Towards a Common Platform for the Support of Routine and Agile Business Processes

Michael Zeising, Stefan Schönig, Stefan Jablonski
Chair for Applied Computer Science IV
University of Bayreuth
Bayreuth, Germany
{michael.zeising, stefan.schoenig, stefan.jablonski}@uni-bayreuth.de

***Abstract*—The spectrum of an organization's business processes ranges from routine processes with a well-defined flow to agile processes with a degree of uncertainty. The Process Navigation platform aims at supporting both types of processes as well as combinations of them. It offers execution support for traditional flow-oriented notations like BPMN as they are well-suited for the routine type of processes. Rule-based notations for agile processes like CMMN are on the way of getting established but still have a number of weaknesses. As a consequence, the platform's agile part does not target one single notation but relies on a rule-based cross-perspective and modal intermediate language. CMMN models are then translated to the intermediate language for execution. The contribution of this paper is built up in three parts: first of all, the overall architecture of the execution platform is explained. In a second step, the intermediate language is evaluated on the basis of a comprehensive and acknowledged framework of business process requirements. And finally, the translation of CMMN to the intermediate language is described by means of an example.**

Keywords-business processes, routine processes, agile processes, procedural modeling, rule-based modeling, cross-perspective modeling, modalities, Workflow Patterns, Process Navigation, CMMN

I. INTRODUCTION

The success of an organization primarily depends upon its ability to accomplish its tasks in a structured and reliable manner. A well accepted method for structuring an organization is business process management. It is usually motivated by the drive towards the implementation of regulatory measures like the Sarbanes-Oxley Act, the implementation of quality management or a general increase of efficiency. [1]

As already recognized about 20 years ago, at least two different types of business processes can be distinguished [2]:

- well-structured *routine processes* of which the exact flow is in focus and is known a priori and
- *agile processes* of which the exact flow cannot be determined completely a priori.

Traditional notations for business process modelling like flow charts [3], EPCs [4] and BPMN [5] rely on an explicit encoding of sequential, alternative and parallel paths. As a result, every possible flow must be known at design time. In the context of programming languages this paradigm is known as procedural (or imperative) programming. The term has been applied to process modelling so that models with an explicit encoding of flow are called *procedural business process models*. Procedural models are well-suited for routine processes [6].

During the mid-1990s, it has been recognized that there are processes of which the exact flow of activities cannot be determined at design time [2]. These processes heavily depend on the human participants, their decisions and their expert knowledge. These information cannot be identified and formalized in a whole. As a result, these processes require highly flexible IT support. [7] In brief, a more flexible business process and IT support means a greater number of alternative paths [8]. However, to make a procedural process more flexible, all additional and alternative paths have to be added explicitly [8]. This results in a tendency to overspecify the process [9]. The requirement, e.g., that, within a process, two activities should never be performed both cannot be expressed directly. Instead, the modeler is forced to provide a detailed strategy that implements this requirement [10].

An alternative to procedural is *rule-based business process modelling*. It prevents overspecification by a paradigm shift. Within a rule-based model initially all paths are considered viable. The more constraints are added to the model the less paths remain. As result, to make a rule-based model more flexible constraints have to be removed or weakened [8]. Moreover, a rule-based model focuses on crosscutting relations instead of the flow of activities [6]. Hence, a rule-based approach is well-suited for modelling agile processes [7].

As mentioned above, rule-based modelling provides means for increasing the number of alternative paths without explicitly modelling them. Given the vast amount of alternatives, a differentiation between mandatory, recommended and forbidden actions in the form of *modalities* is reasonable [11]. They are necessary to reflect, e.g., a legal framework on the one hand and good practice on the other hand. An IT system that supports agile processes is not only a means of automation but a decision support system. A central characteristic of such a system is that it *explains and qualifies* its output [12]. In the context of business process execution, this means that some of the proposed actions may be marked as recommended and that this qualification can be tracked back to the process model and/or other contextual information like, e.g., the organizational structure [13].

Regardless of the modelling paradigm, business processes can be described using five fundamental *perspectives*:

This research was conducted as part of the project *Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement* (KpPQ) which is funded by the *European Regional Development Fund* (ERDF).

COLLABORATECOM 2014, October 22-25, Miami, United States

DOI 10.4108/icst.collaboratecom.2014.257269

- The *functional* perspective describes the functional elements of a process. Atomic activities form the smallest unit of work. Composite activities, however, refer to a sub-process and can be used to reuse sets of activities or to structure large processes.
- The *behavioral* perspective describes the chronological behavior of a process and is covered by the control flow in traditional process models.
- The *organizational* perspective covers the assignment of human tasks to participants.
- The *informational* perspective describes the information entities accessed during process activities. [14]
- The *operational* perspective describes how an atomic activity is implemented, i.e., what exactly is to be done when the execution of a process reaches a certain activity. [15]

A rule-based model focuses on crosscutting relations rather than the exact flow of activities [6]. A typical requirement is that "assuming that it has been created by a trainee, the document must be reviewed by his or her supervisor before it can be published". In this example, the behavior, the information and the organization are interwoven. If a modelling construct is viewed as a rule, then *cross-perspective business process modelling* involves that both the conditions and the consequences may depend on all of the supported perspectives.

To summarize, we claim that an organization's business processes cover the entire range of process types from routine to agile. In order to support such landscapes, the challenge is now to provide IT support for both process types and especially combinations of them. Throughout this paper, we will show that existing approaches do not handle this task sufficiently. As a result, the contribution of this paper consists of the following parts:

- An integrated business process execution platform for routine and agile processes and combination of them will be presented.
- It comprises an alternative solution for agile processes allowing for rule-based, cross-perspective and modal process modelling.
- This solution will be evaluated on the basis of an acknowledged requirements framework for business process management systems.

