UNIVERSIDADE da MADEIRA

Madeira Interactive Technologies Institute

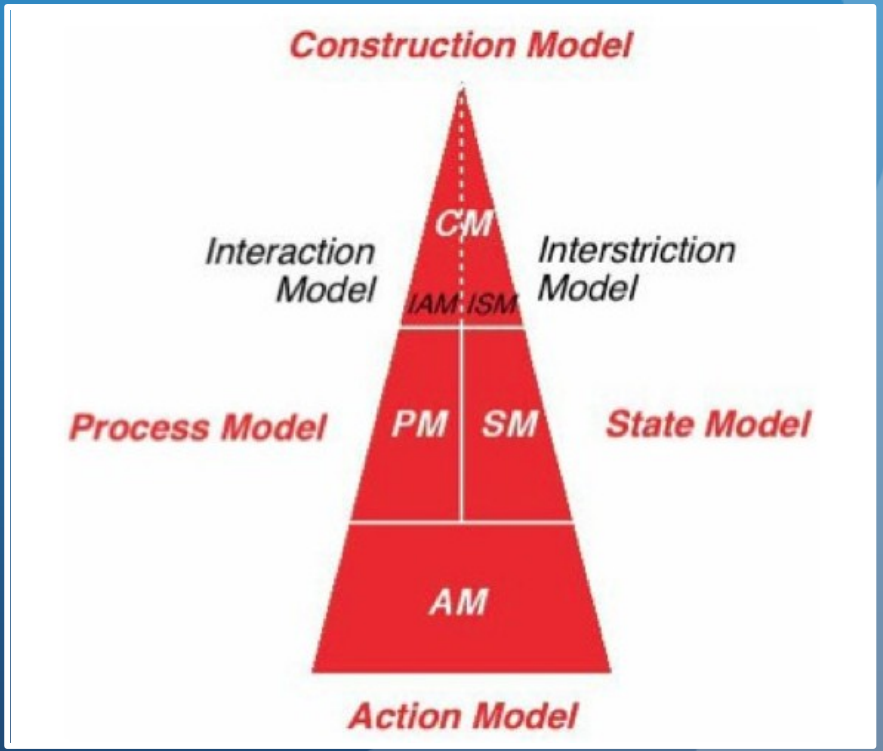
A new Action Rule Syntax for DEmo MOdels Based Automatic worKflow procEss geneRation (DEMOBAKER)

Carlos Figueira and David Aveiro

Research Context
DEMO PSI-Theory and method

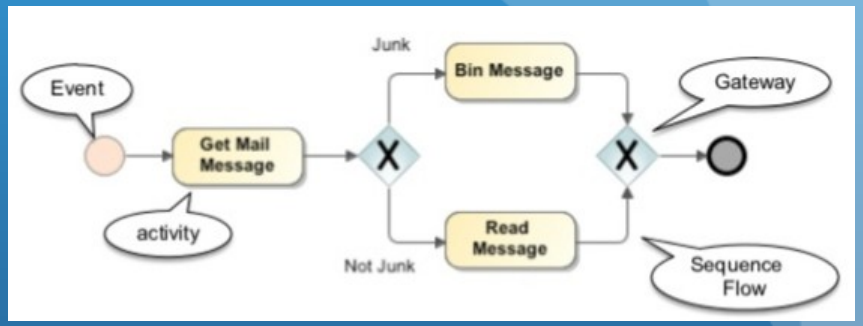
4 Model kinds:

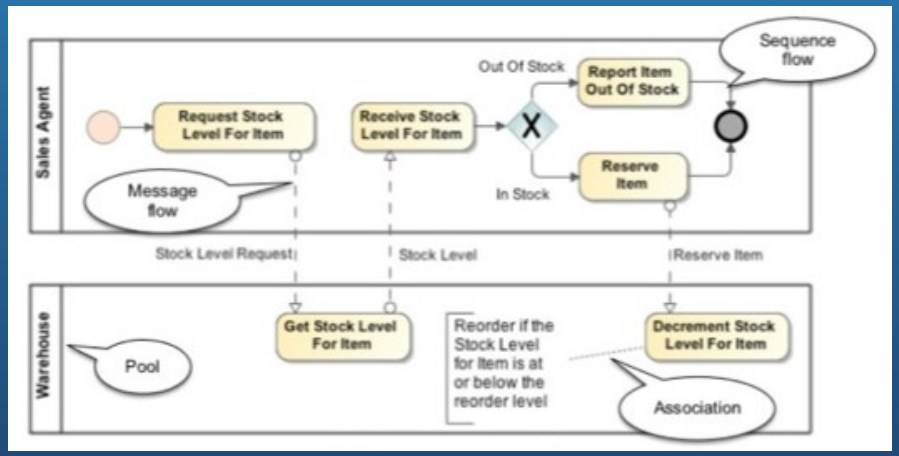
- **Construction Model**: specifies the construction of the organization and its transactions types;
- **Process Model**: contains for every transaction in the CM a specific transaction pattern and their causal and conditional relationships between transactions;
- **Action Model**: is the rule base that serves as guidelines for the actors actions;
- **State Model**: specifies the object classes, fact types, result types and the ontological coexistence rules



