Deliverable 3.2

Bericht zum Prototypen für die Modellbasierte Ansteuerung von Robotern

(Report of the Prototype for Model-based Operation of Robots)

Dokument Version	:	Final
Abgabe	:	31.01.2021
Dissemination Level	:	Öffentlich
Beitrag zu	:	WP3
Dokument Inhaber	:	BOC
Dokument Name	:	ComplAI-D3.2 Prototyp Modellbasierte Ansteuerung von Robotern
Revision	:	1.0

Projektakronym	:	compl@i
Projekttitel	:	Collaborative Model-Based Process Assessment for trustworthy AI in Robotic Platforms
Grant Agreement n.	:	33755860
Call	:	IDEEN LAB 4.0 (2019)
Projektdauer	:	12 Monate, beginnend mit 01/02/2020
Website	:	https://complai.innovation-laboratory.org/

Revision History

REVISION	DATE	INVOLVED PARTNERS	DESCRIPTION
0.1	02/11/2020	BOC, JR	Initial, commented table of content, content collection
0.2	02/11/2020	BOC	First draft, introduction, layers, workflows
0.3	02/11/2020	BOC	First draft, package description, conclusion, executive summaries
0.4	01/12/2020	BOC	Writing and Revision of draft version, extension, appendix Introduction Chimera Chapter
0.5	20/12/2020	BOC	Concluding the document for internal review
0.8	10/01/2021	BOC	Ready for Review
1.0	31/01/2021	BOC, JR	Final Review of Deliverable

List of Contributors:

Wilfrid Utz (BOC), Anna Sumereder (BOC), Damiano Falcioni (BOC), Harald Kühn (BOC)

Bernhard Dieber (JR), Benjamin Breiling (JR)

List of Reviewers:

Benjamin Breiling (JR)

Robert Woitsch (BOC)

Kurzfassung

Dieses Dokument ergänzt das D3.1 „Spezifikation des Anwendungsfalls Support“ indem basierend auf der gewählten Supermarkt Domäne, der Prototyp für die modellbasierte Ansteuerung von Robotern präsentiert wird. Die Demonstration des Prototyps verwendet Laborexperimente, cyber-physischen Systeme und Geräte, welche durch die Partner – BOC und JR – bereitgestellt werden.

Zwei Konzepte, welche essenziell für diesen Prototypen sind wurden identifiziert

(1) die Unterscheidung von Abstraktionsschichten, und

(2) die Ressourcenallokation bei Arbeitsabläufen.

Beide Aspekte sind essenziell, um die semantische Distanz zwischen den verschiedenen Schichten und Plattformen in einem Supermarkt Kontext darzustellen und mit geeigneter Wissensbasierter Technologie zu überbrücken. Basierend auf den drei Plattformen – sozial, virtuell und physisch – welche in D3.1 präsentiert wurden, sind insgesamt vier Abstraktionsschichten identifiziert worden.

Diese sind

(1) der **abstrakte Arbeitsablauf**, welcher plattformunabhängig Sequenzen von Roboteroperationen darstellt, um ein Domänen-spezifisches Ziel zu erreichen,

(2) der **technische Arbeitsablauf**, der durch den Aufruf von externen Operationen oder generischen Subprozessen noch plattformunabhängig ist und eine Sequenz von ausführbaren Roboterfähigkeiten repräsentiert,

(3) die **plattformspezifische Abstraktion**, welche in Form von Konzeptmodellen dargestellt wird und durch spezifische IoT Adaptoren und Schnittstellen plattformspezifische Operationen beschreibt, und

(4) die **Roboterplattform** inklusive IoT Adaptoren und Software, welche plattformspezifische Operationen zur Ausführung von Arbeitsabläufen zur Verfügung stellen.

Weiters sind nicht nur unterschiedliche Plattformen und Abstraktionsschichten notwendig, sondern auch ein Konzept zur Ressourcenallokation bei den erwähnten Arbeitsabläufen. Für diese so genannten Workflows präsentieren wir drei unterschiedliche Algorithmen zur Ressourcenallokation (1) **fix**, (2) **vor**, und (3) **während der Ausführung**. Diese unterscheiden sich im groben durch den Zeitpunkt der Interaktion und der Möglichkeit der Unterbrechung während der Ausführung. Somit kann zwischen vordefinierten Systemen und autonomen Verhalten unterschieden werden. Instanziierungen für die Abstraktionsschichten, sowie für die Ressourcenallokationsszenarien werden auf Basis von verschieden Modellierungssprachen, wie BPMN, Petri Nets und Flowchart, erprobt.

Abschließend wird das Prototyp Paket vorgestellt, um zu evaluieren, wie ein Roboter mittels Modelle angesteuert werden kann. Die notwendigen Materialien werden zum Download bereitgestellt.

Executive Summary

The document extends the D3.1 “Specification of Application Scenario Support” presenting the prototype within the supermarket domain using a model-based approach. The demonstration prototypes are based on laboratory experiments, and cyber-physical system setup and devices provided by the partners, in particular by BOC and JR.

Two major concepts were identified to be crucial for this prototype, these are

(1) different abstraction layers and

(2) various workflow bindings, in order to bridge the semantic distance between layers and platforms within the supermarket domain.

Based on the three platforms – social, virtual and physical – presented in D3.1, four abstraction layers were identified. Those are

(1) **abstract workflow** that represents a sequence of robotic actions without platform binding in order to fulfil the domain specific goal,

(2) **technical workflow**, which is also platform independent by calling external operations or generic subprocesses and represents a sequence of executable capabilities of a robotic platform,

(3) **platform specific abstraction** that is presented in form of concept models characterized by a specific IoT adaptor that encapsulations platform specific APIs, and

(4) **robotic platform**, IoT adaptors and workflow engines that use platform specific commands and operations. Furthermore, not only platform and abstraction layers are required, but also different workflow bindings. These workflow bindings are seen as a means of resource allocation.

We propose (1) **fixed**, (2) **pre**, and (3) **late binding**. Those workflow bindings can be differentiated by having a look at the point of interaction and interruption during the processing. This distinguishes between pre-defined behaviour vs. autonomous system. Instantiations for the abstraction layers as well as for the workflow bindings are presented by using different modelling languages such as BPMN, Petri Nets and flowchart.

The resulting prototype package provides the necessary hands-on material and is provided for download to test the model-based approach.

Table of Contents

Kurzfassung 3
Executive Summary 4
List of Figures 6
List of Abbreviations 7
1. Introduction 8
1.1 Relation to Work Package 10
1.2 Document Structure 10
2. Overview of Model-Driven Robot Execution 11
2.1 Introduction AI 12
2.2 Using Petri Nets for Exeuction 13
2.3 Using Flow Chart for Execution 14
2.4 Using BPMN for Execution 15
3. Adaptive Workflows for Robot Exeuction 16
3.1 Laboratory Setting 16
3.2 Fixed Workflows: Fixed Binding of Robots Execution 17
3.3 Alternative Workflows: Pre-Binding of Robot Execution 18
3.4 Adaptive Workflows: Late-Binding of Robot Execution 19
4. OMiLAB – Proof-of Concept 20
4.1 Installation of Proof-of-Concept Package in OMiLAB 20
4.2 Starting of Proof-of-Concept Package in OMiLAB 21
4.3 Demonstration of Model-based Robot Execution in OMiLAB 22
5. CHIMERA Prototype 25
5.1 Chimera Software stack 26
5.2 How to use and setup the stack 27
5.2.1 WorkflowManager API 27
5.2.2 URsim 27
5.2.3 Webinterface 27
5.2.4 Listing all available workflows 28
5.2.5 How to start a workflow 29
5.2.6 How to check if workflows are running 29
5.3 Workflows 29
6. Conclusion 31
Appendix A 32
Appendix B 49
Appendix C 56
Appendix D 64

List of Figures

Figure 1: Abstraction Layers for the Application Scenarios 11
Figure 2: AI Integration on Abstraction Layers 12
Figure 3: Instantiation of the 4 Abstraction Layers using BPMN and Petri Nets 13
Figure 4: Instantiation of the 4 Abstraction Layers using BPMN and Flowchart 14
Figure 5: Instantiation of the 4 Abstraction Layers using BPMN and Workflows 15
Figure 6: Interconnection of Settings 16
Figure 7: Instantiation of a fixed binding workflow by using Petri Nets 17
Figure 8: Instantiation of a pre binding workflow by using flowcharts 18
Figure 9: Instantiation of a late binding workflow by using BPMN and a workflow engine 19
Figure 10: Prototype Package Structure 20
Figure 11: Starting Bee-Up as Standalone 21
Figure 12: Operating Robots with Workflows 21
Figure 13: Mobile Platform Environment 23
Figure 14: Robot Operation in a Remote Setting 23

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
BPMN	Business Process Modelling Notation
IoT	Internet of Things
SW	Software

1. Introduction

This document describes the prototype for model-based operation of robots.

The demonstration prototypes are based on laboratory experiments, and cyber-physical system setup and devices provided by BOC and JR. In specific, the introduction of the OMiLAB experimentation space (conceptually and physically/virtually) included in D3.1 is the foundation for testing the experiments for this prototype.

According to Retail Austria (2020)[1] there is a variety of digital application in retail. For example, store traffic recognition can be used to observe and visualise store traffic in order to better place expensive premium products or introduce new products. Furthermore, in-store concierges in form of mobile robots can support customer to find the right product. Also, in-store voice commerce and consultancy concepts allow consultation and advertisement of products. Moreover, automatic check-out shopping carts can support with scanning products and checking them out after putting those into the shopping cart and additionally, complementary products can be automatically suggested on the shopping cart screen. Not only the action of shopping can be supported by digital applications, but also smart buildings can facilitate the access management by using biometric data, theft detection or hazard management by using smart cameras and power management by using sensors for energy efficiency. Furthermore, we can see the interest towards a cashier free shop, which is open 24/7.

This demonstrative domain was chosen for the application scenarios as well as the proof-of-concept realisation, as it provides plethora of challenges, where the appropriate ones for the project have been selected. It is important to bridge the semantic distance between layers and platforms. As a recap from D3.1 the three platforms and their preconditions and model-based approaches are summarized below:

1. **Social Platform** – This platform mostly represents the traditional approach of shopping. The preconditions consist of capabilities of the actor. It is assumed that a human actor has a capability to (a) interpret informal and semi-formal models, (b) interact between physical and digital world, (c) complete incomplete and vague models. A model-based approach is followed, as this is easy to understand for humans, simple, ideally provides a graphical representation and no virtualization necessary. Furthermore, incomplete knowledge can be easily dealt with by using hints, heuristics or images. Any kind of adaptivity is automatically performed by the human actor. For example, if a human wants to buy 6 eggs, a set of instinctive decisions like “10 eggs with a long BBD (best before date)” are equally OK in case 6 eggs are not available, “the quality code” of eggs is checked, “taking the egg box and handling with care” is intuitively performed as well as finally “the quality approval” b opening the box and checking if eggs are broken are also intuitively performed by humans.
2. **Virtual Platform** – The virtual platform requires some preparations, for instance an online shop needs to virtualize the provided products, a user account hast to be established, an APIs needs to be available and a workflow engine has to be programmed. Here, a model-based approach can be supportive in the context of formal correct and verifiable sequences of actions that can be executed by a workflow engine using the provided online shop APIs. For example, late binding of products to the shop order or rule-based workflows can be seen as means of adaptivity. The aforementioned “intuitive” actions from humans are exchanged with “trust” in the provider of the market place. The user hence “trusts” that ordering 6 eggs, will result in the delivery of the correct quality and number.
3. **Physical Platform** – For the physical platform robot and sensor devices for the appropriate moves must be installed in order to pick up the provided products. Furthermore, these robot and sensor devices must be secured and work in a legal and ethical correct way. Here, a model-based approach can be supportive in the context of formal correct and verifiable sequences of actions that can be execute by workflow engines and robots in combination with sensors. Adaptivity can be included by means of late binding using rule-based workflows in conjunction with smart sensors. The huge challenge is to develop aforementioned human “intuitive” actions with sensors that e.g. count the eggs,

[1] Enlite.AI (24/01/2020). Innovation talk „AI in Retail“. At Retail Austria.

check the quality using image recognition, approve the code that is printed on the egg using image to text transformation or handle the egg box with care.

The so-called domain model describes independently of the underlying platform, the necessary steps, like that buying a box of egg requires to check the code, the quality, the number and to handle the box with care.

In order to transform a domain model towards an executing platform, various aspects must be considered. For instance, human interpretations, the level of formalisation, the resource allocation and various physical challenges.

The domain model was first specified using the BPMN syntax for business processes. This representation was exchanges also with a UML Flow Chart model as well as a fully specified Petri-Net. BPMN and UML Flow Chart had been found appropriate for modelling the domain specific aspects. Petri-Net was considered as too technical and hence more appropriate for developer.

On the social platform it was enough to add knowledge that is used to perform a BPMN task by a so-called note including e.g., a picture of an egg box pointing out how to check the code, the quality and the number. Whereas on the virtual platform rather a BPMN workflow with additional so-called “Service Level Agreement” (SLA) is required. Such SLA are commonly expressed in files that provide the contract – the Service Level Agreement – in a computer readable format, so that computer programmes can interpret the file and consequently agree or disagree – in our sample to buy or not to buy the box of eggs.

The physical platform requires additional information for sensors and knowledge-based algorithms. For instance, sensors for security, image recognition and text processing or other means of artificial intelligence are required to ensure that a box of eggs is carefully removed form the shelf, it is opened, the stamped code on the eggs is read and interpreted, the eggs are counted and it is checked if an egg is broken.

Ethical, legal, security and safety checks can be conducted for all of the different platforms. For example, ethical issues on a social platform could be the fact that the customer cannot check the eggs (e.g., elderly people may have difficulties to see), whereas on the virtual platform a service level agreement can be used which requires that the user is actually capably to understand which form of agreement was accepted. Technology supports the quality check on the physical level, for instance by using sensors and knowledge-based algorithms.

The aforementioned platforms can be realised in several different ways using different technologies. Hence, finally we talk about a platform-specific model, when a concrete robot or computer programme is addressed. In case a similar robot or computer programme is addressed, then we apply the so-called “abstraction”, that describe the same behaviour on a higher – a more abstract – layer but can be implemented differently. Those different abstraction layers have been worked out in form of different layers or workflows. Some are platform specific and approach a concrete robot platform, and some are abstract that describe a generic behaviour, which needs to be detailed in sub-workflows.

This document describes the abstraction layers including their platform dependency, the workflows for allocating resources and the created prototype package.

1.1 Relation to Work Package

According to the project plan, work package three is structured in two phases:

1. Specification of application scenarios, and
2. Demonstration of the model-based operation of robots.

This deliverable is related to the second phase. It builds upon D3.1 and covers the model-based prototype for operating robots.

1.2 Document Structure

Executive summaries in German as well as in English are provided.

This deliverable is structured in the following chapters: this introductory section provides the context of the work performed and results achieved. Chapter 2 introduces the abstraction layer concept applied within the prototype. Various workflow bindings in connection with different modelling languages are outlined in chapter 3. The prototype package and its usage are introduced in chapter 4. And finally, a concise conclusion in chapter 5 summarizes the results so far.

2. Overview of Model-Driven Robot Execution

As already mentioned, there are three platforms for the application scenarios – social, virtual and physical. However, not only various platforms, but also different underlying abstraction layers, presented in Figure 1, need to be considered in application scenarios like the supermarket domain.

Important questions when working with a robot are (1) is the robot safe or can it hurt people, (2) does the customer know the result, or (3) is artificial intelligence performing any decisions the customer is unaware of. With model-based approaches, we can describe separately the different levels of abstraction.

The project identified four abstraction layers that are introduced in Figure 1. These are (1) abstract workflow, (2) technical workflow, (3) platform specific abstraction, and (4) robotic platform, IoT adaptors and workflow engines. All of the layers are characterized by the abstraction level of operations they offer as well as by their dependency on a platform.

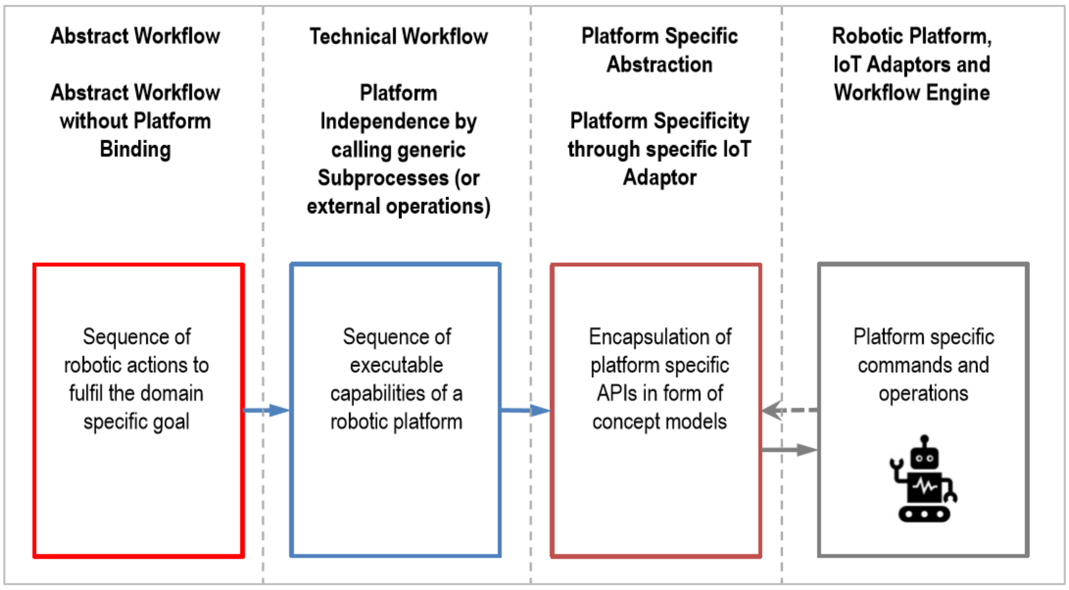

Abstract Workflow
Abstract Workflow without Platform Binding
Technical Workflow
Platform Independence by calling generic Subprocesses (or external operations)
Platform Specific Abstraction
Platform Specificity through specific IoT Adaptor
Robotic Platform, IoT Adaptors and Workflow Engine
Sequence of robotic actions to fulfil the domain specific goal
Sequence of executable capabilities of a robotic platform
Encapsulation of platform specific APIs in form of concept models
Platform specific commands and operations

Figure 1: Abstraction Layers for the Application Scenarios

The abstract workflow is without concrete so-called “binding of resources” and hence only represents a sequence of robotic actions in order to fulfil the goal. For instance, represented in business process modelling notation. This workflow is created in collaboration with a domain expert, as no particular technical know-how is needed but domain specific know-how is required. The technical workflow - that details the abstract workflow - remains platform independent by calling external operations or generic subprocesses, but introduces the technical details. It represents a sequence of executable capabilities of a robotic platform. The platform specific workflow instantiates the technical workflow with concrete commands and API interactions of a concrete technical platform. The exchange of an abstract command like “move to the rights b 90 degree” with a platform specific command “run right engine with speed 2 for 0,3 ms and run left engine with speed -2 for 0,2 ms” is considered as the resource allocation, hence the so-called binding of software artefacts to actions of the workflow. Those platform-specific commands are encapsulated in so-called IoT adaptors that provide platform specific APIs. Platform specific commands and operations are used on the robotic platform, the IoT adaptors and the workflow engine layer.

At all of the stages artificial intelligence might be identified, therefore the corresponding assessment of knowledge-based algorithms is required for each abstraction layer. The different layers may have different dependencies to legal, ethical, safety and security aspects. We assume that lower technical layers are more safety and security oriented, whereas higher more domain specific layers are more legally and ethically oriented.

We propose questionnaires that assess the different models on different abstraction levels with the corresponding legal, ethical, security or safety assessment and ensure via a trusted digital signature after completing the questionnaire that only assessed models are further used.

In the following sections, the possible integration of AI at the different abstraction layers is described, as well as the realisation of the different layers using different modelling languages are presented.

More details on the abstraction layers as well as more examples can be found in Appendix A.

2.1 Introduction AI

There are various means of AI integration, hence different AI technology are appropriate on different layers.

The abstracts workflow layer facilitates rather generic technologies. Simulation and/or optimization can be enabled by AI. This can happen on the model level for instance by using the business process modelling notation. On the technical workflow, as well as on the platform specific abstraction layer mechanisms for product and/or service allocation can be applied. For instance, ontologies or rule-based systems might support the processing not only in a platform independent, but also in a platform specific environment. On the robotic platform layer, sensors as well as services enable the usage of AI technologies like image recognition.

An overview of AI on the various abstraction layers can be found in Figure 2.

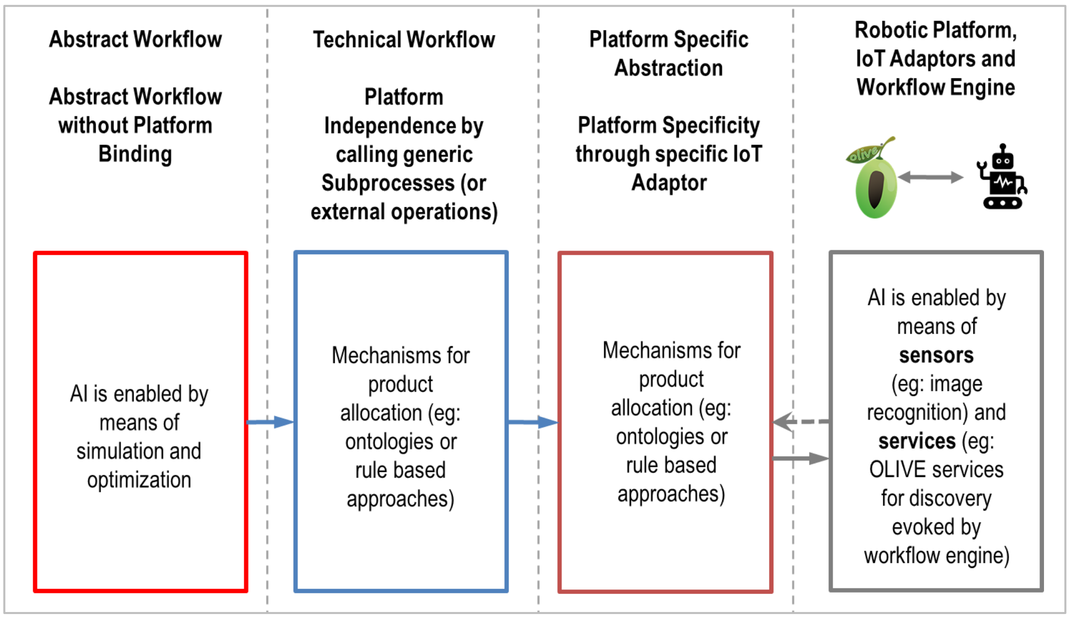

Abstract Workflow
Abstract Workflow without Platform Binding
Technical Workflow
Platform Independence by calling generic Subprocesses (or external operations)
Platform Specific Abstraction
Platform Specificity through specific IoT Adaptor
Robotic Platform, IoT Adaptors and Workflow Engine
olive
AI is enabled by means of simulation and optimization
Mechanisms for product allocation (eg: ontologies or rule based approaches)
Mechanisms for product allocation (eg: ontologies or rule based approaches)
AI is enabled by means of **sensors** (eg: image recognition) and **services** (eg: OLIVE services for discovery evoked by workflow engine)

Figure 2: AI Integration on Abstraction Layers

2.2 Using Petri Nets for Exeuction

The layer concept can be instantiated in different ways by using various modelling languages. Figure 3 shows how to instantiate the four layers – abstract workflow, technical workflow, platform specific abstraction and robotic platform – by using BPMN and Petri Nets. On the abstract workflow layer BPMN is used to model the domain scenario. This sample shows a process of selecting and collecting three fruits within a supermarket. For the technical workflow and the platform specific abstraction layer, we consider BPMN as the targeted workflow languages, however, we also see the Petri-Net notation as a proof-of-concept description that can be used, before finalising workflows on the concrete platforms.

However, in the following sample a separation can be conducted. As described in D3.1, a robot arm – in our case the so-called "Dobot Magician" – is used in the OMiLAB experiment space as a robotic actor.

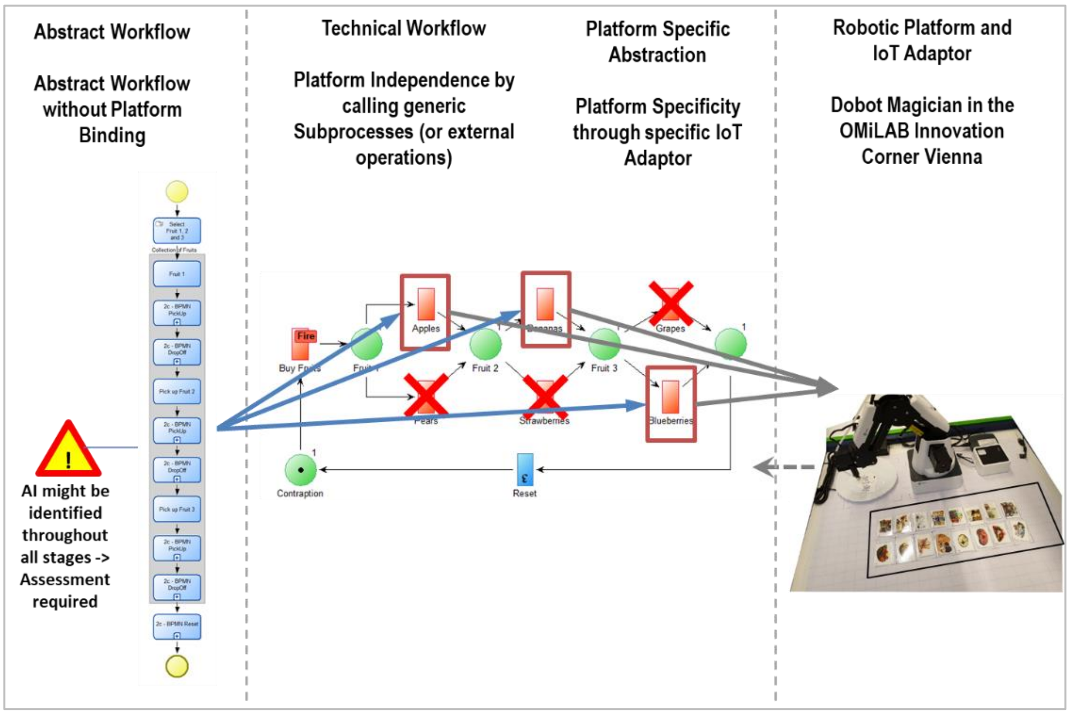

Abstract Workflow
Abstract Workflow without Platform Binding
Technical Workflow
Platform Independence by calling generic Subprocesses (or external operations)
Platform Specific Abstraction
Platform Specificity through specific IoT Adaptor
Robotic Platform and IoT Adaptor
Dobot Magician in the OMiLAB Innovation Corner Vienna
AI might be identified throughout all stages -> Assessment required
Buy Fruits
Apples
Bananas
Grapes
Pears
Strawberries
Blueberries
Fruit 2
Fruit 3
Contraption
Reset

Figure 3: Instantiation of the 4 Abstraction Layers using BPMN and Petri Nets

2.3 Using Flow Chart for Execution

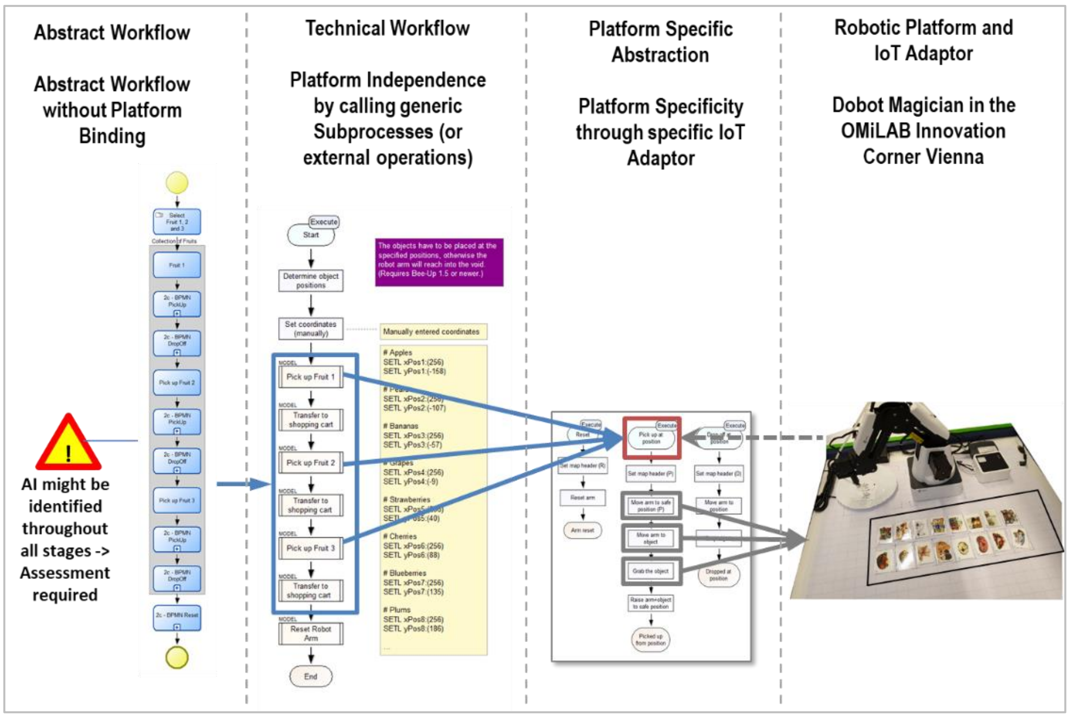

Abstract Workflow
Abstract Workflow without Platform Binding
Technical Workflow
Platform Independence by calling generic Subprocesses (or external operations)
Platform Specific Abstraction
Platform Specificity through specific IoT Adaptor
Robotic Platform and IoT Adaptor
Dobot Magician in the OMiLAB Innovation Corner Vienna
AI might be identified throughout all stages -> Assessment required
Start
Determine object positions
Set coordinates (manually)
Pick up Fruit 1
Transfer to shopping cart
Pick up Fruit 2
Pick up Fruit 3
Reset Robot Arm
End
Manually entered coordinates
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)

Figure 4: Instantiation of the 4 Abstraction Layers using BPMN and Flowchart

Figure 4 shows an example instantiation of the four abstraction layers by using BPMN and Flowcharts. In this sample, again the Dobot Magician in the OMiLAB is used as a robotic actor. The sample shows the selection and collection of three fruits within a supermarket. The IoT adaptor of the robot arm provides an interface, so that basic movements can be summarized in so called concept models that are required for the platform specific abstraction layer. For instance, such a concept model can be composed of the basic movements of moving to the right fruit and picking it up. Another concept model can be used to place the fruit in a shopping basket for instance. These concept models are then used in the technical workflow layer, which is platform independent by using them.