The remainder of this paper is organized as follows. Section II describes the structure of the Process Navigation platform and the reasons for its design. Section III shows how behavioral patterns can be implemented for agile processes on the PN platform. Sections IV and V continue for organizational and informational patterns. Section VI summarizes the evaluation of the platform. Section VII describes how agile processes can be executed on the platform. Section VIII gives an overview of other approaches to supporting routine and agile business processes and Section IX concludes the paper.

II. THE PROCESS NAVIGATION PLATFORM

The goal of the *Process Navigation* (PN) platform is to support both routine and agile business processes in an integrated manner. As mentioned above, procedural flow-oriented process models are considered as well-suited for capturing routine processes. BPMN represents the state of the art of procedural modelling. As a result, routine business processes can be covered by offering execution support for BPMN models. For this, the PN platform implements a variant of the *Process Virtual Machine* (PVM) [16] which also forms the basis of widely used process engines like *Activiti*.

The *Case Management Model and Notation* (CMMN) represents recent efforts to standardize rule-based business process modelling. Instead of relying on an explicit flow, event-condition-action (ECA) rules constrain the entry and/or exit of activities within the model. [17] On the basis of the above mentioned requirements, CMMN still has a number of weaknesses:

- CMMN does not support different modalities so that rules may not be classified, e.g., as only recommended. As a result, mandatory and recommended actions may not be distinguished and, e.g., a legal framework may not be distinguished from good practice.
- CMMN neglects the organizational perspective. The potential performer of a human task can only be selected on the basis of a role and the perspective is completely missing in the graphical representation of CMMN models (diagrams).
- Cross-perspective modelling is limited in CMMN. Rules may only depend on events of informational entities (case file items) and activities and may only constrain the entry and exit of activities. They may not, e.g., depend on or constrain organizational aspects.

For this reason, the rule-based part of PN does not target one single modelling language. Instead, it relies on a rule-based cross-perspective and modal intermediate language on a textual basis. This language is called *Declarative Process Intermediate Language* (DPIL). The details of DPIL will be described throughout the rest of this paper. To support agile business processes, PN transforms CMMN models into DPIL models and executes them on the rule-based part of the engine [18].

Note that an organization's business processes are not all of the same kind, i.e., routine or agile, but rather form a heterogeneous landscape [19]. In order to support such landscapes, procedural and rule-based process models can be combined by nesting them. A BPMN model may reference a CMMN sub-processes and vice versa. As the execution of procedural and rule-based models differs considerably, the PN platform comprises two process virtual machines, each adjusted to the respective paradigm. Crosscutting services like human task management, enterprise content management (ECM) and identity management (IdM) integration, the monitoring interface and the decomposition of process models is provided by a common process infrastructure. This architecture is shown in Fig. 1.

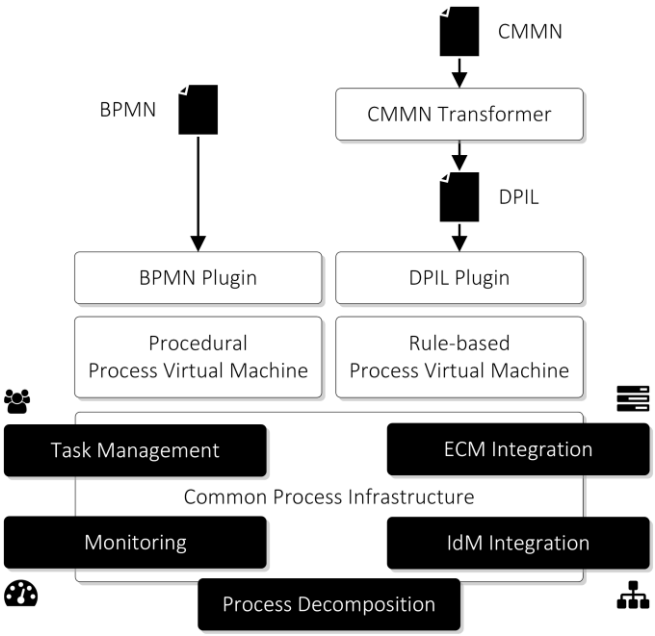

Figure 1. Architecture of the Process Navigation platform

Clearly, the key task now is to evaluate to what extent DPIL itself is suited for business process modelling. For this purpose, the *Workflow Patterns Initiative* has proposed a comprehensive framework of requirements for a business process modelling and execution approach. These so-called *Workflow Patterns* consist of recurring requirements from the behavioral, organizational and informational perspective [20] [21] [22]. These patterns will be used to evaluate DPIL itself on the one hand and to compare it with BPMN on the other hand. Note that the evaluation follows a pragmatic approach. It does not only require a pattern to be implemented at all but with reasonable efforts. This means that an excessively large number of constraints is not a feasible implementation.

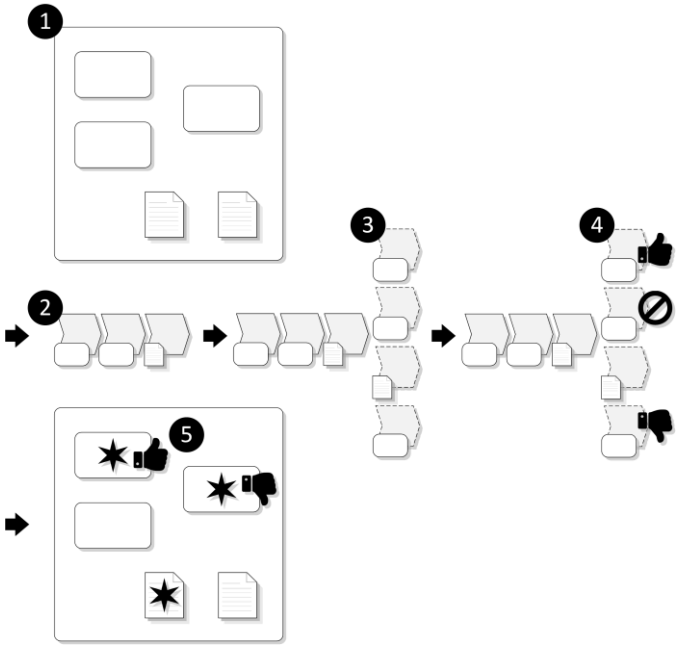

Figure 2. Procedure of rule-based process execution

In order to understand how the business process patterns are implemented in DPIL, its execution principle as shown Fig. 2 will be briefly described in the following. Some of the elements of a DPIL process model (1) undergo a life cycle composed of events that is managed by the engine. A human task, e.g., can be started and completed while a data object can be read or written. The current state of a process is then the series of past events (2).