Research Context

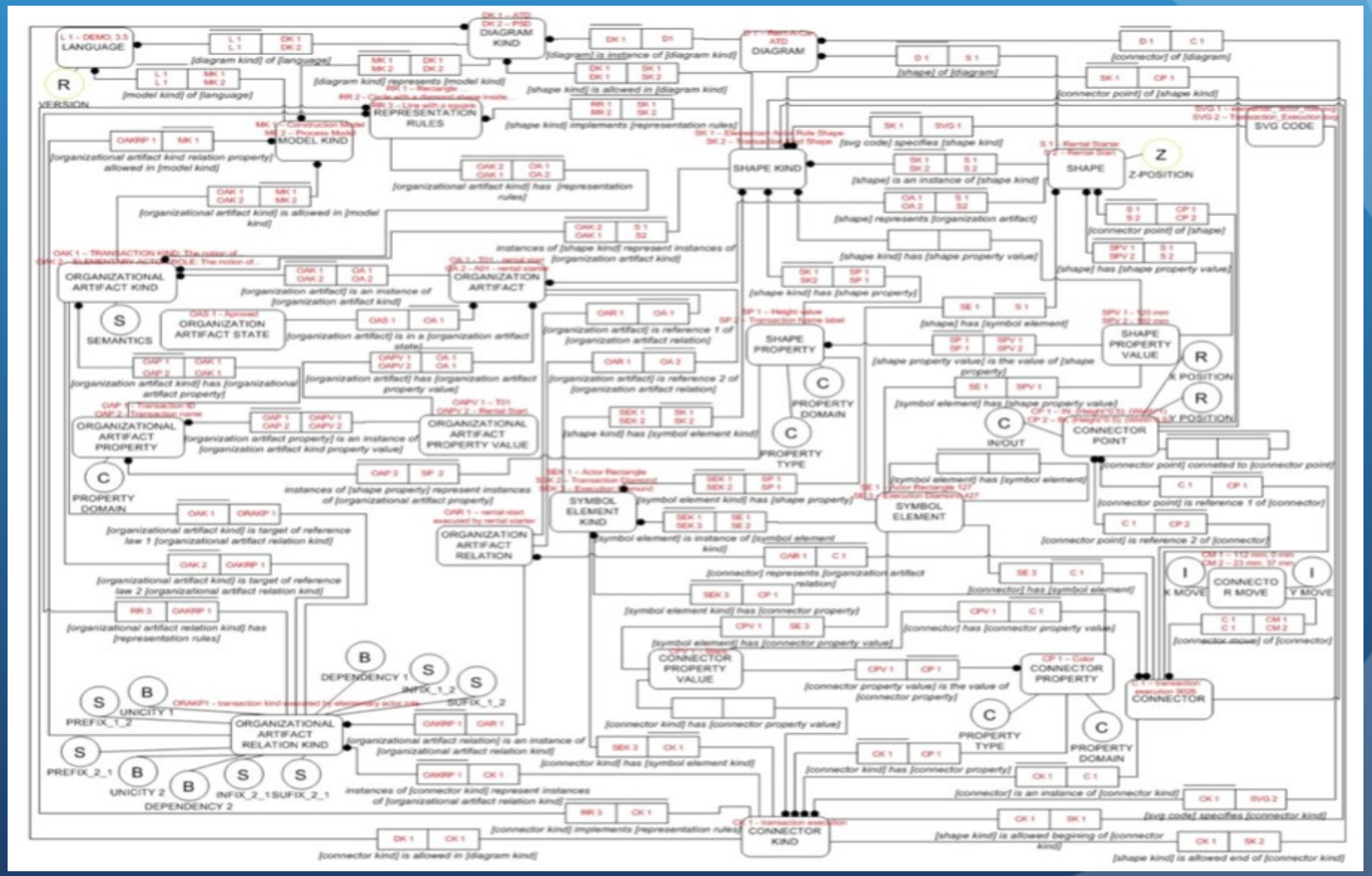
- Business process management
  - BPMN simple constructions