2.4 Using BPMN for Execution

Figure 5 shows an example instantiation of the four abstraction layers by using BPMN and workflows. In this sample, again the Dobot Magician in the OMiLAB is used as a robotic actor. As a sample workflow engine Camunda[2] is used. The workflow engine uses BPMN. Therefore, the notation in the technical workflow and the platform specific abstraction layer is BPMN. Apart from that, the instantiation with flowchart and workflows are similar. As above, the sample shows the selection and collection of three fruits within a supermarket. The IoT adaptor of the robot arm provides an interface, so that basic movements can be summarized in so called concept models that are required for the platform specific abstraction layer. For instance, such a concept model can be composed of the basic movements of moving to the right fruit and picking it up. Another concept model can be used to place the fruit in a shopping basket for instance. These concept models are then used in the technical workflow layer, which is platform independent by using them.

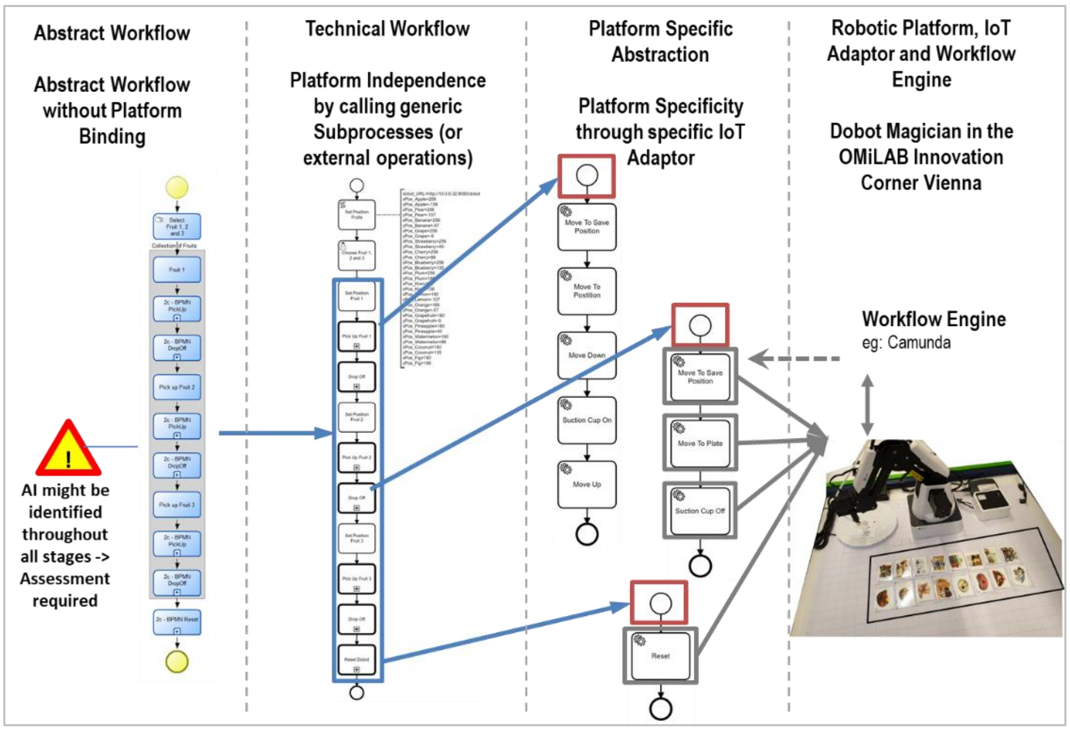

Abstract Workflow
Abstract Workflow without Platform Binding
Technical Workflow
Platform Independence by calling generic Subprocesses (or external operations)
Platform Specific Abstraction
Platform Specificity through specific IoT Adaptor
Robotic Platform, IoT Adaptor and Workflow Engine
Dobot Magician in the OMiLAB Innovation Corner Vienna
Workflow Engine
eg: Camunda
AI might be identified throughout all stages -> Assessment required
Move To Save Position
Move To Position
Move Down
Suction Cup On
Move Up
Move To Plate
Suction Cup Off
Reset

Figure 5: Instantiation of the 4 Abstraction Layers using BPMN and Workflows

[2] https://camunda.com/de/download/ (last visited on 02/11/2020)

3. Adaptive Workflows for Robot Exeuction

This section elaborates how to operate a robot with a model. In order to manage all of the above-mentioned aspect, workflows operating on robots are introduced. A workflow executes and orchestrates a sequence of SW APIs. In the context of robots each SW API is an IoT Adaptor that triggers a pre-programmed actuation – a certain move. The resource allocation or binding of a workflow is typically understood to link a certain SW API to a particular action of the workflow a particular IoT Adaptor – hence a certain move – is bound to a workflow task.

Following three bindings are used and presented in more detail. In particular, the pre- and late binding may apply AI technologies for operating robots.

- **Fixed Binding** – The workflow and all of the moves are predefined, when the user selects the workflow, the user gets exactly the sequence of moves.
- **Pre-Binding:** The workflow and all of the moves are decided just before starting. This allows to react in case a certain move – picking up of a particular object – is not possible and allows to consider alternatives – picking up another object instead.
- **Late Binding:** The workflow and all of the moves are decided just while the workflow is executed. This allows to react on situations just before the actual move is performed. This principle is known from adaptive workflows or self-healing systems. We consider this approach as a promising candidate for autonomous behaviour.

In the following subsections, we show the setting of interconnections between models and robotic platforms and how robots can be operated via workflow models by using the three proposed workflow bindings. More details on the abstraction layers as well as more examples can be found in Appendix A.

3.1 Laboratory Setting

Figure 6 shows how models, robotic platforms and workflow engines are interconnected within a lab and among different locations.

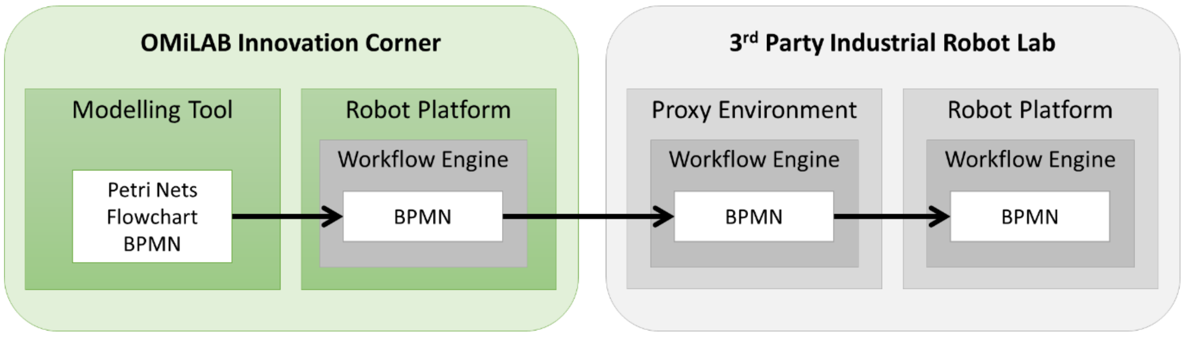


Figure 6: Interconnection of Settings

Within the OMiLAB, there are various modelling tools that allow to model a specific scenario in form of workflows by means of various modelling languages such as Petri Net, Flowchart or BPMN. Based on the above-mentioned supermarket scenario, a domain specific assembly scenario can be modelled. For instance, a dairy product producer might package pallets and boxes with single yoghurt cups. A pallet might be predefined with 20 boxes a 5 blueberry, 5 strawberry and 10 natural yoghurts or it can be packed individually based on order parameters.

In a first step, such a scenario is defined on a high level with Petri Nets and/or flowcharts. Different variants can be defined by applying the three workflow binding scenarios. Furthermore, the basic domain workflows can be refined by using BPMN and lifting the models to a workflow engine in order to bring those models closer to the robotic platform. These first workflows require sensor information – for example about product availability – and decisions might be necessary, especially in case of missing products.

In a second step, the workflows from the OMiLAB are transformed to a 3rd party industrial robot. Whereas the BPMN on the OMiLAB side is rather generic and abstract, the BPMN for the environment must be more specific and concrete. Such a proxy environment architecture allows the processing of an industrial robot within a secured setting. The BPMN is not executed with a workflow engine on the industrial robotic platform if the steps before are not considered to be safe and secure. Notice that this architecture implicitly includes the above-mentioned abstraction levels, which are required to differentiate between the quite different aspects and concerns on each layer.

3.2 Fixed Workflows: Fixed Binding of Robots Execution

In a **fixed binding** workflow, the workflow itself and all the moves are predefined. When the user selects the workflow, the user gets exactly the sequence of moves, which was defined. No intervention during the execution is possible. Figure 7 shows an example of a fixed binding workflow by using Petri Nets. In this example, apples, bananas and grapes are picked up in this sequence. As in a fixed binding workflow, all decisions are taken beforehand, it is not possible to react to any missing products. So, if there are no apples available, the robot will reach into void or an erroneous behaviour. The Petri Net requires step by step user interaction for the execution, as each transition must be fired individually. As it is clearly a problem that the robot reaction cannot be influenced in case of empty products, we propose to use artificial intelligence to overcome this issue. More details and example models for fixed binding can be found in Appendix B.

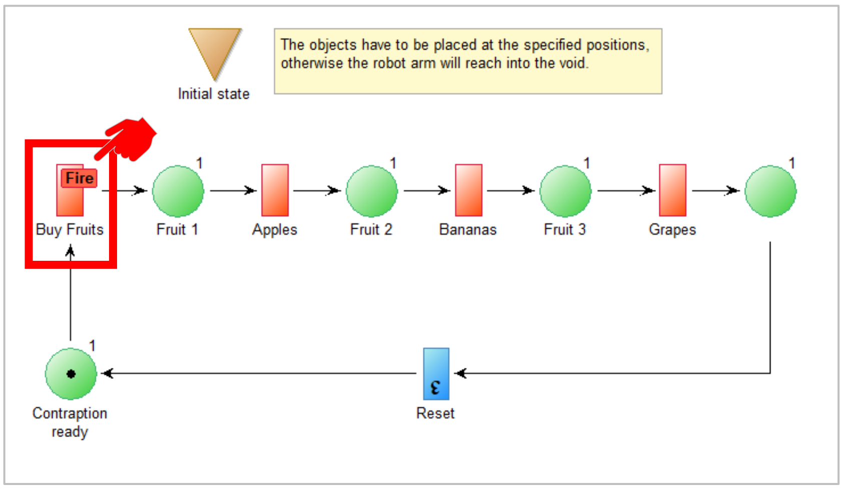
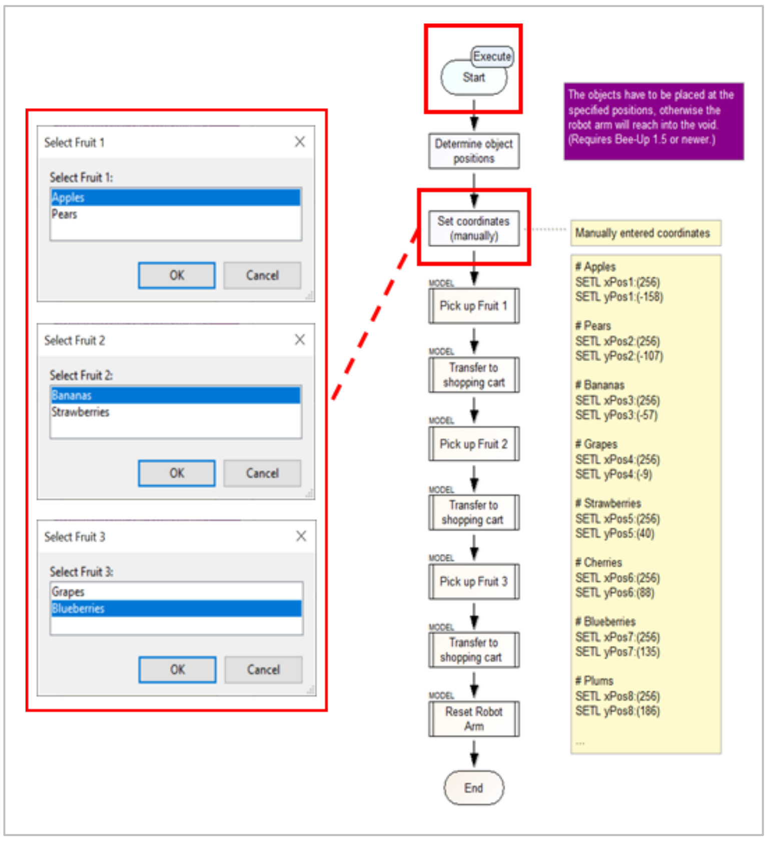

Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Fire
Buy Fruits
1
Fruit 1
Apples
1
Fruit 2
Bananas
1
Fruit 3
Grapes
1
1
Contraption ready
ε
Reset

Figure 7: Instantiation of a fixed binding workflow by using Petri Nets

3.3 Alternative Workflows: Pre-Binding of Robot Execution

In contrary, in a **pre-binding** workflow, the workflow and all the moves are decided just before starting. This allows to react in case a certain move – for instance picking up of a particular object is not possible. It allows the consideration of alternatives – for example picking up another object instead. Intervention at the beginning is possible, whereas it is not possible to interfere during the execution. Figure 8 visualizes a pre-binding workflow by using Flowcharts. Again, we have three objects – in our case apple, bananas and grapes and these fruits are picked up in this sequence. In case there are not enough apples, bananas and grapes available, they are substituted by pears, strawberries and blueberries. In the pre-binding workflow, the product availability for all fruits is checked at the beginning. In case fruits need to be exchanged, there are several AI algorithms that support his decision making. We proposed either a rule-based, or a semantic based matching algorithm. In our sample apples and bananas are available. Grapes are not available and therefore Blueberries are taken. In our sample, we emulated a rule-based approach with a user interaction. Artificial intelligence can be applied on different abstraction layers. In this sample artificial intelligence is applied on the technical workflow layer. More details about the abstraction layers and the value for each layer can be found in section **Error! Reference source not found. Error! Reference source not found.**. Moreover, additional information and example models for pre binding can be found in Appendix C.

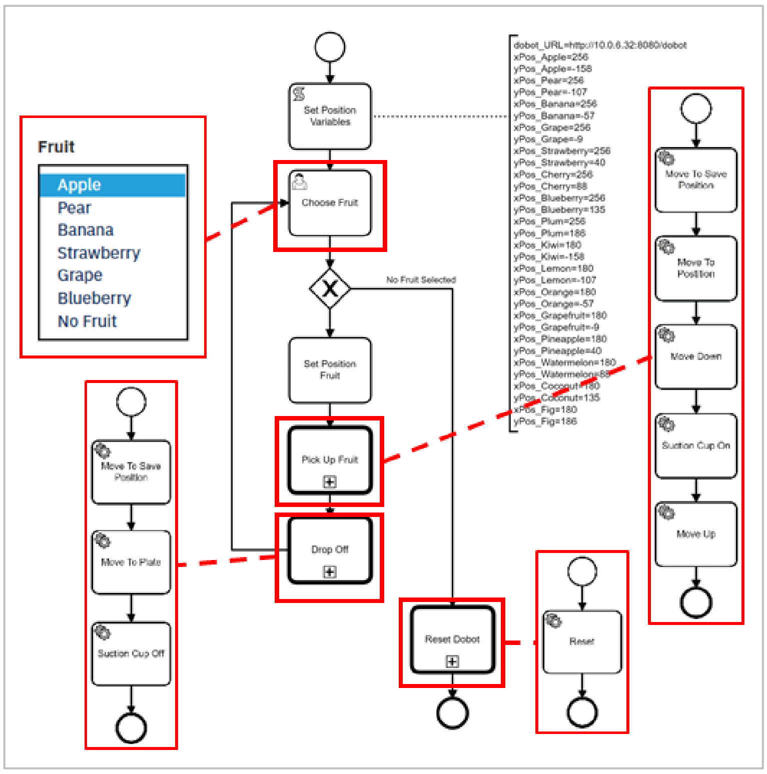

Select Fruit 1
Select Fruit 1:
Apples
Pears
OK
Cancel
Select Fruit 2
Select Fruit 2:
Bananas
Strawberries
OK
Cancel
Select Fruit 3
Select Fruit 3:
Grapes
Blueberries
OK
Cancel
Execute
Start
Determine object positions
Set coordinates (manually)
MODEL
Pick up Fruit 1
MODEL
Transfer to shopping cart
MODEL
Pick up Fruit 2
MODEL
Transfer to shopping cart
MODEL
Pick up Fruit 3
MODEL
Transfer to shopping cart
MODEL
Reset Robot Arm
End
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Manually entered coordinates

```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

...
```

Figure 8: Instantiation of a pre binding workflow by using flowcharts

3.4 Adaptive Workflows: Late-Binding of Robot Execution

The **late-binding** workflow and all the moves are decided just while the workflow is executed. This allows to react on situation just before the actual move is performed. A sample for a late binding workflow by using BPMN can be found in Figure 9. In the late binding workflow, the product availability is checked each time a fruit is chosen. Artificial intelligence can be integrated in the same way as before, when checking the possibility of a certain move. In this sample artificial intelligence can be applied on the robotic platform layer by means of services. Sensor information can be provided to the workflow engine, which invokes a service from the Microservice Framework OLIVE. More details and example models for late binding can be found in Appendix D.



```
dobot_URL=http://10.0.6.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
xPos_Fig=180
yPos_Fig=186
```

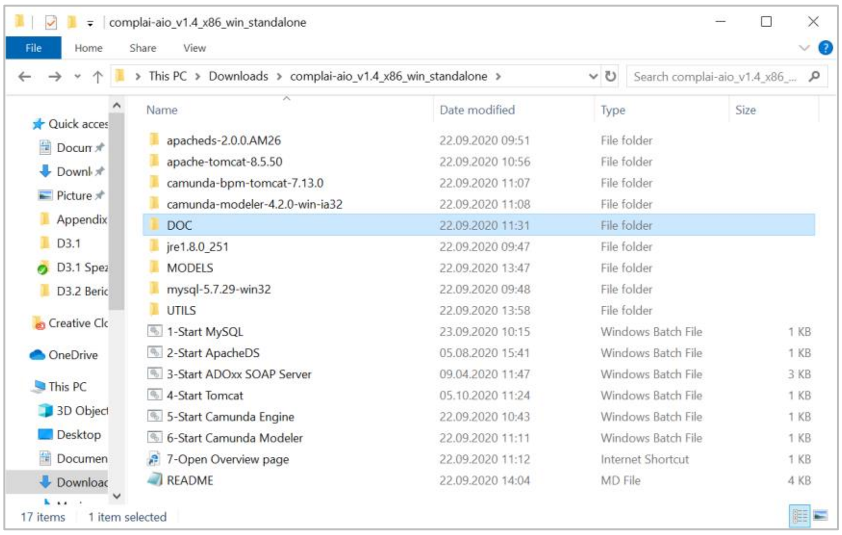
Figure 9: Instantiation of a late binding workflow by using BPMN and a workflow engine

4. OMiLAB – Proof-of Concept

This subchapter provides some more details about the prototype package. First, the prerequisites as well as the first steps are outlined. Afterwards, it is explained how the model-based operation of robots can be tested.

4.1 Installation of Proof-of-Concept Package in OMiLAB

First, the prototype package must be downloaded from here: compl@i prototype package. Notice that the package must be unzipped with 7Zip[3]. The unzipped package includes various libraries, models as well as executable files, see Figure 10: Prototype Package Structure. In general, users use the files in the DOC and the MODELS folder as well as the files numbered from 1 to 7.



complai-aio_v1.4_x86_win_standalone

This PC > Downloads > complai-aio_v1.4_x86_win_standalone >

Search complai-aio_v1.4_x86_...

Name	Date modified	Type	Size
apacheds-2.0.0.AM26	22.09.2020 09:51	File folder	
apache-tomcat-8.5.50	22.09.2020 10:56	File folder	
camunda-bpm-tomcat-7.13.0	22.09.2020 11:07	File folder	
camunda-modeler-4.2.0-win-ia32	22.09.2020 11:08	File folder	
DOC	22.09.2020 11:31	File folder	
jre1.8.0_251	22.09.2020 09:47	File folder	
MODELS	22.09.2020 13:47	File folder	
mysql-5.7.29-win32	22.09.2020 09:48	File folder	
UTILS	22.09.2020 13:58	File folder	
1-Start MySQL	23.09.2020 10:15	Windows Batch File	1 KB
2-Start ApacheDS	05.08.2020 15:41	Windows Batch File	1 KB
3-Start ADOxx SOAP Server	09.04.2020 11:47	Windows Batch File	3 KB
4-Start Tomcat	05.10.2020 11:24	Windows Batch File	1 KB
5-Start Camunda Engine	22.09.2020 10:43	Windows Batch File	1 KB
6-Start Camunda Modeler	22.09.2020 11:11	Windows Batch File	1 KB
7-Open Overview page	22.09.2020 11:12	Internet Shortcut	1 KB
README	22.09.2020 14:04	MD File	4 KB

17 items 1 item selected

Figure 10: Prototype Package Structure

Furthermore, ADOxx must be downloaded and installed[4]. For this reason, a personal licence key is required that can be requested for free in the context of academic purposes on the webpage. Afterwards all required libraries (for the modelling tools/languages) must be imported in the ADOxx development toolkit and the corresponding prepared models must be imported in the ADOxx modelling toolkit. BPMN, Bee-Up and the Questionnaire library must be imported. The BPMN library and models are available inside the package folder MODELS/BPMN. More details for Bee-Up and the standalone version can be found online[5]. The questionnaire library and the models are available inside the folder MODELS/QUESTIONNAIRES. Notice that the Questionnaire library is required for performing the ethical, legal, security and safety assessments. Additional information is provided on importing libraries in ADOxx (credentials: Admin / password), creating a new user and importing the models in ADOxx at "import a library", "create a user" and "import models".

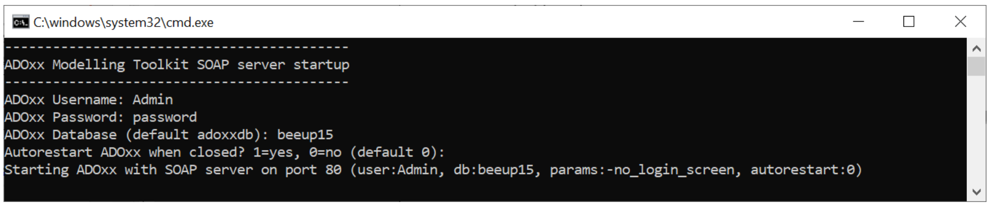
In order to avoid parallel allocation of the robotic devices in the OMiLAB Innovation Corner, the project required that each robot item can be reserved via an online service provided by OLIVE[6]: OMiLAB Innovation Corner - Device Reservation. More details on how to reserve such a device can be found here: Reservation Manual for OMiLAB Innovation Corner Devices.

[3] https://www.7-zip.org/ (last visited on 02/11/2020)
[4] https://www.adoxx.org/live/download-guided (last visited on 02/11/2020)
[5] https://austria.omilab.org/psm/content/bee-up/info (last visited on 02/11/2020)
[6] https://www.adoxx.org/live/olive (last visited on 02/11/2020)

4.2 Starting of Proof-of-Concept Package in OMiLAB

In order to start the portal, following steps must be executed:

1. Execute the "1-Start MySQL.bat" file in the package folder
2. Execute the "2-Start ApacheDS.bat" file in the package folder
3. Execute the "3-Start ADOxx SOAP Server.bat" in the package folder
    a. insert the credentials for the created username and password for the ADOxx modelling toolkit, use the default database if you did not select a specific name during the ADOxx installation process.
    b. or “Admin”, “password” and “beeup15” if you prefer the Bee-Up standalone version, see Figure 11



```
C:\windows\system32\cmd.exe
--------------------------------------------
ADOxx Modelling Toolkit SOAP server startup
--------------------------------------------
ADOxx Username: Admin
ADOxx Password: password
ADOxx Database (default adoxxdb): beeup15
Autorestart ADOxx when closed? 1=yes, 0=no (default 0):
Starting ADOxx with SOAP server on port 80 (user:Admin, db:beeup15, params:-no_login_screen, autorestart:0)
```

Figure 11: Starting Bee-Up as Standalone

4. Execute the "4-Start Tomcat.bat" file in the package folder
5. Execute the "5-Start Camunda Engine.bat" file in the package folder
6. Execute the "6-Start Camunda Modeler.bat" file in the package folder
    a. open the .bpmn files from the folder MODELS/BPMN in the modeler
    b. deploy all of the subprocesses, before one of the main workflows is started
    c. notice that a user task at the beginning is used to insert the reservation token of the device
7. Open the link "7-Open Overview page" in the package folder
    a. prefer Chrome for using the dashboard webpage
    b. go to the “Workflow Engine” tab, the credentials are “demo” / “demo” (see Figure 12)
    c. notice that the tabs “Model Questionnaire” and “ Model Signature” are not needed in this prototype for model-based operation of robots, however, they will be needed in deliverable related to assessment
8. Finally, when you are finished with operating the robots, terminate the execution closing the command console opened in point 1, 2, 3, 4, 5 and 6 (Ctrl+C) as well as the dashboard webpage

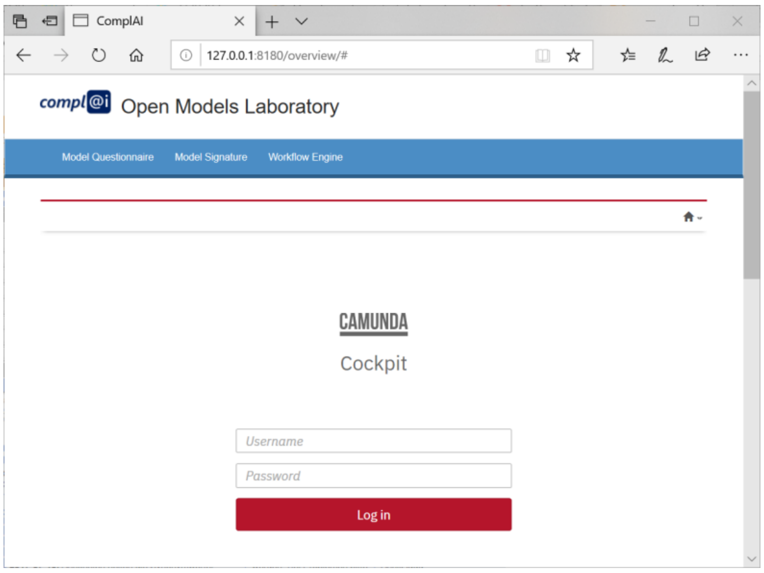


Figure 12: Operating Robots with Workflows

4.3 Demonstration of Model-based Robot Execution in OMiLAB

In particular, we used the following robots:

1. Dobot Magician – robot arm
2. Makeblock mBot – mobile platform

Furthermore, models in various modelling languages with different workflow bindings are provided. Petri Nets as well as Flowcharts can be executed directly in ADOxx by using the Bee-Up library. In contrast, the workflows in BPMN must be executed by using a 3rd party workflow engine.

The different modelling language and workflow binding combinations are presented in short sample videos:

1. Fixed Binding (see Appendix B)
   a. Process with fixed binding using Petri Net
   b. Process with fixed using Flowchart
   c. Process with fixed binding using BPMN
2. Pre-Binding (see Appendix C)
   a. Process with pre binding using Petri Net
   b. Process with pre binding using Flowchart
   c. Process with pre binding using BPMN
3. Late-Binding (see Appendix D)
   a. Process with late binding using Petri Net
   b. Process with late binding using Flowchart
   c. Process with late binding using BPMN

The videos demonstrate the different workflows addressing the robot arm Dobot Magician. The corresponding IoT Adapter – Raspberry-Pi – and corresponding SW – Tomcat Web-Application connected with the Dobot Magician interfaces – are provided. The pre-installed modelling toolkit Bee-Up is used for modelling the Petri Net, the flowchart and the BPMN processes that accesses the IoT Adapter. A 3rd party workflow engine was used. As there is currently no interface between Bee-Up and the 3rd party workflow engine, the BPMN models must be remodelled in the workflow engine modeller before execution.

Similar to the videos demonstrating the use of workflow to steer the robot arm, such workflows can also be used to steer mobile platform. The mobile robot can be seen as a shopping cart that drives through the supermarket and collects products at various stations. The different abstractions of the workflow are equally applicable for the mobile robot.

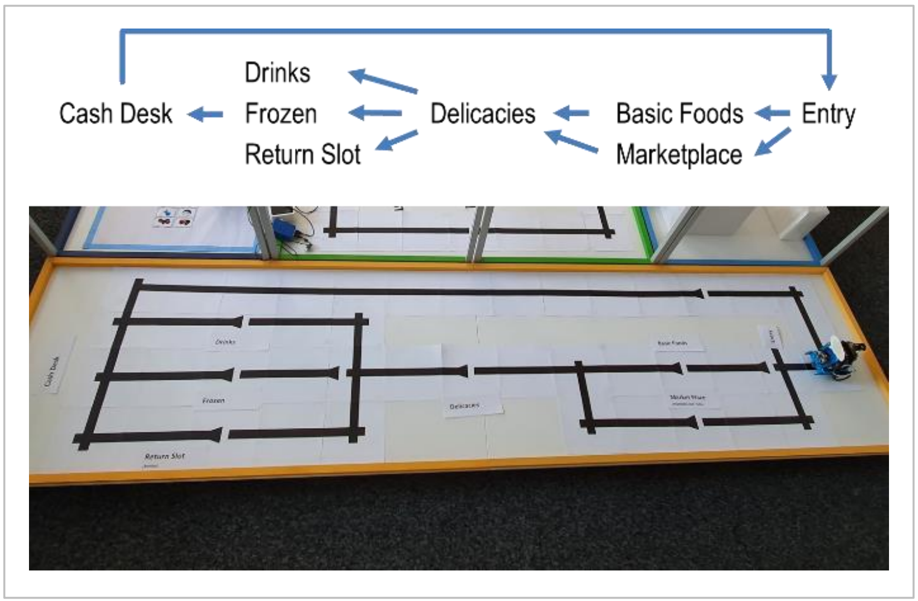


Figure 13: Mobile Platform Environment

The proof-of-concept environment, where robots are accessed by the use of models the project offers two settings: (1) the physical visit in the OMiLAB Innovation Corner, or (2) accessing the OMiLAB Innovation Corner remotely. Such a remote scenario setting looks as follows in Figure 14.