Besides the static elements like human tasks and data objects, a process model may specify rules constraining that series of events. It may, e.g., claim that some data object may only be written after some task has been started. On the one hand, these rules may be hard rules reflecting, e.g., the legal framework. Hard rules can be used to model mandatory and forbidden actions. On the other hand, there are soft rules reflecting, e.g., good practice. Soft rules can be used to model recommendations.

When the model is executed, the engine simulates one event ahead for every model element (3) and evaluates the resulting series of events on the basis of the rules (4). Each simulated event that does not violate any hard rule is related to an action that the engine interprets immediately (5). A simulated start of a task by a certain participant, e.g., is interpreted as the assignment of this task to the participant. If the start event violates a soft rule, the action is marked as not recommended.

III. BEHAVIORAL PATTERNS

The following sections will show selected example implementations of patterns and will highlight alongside the key concepts of the DPIL notation.

Patterns concerning process control flow have been initially proposed in [23] and were then revised and extended in [20] and [24]. The basic patterns cover the classic sequential, alternative and parallel paths. The sequence pattern (WCP1), e.g., claims that an activity is enabled after the completion of another activity. This is implemented by the following DPIL process:

```
sequence(a, b) iff
  start(of b at :t)
  implies complete(of a at < t)

process Example {
  task CaptureCardDetails
  task VerifyAccount

  ensure sequence(CaptureCardDetails,
    VerifyAccount)
}
```

DPIL allows for the definition of macros in order to make it possible for expressions to be reused. The `sequence` macro claims the existence of a start event of task *b* implies the occurrence of a complete event of task *a* before that. The example process comprises the two tasks `CaptureCardDetails` and `VerifyAccount`. A hard rule (type **ensure**) leads to a sequence between the two tasks. Note that the pattern as well as its implementation above only claims a partial order of the activities which differs from the sequence flow as specified in BPMN [5].

A parallel split (WCP2) denotes a point in the process where a single thread of control splits into multiple threads which can be executed in parallel, thus allowing activities to be executed

simultaneously or in any order [20]. This pattern has no equivalent in a rule-based model (code '~') as all activities can be executed simultaneously or in any order without any constraints imposed on the model. The same holds for a simple merge (WCP5) where two or more alternative branches come together without synchronization [20], i.e., without imposing any constraints.

Most of the advanced control flow patterns represent diverse variations of the inclusive join (OR join) and other complex procedural constructs. No attempt was made to transfer these patterns to the rule-based world (code '?') so that they are considered unsupported (code '-') later on. For reasons of simplicity, DPIL does not support multiple instances of an activity. As a result, the according patterns are considered as unsupported as well. Table I lists all behavioral patterns together with their support by DPIL and BPMN according to [25].

TABLE I.
SUPPORT FOR BEHAVIORAL PATTERNS IN DPIL AND BPMN

Basic		**D**	**B**	34	Static Partial Join for Multiple Instances	-	+/-
1	Sequence	+	+	35	Cancelling Partial Join for Multiple Instances	-	+/-
2	Parallel Split	~	+	36	Dynamic Partial Join for Multiple Instances	-	-
3	Synchronization	+	+	**State-based**		**D**	**B**
4	Exclusive Choice	+	+	16	Deferred Choice	+	+
5	Simple Merge	~	+	17	Interleaved Parallel Routing	+	-
Advanced		**D**	**B**	18	Milestone	+	-
6	Multi-Choice	+	+	39	Critical Section	-	-
7	Structured Synchronizing Merge	?	+	40	Interleaved Routing	+	+/-
8	Multi-Merge	?	+	**Cancellation and Force Completion**		**D**	**B**
9	Structured Discriminator	?	+/-	19	Cancel Task	-	+
28	Blocking Discriminator	?	+/-	20	Cancel Case	+	+
29	Cancelling Discriminator	?	+	25	Cancel Region	+/-	+/-
30	Structured Partial Join	?	+/-	26	Cancel Multiple Instance Activity	-	+
31	Blocking Partial Join	?	+/-	27	Complete Multiple Instance Activity	-	-
32	Cancelling Partial Join	?	+/-	**Iteration**		**D**	**B**
33	Generalized AND-Join	?	+	10	Arbitrary Cycles	+	+
37	Local Synchronizing Merge	?	-	21	Structured Loop	+	+
38	General Synchronizing Merge	?	-	22	Recursion	-	-
41	Thread Merge	?	+	**Termination**		**D**	**B**
42	Thread Split	?	+	11	Implicit Termination	-	+
Multiple Instances		**D**	**B**	43	Explicit Termination	+	+
12	without Synchronization	-	+	**Trigger**		**D**	**B**
13	with a Priori Design-Time Knowledge	-	+	23	Transient Trigger	-	-
14	with a Priori Run-Time Knowledge	-	+	24	Persistent Trigger	+	+
15	without a Priori Run-Time Knowledge	-	-				

IV. ORGANIZATIONAL PATTERNS

Patterns concerning the organizational aspects of a business process are summarized in [21] where process participants are referred to as 'resources'. They are divided into patterns that refer to the design phase of the process (creation), to the system's perspective (push), to the participants' perspective (pull), to exceptional situations (detour), to event- or context-based execution (auto-start), to visibility and to multiple participants working on the same task.