Research Context - the Universal Enterprise Adaptive Object Model

UNIVERSIDADE da MADEIRA

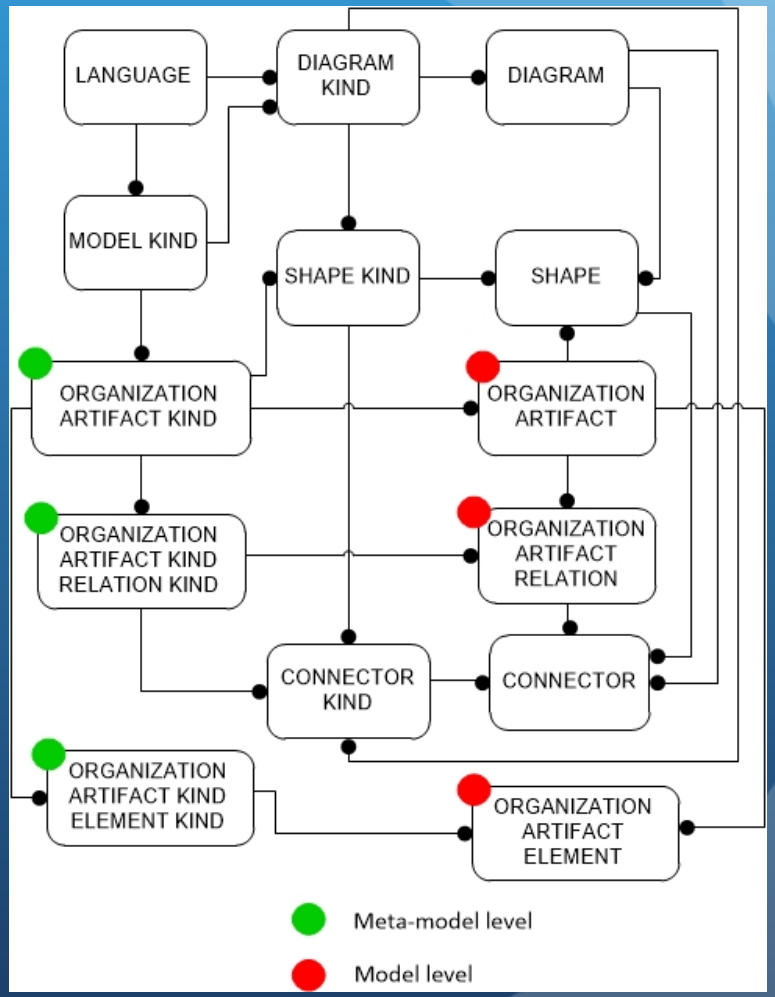


DEMO models based automatic workflow process generation

