For this example, the maximum number of commutations from any trace to a default trace is 2.

Inclusive choice:

In this model (Figure 6), either “A”, “B”, “C” or two or all of the three activities are being executed in parallel, while “C” is the default path.

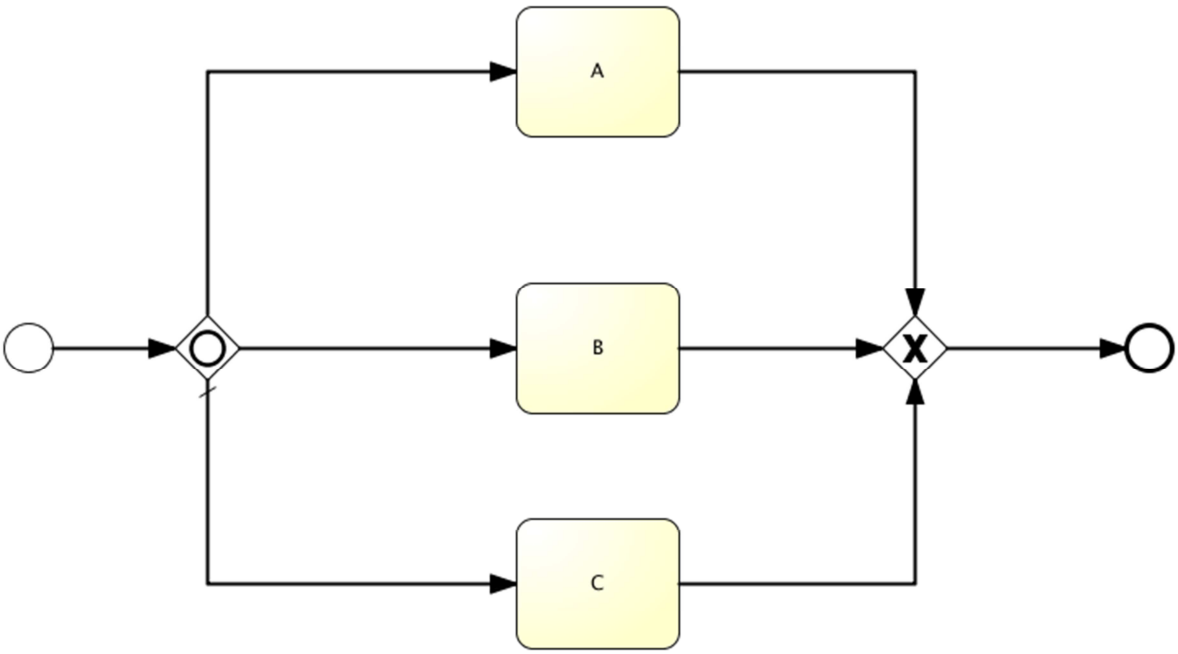

Figure 6: Inclusive choice

Possible traces and probabilities:

- A (0.5 + 0.25 = 0.75)
- B (0.5 + 0.25 = 0.75)
- C (0.5 + 0.25 = 0.75)
- A,B (1) [median case]

  …
- A,B,C (0.25 + 0.25 + 0.25 = 0.75)

  …

The median case here is 2 branches, thus it has the probability 1. All other cases have lower probability.

Dynamic loop with choice

The following model (Figure 7) describes a process in which an arbitrary amount of signals (x) are being triggered in the loop on the upper branch, which In turn lead to the creation of instances of the activities “M” and “N” on the lower branch. Technically, the diagram is not a valid BPMN document, as event-based gateways must be followed by either catching time- or signal-intermediate events or activities. Figure 8 in comparison shows a version that bases on the same principle and that describes a suggestion of how to implement workflow pattern 36 (Dynamic partial join for multiple instances)[4] in SAP NetWeaver BPM 7.11[5].