DPIL assumes a simple organizational metamodel where *identities* and *groups* can be interconnected by *relations* of a certain *relation type* [26]. The information is taken from a central organizational model like, e.g., an LDAP service and is only referenced within the DPIL process model. Human participation in a DPIL process is implemented using the **`task`** type of activity. From the rule-based engine's viewpoint, a task may be started and completed. The task management service of the PN platform extends the life cycle of a task by reserve/release and suspend/resume states.

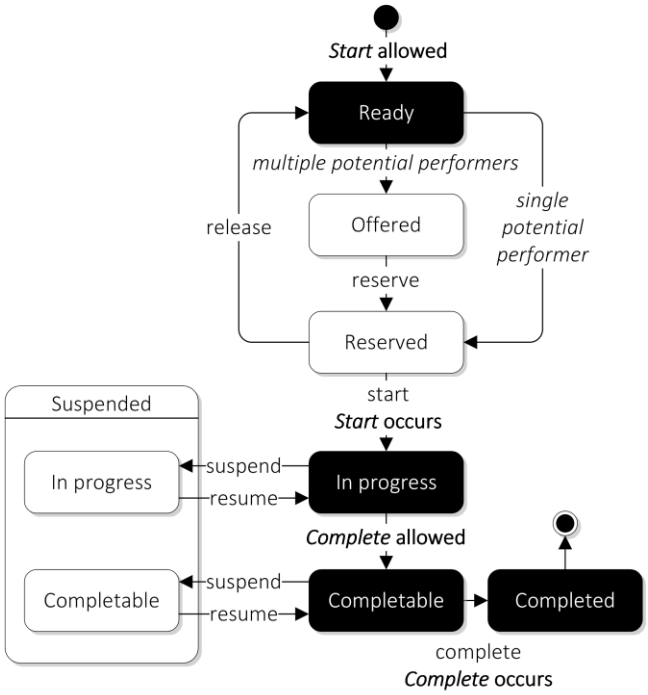

Figure 3. Life cycle of a human task in DPIL

Fig. 3 shows the combined life cycle of a task. States in black are managed by the engine and can therefore be observed and constrained by process rules. States in white are added by the task management service.

The simplest method of distribution is the direct allocation of a participant to a task at design time (WRP1). Besides this, tasks may be distributed on the basis of roles (WRP2), capabilities (WRP8) or organizational relationships (WRP10). The following example process contains each of these patterns:

```
identity Fred
group Manager, Engineer
relationtype hasRole, hasJob, isManagerOf

direct(t, i) iff
  start(of t) implies start(of t by i)

role(t, r) iff
  start(of t by :i)
  implies relation(subject i
    predicate hasRole object r)

process Organisation {
  task FixBentley
  task ApproveTravelRequisition
  task AirframeExamination
  task ClaimExpenditure
  task AuthoriseExpenditure

  advise direct(FixBentley, Fred)

  advise role(ApproveTravelRequisition,
    Manager)

  advise start(of AirframeExamination)
    implies start(of AirframeExamination
      by :i
      by.servicingExperienceInYears >= 10)
    and relation(subject i predicate hasJob
      object Engineer)

  advise start(of AuthoriseExpenditure
      by :authoriser at :t)
    implies start(of ClaimExpenditure
      by :claimer at < t)
    and relation(subject authoriser
      predicate isManagerOf object claimer)
}
```

The above model references the identity `Fred`, two groups and three relation types. The task `FixBentley` should only be performed by `Fred` (WRP1). The rule is soft (type **advise**) so that the assignment to `Fred` is recommended but not mandatory, i.e., other participants are allowed to perform the task but they are advised not to do so. The task `ApproveTravelRequisition` should be performed by a participant having the role of a `Manager` (WRP2). The `AirframeExamination` should be performed by an `Engineer` having at least ten years of servicing experience (WRP8). Finally, `AuthoriseExpenditure` should be performed by the manager of the participant who has performed `ClaimExpenditure` (WRP10). For reasons of simplicity, DPIL does not offer any means of referencing earlier process instances (WRP9) and is not able to perform any scheduling (WRP15-17). Table II lists all organizational patterns together with their support by DPIL and BPMN according to [25].

TABLE II.
SUPPORT FOR ORGANIZATIONAL PATTERNS IN DPIL AND BPMN

	Creation	**D**	**B**	23	Resource-Initiated Execution - Offered Work Item	+	-
1	Direct Distribution	+	+	24	System-Determined Work Queue Content	-	-
2	Role-Based Distribution	+	+	25	Resource-Determined Work Queue Content	+	-
3	Deferred Distribution	+	-	26	Selection Autonomy	+	-
4	Authorization	+	-		**Detour**	**D**	**B**
5	Separation of Duties	+	-	27	Delegation	-	-
6	Case Handling	-	-	28	Escalation	-	-
7	Retain Familiar	+	-	29	Deallocation	+	-
8	Capability-Based Distribution	+	-	30	Stateful Reallocation	-	-
9	History-Based Distribution	+/-	-	31	Stateless Reallocation	-	-
10	Organisational Distribution	+	-	32	Suspension-Resumption	+	-
11	Automatic Execution	+	+	33	Skip	+/-	-
	Push	**D**	**B**	34	Redo	+	-
12	Distribution by Offer - Single Resource	+	-	35	Pre-Do	+/-	-
13	Distribution by Offer - Multiple Resources	+	-		**Auto-Start**	**D**	**B**
14	Distribution by Allocation - Single Resource	+	+	36	Commencement on Creation	+	+
15	Random Allocation	-	-	37	Commencement on Allocation	-	-
16	Round Robin Allocation	-	-	38	Piled Execution	-	-
17	Shortest Queue	-	-	39	Chained Execution	+	+
18	Early Distribution	-	-		**Visibility**	**D**	**B**
19	Distribution on Enablement	+	+	40	Configurable Unallocated Work Item Visibility	-	-
20	Late Distribution	-	-	41	Configurable Allocated Work Item Visibility	+	-
	Pull	**D**	**B**		**Multiple Resources**	**D**	**B**
21	Resource-Initiated Allocation	+	-	42	Simultaneous Execution	+	+
22	Resource-Initiated Execution - Allocated Work Item	+	-	43	Additional Resources	-	-