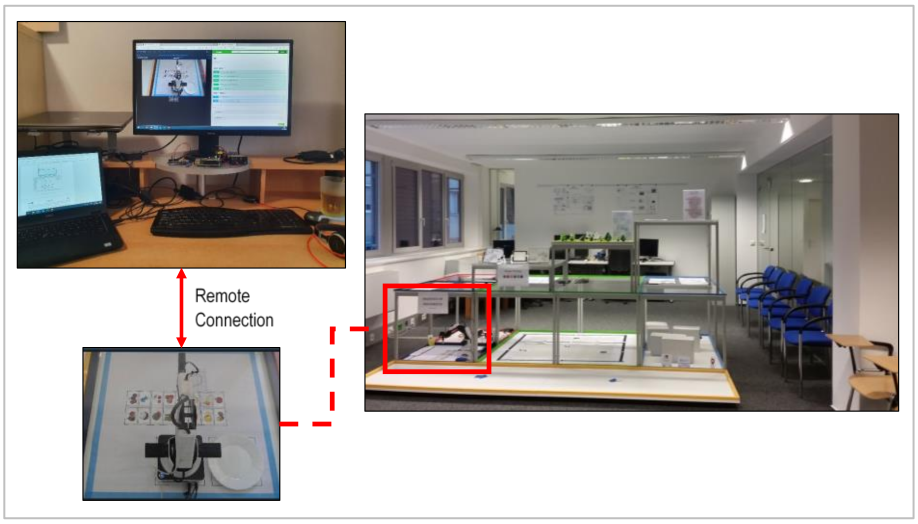
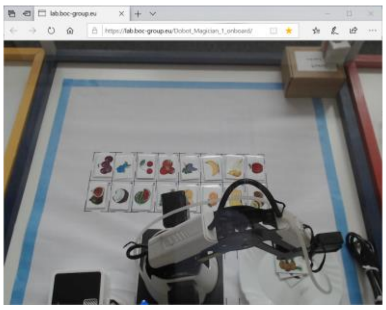
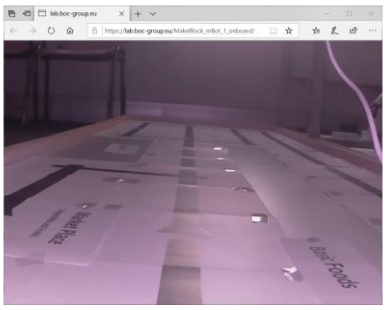
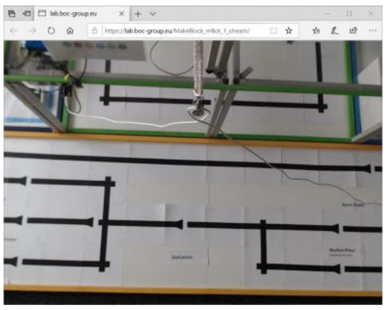


Figure 14: Proof-of-Concept Robots accessed by Remote Setting

In order to ease the testing and operating of robots via models in a remote setting, following live streams provide support.

Dobot Magician Live Stream	Makeblock mBot Live Stream - onboard	Makeblock mBot Live Stream - environment
		

All ADOxx/Bee-Up models can be executed, however, executing the workflows is not as trivial and requires basic knowledge about the workflow and the workflow engine. Therefore, the project provides more details in specific showing how to execute workflows using a workflow engine[7].

In case of running the models via a remote setting there is a task at the beginning, where the user needs to register the reservation token. In general, more detailed documentation can be found in the prototype package in the DOC folder. In case the models should be changed the integration of the „Reservation and Virtual Access Service“ is provided here: integration of device reservation service in Bee-Up models.

[7] https://adoxx.org/live/web/complai/downloads (last visited on 02/11/2020)

5. CHIMERA Prototype

After the proof-of-concept of the executable models within the aforementioned OMiLAB, the workflows have been extended to also steer real-world industrial robots. In the following the sample workflows, the corresponding robot behaviour and the used software stack is introduced.

The robot used in our use-case is a mobile manipulator called CHIMERA (shown in Figure 1). CHIMERA is a mobile manipulator consisting of the mobile industrial platform MiR100 and the industrial robot arm UR10. It is intended for applications like intralogistics, machine tending or location-independent manipulation. Both robots are individual sub-systems that are integrated using dedicated computing hardware and combined in a shared software environment (see below). However, in hard- and software, we designed CHIMERA for the base and the arm to be exchangeable with other robots. To integrate the two robots both physically and logically we employ additional network hardware, an industrial PC as well as PLCs. Apart from that, a separate battery system ensures the endurance for complex industrial applications. The software stack—as described below—runs on the industrial PC communicating with MiR and UR via the internal network. From the software perspective on application layer, CHIMERA provides workflows for moving the arm and for changing the robots location using the mobile base via the API.

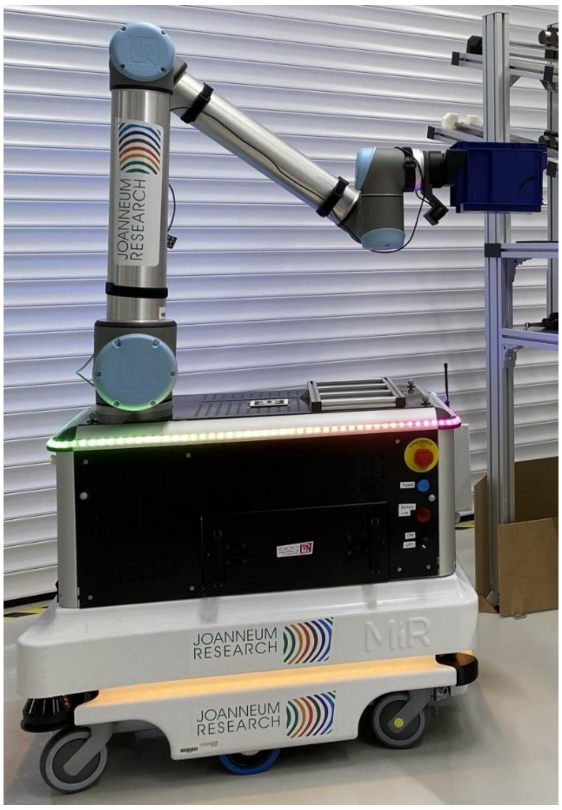


Figure 1: The CHIMERA robot

A video demonstrating an industrial intralogistic use case executed on the CHIMERA can be found at:

https://youtu.be/5w_GUe2lK4Y

5.1 Chimera Software stack

As basis of our use-case, we use our internal software stack further developed from what our group presented in[8]. It is a layered collection of re-usable components that can be applied to various robots and applications (see Figure 2). The bottom layer defines drivers for individual robots and other devices like sensors. In the case of CHIMERA, there are drivers for the UR arm and the MiR robot base (we can either choose a ROS-based driver or a driver that uses the MiR REST interface). Above that, the next layer defines various modules for integration of devices like one for whole-body compliance as presented in[9],[10] or sensor fusion and advanced motion planning.

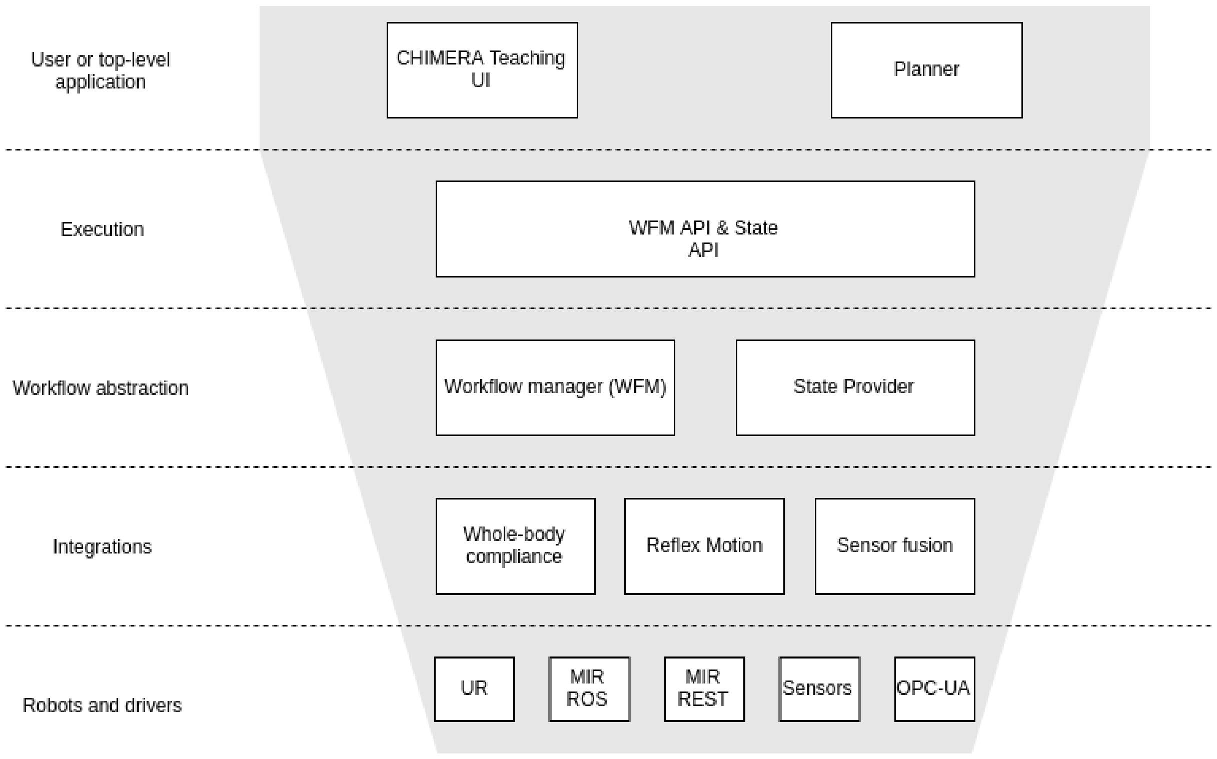


Figure 2: Software stack running on the CHIMERA robot

The workflow manager on the workflow abstraction layer orchestrates the execution of pre-defined sequences of actions. The state provider on the same layer collects information of the internal components and provides that to all other components. Both components are exposed to external entities by REST-based APIs on the Execution layer (the workflow manager API and the state API). The workflow manager API is a re-usable module that can be configured according to the capabilities of the underlying robot system. In the context of the CHIMERA robot, it allows for e.g., triggering movements of the mobile base or the robot arm (in joint and cartesian spaces) as well as support functions like the detection of AR tags using the wrist camera. The API provides asynchronous handling methods by returning an ID for each triggered workflow and the ability to query the status of such a workflow as it is being executed. Similarly, the state API adapts to the state data available from the robot system and exposes this data to the outside. Above that, components like UIs or planners can be used to (i) instrument single robots and (ii) coordinate fleets of robots. Security-wise, the

[8] Haspl, T.; Breiling, B.; Dieber, B.; Pichler, M.; Breitenhuber, G. Flexible industrial mobile manipulation: A software perspective. In Proceedings of the OAGM & ARW Joint Workshop 2019, Steyr, Austria, 9–10 May 2019. doi:10.3217/978-3-85125-663-5-10.
[9] Weyrer, M.; Brandstötter, M.; Mirkovic, D. Intuitive Hand Guidance of a Force-Controlled Sensitive Mobile Manipulator. In Proceedings of the IFToMM Symposium on Mechanism Design for Robotics, Udine, Italy, 11–13 August 2019; Gasparetto, A., Ceccarelli, M., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 361–368.
[10] Weyrer, M.; Brandstötter, M.; Husty, M. Singularity Avoidance Control of a Non-Holonomic Mobile Manipulator for Intuitive Hand Guidance. Robotics 2019, 8, 14, doi:10.3390/robotics8010014.

software stack is configured and deployed according to our recommendations published in[11]. The components below the execution layer are not accessible from outside the stack itself. Thus, the API provides a well defined access channel to the robot which is secured using encrypted communication via https as well as API tokens for authorization. As we mentioned above, sealing ROS applications is more difficult. However, within our software stack, we are able to securely do so. We already presented an architecture for how to securely modularize robotic applications including ROS modules. Our software stack adheres to this concept. We can also use ROS in our stack as individual modules e.g., on the driver level. Above we provide workflow abstractions of individual ROS actions, services and publications to make them accessible in the workflow manager. The abstraction in combination with security measures also enables securing ROS applications with our concept.

5.2 How to use and setup the stack

- Start with docker-compose: `docker-compose up`
- Access WFM-API webinterface via http://127.0.0.1:8081
- Access UR simulation webinterface via http://127.0.0.1:8080
- Stop and shutdown with docker-compose: `docker-compose down`

5.2.1 WorkflowManager API

When you successfully started the *WorkflowManager API* you should see the following output:

```
...

wfm-api    | Now listening on: http://0.0.0.0:8081

wfm-api    | Application started. Press Ctrl+C to shut down.
```

5.2.2 URsim

The robot simulator for Universal robots will run on port 8080.

You have to boot and start the robot first. On the lower left corner is a state indicator that will be red initially and showing "power off". Click it, then in the next window click On. When the button changes to "start", click it again.

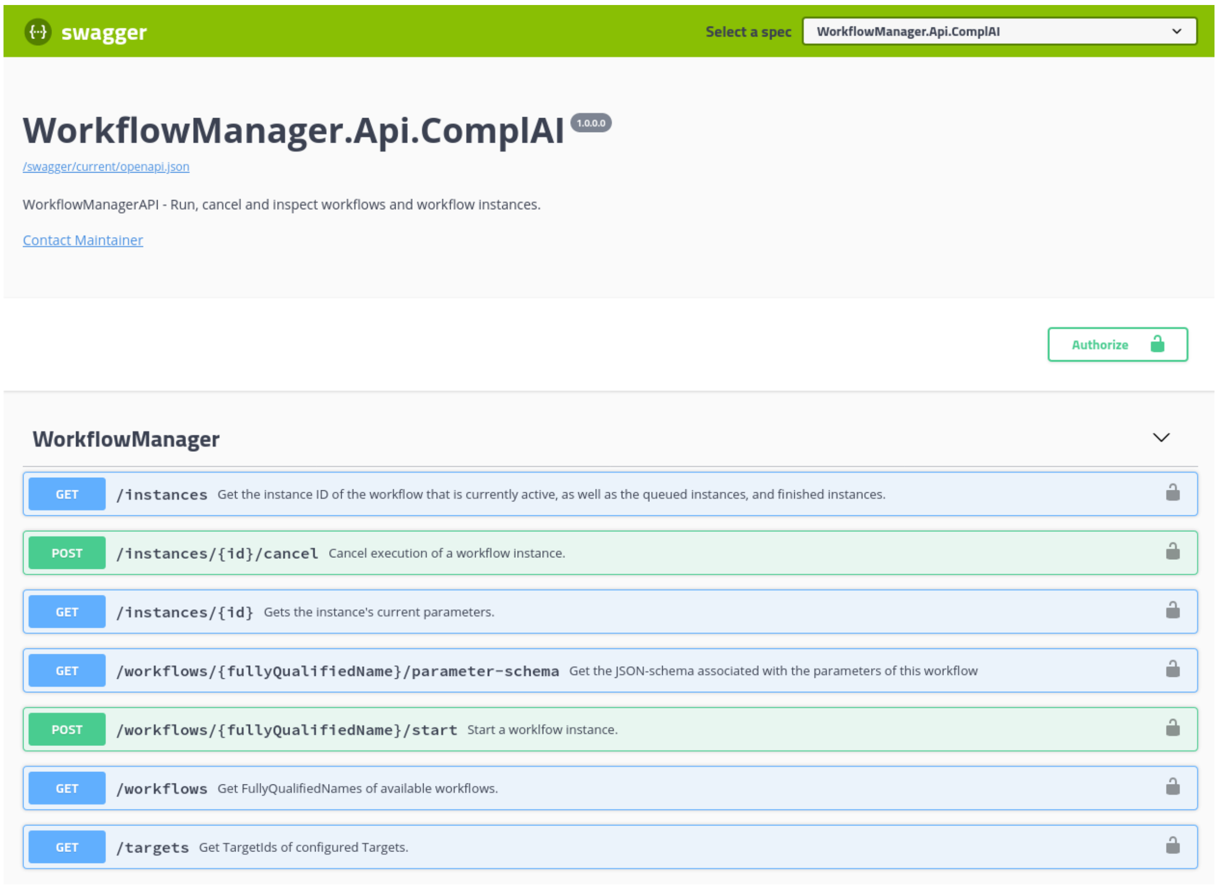
Press "Exit" and on the top row of symbols, click "Move" to see the robot.

5.2.3 Webinterface

The webinterface can be used to interact with the WorkflowManager. To access the webinterface of the WorkflowManager API, open http://127.0.0.1:8081 in your browser.
The webinterface provides:

- the possibility to directly use the WorkflowManager with the available API methods
- the openapi.json file to generate an API client for your preferred programming language

[11] Dieber, B.; Breiling, B. Security considerations in modular mobile manipulation. In Proceedings of the 3rd International Conference on Robotic Computing, Naples, Italy, 25–27 February 2019; IEEE: Naples, Italy, 2019; pp. 70–77, doi:10.1109/IRC.2019.00019.


swagger
Select a spec WorkflowManager.Api.ComplAI

WorkflowManager.Api.ComplAI 1.0.0.0
/swagger/current/openapi.json

WorkflowManagerAPI - Run, cancel and inspect workflows and workflow instances.

Contact Maintainer

Authorize

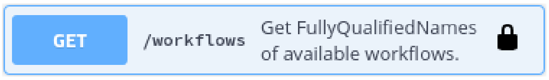
WorkflowManager

GET	/instances	Get the instance ID of the workflow that is currently active, as well as the queued instances, and finished instances.
POST	/instances/{id}/cancel	Cancel execution of a workflow instance.
GET	/instances/{id}	Gets the instance's current parameters.
GET	/workflows/{fullyQualifiedName}/parameter-schema	Get the JSON-schema associated with the parameters of this workflow
POST	/workflows/{fullyQualifiedName}/start	Start a worklfow instance.
GET	/workflows	Get FullyQualifiedNames of available workflows.
GET	/targets	Get TargetIds of configured Targets.

5.2.4 Listing all available workflows

1. First, you need to authorize yourself by clicking the button below and entering a valid API-key.

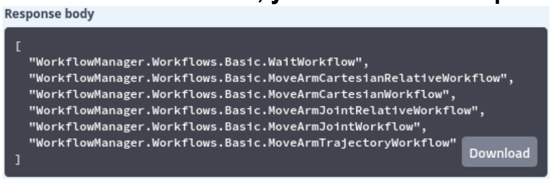
2. Then you can use any of the available API methods. E.g. If you want to list all available workflows, click on the **GET /workflows** button in the webinterface, like shown below



3. Then click on **Try it out**

4. And **Execute**

5. **After a successful call, you can see the response of the API method in the response body.**

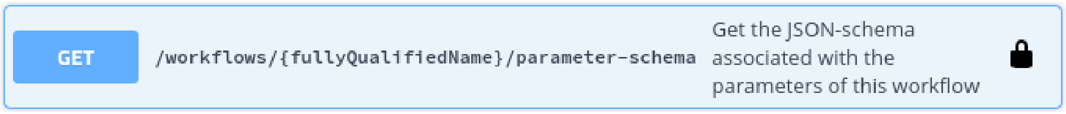

Response body

```
[
  "WorkflowManager.Workflows.Basic.WaitWorkflow",
  "WorkflowManager.Workflows.Basic.MoveArmCartesianRelativeWorkflow",
  "WorkflowManager.Workflows.Basic.MoveArmCartesianWorkflow",
  "WorkflowManager.Workflows.Basic.MoveArmJointRelativeWorkflow",
  "WorkflowManager.Workflows.Basic.MoveArmJointWorkflow",
  "WorkflowManager.Workflows.Basic.MoveArmTrajectoryWorkflow"
]
```

5.2.5 How to start a workflow

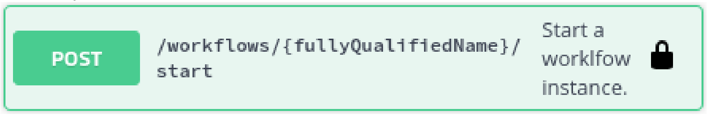
1. First, you need to authorize yourself by clicking the button below and entering a valid API-key.

2. Then you need to find out what parameters the workflow needs to run successfully. Therefore click on the **GET /workflows/{fullyQualifiedName}/parameter-schema** button and provide the full qualified name of the workflow.


GET /workflows/{fullyQualifiedName}/parameter-schema Get the JSON-schema associated with the parameters of this workflow

The response body contains the **properties** section, which describes the parameters of the workflow as JSON-schema.

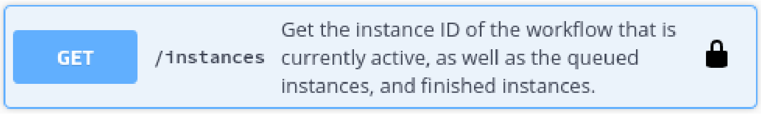
3. Start a workflow with the button **GET /workflows/{fullyQualifiedName}/start**. Therefore, you have to provide the fully qualified name of the workflow and the workflow parameters as json.


POST /workflows/{fullyQualifiedName}/start Start a worklfow instance.

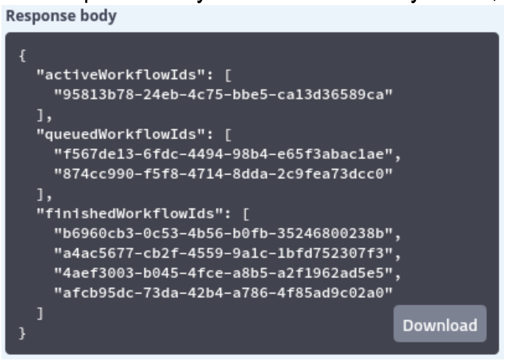
5.2.6 How to check if workflows are running

1. First, you need to authorize yourself by clicking the button below and entering a valid API-key.

2. Use the **GET /instances** method to list all workflow instances.


GET /instances Get the instance ID of the workflow that is currently active, as well as the queued instances, and finished instances.

The response body shows the currently active, as well as the queued, and finished instances.


Response body

```
{
  "activeWorkflowIds": [
    "95813b78-24eb-4c75-bbe5-ca13d36589ca"
  ],
  "queuedWorkflowIds": [
    "f567de13-6fdc-4494-98b4-e65f3abac1ae",
    "874cc990-f5f8-4714-8dda-2c9fea73dcc0"
  ],
  "finishedWorkflowIds": [
    "b6960cb3-0c53-4b56-b0fb-35246800238b",
    "a4ac5677-cb2f-4559-9a1c-1bfd752307f3",
    "4aef3003-b045-4fce-a8b5-a2f1962ad5e5",
    "afcb95dc-73da-42b4-a786-4f85ad9c02a0"
  ]
}
```

5.3 Workflows

The following workflows are available:

- WorkflowManager.Workflows.Basic.WaitWorkflow
- WorkflowManager.Workflows.Basic.MoveArmCartesianRelativeWorkflow
- WorkflowManager.Workflows.Basic.MoveArmCartesianWorkflow
- WorkflowManager.Workflows.Basic.MoveArmJointRelativeWorkflow
- WorkflowManager.Workflows.Basic.MoveArmJointWorkflow

- WorkflowManager.Workflows.Basic.MoveArmTrajectoryWorkflow

but the following are not working:

- WorkflowManager.Workflows.Basic.MoveArmCartesianRelativeWorkflow
- WorkflowManager.Workflows.Basic.MoveArmJointRelativeWorkflow

An instruction video for setting up and using the environment can be found at:

https://youtu.be/jK-HEwzZqdI

6. Conclusion

This deliverable shows how to operate robots by using model-based approach.

Three types of platforms – social, virtual and physical – were presented to explain the different requirements for knowledge-based algorithms. Algorithms that are interpreted by humans can be satisfactorily expressed by semi-formal models, whereas algorithms that are interpreted by machines requires the expression of well-formalised models.

Independent of the executing platform, different abstraction layers introduce the separation of concerns, which is needed to assess according ethical, legal, secure or safety perspectives.

The project identified the following abstractions layers:

(1) abstract workflow,

(2) technical workflow,

(3) platform specific abstraction, and

(4) robotic platform, IoT adaptors and workflow engines.

All abstractions layers are described by sample workflows and proof-of-concept robot realisation are presented.

In order to technically realise the autonomous behaviour different forms or workflow bindings are required. Those workflow bindings enable autonomous or smart behaviour by different "resource allocation", which is understood as different invocations of IoT Adapters.

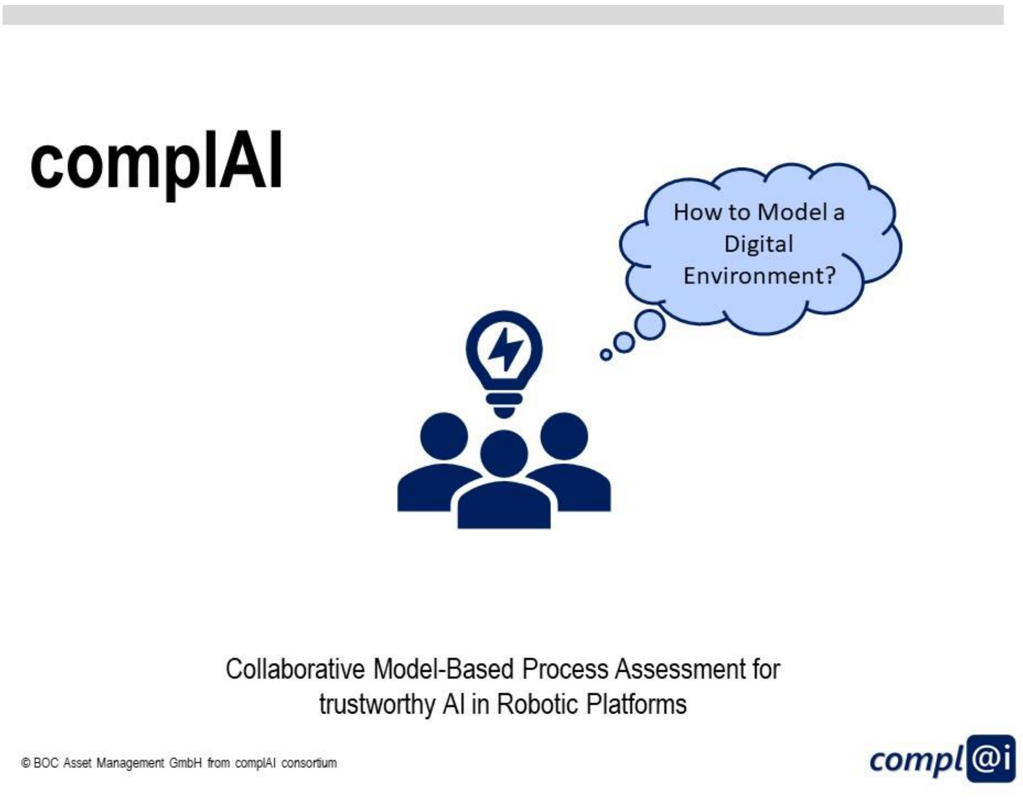
The project elaborated with (1) fixed, (2) pre-, and (3) late-binded workflows.

Sample workflows demonstrate how robots can be accessed by using different modelling languages.

Finally, the prototype package can be downloaded. All models presented in this deliverable can be downloaded and tested using the remote environment of the OMiLAB. Following the prerequisites and getting started steps provide the initial hands-on-experience. In order to support the installation, adaptation and use in the virtual environment the faq@adoxx.org email can be provided.

This deliverable is seen as the basis for the model-based assistant system prototype, which is outlined in "D4.1 Modellbasierter Prototyp eines Assistenzsystems".

Appendix A

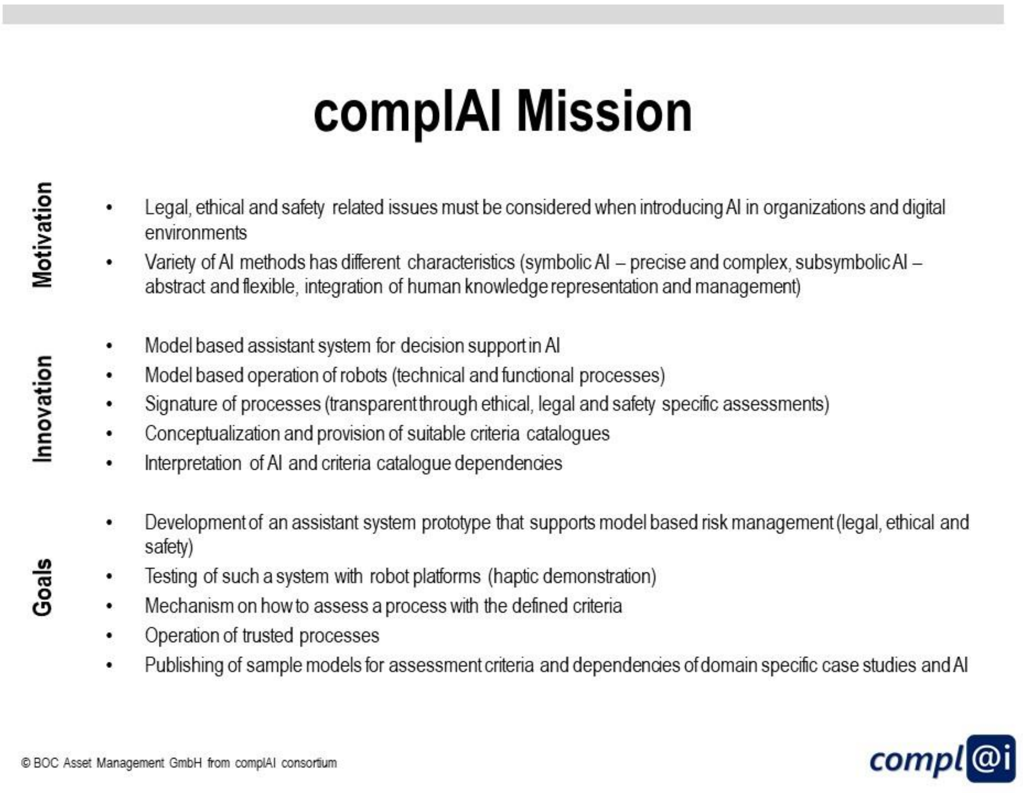


complAI

How to Model a Digital Environment?