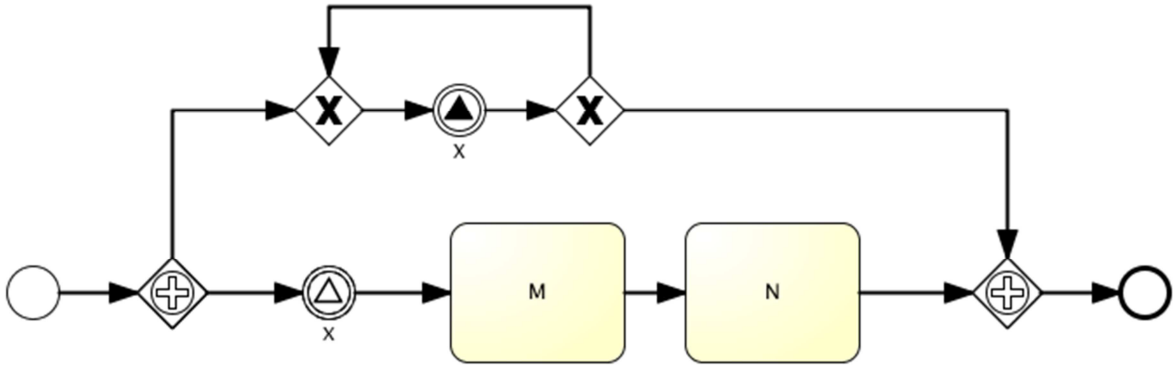

Figure 7: Dynamic loop with choice

[4] http://www.workflowpatterns.com/patterns/control/new/wcp36.php
[5] http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/8026f177-f32f-2c10-b7b0-9cc31d92984d?overridelayout=true

Possible traces and probabilities:

- M,N (0.66*1=0.66)
- M,N,M,N (0.83*0.5=0.41) [1 change to default trace]
- M,M,N,N (0.83*1.0=0.83) [default trace for 2 iterations]
- M,N,M,N,M,N (1.0*0.25=0.25) [2 changes to default trace]
- M,M,N,N,M,N (1.0*0.25=0.25) [2 changes to default trace]
- M,M,N,M,N,N (1.0*0.625=0.625) [1 change to default trace]
- M,M,M,N,N,N (1.0*1.0=1.0) [default trace for 3 iterations]
- …

This case is very special in several aspects:

- The loop allows for 1 up to 5 parallel executions of M,N
- Each case (1,2,3,4,5) of parallel executions is scored with a probability according to the schema presented above.
- After the individual scores of the parallel execution cases (1,2,3,4,5 parallel executions), each resulting trace is weighed by the loop schema presented above.
- Note that because only “M” and “N” are executed in parallel, the number of possible changes to go from an arbitrary case to the default trace is lower.

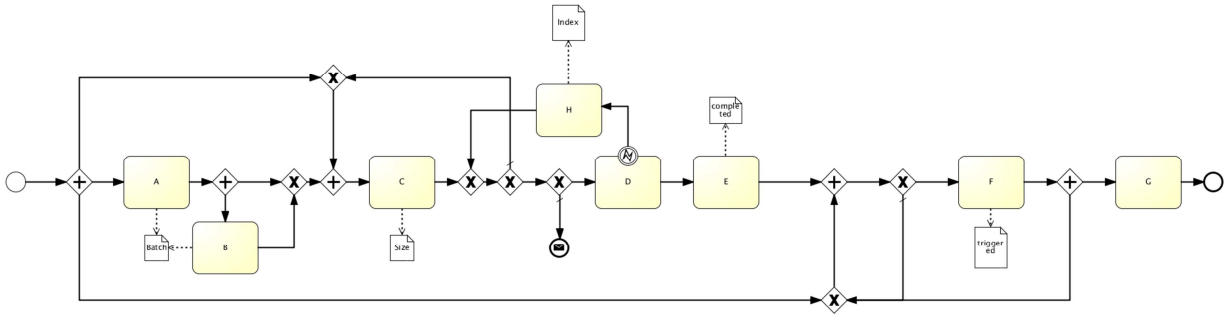

Figure 8: Dynamic Partial Join for Multiple Instances as Elaborated by SAP

2.3 Probability that Traces Might Occur

For a given process model a very long list of possible traces might be possible. For example for three parallel branches, each with 3 elements that are executed in order:

Branch 1: {a, b, c}
Branch 2: {d, e, f}
Branch 3: {g, h, i}

yields possible 1680 traces, according to the formula

$$\text{number of traces} = \frac{(total\ number\ of\ elements)!}{for\ all\ branches\ i: \prod(number\ of\ elements\ in\ i)!}$$

Concrete:

$$1680 = \frac{7!}{3! * 3! * 3!}$$

Not every trace in this huge number of traces is equally probable to occur in an execution. Thus we define a default case with the probability of 1: {a,d,g,b,e,h,c,f,i}. In this default case we assume that every activity has the same duration.