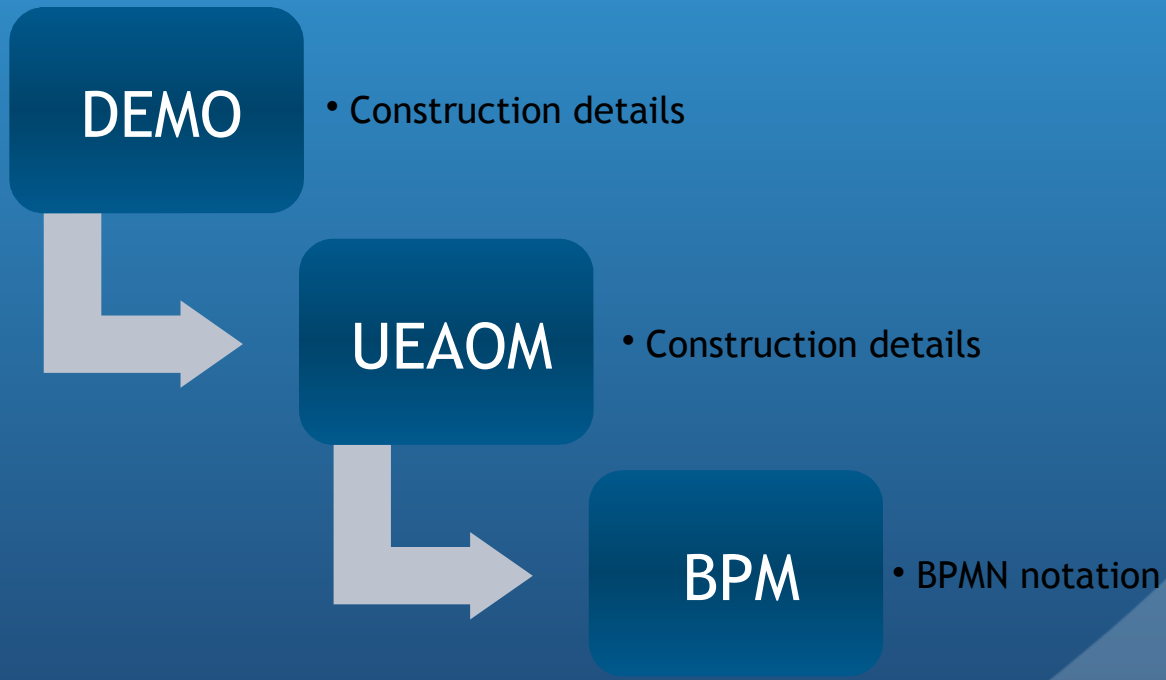
Research Context - the Universal Enterprise Adaptive Object Model

UNIVERSIDADE da MADEIRA

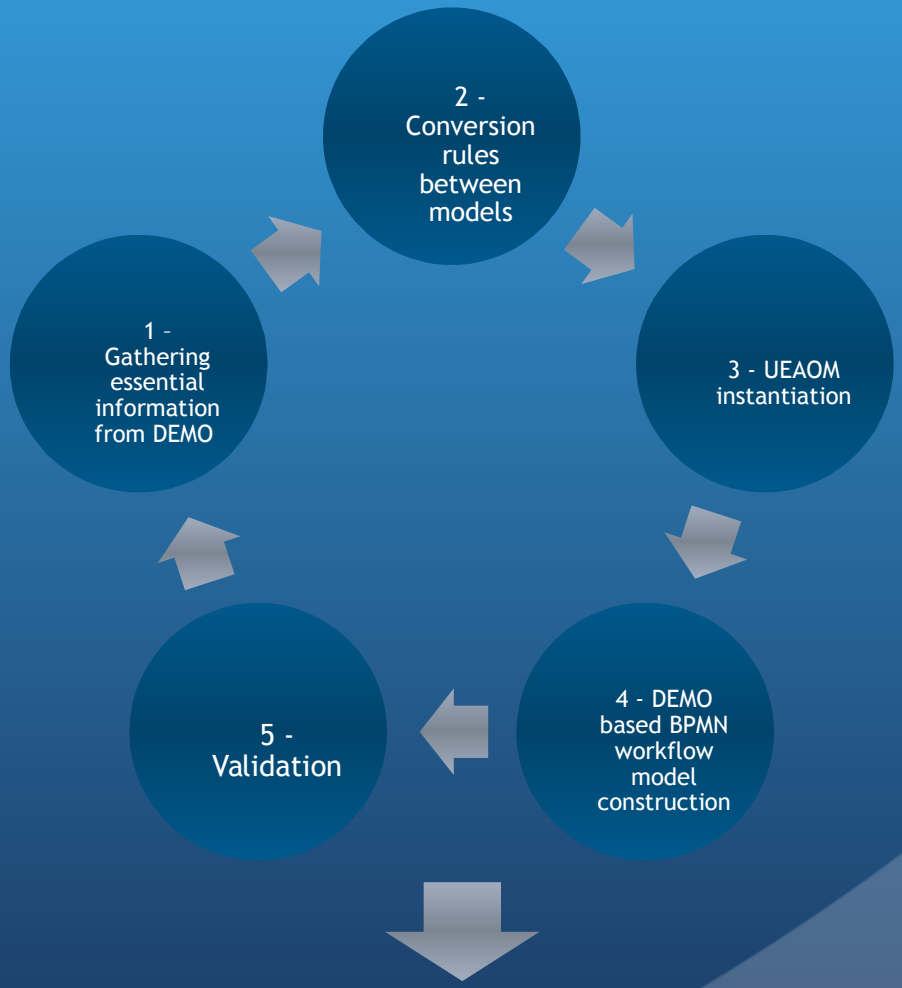
- UEAOM core classes:



Research approach



Research approach



Solution development

- Essential concepts in DEMO for Workflow generation:
  - Transactions and related actors (Construction Model)

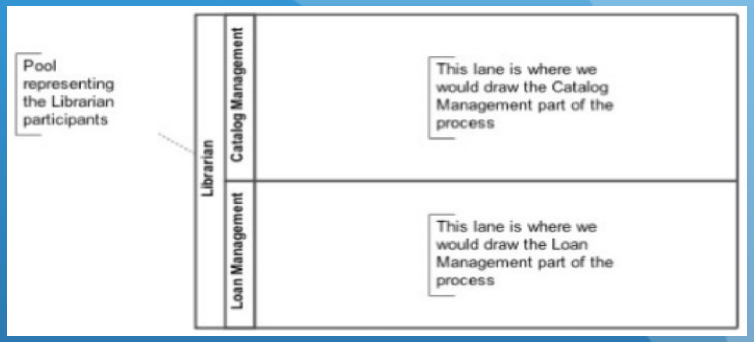
		Transaction kind	Transaction result	Initiator	Executor
	T01	rental start	[rental] has been started	renter	rental starter
	T02	rental end	[rental] has been ended	renter	rental ender
	T03	car pick-up	the car of [rental] has been picked-up	rental starter	driver
	T04	car drop-off	the car of [rental] has been dropped-off	rental starter	driver
	T05	rental payment	[rental] has been paid	rental ender	payer

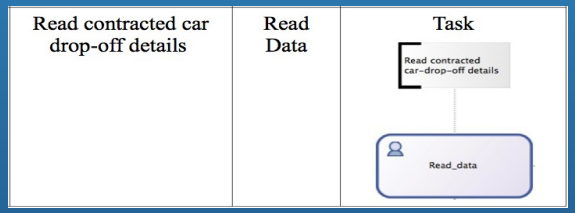
  - Action Rules (Action Model)

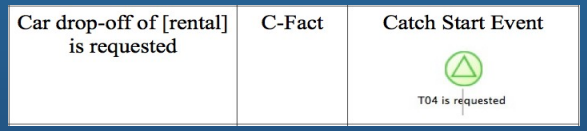
B-A01 rental-starter 5	WHEN	car drop-of of [rental] is stated
	with	the actual drop-off branch of [rental] is [branch]
	then	car drop-of of [rental] must be accepted

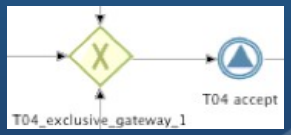
Solution development

- Conversion rules between models:
  - Each DEMO concept needs to have a 1 to 1 correspondence to a BPMN concept;
    - Transaction < > Pool
    - Actors < > Lane
    - Actions, Flows, Conditions < > Tasks/Activities
    - Coordination-facts/Production-facts, Action type C-Act/P-Act < > Signals (Throw and Catch Event)
    - Sequence merging and splitting < > Parallel and Exclusive Gateways









Solution development

- Based on the EU-Rent case
  - DEMO elements gathered from the case
    - Transactions (5 transactions):
    - Actors and their respective roles on each transaction:
    - Action Rules (11 action rules):

UNIVERSIDADE da MADEIRA

Solution development

- Action rule 05: T04 – car drop-off of [rental] is stated

B-A01 rental-starter 5	WHEN	car drop-of of [rental] is stated
	with	the actual drop-off branch of [rental] is [branch]
	then	car drop-of of [rental] must be accepted

  - Problem
    - Only verified fact is if the branch where the car is delivered is the same as the contracted one, but no action is specified for the case it is not
    - What is the meaning of the construct with?
      - We find it in many action rules and, apparently, with different functions: creating new facts, verifying new facts, etc
    - Dietz: “the syntax and the formal semantics of the action rules need to be elaborated yet”