Collaborative Model-Based Process Assessment for trustworthy AI in Robotic Platforms

© BOC Asset Management GmbH from complAI consortium



complAI Mission

Motivation

- Legal, ethical and safety related issues must be considered when introducing AI in organizations and digital environments
- Variety of AI methods has different characteristics (symbolic AI – precise and complex, subsymbolic AI – abstract and flexible, integration of human knowledge representation and management)

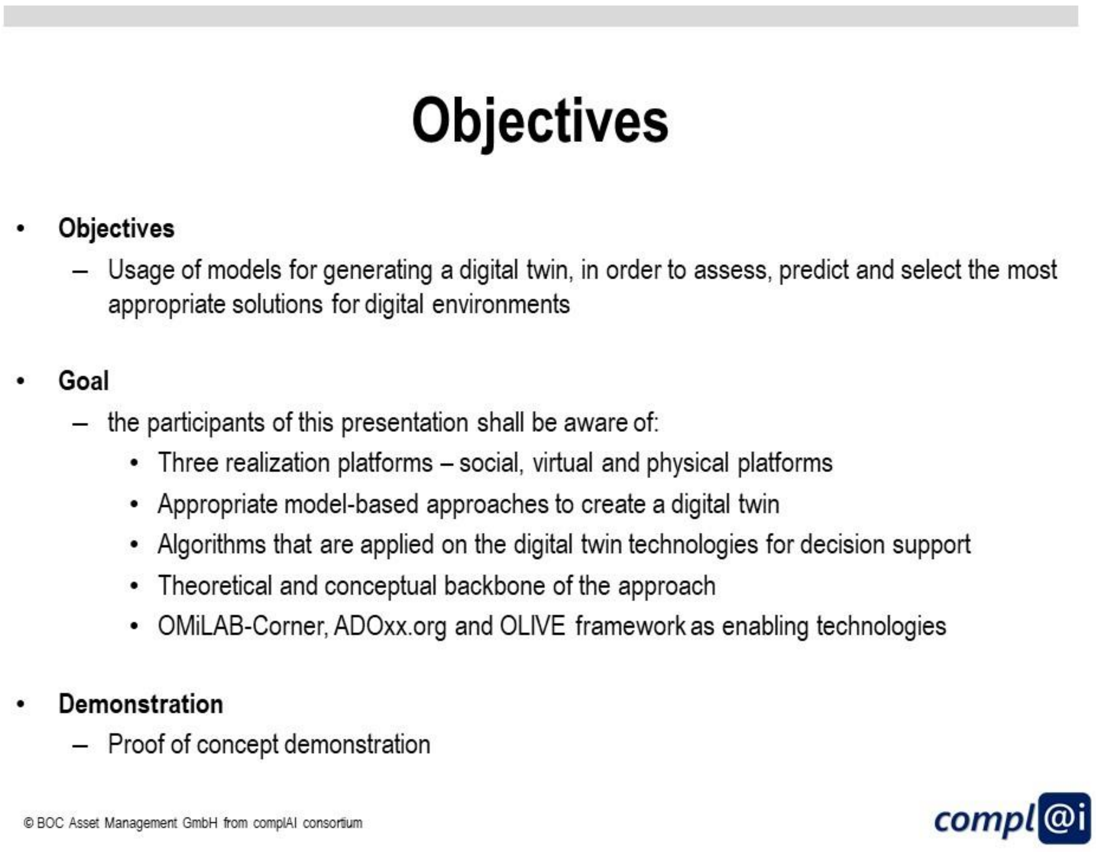
Innovation

- Model based assistant system for decision support in AI
- Model based operation of robots (technical and functional processes)
- Signature of processes (transparent through ethical, legal and safety specific assessments)
- Conceptualization and provision of suitable criteria catalogues
- Interpretation of AI and criteria catalogue dependencies

Goals

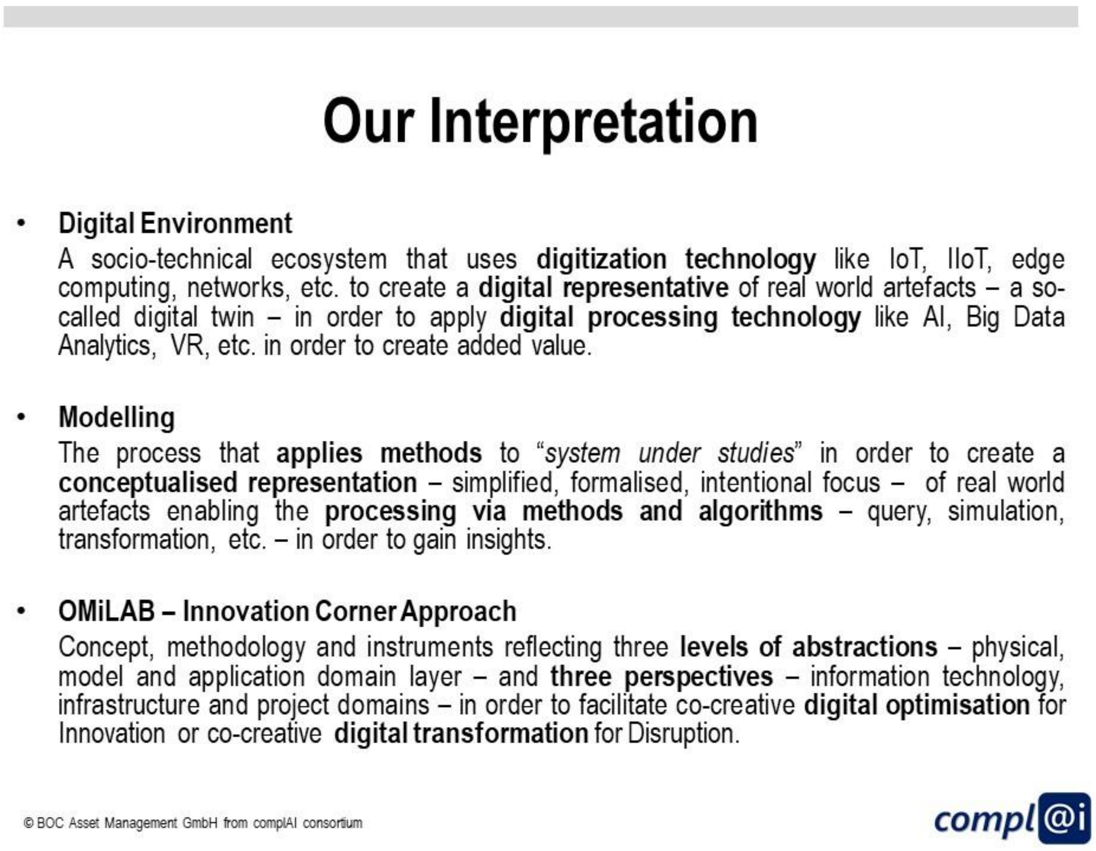
- Development of an assistant system prototype that supports model based risk management (legal, ethical and safety)
- Testing of such a system with robot platforms (haptic demonstration)
- Mechanism on how to assess a process with the defined criteria
- Operation of trusted processes
- Publishing of sample models for assessment criteria and dependencies of domain specific case studies and AI

© BOC Asset Management GmbH from complAI consortium



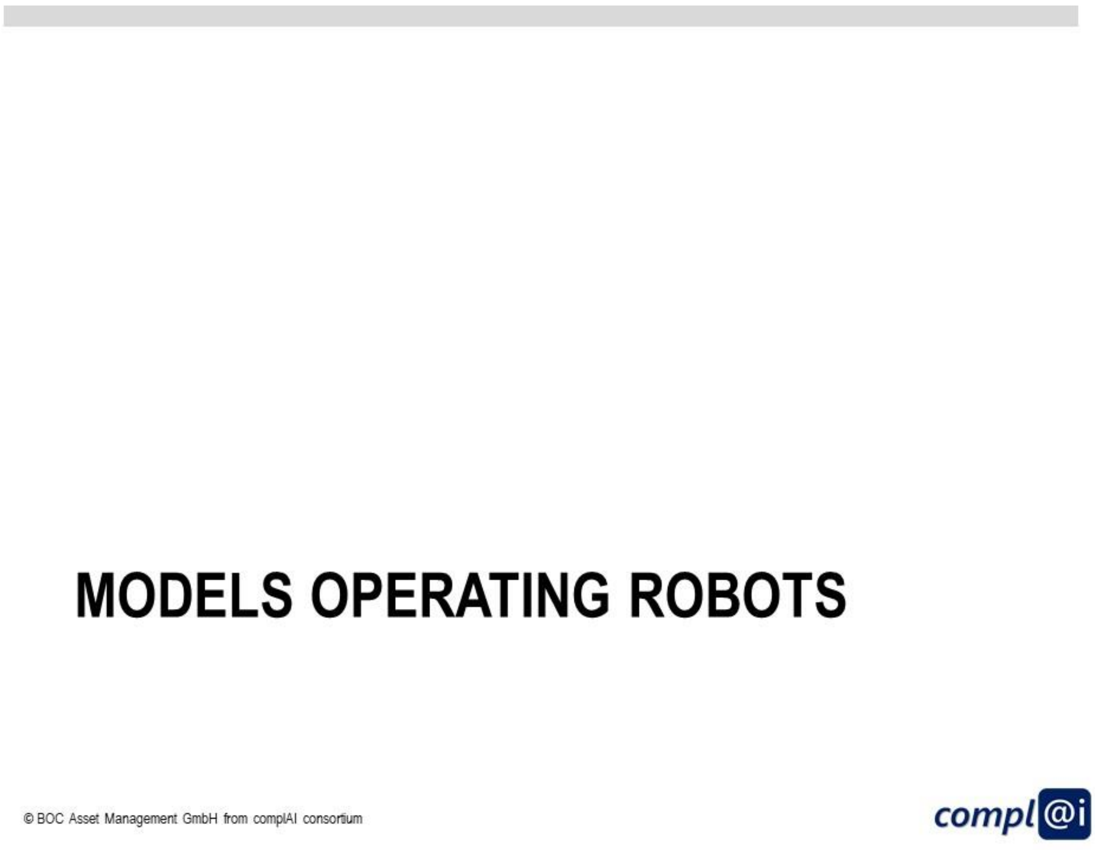
Objectives

- **Objectives**
  - Usage of models for generating a digital twin, in order to assess, predict and select the most appropriate solutions for digital environments
- **Goal**
  - the participants of this presentation shall be aware of:
    - Three realization platforms – social, virtual and physical platforms
    - Appropriate model-based approaches to create a digital twin
    - Algorithms that are applied on the digital twin technologies for decision support
    - Theoretical and conceptual backbone of the approach
    - OMiLAB-Corner, ADOxx.org and OLIVE framework as enabling technologies
- **Demonstration**
  - Proof of concept demonstration



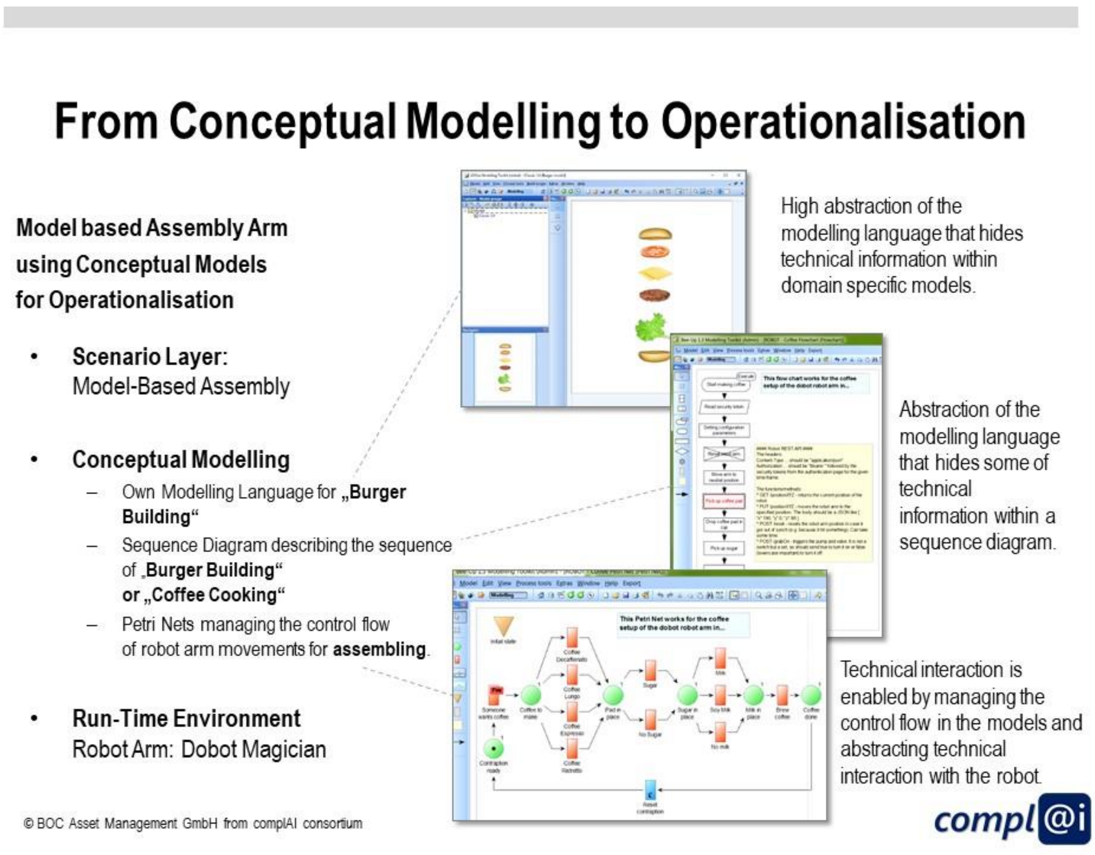
Our Interpretation

- **Digital Environment**
  A socio-technical ecosystem that uses **digitization technology** like IoT, IIoT, edge computing, networks, etc. to create a **digital representative** of real world artefacts – a so-called digital twin – in order to apply **digital processing technology** like AI, Big Data Analytics, VR, etc. in order to create added value.
- **Modelling**
  The process that **applies methods** to "*system under studies*" in order to create a **conceptualised representation** – simplified, formalised, intentional focus – of real world artefacts enabling the **processing via methods and algorithms** – query, simulation, transformation, etc. – in order to gain insights.
- **OMiLAB – Innovation Corner Approach**
  Concept, methodology and instruments reflecting three **levels of abstractions** – physical, model and application domain layer – and **three perspectives** – information technology, infrastructure and project domains – in order to facilitate co-creative **digital optimisation** for Innovation or co-creative **digital transformation** for Disruption.



MODELS OPERATING ROBOTS

© BOC Asset Management GmbH from complAI consortium



From Conceptual Modelling to Operationalisation

Model based Assembly Arm using Conceptual Models for Operationalisation

- **Scenario Layer:** Model-Based Assembly
- **Conceptual Modelling**
  - Own Modelling Language for **„Burger Building“**
  - Sequence Diagram describing the sequence of **„Burger Building“ or „Coffee Cooking“**
  - Petri Nets managing the control flow of robot arm movements for **assembling**.
- **Run-Time Environment** Robot Arm: Dobot Magician

High abstraction of the modelling language that hides technical information within domain specific models.

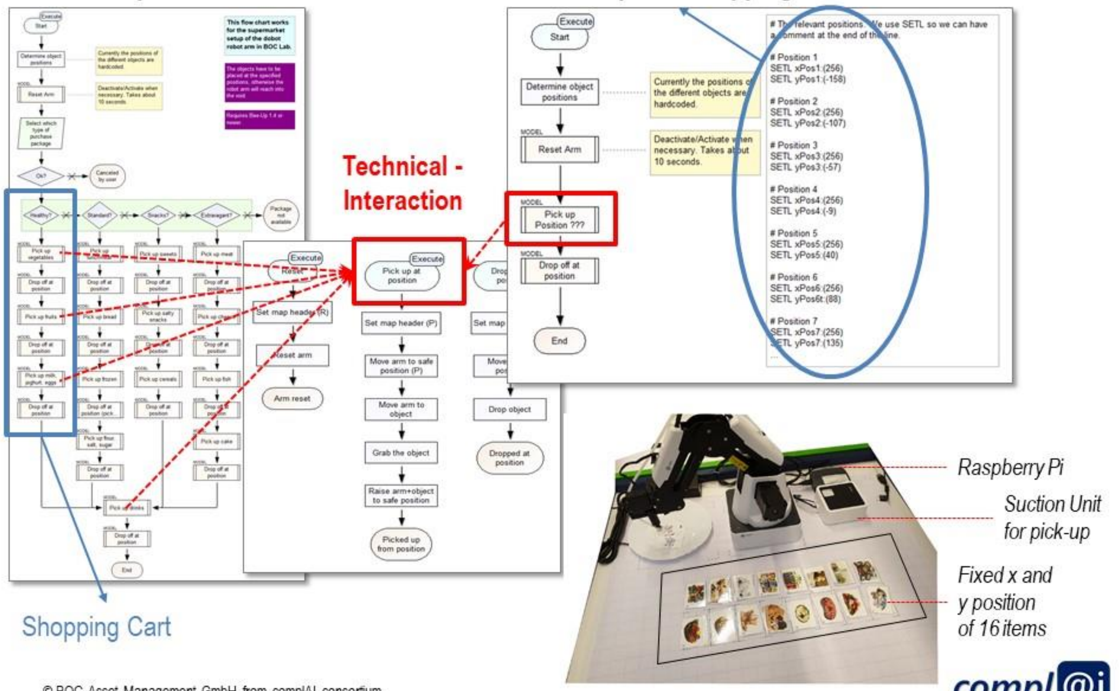
Abstraction of the modelling language that hides some of technical information within a sequence diagram.

Technical interaction is enabled by managing the control flow in the models and abstracting technical interaction with the robot.

© BOC Asset Management GmbH from complAI consortium

Operating Assembly Arm using Models

Domain-specific Model | **Domain-specific Mapping**



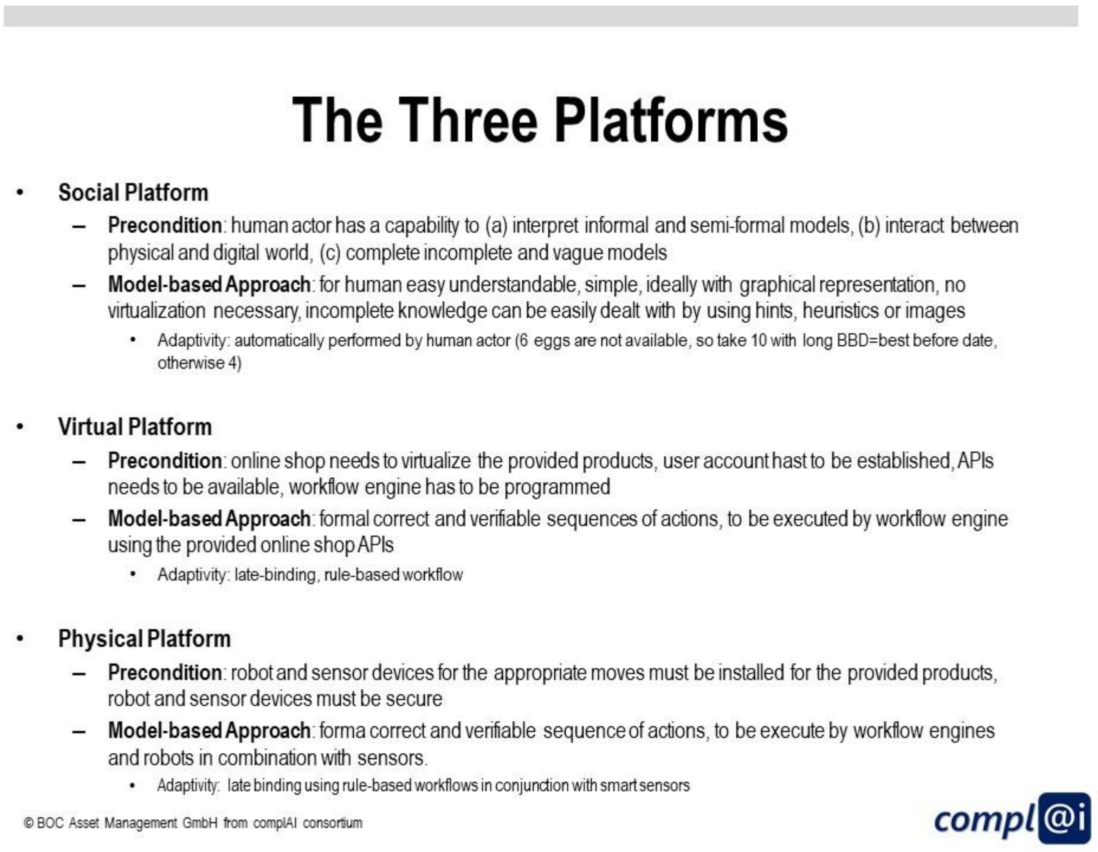
```
# The relevant positions. We use SETL so we can have
a comment at the end of the line.

# Position 1
SETL xPos1:(256)
SETL yPos1:(-158)

# Position 2
SETL xPos2:(256)
SETL yPos2:(-107)

# Position 3
SETL xPos3:(256)
SETL yPos3:(-57)

# Position 4
SETL xPos4:(256)
SETL yPos4:(-9)

# Position 5
SETL xPos5:(256)
SETL yPos5:(40)

# Position 6
SETL xPos6:(256)
SETL yPos6t:(88)

# Position 7
SETL xPos7:(256)
SETL yPos7:(135)
...
```

compl@i

MOTIVATION

BRIDGING SEMANTIC DISTANCE BETWEEN LAYERS AND PLATFORMS

compl@i



The Three Platforms

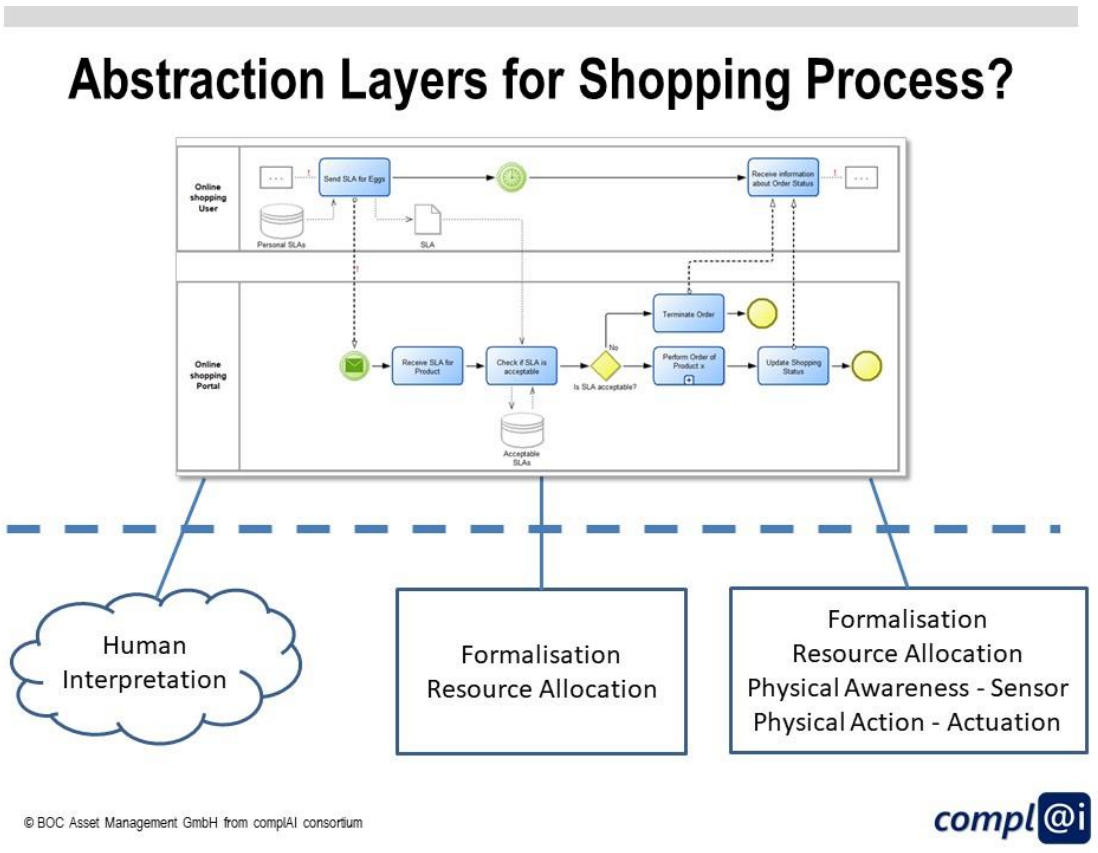
- **Social Platform**
  - **Precondition**: human actor has a capability to (a) interpret informal and semi-formal models, (b) interact between physical and digital world, (c) complete incomplete and vague models
  - **Model-based Approach**: for human easy understandable, simple, ideally with graphical representation, no virtualization necessary, incomplete knowledge can be easily dealt with by using hints, heuristics or images
    - Adaptivity: automatically performed by human actor (6 eggs are not available, so take 10 with long BBD=best before date, otherwise 4)

- **Virtual Platform**
  - **Precondition**: online shop needs to virtualize the provided products, user account hast to be established, APIs needs to be available, workflow engine has to be programmed
  - **Model-based Approach**: formal correct and verifiable sequences of actions, to be executed by workflow engine using the provided online shop APIs
    - Adaptivity: late-binding, rule-based workflow

- **Physical Platform**
  - **Precondition**: robot and sensor devices for the appropriate moves must be installed for the provided products, robot and sensor devices must be secure
  - **Model-based Approach**: forma correct and verifiable sequence of actions, to be execute by workflow engines and robots in combination with sensors.
    - Adaptivity: late binding using rule-based workflows in conjunction with smart sensors

© BOC Asset Management GmbH from complAI consortium

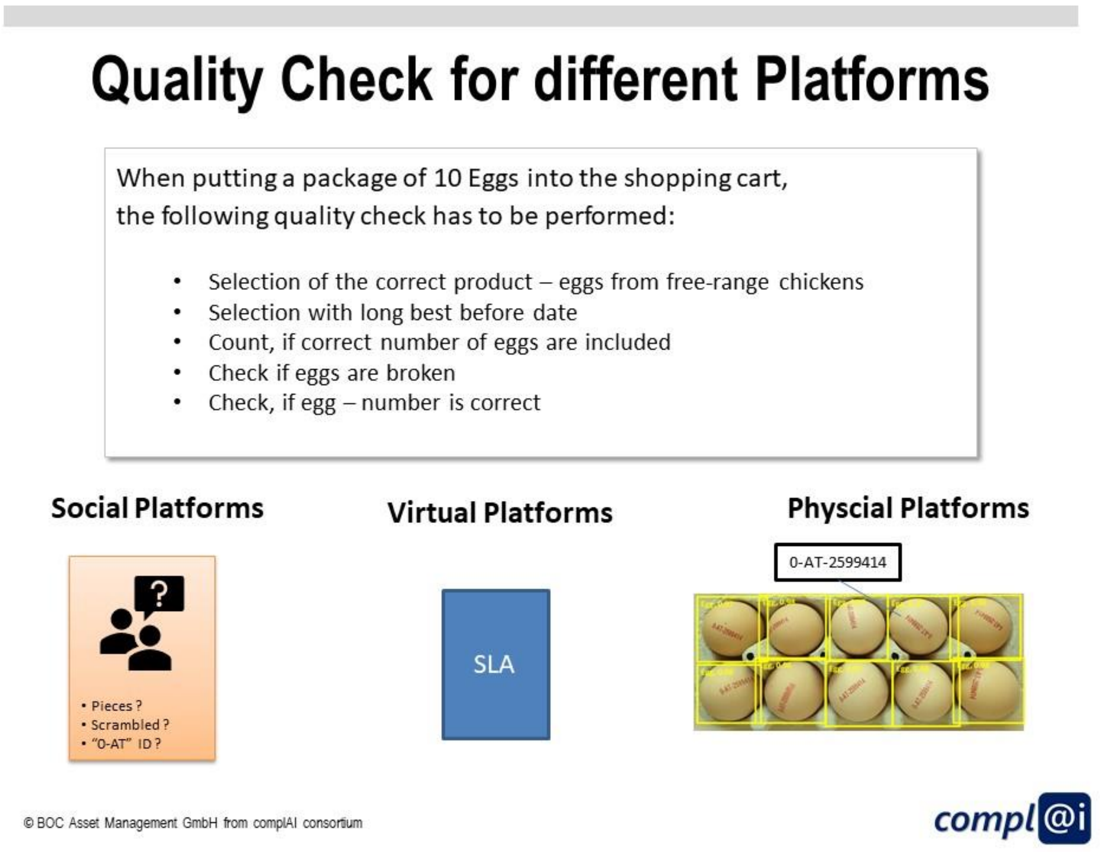
compl@i



Abstraction Layers for Shopping Process?

© BOC Asset Management GmbH from complAI consortium

compl@i



Quality Check for different Platforms

When putting a package of 10 Eggs into the shopping cart, the following quality check has to be performed:

- Selection of the correct product – eggs from free-range chickens
- Selection with long best before date
- Count, if correct number of eggs are included
- Check if eggs are broken
- Check, if egg – number is correct

Social Platforms

- Pieces ?
- Scrambled ?
- "0-AT" ID ?