V. INFORMATIONAL PATTERNS

Patterns concerning the informational aspects of a business process are collected in [22]. DPIL distinguishes two types of data objects within processes. A *variable* is bound to the lifetime of the process instance and references an object within the address space of the engine. A variable represents process instance-specific data. Instead, a *document* references a file managed externally by an *Enterprise Content Management* (ECM) system and is therefore independent from the lifetime of the process instance. The PN platform implements the *Content Management Interoperability Services* (CMIS) standard so that it may communicate with a large number of current ECM systems [27]. Both types of data objects may be read or written by an identity involved into the process. These read and write events may be observed and constrained within process rules.

A variable is visible to instances of the surrounding process (WDP5) and those of its sub-processes (WDP2). Read and write events of a data object may be constrained so that the object may only be accessed during the execution of a certain task (WDP1). This pattern is claimed by the `local` macro in the following process:

```
local(task, object) iff
  read(of object at :tr)
  implies start(of task at < tr at :ts)
  and not complete(of task at > ts at < tr)

process Example {
  task CalculateFlightPath
  variable WorkingTrajectory

  advise local(CalculateFlightPath,
    WorkingTrajectory)
}
```

The above process claims that the `WorkingTrajectory` may only be read during the execution of the `CalculateFlightPath` task. Data-based pre- and postconditions on tasks can be implemented by constraining their start and complete events. They may depend on a certain data value or just its existence (WDP34-37) like in the following:

```
consumes(consumer, incoming) iff
  start(of consumer)
  implies write(of incoming)

produces(producer, outgoing) iff
  complete(of producer)
  implies write(of outgoing)

process Example {
  task RocketInitiation

  variable Countdown
  variable IgnitionData

  ensure consumes(RocketInitiation,
    Countdown)

  ensure start(of RocketInitiation)
    implies write(of Countdown value 2)
  ensure produces(RocketInitiation,
    IgnitionData)
}
```

In the above model, the task `RocketInitiation` requires the `Countdown` variable to exist, the variable `IgnitionData` must be updated during the `RocketInitiation` task and the `RocketInitiation` may only start after the `Countdown` has a value of 2. The `consumes` and `produces` macros impose the classic data existence pre- and postconditions for modelling data flows within a process.

A business process may be partially performed by pieces of software that might simply be termed 'services'. Calling such services from a DPIL process is supported by the **`operation`** type of activity. Operations may access data objects of the process as parameters (**`with`**) and may write their result into data objects again (**`to`**). The life cycle of an operation comprises the event of its invocation (event **`invoke`**) and its return (event **`return`**). Operations may be used to implement data transformations (WDP32, 33) like in the following:

```
process Example {
  operation FromKmhToMph
    "http://service.org/kmhToMph"
    with SpeedKmh as speed
    to SpeedMph as result

  task ReviewSpeed

  variable SpeedKmh
  variable SpeedMph

  ensure start(of ReviewSpeed)
    implies return(of FromKmhToMph)

  ensure invoke(of FromKmhToMph)
    implies write(of SpeedKmh)
}
```

The `FromKmhToMph` is invoked immediately after the `SpeedKmh` has been updated and the `ReviewSpeed` task can only be started after the operation has returned.

Service operations can be used to implement most of the external interaction patterns. Table III lists all informational patterns together with their support by DPIL and BPMN according to [25].

TABLE III.
SUPPORT FOR INFORMATIONAL PATTERNS BY DPIL AND BPMN

	Visibility	D	B				
				21	Environment to Case - Push-Oriented	+	-
1	Task Data	+/-	+	22	Case to Environment - Pull-Oriented	+	-
2	Block Data	+	+	23	Workflow to Environment - Push-Oriented	+	-
3	Scope Data	-	-	24	Environment to Workflow - Pull-Oriented	-	-
4	Multiple Instance Data	-	+/-	25	Environment to Workflow - Push-Oriented	-	-

5	Case Data	+	+	26	Workflow to Environment - Pull-Oriented	+	-
6	Folder Data	+	-	**Transfer**		**D**	**B**
7	Workflow Data	+	-	27	Transfer by Value - Incoming	-	+
8	Environment Data	+/-	-	28	Transfer by Value - Outgoing	-	+
Internal Interaction		**D**	**B**	29	Transfer - Copy In/Copy Out	-	+/-
9	Task to Task	+	+	30	Transfer by Reference - Unlocked	+	-
10	Block Task to Sub-Workflow Decomposition	+	+	31	Transfer by Reference - With Lock	+	+
11	Sub-Workflow Decomposition to Block Task	+	+	32	Transformation - Input	+	+/-
12	to Multiple Instance Task	-	-	33	Transformation - Output	+	+/-
13	from Multiple Instance Task	-	-	**Data-based Routing**		**D**	**B**
14	Case to Case	+	-	34	Task Precondition - Data Existence	+	+
External Interaction		**D**	**B**	35	Task Precondition - Data Value	+	-
15	Task to Environment - Push-Oriented	+	+	36	Task Postcondition - Data Existence	+	+
16	Environment to Task - Pull-Oriented	+	+	37	Task Postcondition - Data Value	+	-
17	Environment to Task - Push-Oriented	+	+	38	Event-based Task Trigger	+	+
18	Task to Environment - Pull-Oriented	+	+	39	Data-based Task Trigger	+	+
19	Case to Environment - Push-Oriented	+	-	40	Data-based Routing	+	+
20	Environment to Case - Pull-Oriented	+	-				

VI. SUMMARY

Fig. 4 shows the proportion of supported patterns from the three perspectives for DPIL and BPMN. DPIL supports 52% of the behavioral, 62% of the organizational and 75% of the informational patterns. BPMN supports 67% of the behavioral, 19% of the organizational and 75% of the informational patterns.