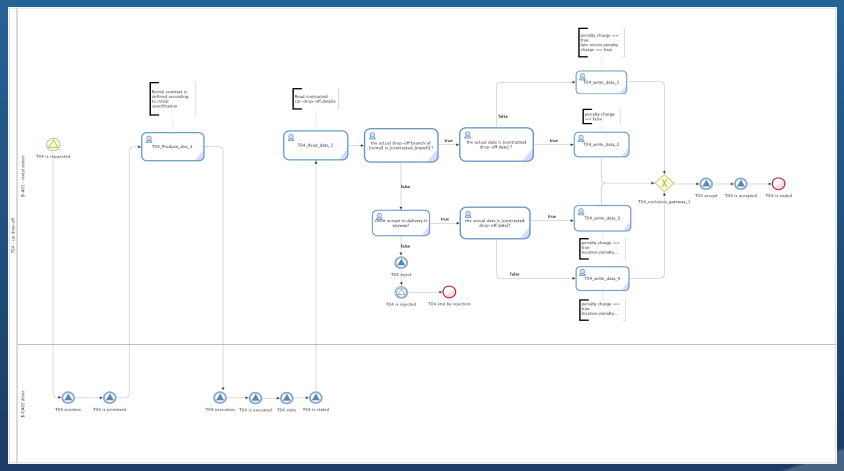
UNIVERSIDADE da MADEIRA

Solution evaluation

- Solution iteration:

AR05					
WHEN	car drop-off of [rental] is stated	C-FACT	ARCW01 (ARCET01)		
ACTION (FIRST, ATOMIC)	Read contracted car drop-off details	READ_DATA	ARCA01		
ACTION (FLOW)	If the actual drop-off branch of [rental] is [contracted_branch]	fact evaluation	ARCA02 (ARCIF01(ARCC01))		
then ACTION (FLOW)	If the actual date is [contracted drop-off date]	fact evaluation	ARCT01 (ARCA03(ARCIF02(ARCC02)))		
	then ACTION (FIRST, BLOCK)	ARCT02(ARCA04)			
	ACTION (FIRST, ATOMIC)	penalty charge == false	WRITE_DATA	ARCA05	
	ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA06	
	else ACTION (FIRST, BLOCK)	ARCE01(ARCA07)			
	ACTION (FIRST, ATOMIC)	penalty charge == true	WRITE_DATA	ARCA08	
	ACTION (AFTER, ATOMIC)	late return penalty charge == true	WRITE_DATA	ARCA09	
	ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA10	
else ACTION (FLOW)	If client accept to delivery it anyway	condition evaluation	ARCE02 (ARCA11(ARCIF03(ARCC03)))		
then ACTION (FLOW)	If the actual date is [contracted drop-off date]	fact evaluation	ARCT03(ARCA12(ARCIF04(ARCC02)))		
	then ACTION (FIRST, BLOCK)	ARCT04 (ARCA13)			
	ACTION (FIRST, ATOMIC)	penalty charge == true	WRITE_DATA	ARCA14	
	ACTION (AFTER, ATOMIC)	location penalty charge == true	WRITE_DATA	ARCA15	
	ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA16	
	else ACTION (FIRST, BLOCK)	ARCE03(ARCA17)			
	ACTION (FIRST, ATOMIC)	penalty charge == true	WRITE_DATA	ARCA18	
	ACTION (AFTER, ATOMIC)	location penalty charge == true	WRITE_DATA	ARCA19	
	ACTION (AFTER, ATOMIC)	late return penalty charge == true	WRITE_DATA	ARCA20	
	ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA21	
else ACTION (FIRST, ATOMIC)	car drop-off of [rental] must be rejected	C-ACT	ARCE04(ARCA22)		