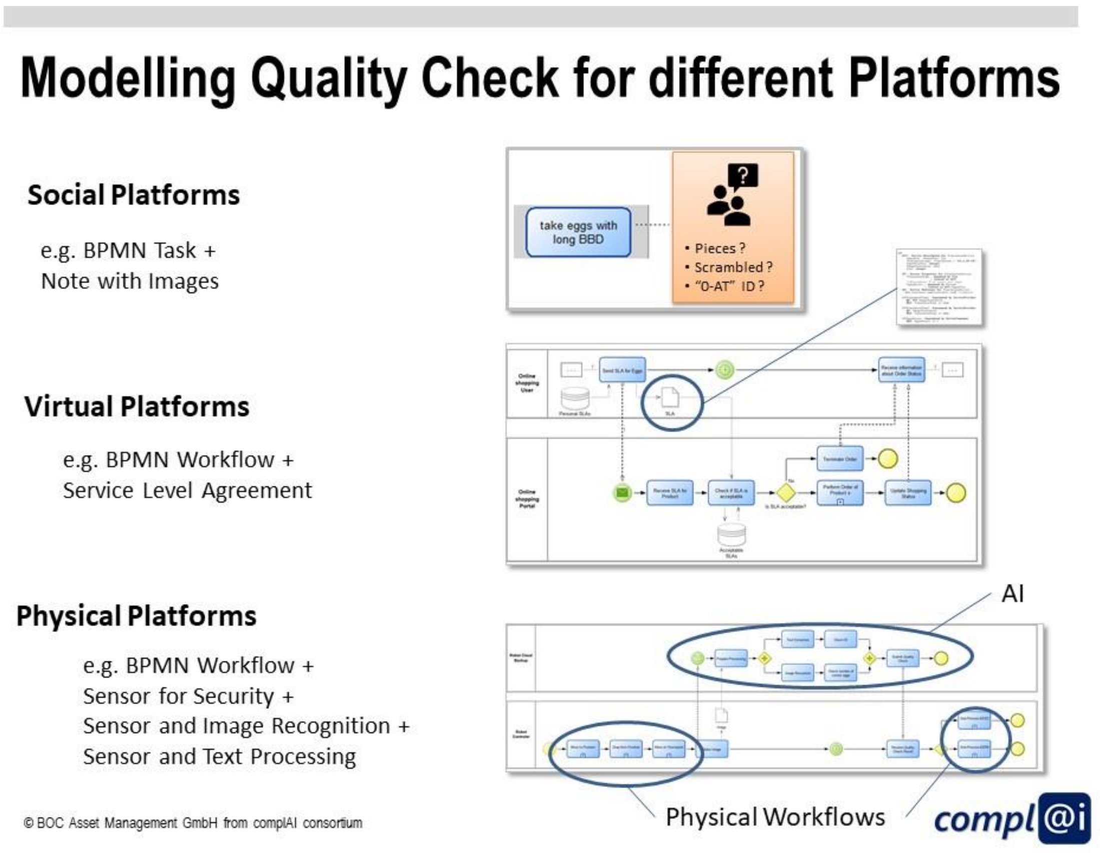
Virtual Platforms

SLA

Physcial Platforms

0-AT-2599414

compl@i



Modelling Quality Check for different Platforms

Social Platforms

e.g. BPMN Task + Note with Images

take eggs with long BBD

- Pieces ?
- Scrambled ?
- "0-AT" ID ?

Virtual Platforms

e.g. BPMN Workflow + Service Level Agreement

Physical Platforms

e.g. BPMN Workflow + Sensor for Security + Sensor and Image Recognition + Sensor and Text Processing

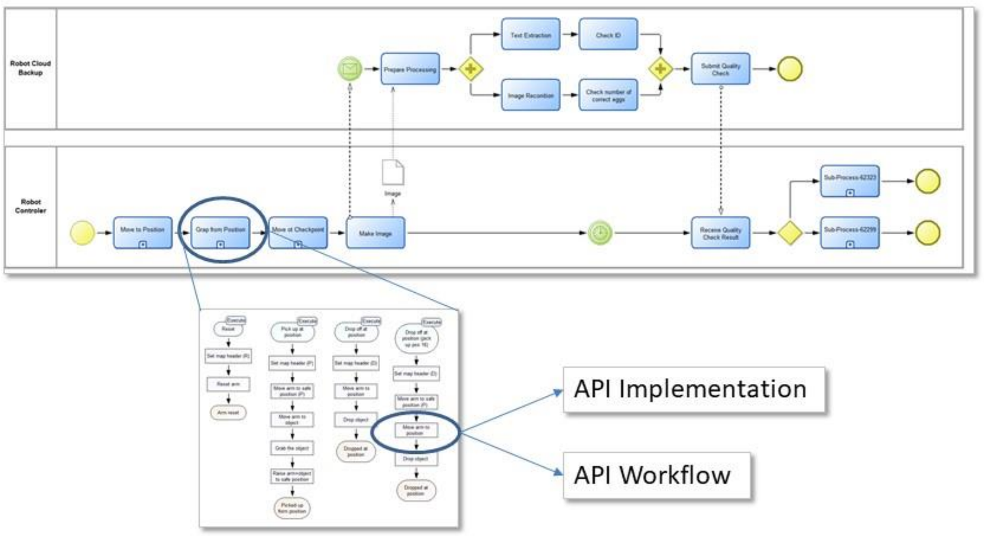
AI

Physical Workflows

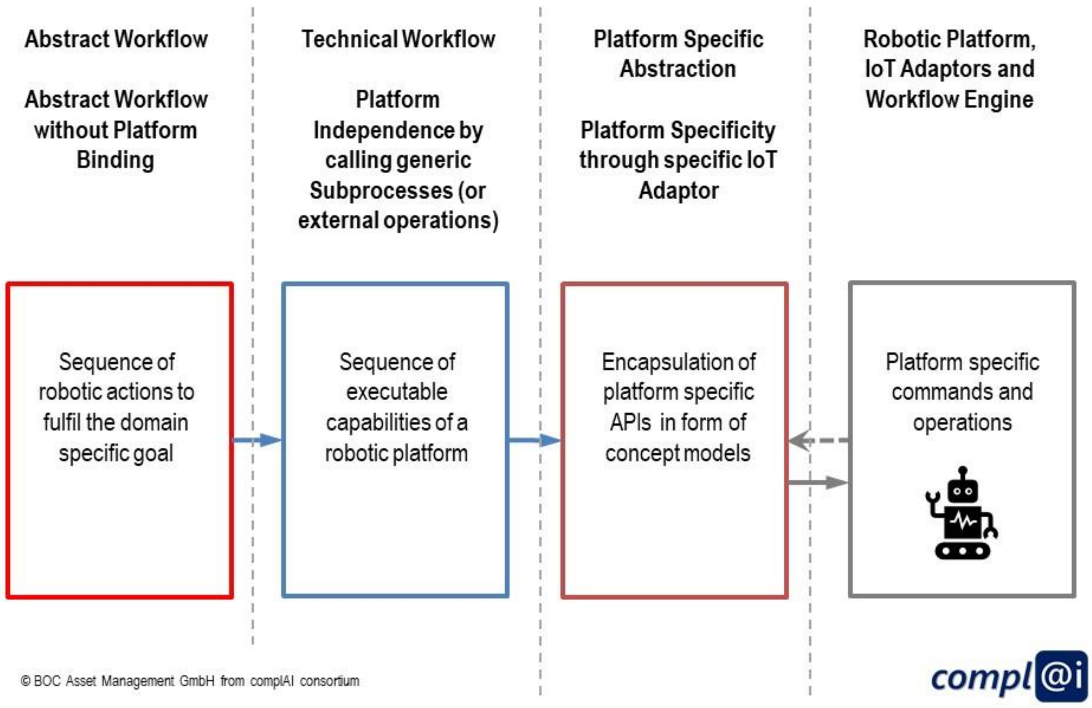
compl@i

Different Abstraction Layers

Physical Interaction for taking a product with quality check into the shopping cart:



Introduction of the 4 Abstraction Layers



Abstract Workflow

Abstract Workflow without Platform Binding

Sequence of robotic actions to fulfil the domain specific goal

Technical Workflow

Platform Independence by calling generic Subprocesses (or external operations)

Sequence of executable capabilities of a robotic platform

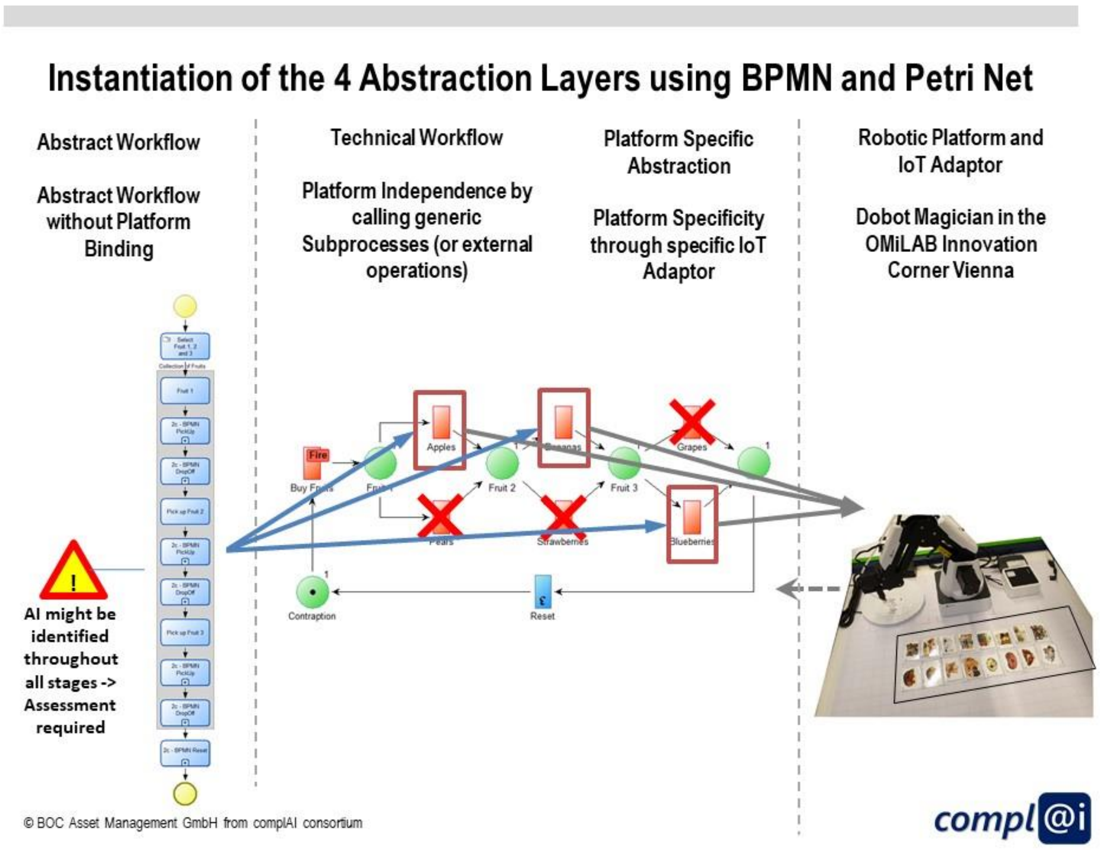
Platform Specific Abstraction

Platform Specificity through specific IoT Adaptor

Encapsulation of platform specific APIs in form of concept models

Robotic Platform, IoT Adaptors and Workflow Engine

Platform specific commands and operations



Instantiation of the 4 Abstraction Layers using BPMN and Petri Net

Abstract Workflow

Abstract Workflow without Platform Binding

Technical Workflow

Platform Independence by calling generic Subprocesses (or external operations)

Platform Specific Abstraction

Platform Specificity through specific IoT Adaptor

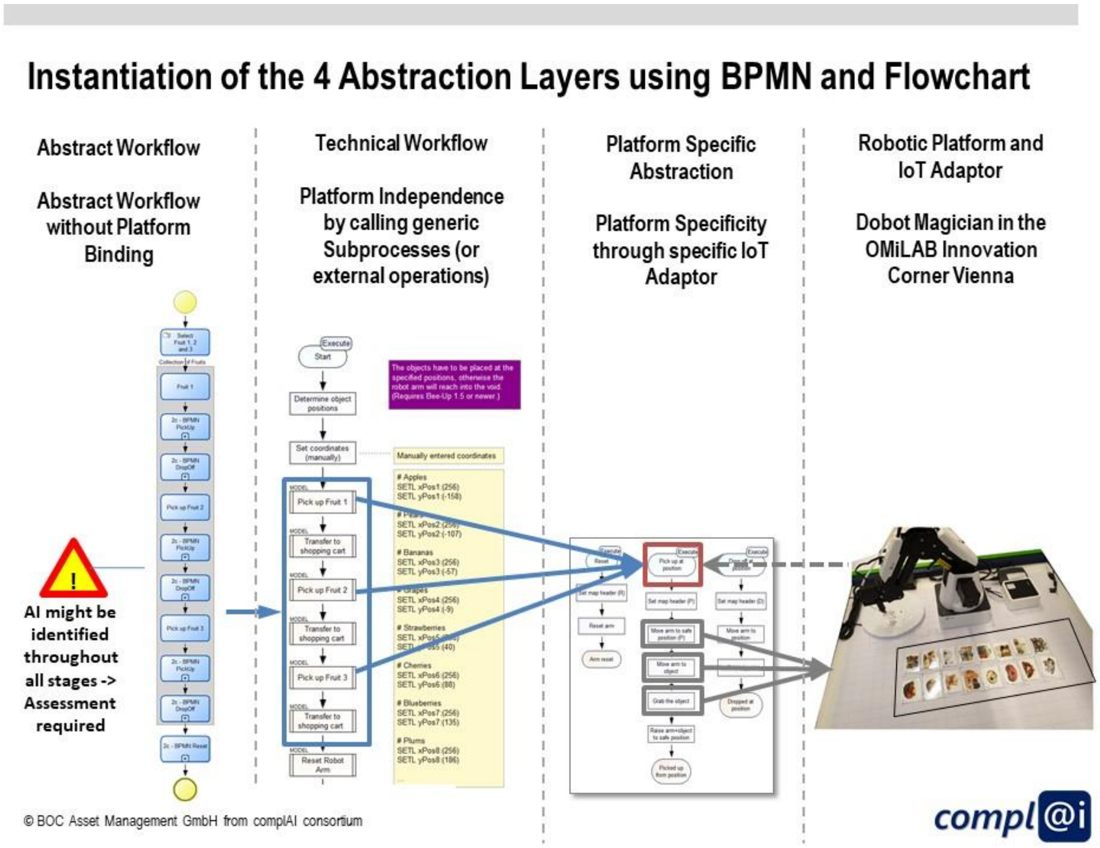
Robotic Platform and IoT Adaptor

Dobot Magician in the OMiLAB Innovation Corner Vienna

AI might be identified throughout all stages -> Assessment required

© BOC Asset Management GmbH from complAI consortium

compl@i



Instantiation of the 4 Abstraction Layers using BPMN and Flowchart

Abstract Workflow

Abstract Workflow without Platform Binding

Technical Workflow

Platform Independence by calling generic Subprocesses (or external operations)

Platform Specific Abstraction

Platform Specificity through specific IoT Adaptor

Robotic Platform and IoT Adaptor

Dobot Magician in the OMiLAB Innovation Corner Vienna

AI might be identified throughout all stages -> Assessment required

The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)

Manually entered coordinates

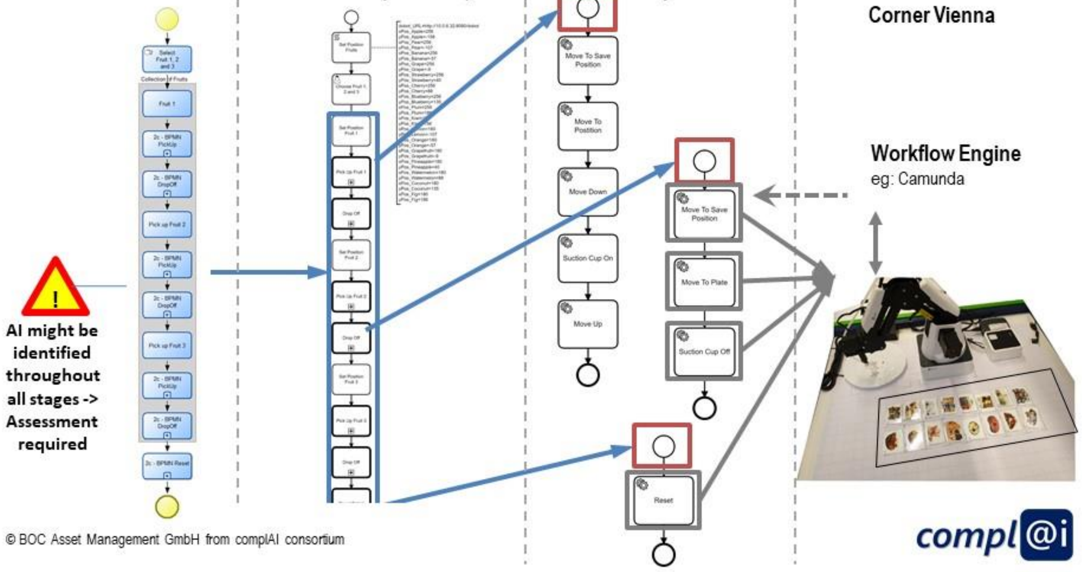
```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

...
```

© BOC Asset Management GmbH from complAI consortium

compl@i

Instantiation of the 4 Abstraction Layers using BPMN and Workflows

Abstract Workflow	Technical Workflow	Platform Specific Abstraction	Robotic Platform, IoT Adaptor and Workflow Engine
Abstract Workflow without Platform Binding	Platform Independence by calling generic Subprocesses (or external operations)	Platform Specificity through specific IoT Adaptor	Dobot Magician in the OMiLAB Innovation Corner Vienna



WORKFLOWS OPERATING ROBOTS

FIXED, PRE & LATE BINDING WORKFLOWS

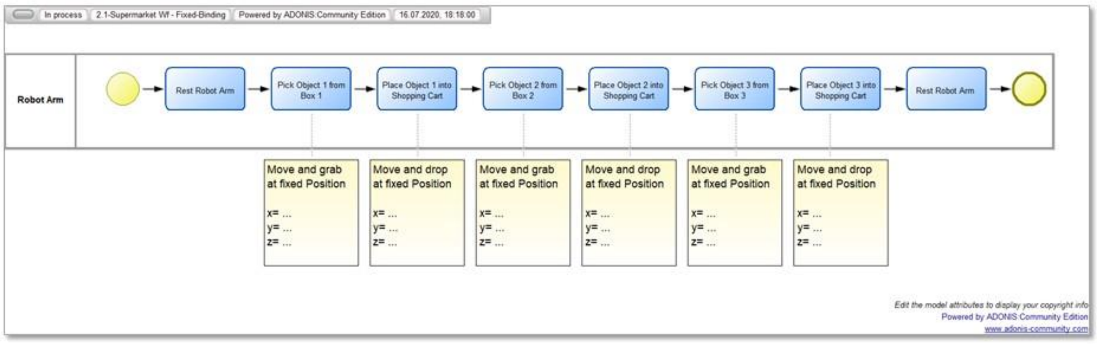
Explanation of different Workflow Bindings

- A Workflow executes and orchestrates a sequence of SW APIs, in the context of robots each SW API is an IoT Adaptor that triggers a pre-programmed actuation – a certain move – of the robot arm.
- The resource allocation or binding of a workflow is typically understood to link a certain SW API to a particular action of the workflow, in our case of the robot arm, a particular IoT Adaptor – hence a certain move – is bound to a workflow task.
- We use the supermarket scenario to demonstrate different ways of binding a SW API – in our case a robot arm move – to a workflow action.
  - Fixed binding: The workflow and ALL the moves are predefined, when the user selects the workflow, the user gets exactly the sequence of moves.
  - Pre binding: The workflow and ALL the moves are decided just before starting with the first move. This allows to react in case a certain move – picking up of a particular object – is not possible and allows to consider alternatives – picking up another object instead.
  - Late binding: The workflow and ALL the moves are decided just while the workflow is executed. This allows to react on situation just before the actual move is performed

compl@i

The Fixed Binding Workflow

Three objects – in our case apple, bananas and grapes are picked up in this sequence and put into the shopping cart.


In process
2.1-Supermarket Wf - Fixed-Binding
Powered by ADONIS:Community Edition
16.07.2020, 18:18:00
Robot Arm
Rest Robot Arm
Pick Object 1 from Box 1
Place Object 1 into Shopping Cart
Pick Object 2 from Box 2
Place Object 2 into Shopping Cart
Pick Object 3 from Box 3
Place Object 3 into Shopping Cart
Rest Robot Arm
Move and grab at fixed Position x= ... y= ... z= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at fixed Position x= ... y= ... z= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at fixed Position x= ... y= ... z= ...
Move and drop at fixed Position x= ... y= ... z= ...
Edit the model attributes to display your copyright info
Powered by ADONIS:Community Edition
www.adonis-community.com

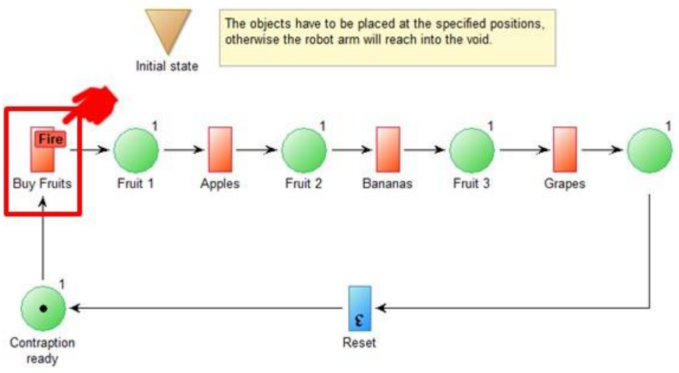
Assessment on:

- Is the robot arm safe, can people be hurt?
- Does the customer know the result?
- Is AI performing any decision the customer is unaware of?

compl@i

Instantiation of the Fixed Binding Workflow using Petri Nets

Three objects – in our case apples, bananas and grapes are picked up in this sequence and put into the shopping cart.


Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Fire
Buy Fruits
1
Fruit 1
Apples
1
Fruit 2
Bananas
1
Fruit 3
Grapes
1
1
Contraption ready
ε
Reset

In a fixed binding workflow, all decisions are taken beforehand, so if there are no apples available, the robot will reach into void or an erroneous behaviour. The Petri Net requires step by step user interaction for the execution.

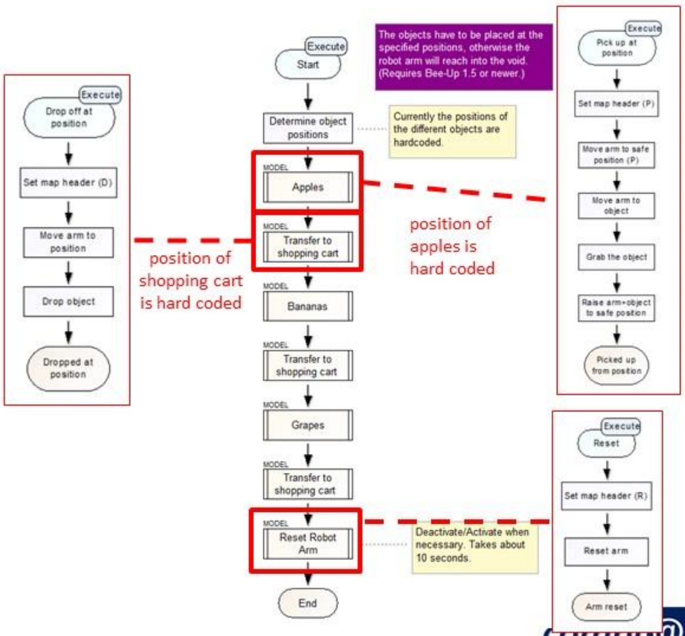
How does the robot react, if there are no apples available? → **proposed solution AI**

Instantiation of the Fixed Binding Workflow using Flowcharts

Three objects – in our case apples, bananas and grapes are picked up in this sequence and put into the shopping cart.

In a fixed binding workflow, all decisions are taken beforehand, so if there are no apples available, the robot will reach into void or an erroneous behaviour. If the fixed binding flowchart execution is started, it runs through without any user interaction.

How does the robot react, if there are no apples available?
→ **proposed solution AI**

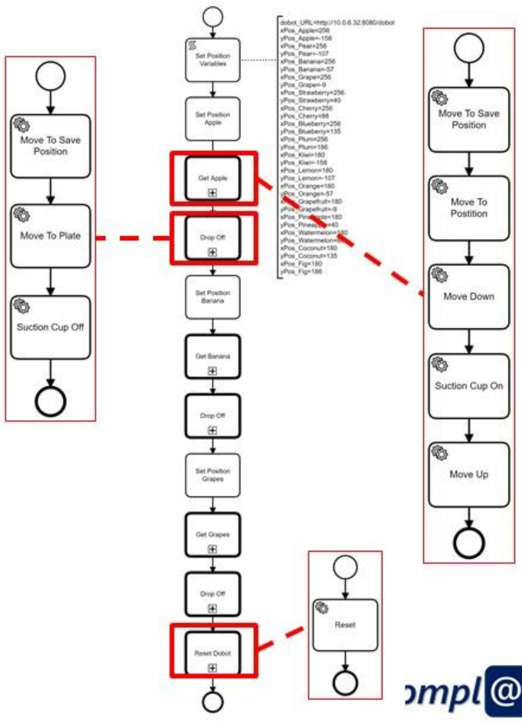

Execute
Start
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Determine object positions
Currently the positions of the different objects are hardcoded.
MODEL
Apples
MODEL
Transfer to shopping cart
MODEL
Bananas
MODEL
Transfer to shopping cart
MODEL
Grapes
MODEL
Transfer to shopping cart
MODEL
Reset Robot Arm
Deactivate/Activate when necessary. Takes about 10 seconds.
End
Execute
Drop off at position
Set map header (D)
Move arm to position
Drop object
Dropped at position
position of shopping cart is hard coded
position of apples is hard coded
Execute
Pick up at position
Set map header (P)
Move arm to safe position (P)
Move arm to object
Grab the object
Raise arm+object to safe position
Picked up from position
Execute
Reset
Set map header (R)
Reset arm
Arm reset

Instantiation of the Fixed Binding Workflow using BPMN

Three objects – in our case apples, bananas and grapes are picked up in this sequence and put into the shopping cart.

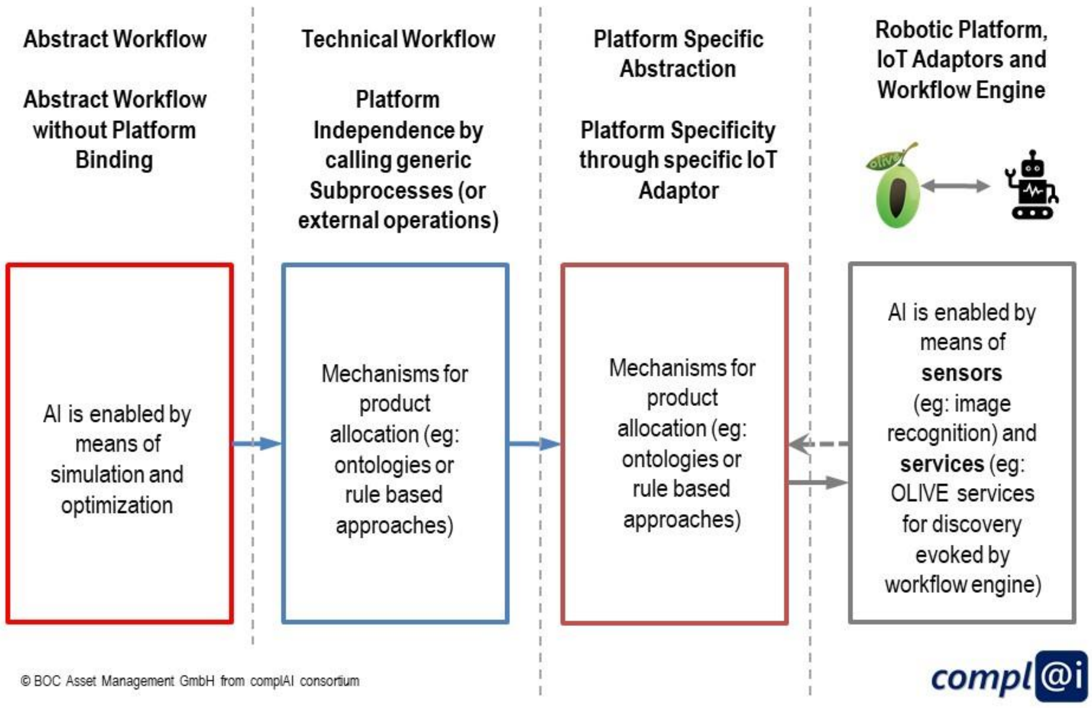
In a fixed binding workflow, all decisions are taken beforehand, so if there are no apples available, the robot will reach into void or an erroneous behaviour. If the fixed workflow execution is triggered, it handled fully automatically by the workflow engine.

How does the robot react, if there are no apples available? → **proposed solution AI**



```
dobot_URL=http://10.0.6.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
xPos_Fig=180
yPos_Fig=186
```

AI Integration on the 4 Abstraction Layers



Abstract Workflow

Abstract Workflow without Platform Binding

AI is enabled by means of simulation and optimization

Technical Workflow

Platform Independence by calling generic Subprocesses (or external operations)

Mechanisms for product allocation (eg: ontologies or rule based approaches)

Platform Specific Abstraction

Platform Specificity through specific IoT Adaptor

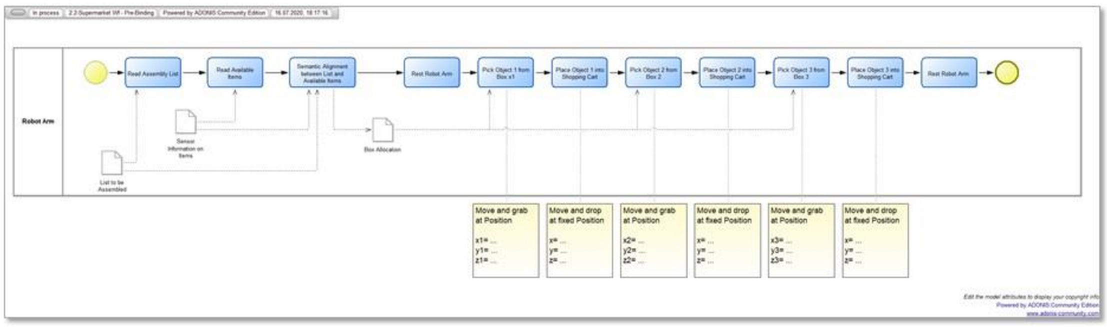
Mechanisms for product allocation (eg: ontologies or rule based approaches)

Robotic Platform, IoT Adaptors and Workflow Engine

AI is enabled by means of **sensors** (eg: image recognition) and **services** (eg: OLIVE services for discovery evoked by workflow engine)

The Pre Binding Workflow

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used; in case no bananas are available a strawberry is used; in case no grapes are available – blue berries are taken.


in process
2.2 Supermarket Wf - Pre-Binding
Powered by ADONIS Community Edition
16.07.2020, 18:17:16
Robot Arm
Read Assembly List
Read Available Items
Semantic Alignment between List and Available Items
Rest Robot Arm
Pick Object 1 from Box x1
Place Object 1 into Shopping Cart
Pick Object 2 from Box 2
Place Object 2 into Shopping Cart
Pick Object 3 from Box 3
Place Object 3 into Shopping Cart
Rest Robot Arm
Sensor Information on Items
Box Allocation
List to be Assembled
Move and grab at Position x1= ... y1= ... z1= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at Position x2= ... y2= ... z2= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at Position x3= ... y3= ... z3= ...
Move and drop at fixed Position x= ... y= ... z= ...
Edit the model attributes to display your copyright info
Powered by ADONIS Community Edition
www.adonis-community.com

Assessment on:

- Is the robot arm safe, can people be hurt?
- Does the customer know the result?
- Is AI performing any decision the customer is unaware of?