We then measure the deviation from this default trace, i.e. the number of swaps necessary to transform a trace into the default trace. In order to do so we utilize the Damerau-Levenshtein distance[6,7] algorithm (typically used for word similarities and RNA (Ribonucleid acid) permutation checking), to calculate the possibility that a trace might occur for transmutations of parallel branches.

For sequences, loops and decisions, probabilities are calculated as follows:

- Each activity of sequence has itself the probability 1
- Without additional information we assume that loops are executed max. 5 times
  - We assume that 3 times execution has the probability of 1
  - We assume that 2 or 4 times execution has the probability of 0.83
  - We assume that 1 or 5 times execution has the probability of 0.66
  - We assume that 0 times execution has the probability of 0.5.

[6] http://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance

[7] F. J. Damerau, “A technique for computer detection and correction of spelling errors,” *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

    - **THUS:** the probability of all activities in a loop is modified by this probability. As other constructs that yielded multiple traces with lower probabilities than 1 might have occurred in the loop, different traces might have different probabilities.
- Without additional information, we assume that for exclusive decision, every branch (including the default branch) is equally probable: 1 / #of branches. E.g. for 4 branches a probability of 0.25.
- Without additional information, we assume inclusive decisions follow the same principle as loops:
    - #branches / 2 out of #branches active: probability 1
    - Each branch has the assigned probability of 1 / (#branches + 1)
    - If smaller #branches / 2 are executed the probability is 0.5 + sum of probabilities per branch
    - If larger #branches / 2 are executed the probability is sum of probabilities per branch

As already shown above (e.g. in the loop example that is in sequence with an activity before and after the loop), each activity in a trace has its own assigned probability.

- For the probability of a trace, probabilities of sequential activities are multiplied.
- If e.g. a loop is inside a parallel branch, then traces the result in a branch a weighted by the probabilities of the trace.

In order to create better simulation (and optimization) results which not rely on artificial rules to create probabilities, process models need additional information:

1. The probability of execution for each branch of a decision.
2. The average number of executions for loops.
3. Average duration of each task, in order to derive the default traces in parallel branches.

This information might come from two sources:

- Through annotation by the ADVENTURE Broker when the model is created, for points 1 and 2 (above).
- Through a simulation that yields durations (for partners) for point 3 (above). Depending on the optimization results (selection of particular partners), precise probabilities can then be assigned to branches. In the scenario of dynamic partner selection (from a list a runtime) the average duration of an activity can also be calculated on the basis of the duration forecast results for all partners.

2.4 Rationale & Advantages of the Traces Approach

ADVENTURE employs a translation of process model to execution model. This translation has the following purposes:

- BPMN 2.x has no well-defined execution semantics. Translating to a language with a well-defined execution semantic allows for streamlined testing, reduction of elements and thus leaner execution.
- ADVENTURE adds custom elements to BPMN (i.e. related to sensor integration), which define their own dependencies to the process model.
- ADVENTURE tries to ensure that arbitrary execution engines (even ones that don't support a particular execution semantic of BPMN 2.x) can be used.

For the above three reasons a translation is a logical conclusion. Yet the decision to not rely on the execution semantics of a single engine forces the way we approach Simulation:

- Different engines might have slightly different execution semantics, depending on the translation.
- Simulation is a tool to show the user how the execution should work.
- THUS, the traces should be derived from the execution (engine), not be realized as another translation from the process model.

In order to ensure the consistency of the ADVENTURE platform, we thus require the process engine to actually be able to:

- Dry-run a process model (without invoking the services). Dry-running includes running through every (reachable) code-path (ignoring decisions and loop criteria).
- Emit specific simulation signals during a dry-run, to let an external component build an execution related process model – for this prototype a structured process tree.