The evaluation reveals two valuable insights. On the one hand it becomes clear that DPIL and the PN platform meet around 50% more of the requirements than BPMN does. On the other hand it confirms that both approaches occupy their niches. Where the classic control flow patterns are concerned, BPMN is superior to DPIL with a coverage of 67% instead of 52%. As expected, when exactly specified complex control flow patterns are in focus then BPMN is best-suited. Informational and organizational relationships, however, can be expressed more comprehensively and more precisely in DPIL.

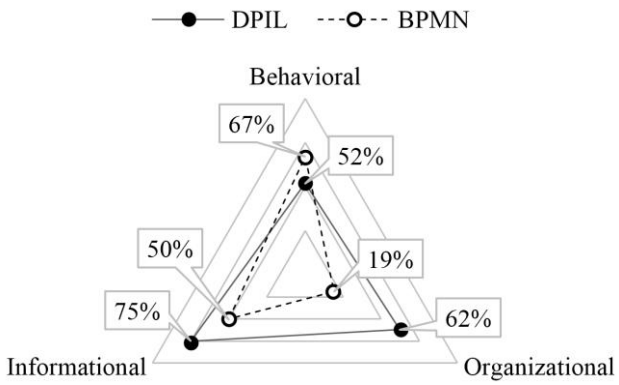

Figure 4. Support for workflow patterns from three different perspectives in DPIL and BPMN

The combination of both approaches makes the Process Navigation platform a comprehensive platform for the support of both routine and agile processes.

VII. EXECUTING CMMN MODELS

Although, as pointed out above, CMMN still has a number of weaknesses, it is considered as the most advanced notation for modelling agile business processes. A model in CMMN is called a *case* and is built from at least one *stage* which is the equivalent of a sub-process. A stage in turn may consist of, again, stages, *human tasks*, routine sub-processes (*process tasks*) or *milestones*. In addition, a case defines a *case file* comprising *case file items* which are the equivalents of data objects. *Plan items*, i.e., stages, tasks and milestones may declare entry and exit criteria using *sentries*. A sentry is triggered by an *event* in turn emitted by a plan item or case file item and may declare a *condition* based only on a case file item.

A stage is the equivalent of a DPIL process and a case file item maps to a DPIL data object. The entry and exit criteria of plan items are translated to the according DPIL rules. A sentry may depend on one or more events. All of them must occur and the condition, if any, must hold to trigger the sentry. Therefore, multiple events for the same sentry have a logically conjunctive semantics (and). If, however, multiple sentries are used for the same plan item, only one of them must be triggered to enter or exit the item. Therefore, multiple sentries for the same item have a logically disjunctive semantics (or).

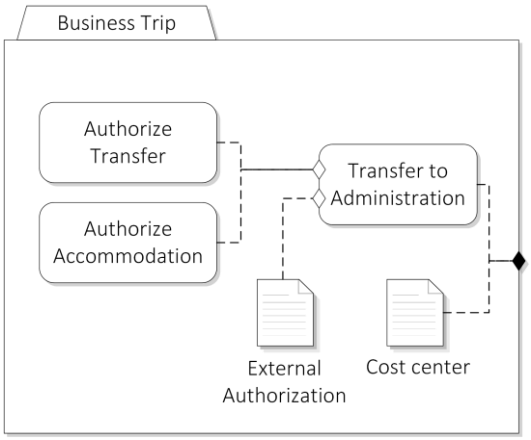

Figure 5. Example CMMN process model to be translated to DPIL and executed on the PN platform

Fig. 5 shows an example CMMN diagram that is to be translated to a DPIL model for execution. Entry criteria are depicted by shallow and exit criteria by filled diamond shapes. The model claims that the task *Transfer to Administration* can only be performed after *Authorize Transfer* and *Authorize Accommodation* or after the *External Authorization* is present. The stage is finished after the transfer is done and the *Cost Center* is present. Note that the exact events are not represented within the diagram.

The above model can be translated to the following DPIL model and executed on the PN platform:

```
process BT "Business Trip" {
  task AT "Authorize Transfer"
  task AA "Authorize Accomodation"
  task TA "Transfer to Administration"

  variable EA "External Authorization"
  variable CC "Cost Center"

  advise "Authorization Required":
    start(of TA)
    implies (complete(of AT)
      and complete(of AA))
    or write(of EA)

  milestone complete(of TA) and write(of CC)
}
```

The two case file items are interpreted as process-specific variables. The two sentries that form the entry criteria for *Transfer to Administration* result in one rule respecting their different semantics (conjunctive and disjunctive). Note that, even though CMMN does not support different modalities, the entry criteria are implemented as soft rules to illustrate the explanation capabilities of the platform later on. The exit criterion sentry of the stage is translated to a milestone which terminates the process.

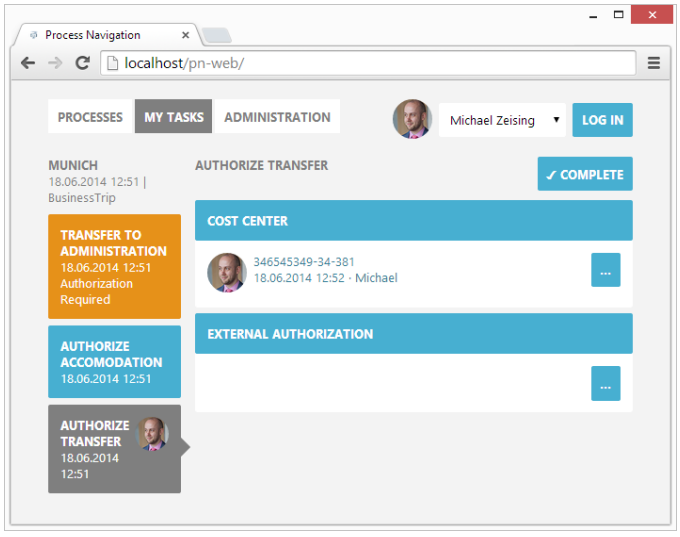

Figure 6. Process participant's view

A prototypical web-based frontend to PN's human task management component enables people to communicate with process instances. Fig. 6 shows a participant's view on an instance of the *Business Trip* process.