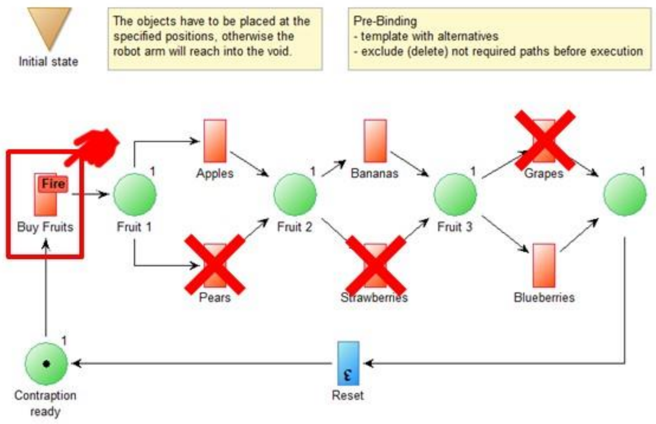
compl@i

Instantiation of the Pre Binding Workflow using Petri Nets

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used; in case no bananas are available a strawberry is used; in case no grapes are available – blue berries are taken.

In the pre binding workflow, the product availability for all fruits is checked at the beginning.

Apples and bananas are available. Grapes are not available, therefore Blueberries are taken.


Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Pre-Binding
- template with alternatives
- exclude (delete) not required paths before execution
Fire
Buy Fruits
Fruit 1
Apples
Pears
Fruit 2
Bananas
Strawberries
Fruit 3
Grapes
Blueberries
1
ε
Reset
Contraption ready

AI integration for checking an allocation by applying a rule based approach with user interaction. In this sample **AI** is applied on the **technical workflow layer.**

compl@i

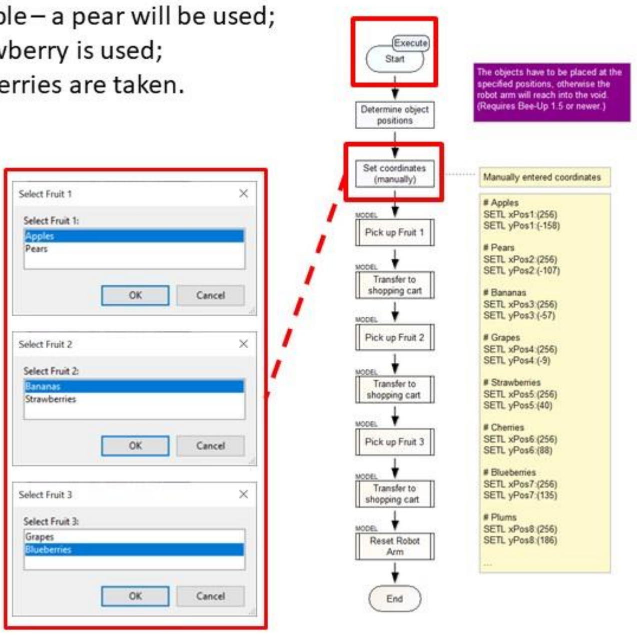
Instantiation of the Pre Binding Workflow using Flowcharts

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used;
in case no bananas are available a strawberry is used;
in case no grapes are available – blue berries are taken.

In the pre binding workflow, the product availability for all fruits is checked at the beginning.

Apples and bananas are available. Grapes are not available, therefore Blueberries are taken.

AI integration for checking an allocation by applying a rule based approach with user interaction. In this sample **AI** is applied on the **technical workflow layer.**


Execute
Start
Determine object positions
Set coordinates (manually)
MODEL
Pick up Fruit 1
Transfer to shopping cart
Pick up Fruit 2
Transfer to shopping cart
Pick up Fruit 3
Transfer to shopping cart
Reset Robot Arm
End
OK
Cancel

Select Fruit 1
Select Fruit 1:
Apples
Pears

Select Fruit 2
Select Fruit 2:
Bananas
Strawberries

Select Fruit 3
Select Fruit 3:
Grapes
Blueberries

The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)

Manually entered coordinates

```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

....
```

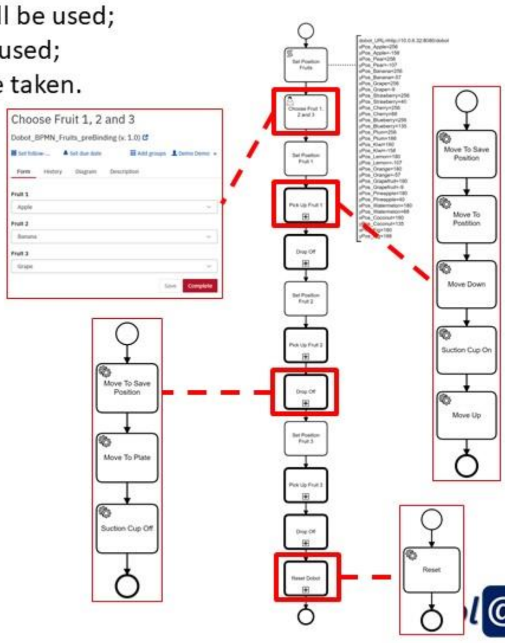
Instantiation of the Pre Binding Workflow using BPMN

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used;
in case no bananas are available a strawberry is used;
in case no grapes are available – blue berries are taken.

In the pre binding workflow, the product availability for all fruits is checked at the beginning.

Apples and bananas are available. Grapes are not available, therefore Blueberries are taken.

AI integration for checking an allocation by applying a rule based approach with user interaction. In this sample **AI** is applied on the **technical workflow layer.**

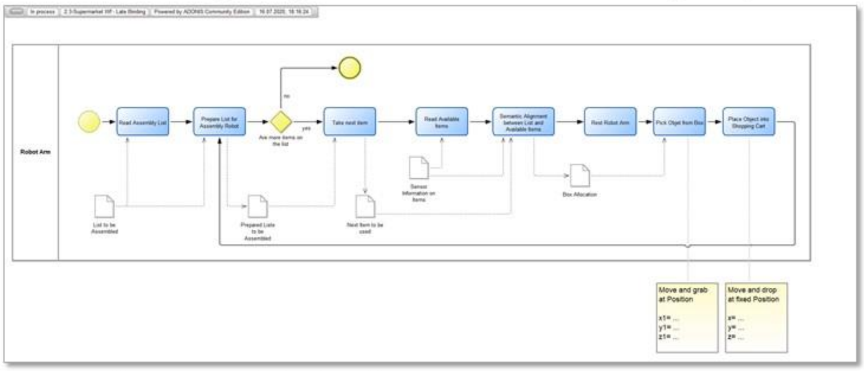


Choose Fruit 1, 2 and 3
Dobot_BPMN_Fruits_preBinding (v. 1.0)
Form History Diagram Description
Fruit 1
Apple
Fruit 2
Banana
Fruit 3
Grape
Save Complete

```
dobot_URL=http://10.0.8.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
[illegible]=180
[illegible]=186
```

The Late Binding Workflow

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used; in case no bananas are available a strawberry is used; in case no grapes are available – blue berries are taken. The availability is not checked at the beginning of the workflow but at the beginning of each robot arm move.


Robot Arm
Read Assembly List
Prepare List for Assembly Robot
Are more items on the list
yes
no
Take next item
Read Available Items
Semantic Alignment between List and Available Items
Rest Robot Arm
Pick Objet from Box
Place Object into Shopping Cart
List to be Assembled
Prepared Lists to be Assembled
Next Item to be used
Sensor Information on Items
Box Allocation
Move and grab at Position
x1= ...
y1= ...
z1= ...
Move and drop at fixed Position
x= ...
y= ...
z= ...

Assessment on:

- Is the robot arm safe, can people be hurt?
- Does the customer know the result?
- Is AI performing any decision the customer is unaware of?

© BOC Asset Management GmbH from complAI consortium

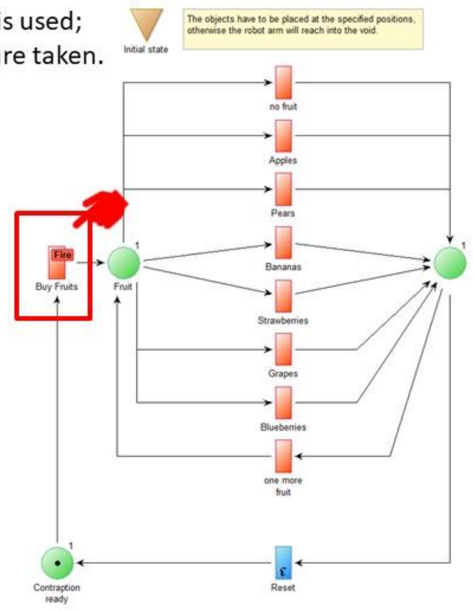
compl@i

Instantiation of the Late Binding Workflow using Petri Nets

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart.
In case there are no apples available – a pear will be used;
in case no bananas are available a strawberry is used;
in case no grapes are available – blue berries are taken.

In the late binding workflow, the product availability is checked each time a fruit is chosen.

AI integration for checking an allocation by applying a rule based approach. In this sample **AI** is applied on the **platform specific abstraction layer.**



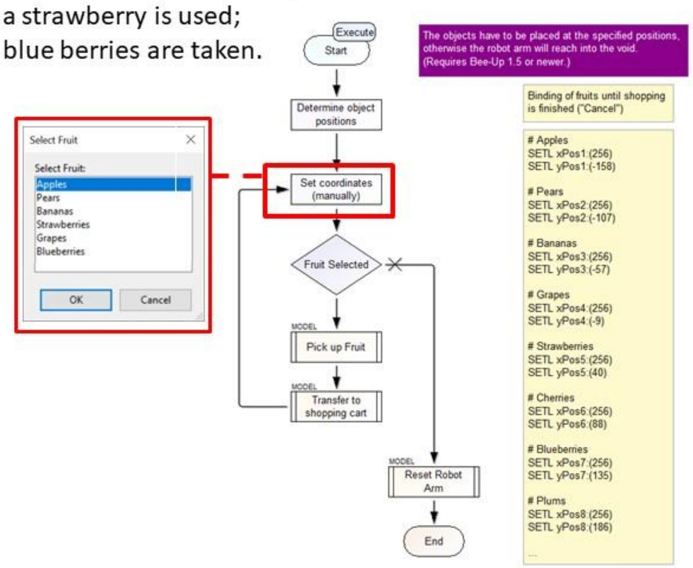
© BOC Asset Management GmbH from complAI consortium

Instantiation of the Late Binding Workflow using Flowcharts

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used;
in case no bananas are available a strawberry is used;
in case no grapes are available – blue berries are taken.

In the late binding workflow, the product availability is checked each time a fruit is chosen.

AI integration for checking an allocation by applying a rule based approach. In this sample **AI** can be applied on the **robotic platform layer by means of services.**


Execute
Start
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Binding of fruits until shopping is finished ("Cancel")
Determine object positions
Select Fruit
Select Fruit:
Apples
Pears
Bananas
Strawberries
Grapes
Blueberries
OK
Cancel
Set coordinates (manually)
Fruit Selected
MODEL
Pick up Fruit
MODEL
Transfer to shopping cart
MODEL
Reset Robot Arm
End

```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

...
```

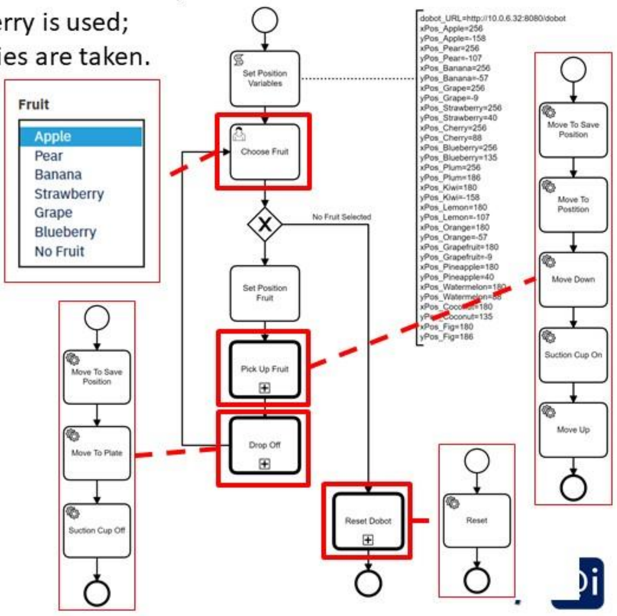
© BOC Asset Management GmbH from complAI consortium

Instantiation of the Late Binding Workflow using BPMN

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used;
in case no bananas are available a strawberry is used;
in case no grapes are available – blue berries are taken.

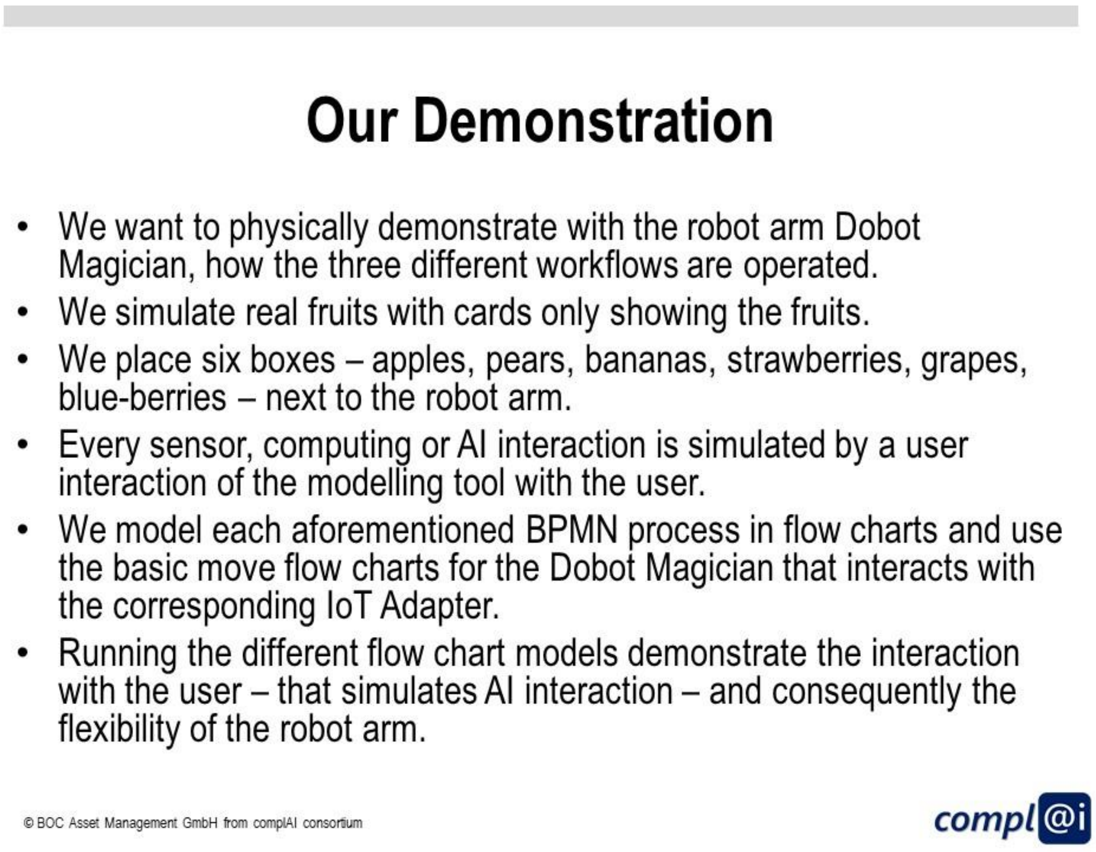
In the late binding workflow, the product availability is checked each time a fruit is chosen.

AI integration for checking an allocation by applying a rule based approach. In this sample **AI** can be applied on the **robotic platform layer by means of services. Sensor information can be provided to the workflow engine, which invokes an OLIVE service for instance.**



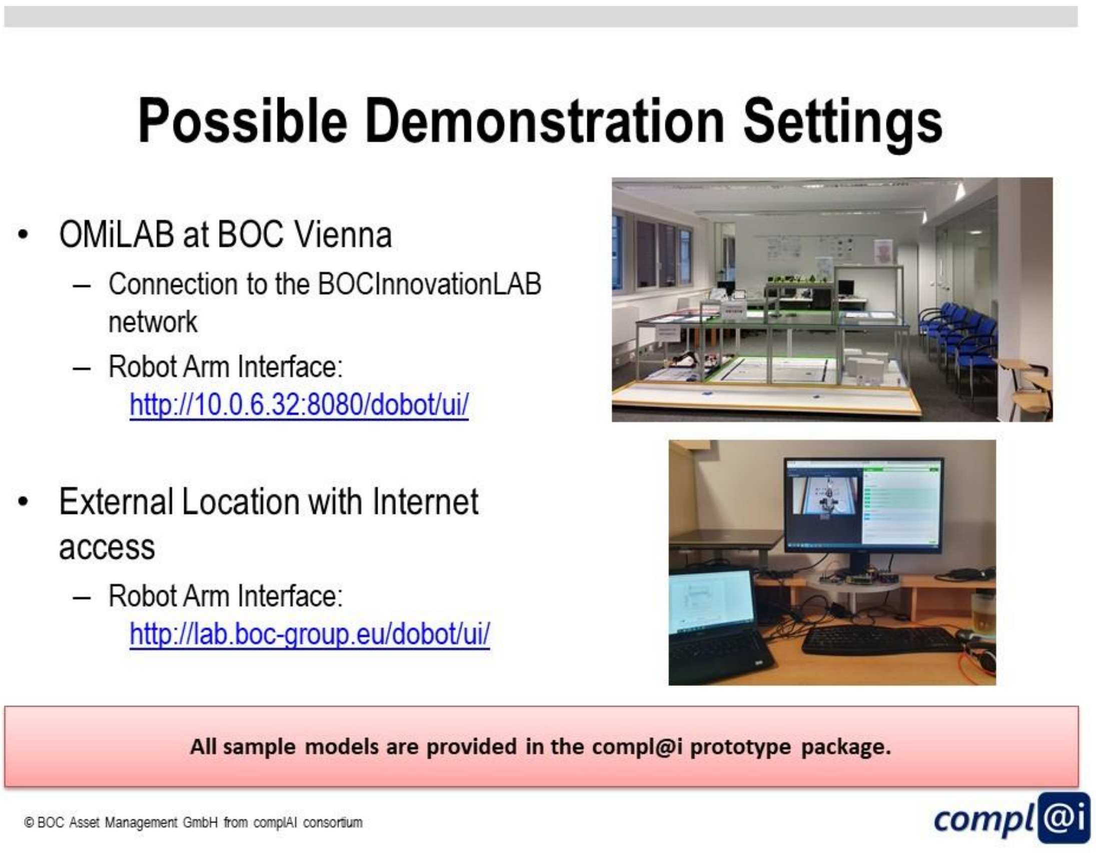
```
dobot_URL=http://10.0.6.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
xPos_Fig=180
yPos_Fig=186
```

© BOC Asset Management GmbH from complAI consortium



Our Demonstration

- We want to physically demonstrate with the robot arm Dobot Magician, how the three different workflows are operated.
- We simulate real fruits with cards only showing the fruits.
- We place six boxes – apples, pears, bananas, strawberries, grapes, blue-berries – next to the robot arm.
- Every sensor, computing or AI interaction is simulated by a user interaction of the modelling tool with the user.
- We model each aforementioned BPMN process in flow charts and use the basic move flow charts for the Dobot Magician that interacts with the corresponding IoT Adapter.
- Running the different flow chart models demonstrate the interaction with the user – that simulates AI interaction – and consequently the flexibility of the robot arm.



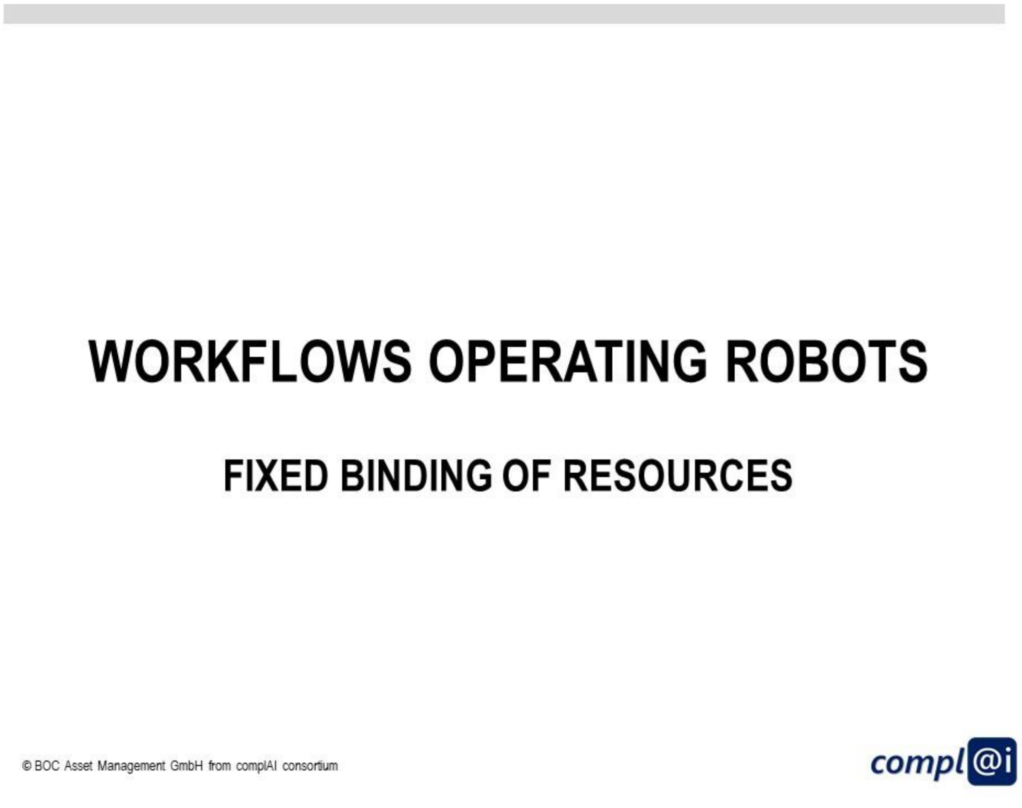
Possible Demonstration Settings

- OMiLAB at BOC Vienna
  - Connection to the BOCInnovationLAB network
  - Robot Arm Interface:
    http://10.0.6.32:8080/dobot/ui/

- External Location with Internet access
  - Robot Arm Interface:
    http://lab.boc-group.eu/dobot/ui/

All sample models are provided in the compl@i prototype package.

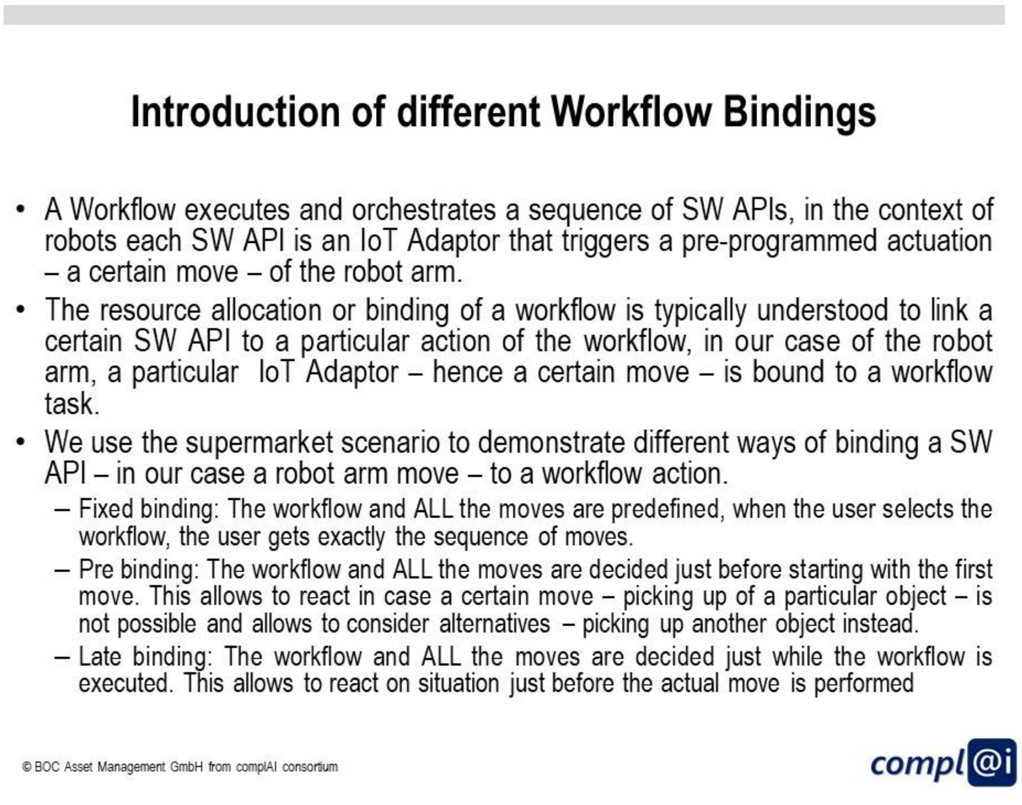
Appendix B



WORKFLOWS OPERATING ROBOTS

FIXED BINDING OF RESOURCES

© BOC Asset Management GmbH from complAI consortium



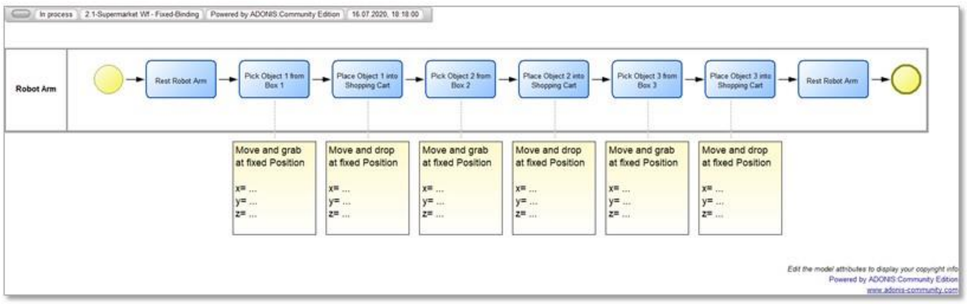
Introduction of different Workflow Bindings

- A Workflow executes and orchestrates a sequence of SW APIs, in the context of robots each SW API is an IoT Adaptor that triggers a pre-programmed actuation – a certain move – of the robot arm.
- The resource allocation or binding of a workflow is typically understood to link a certain SW API to a particular action of the workflow, in our case of the robot arm, a particular IoT Adaptor – hence a certain move – is bound to a workflow task.
- We use the supermarket scenario to demonstrate different ways of binding a SW API – in our case a robot arm move – to a workflow action.
  - Fixed binding: The workflow and ALL the moves are predefined, when the user selects the workflow, the user gets exactly the sequence of moves.
  - Pre binding: The workflow and ALL the moves are decided just before starting with the first move. This allows to react in case a certain move – picking up of a particular object – is not possible and allows to consider alternatives – picking up another object instead.
  - Late binding: The workflow and ALL the moves are decided just while the workflow is executed. This allows to react on situation just before the actual move is performed

© BOC Asset Management GmbH from complAI consortium

The Fixed Binding Workflow

Three objects – in our case apple, bananas and grapes are picked up in this sequence and put into the shopping cart.



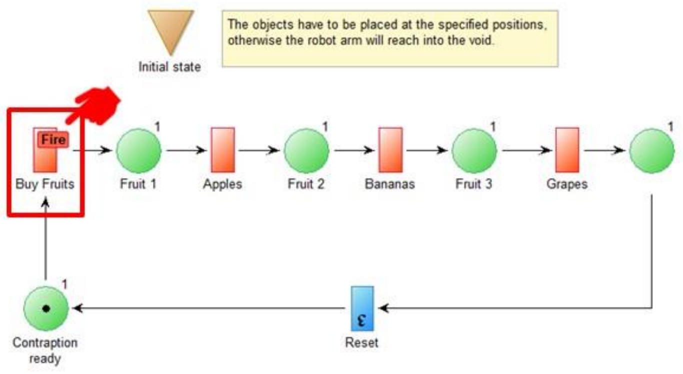
In process
2.1-Supermarket Wf - Fixed-Binding
Powered by ADONIS Community Edition
16.07.2020, 18:18:00
Robot Arm
Rest Robot Arm
Pick Object 1 from Box 1
Place Object 1 into Shopping Cart
Pick Object 2 from Box 2
Place Object 2 into Shopping Cart
Pick Object 3 from Box 3
Place Object 3 into Shopping Cart
Rest Robot Arm
Move and grab at fixed Position x= ... y= ... z= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at fixed Position x= ... y= ... z= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at fixed Position x= ... y= ... z= ...
Move and drop at fixed Position x= ... y= ... z= ...
Edit the model attributes to display your copyright info
Powered by ADONIS Community Edition
www.adonis-community.com

Assessment on:

- Is the robot arm safe, can people be hurt?
- Does the customer know the result?
- Is AI performing any decision the customer is unaware of?