The traces concept itself has multiple advantages:

- Makes Simulation and optimization very easy; Simulation is as simple as a sequence of calls, performed multiple times per trace, as different partner combinations have to be tested. No messing around with control structures.
- Allows for simple reasoning should performance improvements be necessary: do not run simulation for traces with lower probability.
- Allows for simple optimization when running simulations / optimizations for changed process models: the original and the changed process model might share a set of traces that has NOT changed, thus not a full new simulation run is necessary.

3 Requirements & Preparations

3.1 For Users

Users will not directly use the prototype, as it is an internal component only used by other ADVENTURE components.

3.2 For Developers

This version of the prototype is exposed as a suite of test tools, that in junction with the Smart Process Engine (process-execution@fp7-adventure.eu) as an XMPP services allows for the generation of traces for the above specified test scenarios. As the SPE is not yet connected to the Process Designer (PD) component, the sample scenarios have been specified in the internal execution representation of the SPE.

4 Installation (Deployment)

For the ADVENTURE consortium a hosted version of the prototype is readily available by contacting the Message Routing Component (see prototype D4.4.1): process-execution@fp7-adventure.eu.

The following installation steps are only necessary for local installation (explained on the basis of an up-to-date Linux/Ubuntu) on an own server or client.

- Installation of an infrastructure for compiling C code:
  - `sudo apt-get install build essential`
- Installation of the ruby virtual machine, package management and development libraries
  - `sudo apt-get install ruby ruby-dev rubygems`
  - `echo 'export PATH=$PATH:/var/lib/gems/1.8/bin' | tee --append ~/.bashrc`
  - `echo 'export PATH=$PATH:/var/lib/gems/1.9/bin' | tee --append ~/.bashrc`
  - `export PATH=$PATH:/var/lib/gems/1.8/bin`
  - `export PATH=$PATH:/var/lib/gems/1.9/bin`
- Add ADVENTURE installation package repository:
  - `sudo gem sources -a http://fp7-adventure.eu/installation/spe`
- Installation of the ADVENTURE CPEE Handler (has CPEE as a dependency)
  - `sudo gem install adventure-cpee`
- Installation of the correlation handler:
  - `sudo gem install adventure-simulation`

After installation the components have to be started with:

- `cpee start`
- `adventure-simulation start`

After starting a series of tests is executed automatically resulting in lists of traces and probabilities.

5 Execution & Usage

The usage and interfaces of the Simulation have been described in D3.3. In order to run the test suite (including generic and ADVENTURE specific tests), the following command has to be issued:

- cpee adventure-simulation

This command will yield the following output:

```
Testset:
  Branch: [:m, :n]
  Branch: [:m, :n]
  Branch: [:m, :n]
Resulting Traces:
[:m, :n, :m, :n, :m, :n] 2 0.25
[:m, :m, :n, :n, :m, :n] 2 0.25
[:m, :m, :n, :m, :n, :n] 1 0.625
[:m, :m, :m, :n, :n, :n] 0 1.0
Number of default traces: 1
Total number of traces 4
```

The full suite of tests (which contains lots of code samples) can be inspected under:

- `/var/lib/gems/x.x/gems/adventure-simulation-x.x.x/tests`

6 Limitations & Further developments

Currently we realized the generation of traces for simple case (patterns 1-9 in http://www.workflowpatterns.com/patterns/control/). Yet the current SPE (based on the CPEE as described in D3.3) supports many more patterns, which are (1) not directly supported by BPMN 2.0 or (2) have no execution semantics defined in BPMN. We currently work on supporting more esoteric patterns (like the “Dynamic Partial Join for Multiple Instances” shown above), to ensure the future proofness of ADVENTURE. This work is on-going, and includes nested combinations of patterns.

Future prototypes will include:

- The simple to realize simulation (interaction through gateways).
- Refined mechanisms to assign probabilities to traces.

7 Summary

The first Forecasting and Simulation Prototype offers the possibility to generate sets of possible execution traces for a variety of different process models, together with probabilities that define how probable the execution of a specific trace is. The prototype D5.2.1 represents the main outcome of task T5.2 Process Forecasting and Simulation.