Current tasks appear on the left-hand side of the screen. The example illustrates the explanation capabilities of the platform: performing the *Transfer to Administration* task is possible but not recommended because it would violate the *Authorization Required* advice (not recommended). This is indicated by the orange background and the potentially violated rule written below the task name. *Authorize Accommodation*, instead, would not violate any constraint (neutral) and is therefore written on a blue background. Currently accessible data objects are shown on the right-hand side of the screen.

The color coding is used throughout the entire user interface: a blue color tone indicates a neutral action, the orange color tone indicates a not recommended action and a green color tone indicates a recommended action (not shown in the example). This color coding together with the traceability of actions to the process rules form the decision support capabilities of Process Navigation platform. They are the main benefit to the process participant. Up to the authors' knowledge, these capabilities are unique to the platform and cannot be found in traditional business process execution solutions.

VIII. Related Work

Within the scope of the *MOBILE* project, all five fundamental process perspectives where combined in one workflow management system for the first time. Moreover, a combination of procedural and rule-based modelling was proposed. Procedural models comprised the classic elements for serial, alternative and parallel execution while rule-based models where built from deadline, delay and existence constraints. Both types of models could be nested. [2] However, the rule-based processes where mentioned only once and never again in any subsequent publication of the group.

The concept of *Pockets of Flexibility* allows a procedural process model to contain special sub-processes that impose no control flow [28]. However, the concept does not support the specification of rules within the "pockets" and an inverse nesting is not intended. As a result, this is a very limited approach.

The *Declare* framework was designed for modelling and executing rule-based business processes. In its combination with the procedural modelling and execution system YAWL it represents an alternative to the Process Navigation platform. In its most publicized variant, a Declare process model is built from a set of rule templates each of which is mapped to an expression in *Linear Temporal Logic* (LTL). One such template is, e.g., *response*(a, b) with the LTL mapping $\Box(a \rightarrow \Diamond b)$ claiming that whenever a is performed then b must be performed eventually. The resulting LTL formula is then converted to an automaton for execution. [29] Declare only constrains the starts of activities and interrelates them temporally. However, a rule as a whole may depend on arbitrary conditions including informational and organizational context. The rule *response*(a, b) with the condition $x < 3$ claims that whenever a is performed then b must be performed eventually if $x < 3$. However, a requirement like "assuming that it has been created by a trainee, the document must be reviewed by his or her supervisor before it can be published" cannot be expressed in this way. Hence, Declare does not support cross-perspective modelling on the basis of the requirements and is therefore not suited for the rule-based part of the PN platform.

The *Engine for Semantic Process Navigation* (ESProNa) is a concept for the execution of rule-based business processes [30]. It has no specific rule language and is rather a programming library for developing business processes in the Prolog language. As a result, ESProNa processes may not be validated in any way but only checked for correct Prolog syntax. In addition, ESProNa does not represent a process execution engine [23] but only covers the evaluation of the process rules. Concepts like process instance management, rules for terminating instances and nesting of process models are missing. ESProNa may not cover the rule-based part of the PN platform but is rather a predecessor of the approach at hand.

The *EM-BrA²CE* project [31] represents a first step towards the unification of business rules and processes. It extends the *Semantics of Business Vocabulary and Business Rules* (SBVR) framework [32] by concepts like activities, states and participants and therefor allows for the specification of business processes in SBVR. The main difficulty lies within the enforcement of these process rules, i.e., the execution of such processes. For this, the SBVR rules are translated to event-condition-action (ECA) rules using templates [33]. As only the rules covered by a template can be translated and executed, the original advantage of the use of SBVR becomes worthless. Process are effectively modelled using the ECA templates. In addition, the authors do not mention how different modalities are handled during translation. As a result, neither *EM-BrA²CE* is not a candidate for the rule-based part of the PN platform.

IX. Conclusion

This paper introduced the Process Navigation platform supporting both routine and agile business processes. Routine processes can be modelled by traditional flow-oriented notations like BPMN and executed. This paper now focusses on agile business processes that require a rule-based cross-perspective modal way of modelling. The recent standard CMMN introduces rule-based modelling with a graphical representation but still has a number of weaknesses. As a result, the agile part of the platform relies on the intermediate language DPIL that meets the requirements. CMMN models can be translated to DPIL for executing them on the platform. Furthermore, the paper evaluated DPIL concerning its suitability for business process modelling. It could be shown that, in total, DPIL even meets more requirements than BPMN. However, BPMN is better suited for routine processes as it outperforms DPIL concerning flow-oriented process patterns. DPIL in turn excels BPMN concerning organizational and informational patterns which makes it more suitable for context-focused agile processes. Finally, it was described how CMMN process models can be translated to DPIL models and executed on the Process Navigation platform.

CMMN already represents the state of the art of rule-based business process modelling and is certainly a step in the right direction. However, as mentioned above, a number of weaknesses have been identified. The authors intend to address these weaknesses and propose improvements concerning the organizational perspective, cross-perspective modelling and modalities.

References

[1] M. Hammer and J. Champy, *Reengineering the Corporation: A Manifesto for Business Revolution*, 3rd ed. HarperBusiness, 1993.