INSTANTIATION OF THE FIXED BINDING WORKFLOW

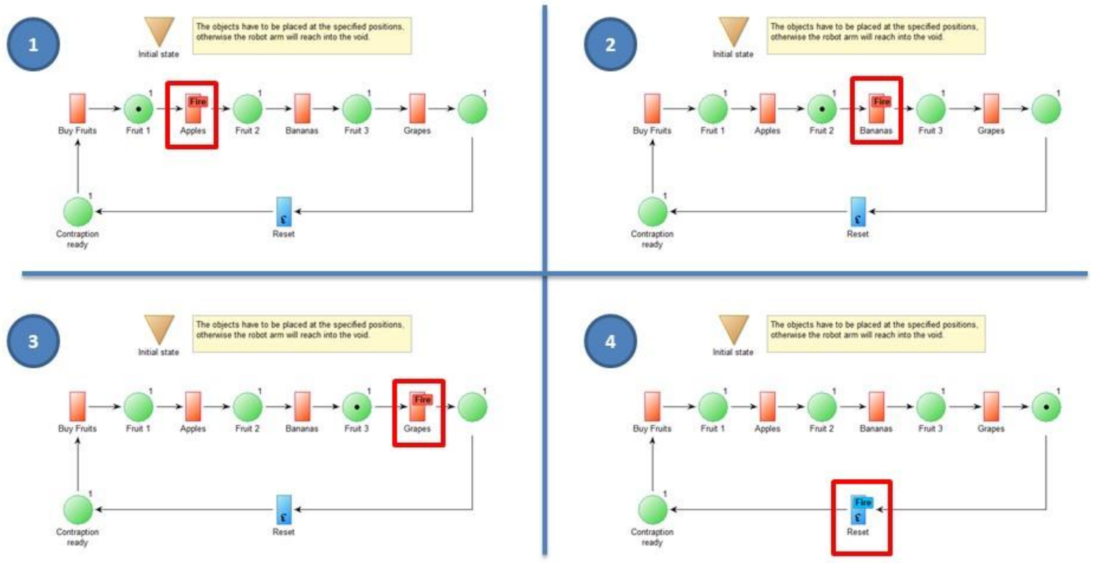
Instantiation using Petri Nets


Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Fire
Buy Fruits
Fruit 1
Apples
Fruit 2
Bananas
Fruit 3
Grapes
1
Contraption ready
Reset
ε

© BOC Asset Management GmbH from complAI consortium

compl@i

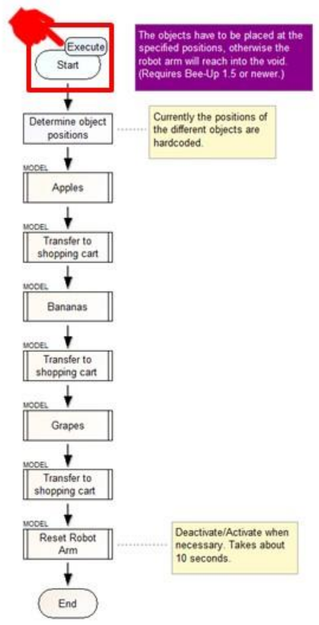
Instantiation using Petri Nets


1
2
3
4
Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Fire
Buy Fruits
Fruit 1
Apples
Fruit 2
Bananas
Fruit 3
Grapes
Contraption ready
Reset

© BOC Asset Management GmbH from complAI consortium

compl@i

Instantiation using Flowcharts

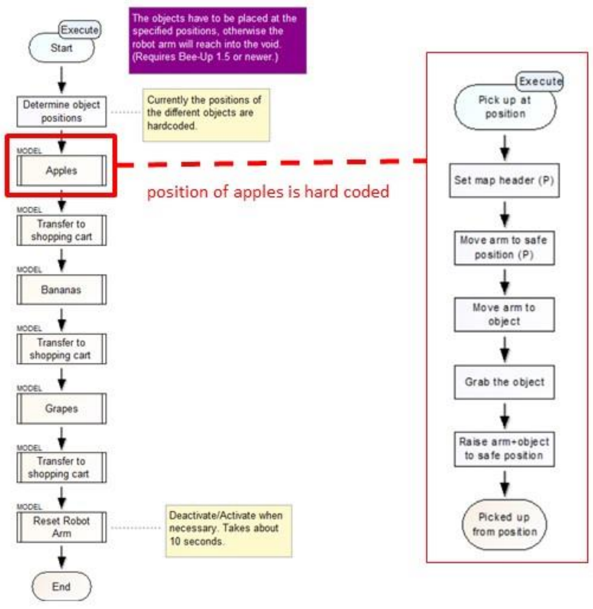

Execute
Start
Determine object positions
MODEL
Apples
MODEL
Transfer to shopping cart
MODEL
Bananas
MODEL
Transfer to shopping cart
MODEL
Grapes
MODEL
Transfer to shopping cart
MODEL
Reset Robot Arm
End

The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)

Currently the positions of the different objects are hardcoded.

Deactivate/Activate when necessary. Takes about 10 seconds.

Instantiation using Flowcharts

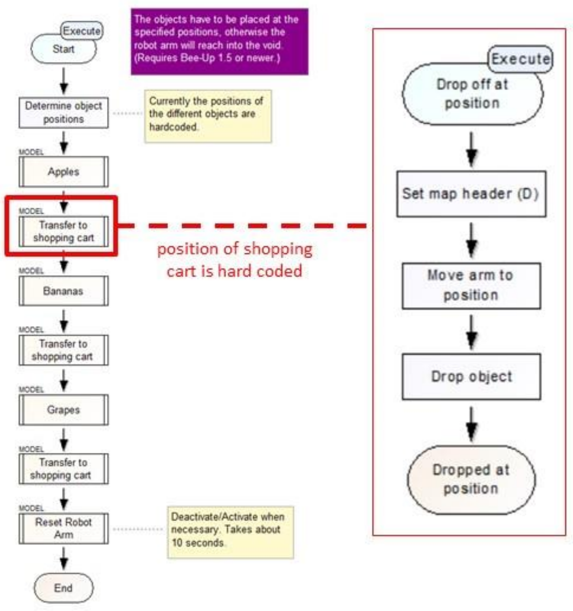

Execute
Start
Determine object positions
MODEL
Apples
MODEL
Transfer to shopping cart
MODEL
Bananas
MODEL
Transfer to shopping cart
MODEL
Grapes
MODEL
Transfer to shopping cart
MODEL
Reset Robot Arm
End
position of apples is hard coded
Execute
Pick up at position
Set map header (P)
Move arm to safe position (P)
Move arm to object
Grab the object
Raise arm+object to safe position
Picked up from position

The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)

Currently the positions of the different objects are hardcoded.

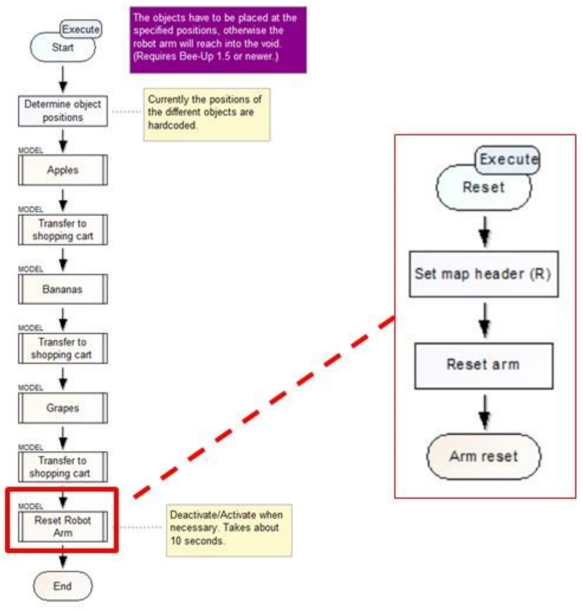
Deactivate/Activate when necessary. Takes about 10 seconds.

Instantiation using Flowcharts


Execute
Start
Determine object positions
MODEL
Apples
Transfer to shopping cart
Bananas
Transfer to shopping cart
Grapes
Transfer to shopping cart
Reset Robot Arm
End
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Currently the positions of the different objects are hardcoded.
Deactivate/Activate when necessary. Takes about 10 seconds.
position of shopping cart is hard coded
Execute
Drop off at position
Set map header (D)
Move arm to position
Drop object
Dropped at position

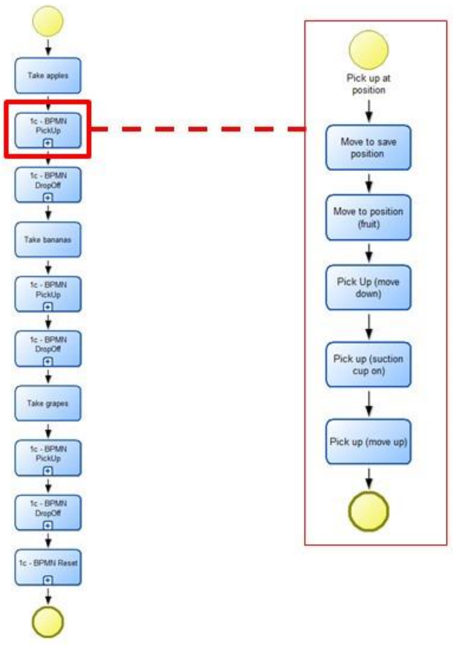
© BOC Asset Management GmbH from complAI consortium

Instantiation using Flowcharts

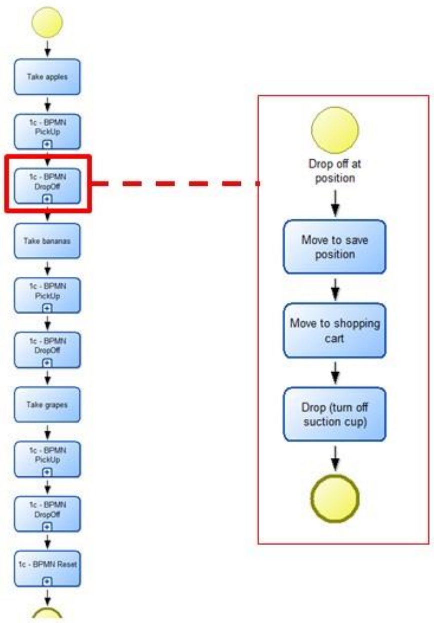

Execute
Start
Determine object positions
MODEL
Apples
Transfer to shopping cart
Bananas
Transfer to shopping cart
Grapes
Transfer to shopping cart
Reset Robot Arm
End
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Currently the positions of the different objects are hardcoded.
Deactivate/Activate when necessary. Takes about 10 seconds.
Execute
Reset
Set map header (R)
Reset arm
Arm reset

© BOC Asset Management GmbH from complAI consortium

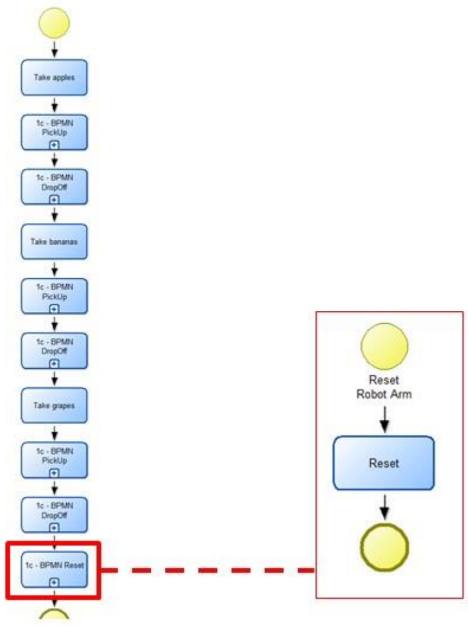
Instantiation using BPMN



Instantiation using BPMN

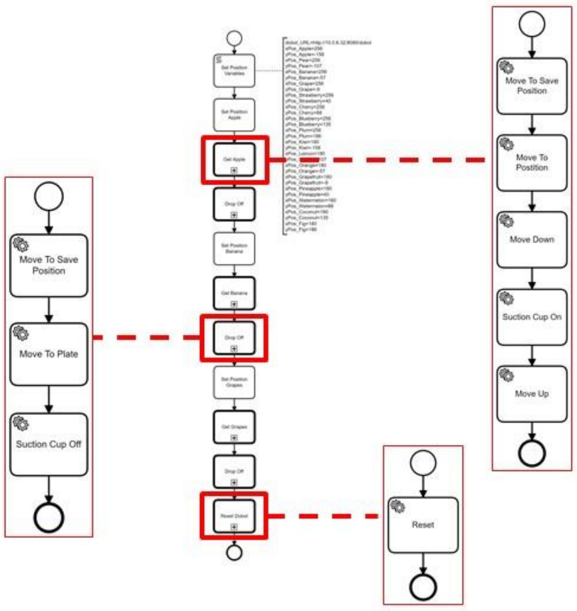


Instantiation using BPMN



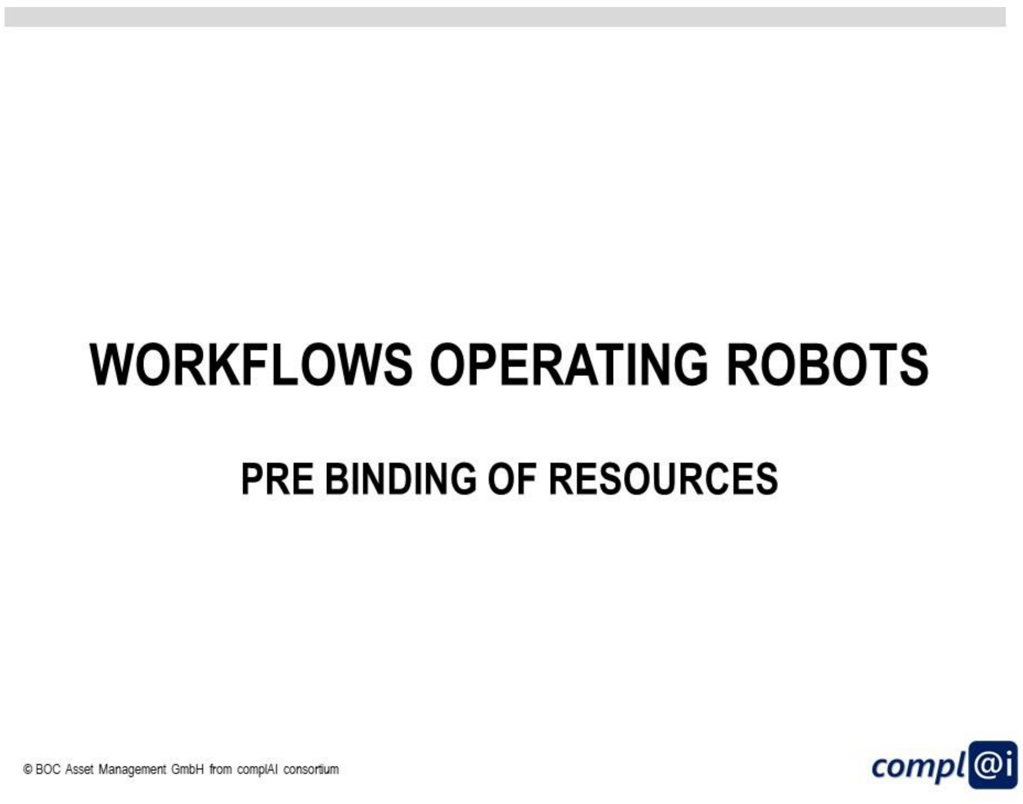
compl@i

Instantiation using a Workflow Engine



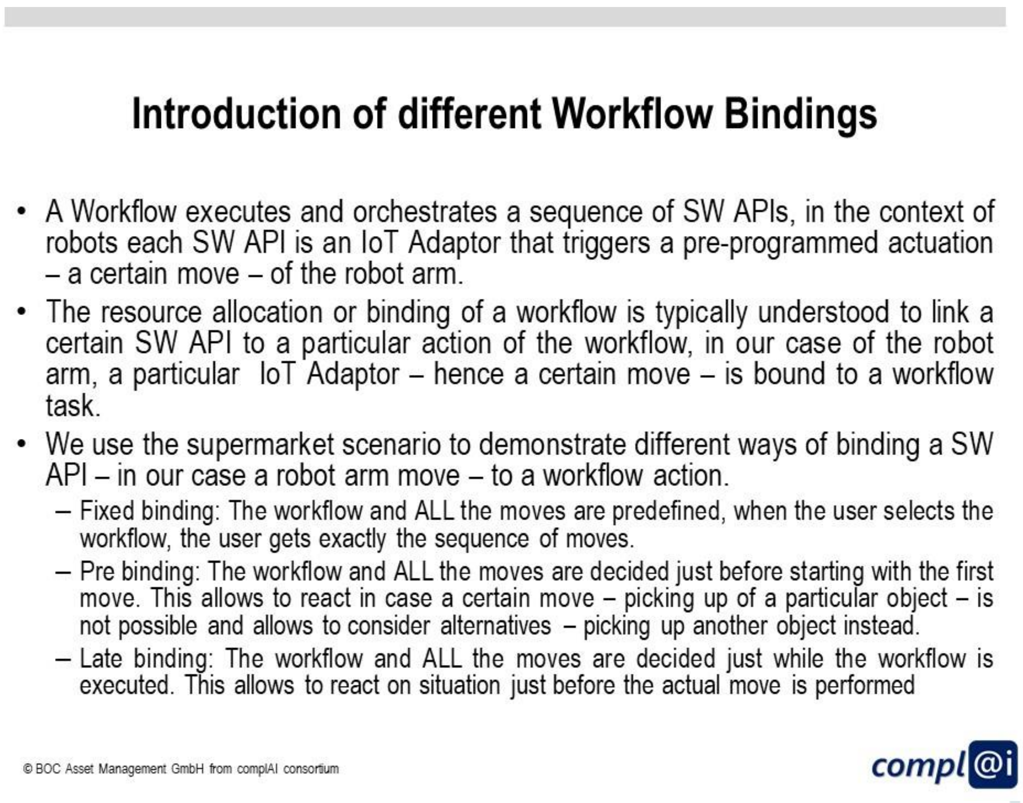
compl@i

Appendix C



WORKFLOWS OPERATING ROBOTS

PRE BINDING OF RESOURCES

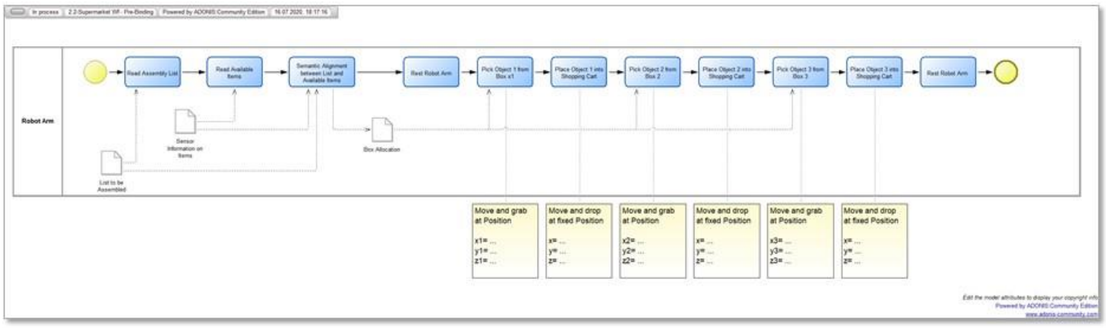


Introduction of different Workflow Bindings

- A Workflow executes and orchestrates a sequence of SW APIs, in the context of robots each SW API is an IoT Adaptor that triggers a pre-programmed actuation – a certain move – of the robot arm.
- The resource allocation or binding of a workflow is typically understood to link a certain SW API to a particular action of the workflow, in our case of the robot arm, a particular IoT Adaptor – hence a certain move – is bound to a workflow task.
- We use the supermarket scenario to demonstrate different ways of binding a SW API – in our case a robot arm move – to a workflow action.
  - Fixed binding: The workflow and ALL the moves are predefined, when the user selects the workflow, the user gets exactly the sequence of moves.
  - Pre binding: The workflow and ALL the moves are decided just before starting with the first move. This allows to react in case a certain move – picking up of a particular object – is not possible and allows to consider alternatives – picking up another object instead.
  - Late binding: The workflow and ALL the moves are decided just while the workflow is executed. This allows to react on situation just before the actual move is performed

The Pre Binding Workflow

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used; in case no bananas are available a strawberry is used; in case no grapes are available – blue berries are taken.

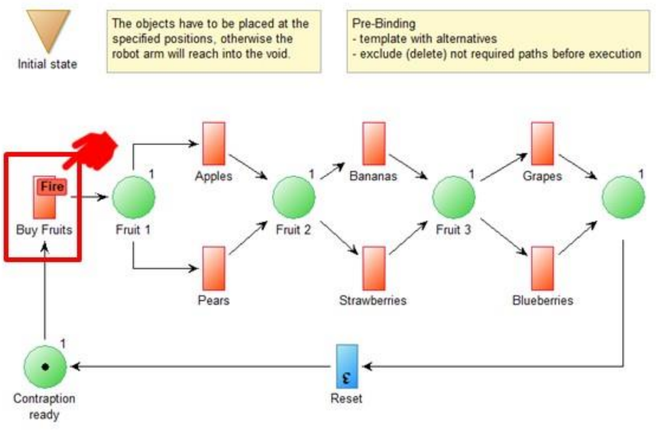

In process
2.2 Supermarket Wf - Pre-Binding
Powered by ADONIS Community Edition
16.07.2020, 18:17:16
Robot Arm
Read Assembly List
Read Available Items
Semantic Alignment between List and Available Items
Rest Robot Arm
Pick Object 1 from Box x1
Place Object 1 into Shopping Cart
Pick Object 2 from Box 2
Place Object 2 into Shopping Cart
Pick Object 3 from Box 3
Place Object 3 into Shopping Cart
Rest Robot Arm
Sensor Information on Items
List to be Assembled
Box Allocation
Move and grab at Position x1= ... y1= ... z1= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at Position x2= ... y2= ... z2= ...
Move and drop at fixed Position x= ... y= ... z= ...
Move and grab at Position x3= ... y3= ... z3= ...
Move and drop at fixed Position x= ... y= ... z= ...
Edit the model attributes to display your copyright info
Powered by ADONIS Community Edition
www.adonis-community.com

Assessment on:

- Is the robot arm safe, can people be hurt?
- Does the customer know the result?
- Is AI performing any decision the customer is unaware of?

INSTANTIATION OF THE PRE BINDING WORKFLOW

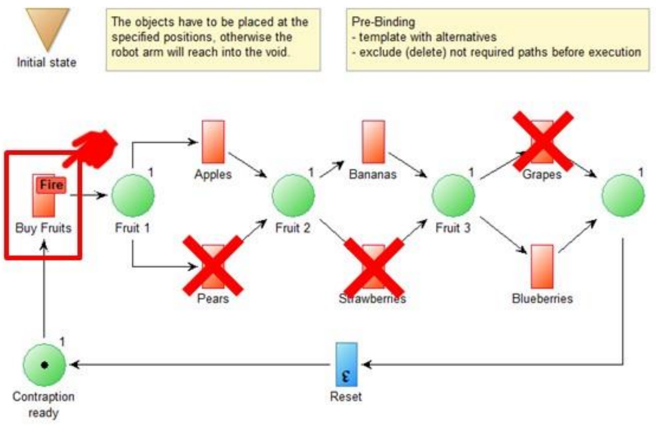
Instantiation using Petri Nets


Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Pre-Binding
- template with alternatives
- exclude (delete) not required paths before execution
Fire
Buy Fruits
Fruit 1
Apples
Pears
Fruit 2
Bananas
Strawberries
Fruit 3
Grapes
Blueberries
1
Contraption ready
Reset

Instantiation using Petri Nets

Pre-Binding

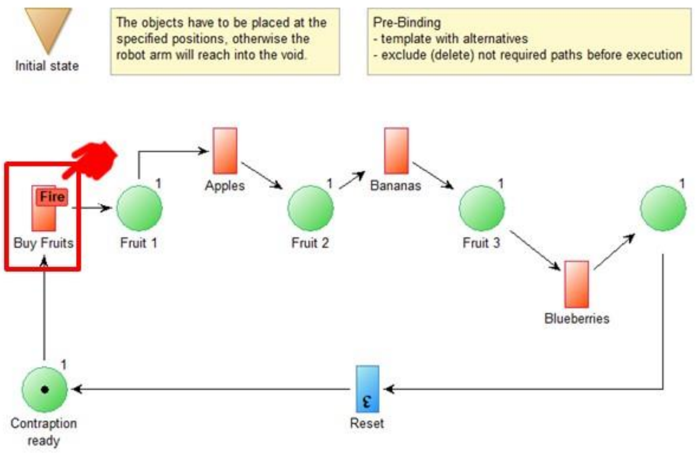
- Apples available
- Bananas available
- Grapes NOT available -> take blueberries


Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Pre-Binding
- template with alternatives
- exclude (delete) not required paths before execution
Fire
Buy Fruits
Fruit 1
Apples
Pears
Fruit 2
Bananas
Strawberries
Fruit 3
Grapes
Blueberries
1
Contraption ready
Reset

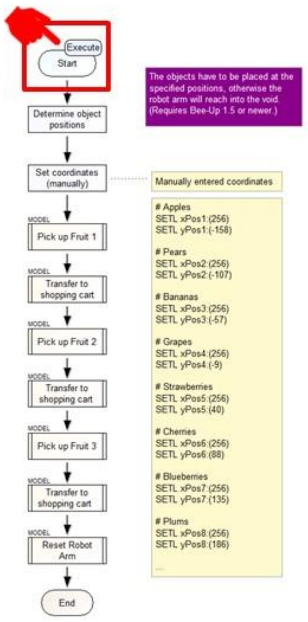
Instantiation using Petri Nets

Petri Net after Pre-Binding selection

- Apples available
- Bananas available
- Grapes NOT available -> take blueberries

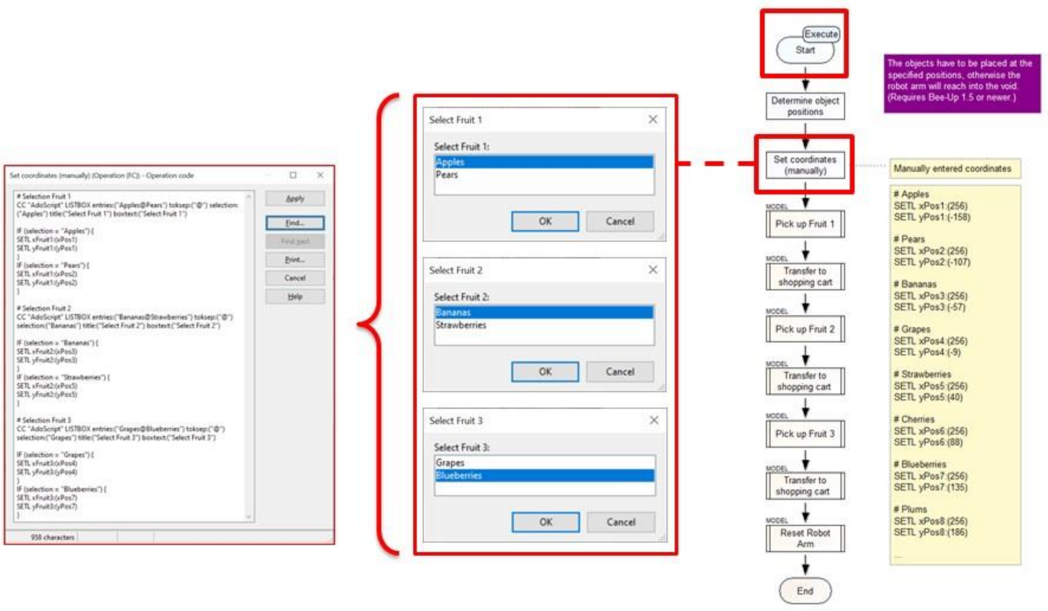

Initial state
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void.
Pre-Binding
- template with alternatives
- exclude (delete) not required paths before execution
Fire
Buy Fruits
Fruit 1
Apples
Fruit 2
Bananas
Fruit 3
Blueberries
1
Contraption ready
Reset
ε

Instantiation using Flowcharts


Execute
Start
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Determine object positions
Set coordinates (manually)
Manually entered coordinates
MODEL
Pick up Fruit 1
Transfer to shopping cart
Pick up Fruit 2
Transfer to shopping cart
Pick up Fruit 3
Transfer to shopping cart
Reset Robot Arm
End

```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

...
```

Instantiation using Flowcharts

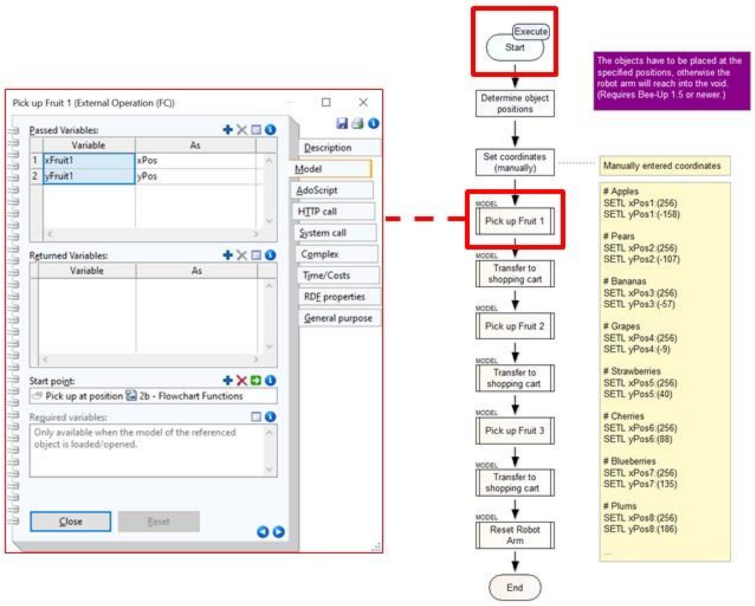

Set coordinates (manually) (Operation (FC)) - Operation code

```
# Selection Fruit 1
CC "AdoScript" LISTBOX entries:("Apples@Pears") toksep:("@") selection:
("Apples") title:("Select Fruit 1") boxtext:("Select Fruit 1")

IF (selection = "Apples") {
SETL xFruit1:(xPos1)
SETL yFruit1:(yPos1)
}
IF (selection = "Pears") {
SETL xFruit1:(xPos2)
SETL yFruit1:(yPos2)
}

# Selection Fruit 2
CC "AdoScript" LISTBOX entries:("Bananas@Strawberries") toksep:("@")
selection:("Bananas") title:("Select Fruit 2") boxtext:("Select Fruit 2")

IF (selection = "Bananas") {
SETL xFruit2:(xPos3)
SETL yFruit2:(yPos3)
}
IF (selection = "Strawberries") {
SETL xFruit2:(xPos5)
SETL yFruit2:(yPos5)
}

# Selection Fruit 3
CC "AdoScript" LISTBOX entries:("Grapes@Blueberries") toksep:("@")
selection:("Grapes") title:("Select Fruit 3") boxtext:("Select Fruit 3")

IF (selection = "Grapes") {
SETL xFruit3:(xPos4)
SETL yFruit3:(yPos4)
}
IF (selection = "Blueberries") {
SETL xFruit3:(xPos7)
SETL yFruit3:(yPos7)
}
```

958 characters

```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

...
```

Instantiation using Flowcharts


Pick up Fruit 1 (External Operation (FC))

Passed Variables:

	Variable	As
1	xFruit1	xPos
2	yFruit1	yPos

Returned Variables:

Variable	As

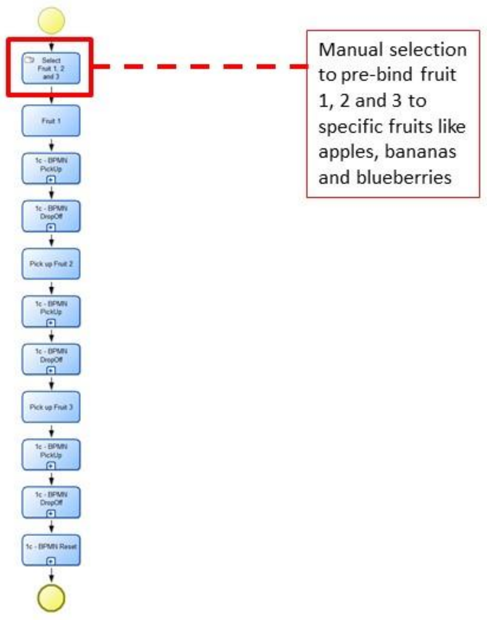
Start point: Pick up at position 2b - Flowchart Functions

Required variables: Only available when the model of the referenced object is loaded/opened.

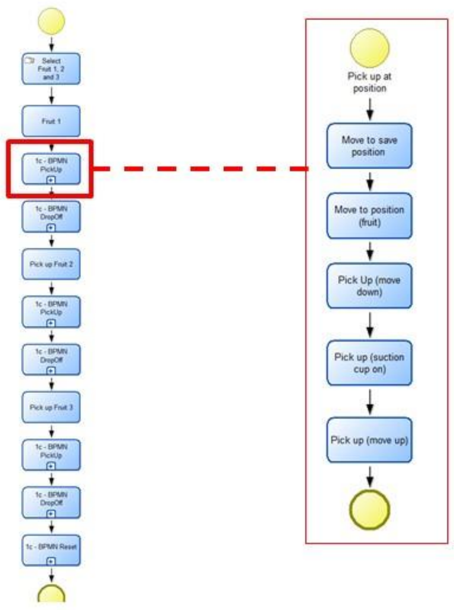
Description
Model
AdoScript
HTTP call
System call
Complex
Time/Costs
RDF properties
General purpose
Close
Reset
Execute
Start
Determine object positions
Set coordinates (manually)
MODEL
Pick up Fruit 1
MODEL
Transfer to shopping cart
MODEL
Pick up Fruit 2
MODEL
Transfer to shopping cart
MODEL
Pick up Fruit 3
MODEL
Transfer to shopping cart
MODEL
Reset Robot Arm
End
The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)
Manually entered coordinates

```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)
```

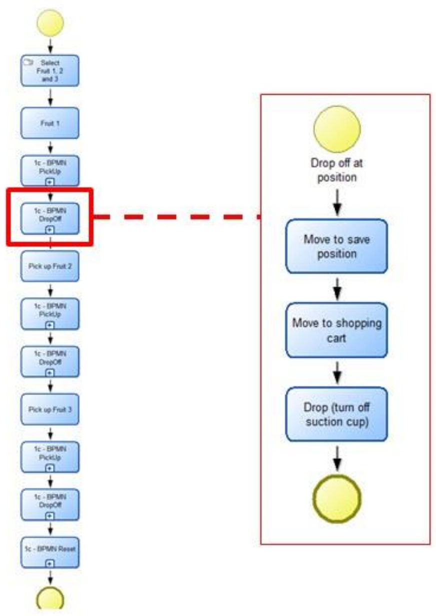
Instantiation using BPMN



Instantiation using BPMN

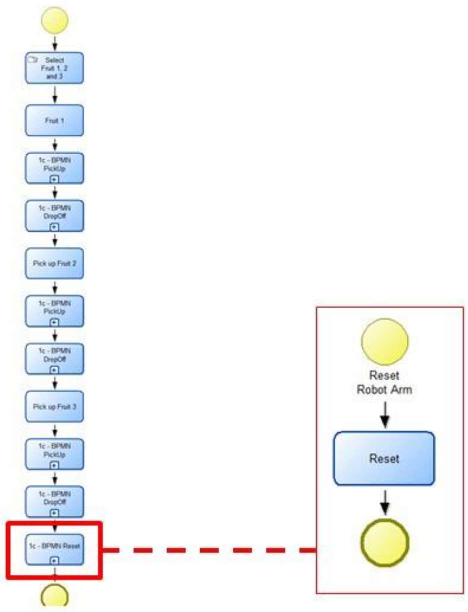


Instantiation using BPMN



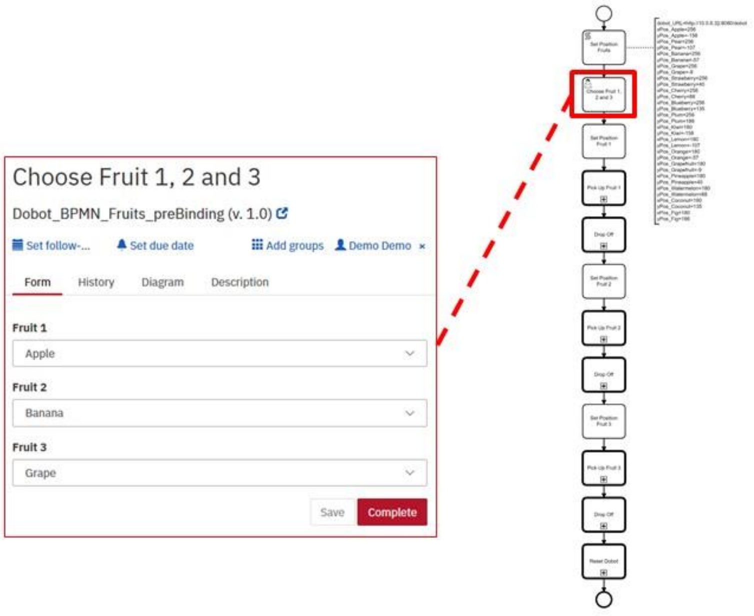
© BOC Asset Management GmbH from complAI consortium

Instantiation using BPMN



© BOC Asset Management GmbH from complAI consortium

Instantiation using a Workflow Engine



Choose Fruit 1, 2 and 3

Dobot_BPMN_Fruits_preBinding (v. 1.0)

Set follow-... Set due date Add groups Demo Demo ×

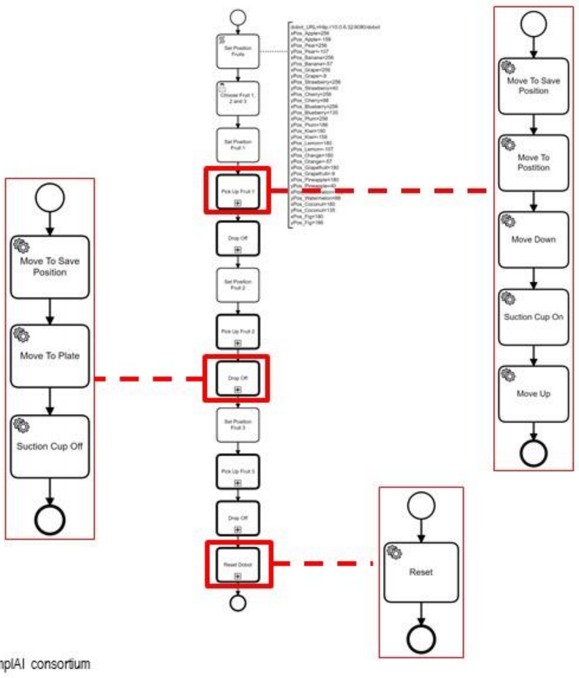
Form History Diagram Description

Fruit 1	Apple
Fruit 2	Banana
Fruit 3	Grape

Save Complete

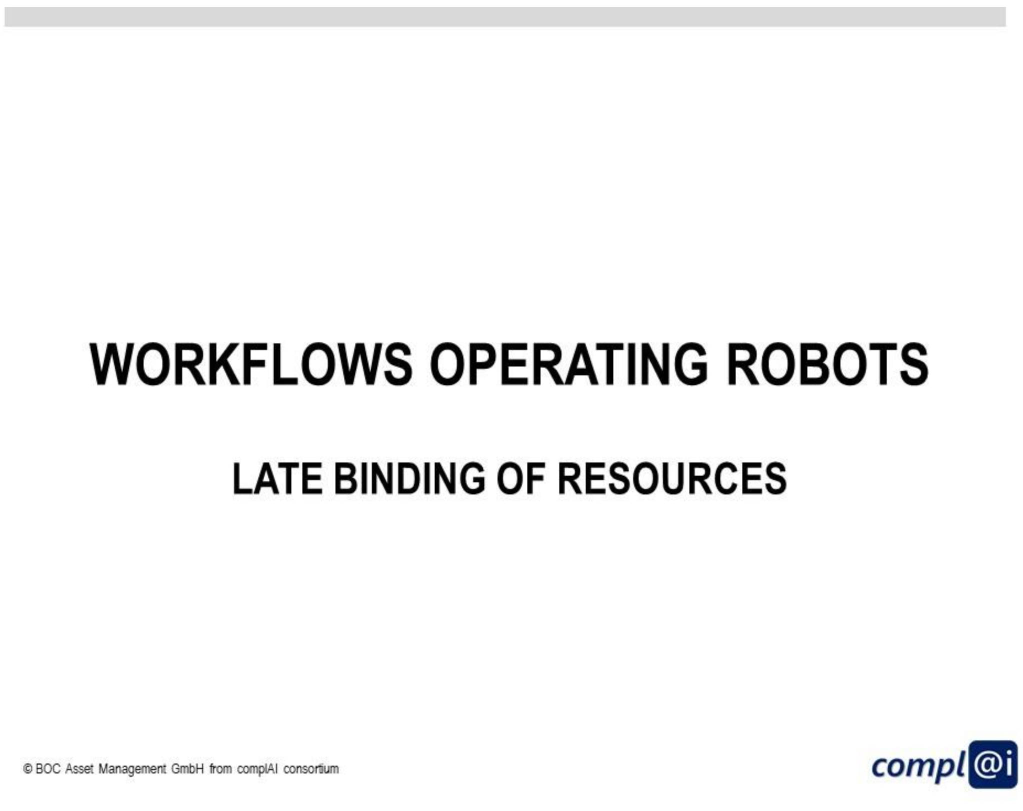
© BOC Asset Management GmbH from complAI consortium

Instantiation using a Workflow Engine



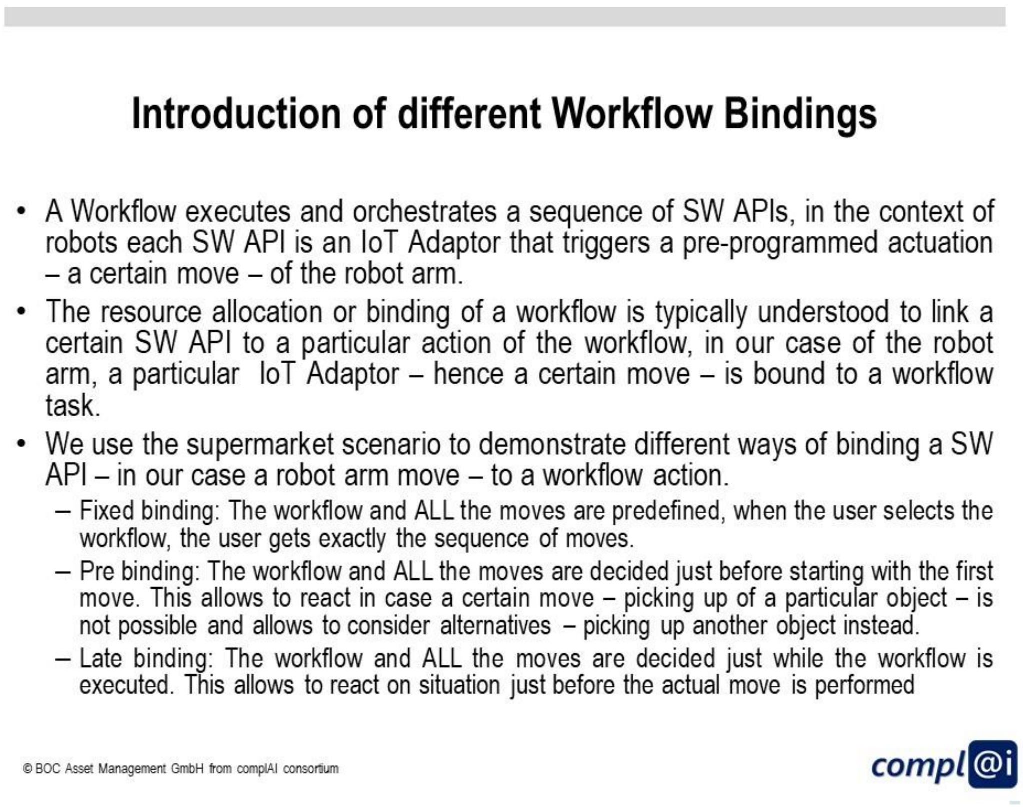
© BOC Asset Management GmbH from complAI consortium

Appendix D



WORKFLOWS OPERATING ROBOTS

LATE BINDING OF RESOURCES

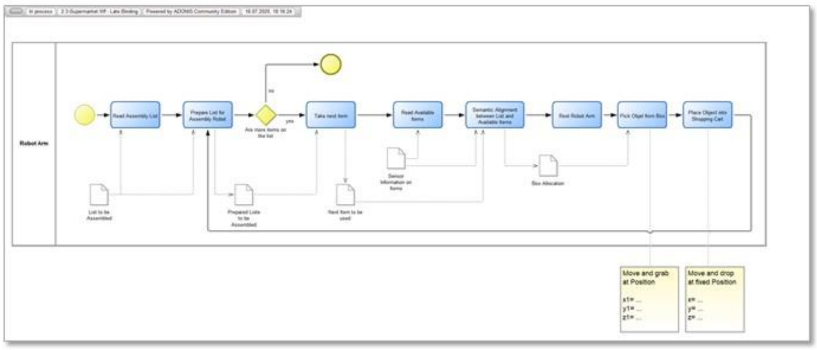


Introduction of different Workflow Bindings

- A Workflow executes and orchestrates a sequence of SW APIs, in the context of robots each SW API is an IoT Adaptor that triggers a pre-programmed actuation – a certain move – of the robot arm.
- The resource allocation or binding of a workflow is typically understood to link a certain SW API to a particular action of the workflow, in our case of the robot arm, a particular IoT Adaptor – hence a certain move – is bound to a workflow task.
- We use the supermarket scenario to demonstrate different ways of binding a SW API – in our case a robot arm move – to a workflow action.
  - Fixed binding: The workflow and ALL the moves are predefined, when the user selects the workflow, the user gets exactly the sequence of moves.
  - Pre binding: The workflow and ALL the moves are decided just before starting with the first move. This allows to react in case a certain move – picking up of a particular object – is not possible and allows to consider alternatives – picking up another object instead.
  - Late binding: The workflow and ALL the moves are decided just while the workflow is executed. This allows to react on situation just before the actual move is performed

The Late Binding Workflow

Three objects – in our case apple, bananas and grapes are picked up in this sequence – in case there are enough apples, bananas and grapes available - and put into the shopping cart. In case there are no apples available – a pear will be used; in case no bananas are available a strawberry is used; in case no grapes are available – blue berries are taken. The availability is not checked at the beginning of the workflow but at the beginning of each robot arm move.

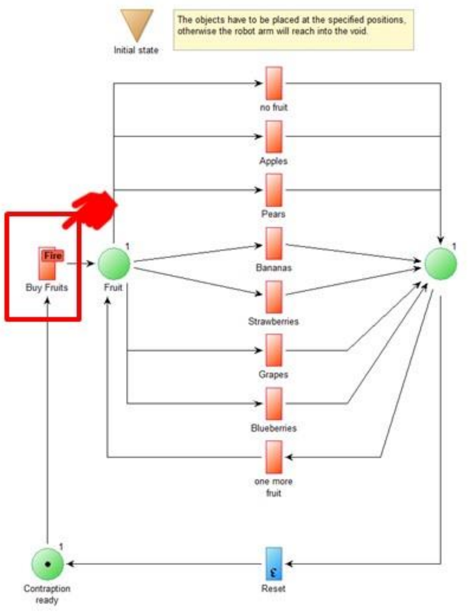

Robot Arm
Read Assembly List
Prepare List for Assembly Robot
Are more items on the list
no
yes
Take next item
Read Available Items
Semantic Alignment between List and Available Items
Read Robot Arm
Pick Objet from Box
Place Object into Shopping Cart
List to be Assembled
Prepared Lists to be Assembled
Next Item to be used
Sensor Information on Items
Box Allocation
Move and grab at Position
x1= ...
y1= ...
z1= ...
Move and drop at fixed Position
x= ...
y= ...
z= ...

Assessment on:

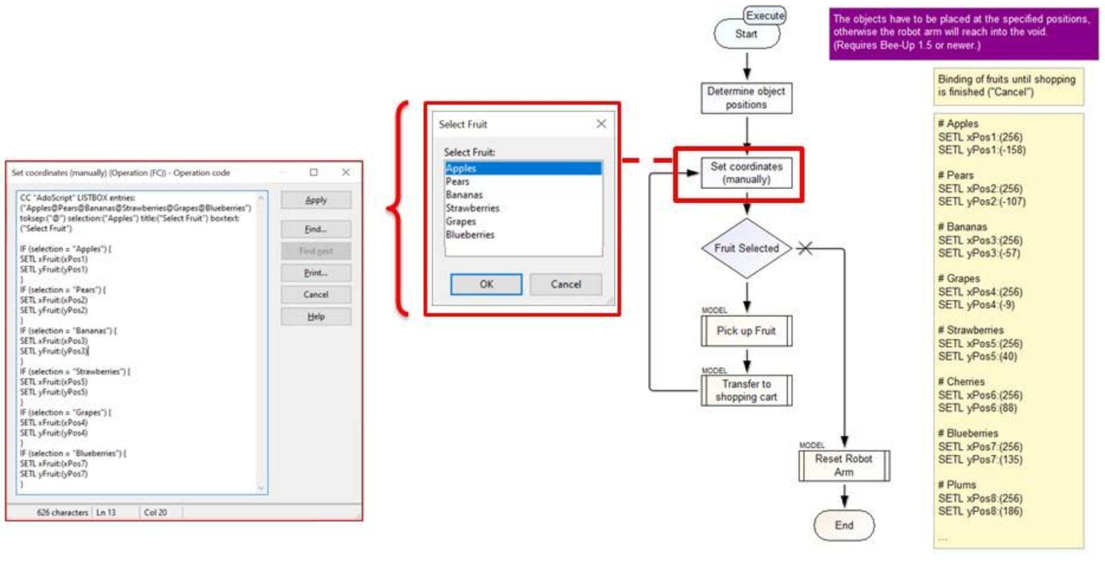
- Is the robot arm safe, can people be hurt?
- Does the customer know the result?
- Is AI performing any decision the customer is unaware of?

INSTANTIATION OF THE LATE BINDING WORKFLOW

Instantiation using Petri Nets



Instantiation using Flowcharts



Set coordinates (manually) (Operation (FC)) - Operation code

```
CC "AdoScript" LISTBOX entries:
("Apples@Pears@Bananas@Strawberries@Grapes@Blueberries")
toksep:("@") selection:("Apples") title:("Select Fruit") boxtext:
("Select Fruit")

IF (selection = "Apples") {
SETL xFruit:(xPos1)
SETL yFruit:(yPos1)
}
IF (selection = "Pears") {
SETL xFruit:(xPos2)
SETL yFruit:(yPos2)
}
IF (selection = "Bananas") {
SETL xFruit:(xPos3)
SETL yFruit:(yPos3)
}
IF (selection = "Strawberries") {
SETL xFruit:(xPos5)
SETL yFruit:(yPos5)
}
IF (selection = "Grapes") {
SETL xFruit:(xPos4)
SETL yFruit:(yPos4)
}
IF (selection = "Blueberries") {
SETL xFruit:(xPos7)
SETL yFruit:(yPos7)
}
```

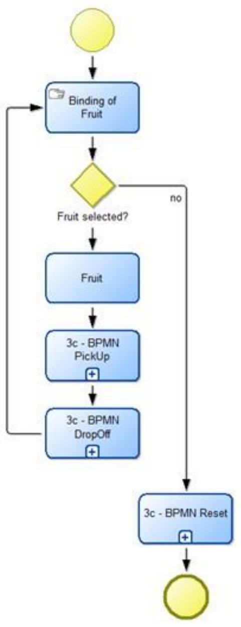
626 characters | Ln 13 | Col 20

The objects have to be placed at the specified positions, otherwise the robot arm will reach into the void. (Requires Bee-Up 1.5 or newer.)

Binding of fruits until shopping is finished ("Cancel")

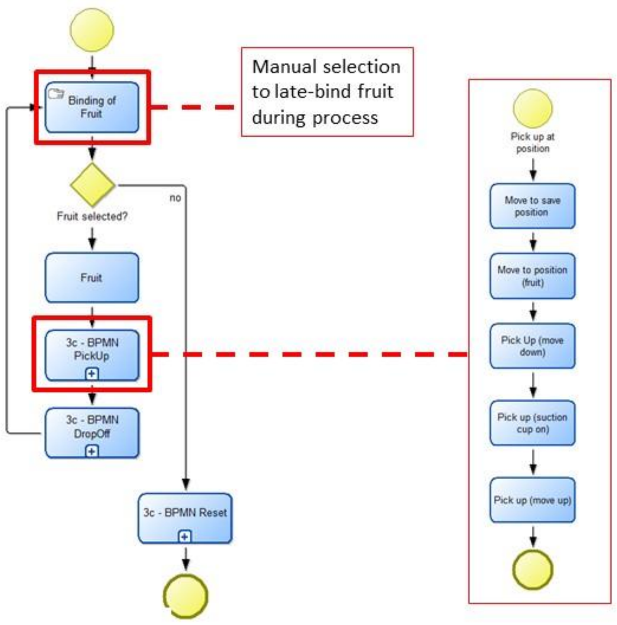
```
# Apples
SETL xPos1:(256)
SETL yPos1:(-158)

# Pears
SETL xPos2:(256)
SETL yPos2:(-107)

# Bananas
SETL xPos3:(256)
SETL yPos3:(-57)

# Grapes
SETL xPos4:(256)
SETL yPos4:(-9)

# Strawberries
SETL xPos5:(256)
SETL yPos5:(40)

# Cherries
SETL xPos6:(256)
SETL yPos6:(88)

# Blueberries
SETL xPos7:(256)
SETL yPos7:(135)

# Plums
SETL xPos8:(256)
SETL yPos8:(186)

...
```

Instantiation using BPMN



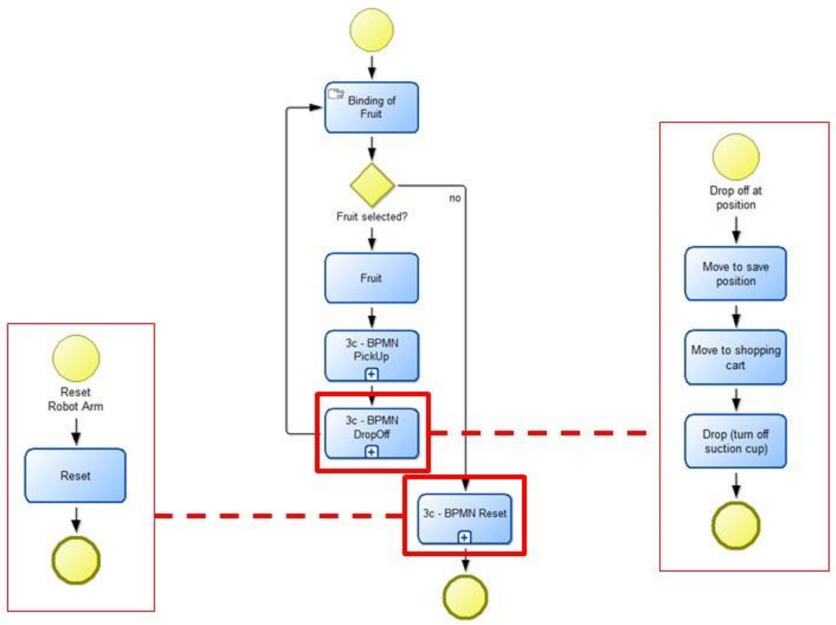
© BOC Asset Management GmbH from complAI consortium

Instantiation using BPMN



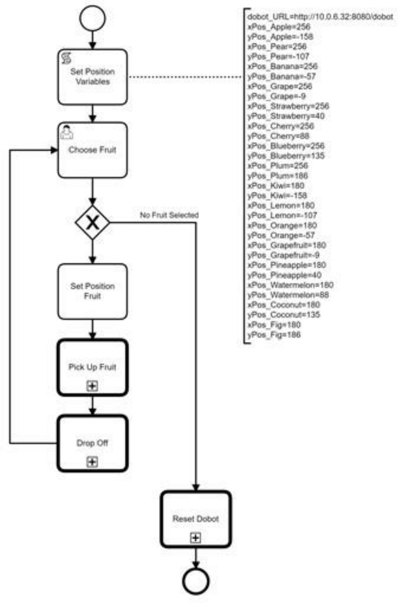
© BOC Asset Management GmbH from complAI consortium

Instantiation using BPMN



compl@i

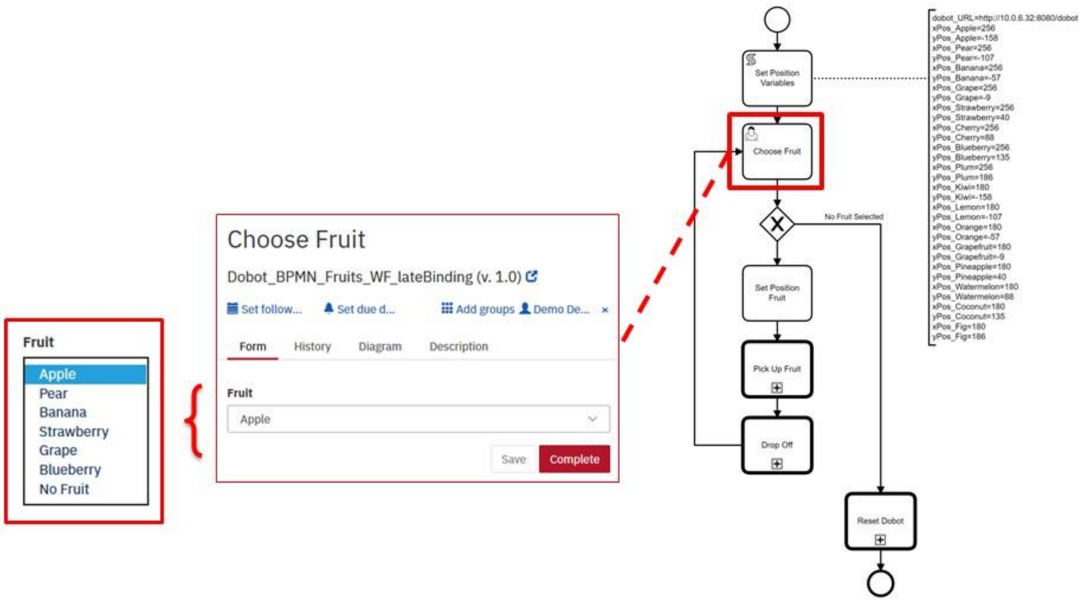
Instantiation using a Workflow Engine



```
dobot_URL=http://10.0.6.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
xPos_Fig=180
yPos_Fig=186
```

compl@i

Instantiation using a Workflow Engine



Fruit

Apple
Pear
Banana
Strawberry
Grape
Blueberry
No Fruit

Choose Fruit

Dobot_BPMN_Fruits_WF_lateBinding (v. 1.0)

Set follow... Set due d... Add groups Demo De... ×

Form History Diagram Description

Fruit: Apple

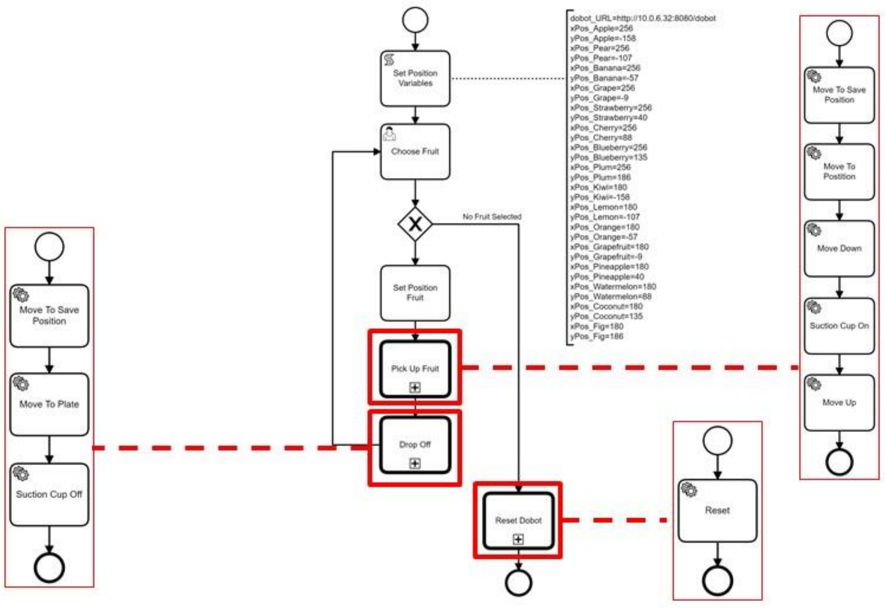
Save Complete

```
dobot_URL=http://10.0.6.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
xPos_Fig=180
yPos_Fig=186
```

© BOC Asset Management GmbH from complAI consortium

compl@i

Instantiation using a Workflow Engine



```
dobot_URL=http://10.0.6.32:8080/dobot
xPos_Apple=256
yPos_Apple=-158
xPos_Pear=256
yPos_Pear=-107
xPos_Banana=256
yPos_Banana=-57
xPos_Grape=256
yPos_Grape=-9
xPos_Strawberry=256
yPos_Strawberry=40
xPos_Cherry=256
yPos_Cherry=88
xPos_Blueberry=256
yPos_Blueberry=135
xPos_Plum=256
yPos_Plum=186
xPos_Kiwi=180
yPos_Kiwi=-158
xPos_Lemon=180
yPos_Lemon=-107
xPos_Orange=180
yPos_Orange=-57
xPos_Grapefruit=180
yPos_Grapefruit=-9
xPos_Pineapple=180
yPos_Pineapple=40
xPos_Watermelon=180
yPos_Watermelon=88
xPos_Coconut=180
yPos_Coconut=135
xPos_Fig=180
yPos_Fig=186
```

© BOC Asset Management GmbH from complAI consortium

compl@i