[2] S. Jablonski, “MOBILE: A Modular Workflow Model and Architecture,” in *4th International Working Conference on Dynamic Modelling and Information Systems*, 1994, pp. 1–30.

[3] European Computer Manufacturers Association (ECMA), “Standard ECMA-4 - Flow Charts (2nd Edition, withdrawn).” Geneva, CH, 1966.

[4] Software AG, “ARIS Method - ARIS Platform Version 7.2 - Service Release 3.” Software AG, Darmstadt, 2012.

[5] Object Management Group Inc., “Business Process Model and Notation (BPMN) Version 2.0.” 2011.

[6] D. Fahland, D. Lübke, J. Mendling, H. Reijers, B. Weber, M. Weidlich, and S. Zugal, “Declarative versus Imperative Process Modeling Languages: The Issue of Understandability,” in *Enterprise, Business-Process and Information Systems Modeling (10th International Workshop, BPMDS 2009, and 14th International Conference, EMMSAD 2009, held at CAiSE 2009)*, 2009, pp. 353–366.

[7] R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukaviriya, “Declarative Business Artifact Centric Modeling of Decision and Knowledge Intensive Business Processes,” in *15th IEEE International Enterprise Computing Conference (EDOC 2011)*, 2011, no. Edoc, pp. 151–160.

[8] H. Schonenberg, R. S. Mans, N. Russell, N. A. Mulyar, W. M. P. van der Aalst, J. L. G. Dietz, A. Albani, and J. Barjis, “Process Flexibility: a Survey of Contemporary Approaches,” in *Advances in Enterprise Engineering I, 4th International Workshop CIAO! and 4th International Workshop EOMAS, CAiSE 2008*, 2008, vol. 10, pp. 16–30.

[9] H. A. Reijers, T. Slaats, and C. Stahl, “Declarative Modeling — An Academic Dream or the Future for BPM?,” in *11th International Conference on Business Process Management (BPM 2013)*, 2013, pp. 307–322.

[10] M. Pešić, H. Schonenberg, and W. M. P. van der Aalst, “DECLARE: Full Support for Loosely-Structured Processes,” in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, 2007, p. 287.

[11] G. Regev and A. Wegmann, “A Regulation-Based View on Business Process and Supporting System Flexibility,” in *17th International Conference on Advanced Information Systems Engineering (CAiSE ’05) Workshops*, 2005, pp. 35–42.

[12] E. Turban, R. Sharda, and D. Delen, *Decision Support and Business Intelligence Systems*, 9th ed. Prentice Hall, 2010.

[13] M. Zeising, S. Schönig, and S. Jablonski, “Improving Collaborative Business Process Execution by Traceability and Expressiveness,” in *8th International Conference Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2012)*, 2012, pp. 435–442.

[14] B. Curtis, “Process Modeling,” *Commun. ACM*, vol. 35, no. 9, pp. 75–90, Sep. 1992.

[15] S. Jablonski and C. Bußler, *Workflow Management: Modeling Concepts, Architecture and Implementation*. London: Thomson, 1996.

[16] T. Baeyens and M. V. Faura, “The Process Virtual Machine,” 2007. [Online]. Available: http://docs.jboss.com/jbpm/pvm/article/.

[17] Object Management Group, “Case Management Model and Notation (CMMN) - Version 1.0,” no. May. 2014.

[18] S. Schönig, M. Zeising, and S. Jablonski, “Supporting Collaborative Work by Learning Process Models and Patterns from Cases,” in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2013)*, 2013, pp. 60–69.

[19] M. Reichert and B. Weber, *Enabling Flexiblity in Process-Aware Information Systems*. Springer-Verlag Berlin Heidelberg, 2012.

[20] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow Patterns,” *Distrib. Parallel Databases*, vol. 14, no. 3, pp. 5–51, 2003.

[21] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond, “Workflow Resource Patterns: Identification,

Representation and Tool Support," in *17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, 2005, pp. 216–232.
[22] N. Russell, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst, "Workflow Data Patterns: Identification, Representation and Tool Support," in *24nd International Conference on Conceptual Modeling (ER 2005)*, 2005, pp. 353–368.
[23] Workflow Management Coalition, "Workflow Management Coalition: Terminology & Glossary 3.0 (WFMC-TC-1011)." Workflow Management Coalition, 1999.
[24] N. Russell, A. H. M. ter Hofstede, W. M. P. van der Aalst, and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View (BPM Center Report BPM-06-22)," 2006.
[25] Workflow Patterns Initiative, "Workflow Patterns Home Page," 2011. [Online]. Available: http://www.workflowpatterns.com/.
[26] C. Bußler, *Organisationsverwaltung in Workflow-Management-Systemen*. Wiesbaden: DUV, 1998.
[27] OASIS, "Content Management Interoperability Services (CMIS) Version 1.1." 2012.
[28] S. Sadiq, W. Sadiq, and M. Orlowska, "Pockets of Flexibility in Workflow Specification," in *20th International Conference on Conceptual Modeling (ER'2001)*, 2001, pp. 513–526.
[29] M. Pešić, "Constraint-Based Workflow Management Systems: Shifting Control to Users," Technische Universiteit Eindhoven, 2006.
[30] M. Igler, "ESProNa - Eine Constraintsprache zur multimodalen Prozessmodellierung und navigationsgestützten Ausführung," Universität Bayreuth, 2011.
[31] S. Goedertier, R. Haesen, and J. Vanthienen, "Rule-based Business Process Modelling and Enactment," *Bus. Process Integr. Manag.*, vol. 3, no. 3, pp. 194–207, 2008.
[32] Object Management Group, "Semantics of Business Vocabulary and Business Rules (SBVR) - Version 1.2." 2013.
[33] W. De Roover, F. Caron, and J. Vanthienen, "A Prototype Tool for the Event-Driven Enforcement of SBVR Business Rules," in *Business Process Management Workshops, BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*, Springer Berlin Heidelberg, 2012, pp. 446–457.