Organization Artifact Kind	**Action Rule**	
OAK Relation Kind	**executing actor**	
OAK Element Kind	action rule id	<string>
Organization Artifact Kind	**AR-Component-WHEN**	
OAK Relation Kind	**when component part of action rule**	
OAK Element Kind	arcw-id	<string>
Organization Artifact Kind	**AR-Component-ENACTING TRANSACTION**	
OAK Relation Kind	**enacting transaction component part of when component**	
OAK Relation Kind	**enacting transaction**	
OAK Element Kind	enacting transaction C-fact	<requested, promised, stated, accepted, revoke request requested, revoke resquet allowed, revoke request refused, revoke promise requested, revoke promise allowed, revoke promise refused, revoke statement requested, revoke statement allowed, revoke statement refused, revoke acceptance requested, revoke acceptance allowed, revoke acceptance refused, rejected, declined>
OAK Element Kind	arcet-id	<string>
Organization Artifact Kind	**AR-Component-CONDITION**	
OAK Relation Kind	**subcondition of condition**	
OAK Element Kind	condition type	<AND, OR, NOT, EXPRESSION>
OAK Element Kind	expression	<string>
OAK Element Kind	arcc-id	<string>
Organization Artifact Kind	**AR-Component-IF**	
OAK Relation Kind	**flow component of action**	
OAK Relation Kind	**evaluated condition**	
OAK Element Kind	arci-id	<string>
Organization Artifact Kind	**AR-Component-FOREACH**	
OAK Relation Kind	**flow component of action**	
OAK Relation Kind	**type of element**	
OAK Element Kind	arcf-id	<string>
Organization Artifact Kind	**AR-Component-WHILE**	
OAK Relation Kind	**flow component of action**	
OAK Relation Kind	**evaluated condition**	
OAK Element Kind	arcwhile-id	<string>
Organization Artifact Kind	**AR-Component-THEN**	
OAK Relation Kind	**then component part of if component**	
OAK Element Kind	ar-et-id	<string>
Organization Artifact Kind	**AR-Component-ELSE**	
OAK Relation Kind	**else component part of if component**	
OAK Element Kind	arce-id	<string>
Organization Artifact Kind	**AR-Component-ACTION**	
OAK Relation Kind	**action component part of component**	
OAK Relation Kind	**previous action**	
OAK Relation Kind	**fact creation**	
OAK Element Kind	precedence type	<FIRST, AFTER>
OAK Element Kind	action type	<ATOMIC, BLOCK, FLOW>
OAK Element Kind	atomic action type	<WRITE_DATA, READ_DATA, TRANSMIT_DATA, SPECIFY_DATA, PRODUCE_DATA, PRODUCE_DOC, COPY_DOC, STORE_DOC, GET_DOC, READ_DOC, C-ACT, P-ACT>
OAK Element Kind	action description	<string>
OAK Element Kind	generic requirement	<string>
OAK Element Kind	implementation requirement	<string>
OAK Element Kind	arca-id	<string>



UNIVERSIDADE da MADEIRA

Solution

B-A01 rental-starter 5	WHEN	car drop-of of [rental] is stated
	with	the actual drop-off branch of [rental] is [branch]
	then	car drop-of of [rental] must be accepted

- EU-Rent action rule solution:

AR05						
WHEN	car drop-off of [rental] is stated	C-FACT	ARCW01 (ARCET01)			
ACTION (FIRST, ATOMIC)	Read contracted car drop-off details	READ_DATA	ARCA01			
ACTION (FLOW)	**if** the actual drop-off branch of [rental] is [contracted_branch]	fact evaluation	ARCA02 (ARCIF01(ARCC01))			
	then ACTION (FLOW)	**if** the actual date is [contracted drop-off date]	fact evaluation	ARCT01 (ARCA03(ARCIF02(ARCC02)))		
		then ACTION (FIRST, BLOCK)	ARCT02(ARCA04)			
		ACTION (FIRST, ATOMIC)	penalty charge == false	WRITE_DATA	ARCA05	
		ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA06	
		else ACTION (FIRST, BLOCK)	ARCE01(ARCA07)			
		ACTION (FIRST, ATOMIC)	penalty charge == true	WRITE_DATA	ARCA08	
		ACTION (AFTER, ATOMIC)	late return penalty charge == true	WRITE_DATA	ARCA09	
		ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA10	
	else ACTION (FLOW)	**if** client accept to delivery it anyway	condition evaluation	ARCE02 (ARCA11(ARCIF03(ARCC03)))		
		then ACTION (FLOW)	**if** the actual date is [contracted drop-off date]	fact evaluation	ARCT03(ARCA12(ARCIF04(ARCC02)))	
			then ACTION (FIRST, BLOCK)	ARCT04 (ARCA13)		
			ACTION (FIRST, ATOMIC)	penalty charge == true	WRITE_DATA	ARCA14
			ACTION (AFTER, ATOMIC)	location penalty charge == true	WRITE_DATA	ARCA15
			ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA16
			else ACTION (FIRST, BLOCK)	ARCE03(ARCA17)		
			ACTION (FIRST, ATOMIC)	penalty charge == true	WRITE_DATA	ARCA18
			ACTION (AFTER, ATOMIC)	location penalty charge == true	WRITE_DATA	ARCA19
			ACTION (AFTER, ATOMIC)	late return penalty charge == true	WRITE_DATA	ARCA20
			ACTION (AFTER, ATOMIC)	car drop-off of [rental] must be accepted	C-ACT	ARCA21
		else ACTION (FIRST, ATOMIC)	car drop-off of [rental] must be rejected	C-ACT	ARCE04(ARCA22)	

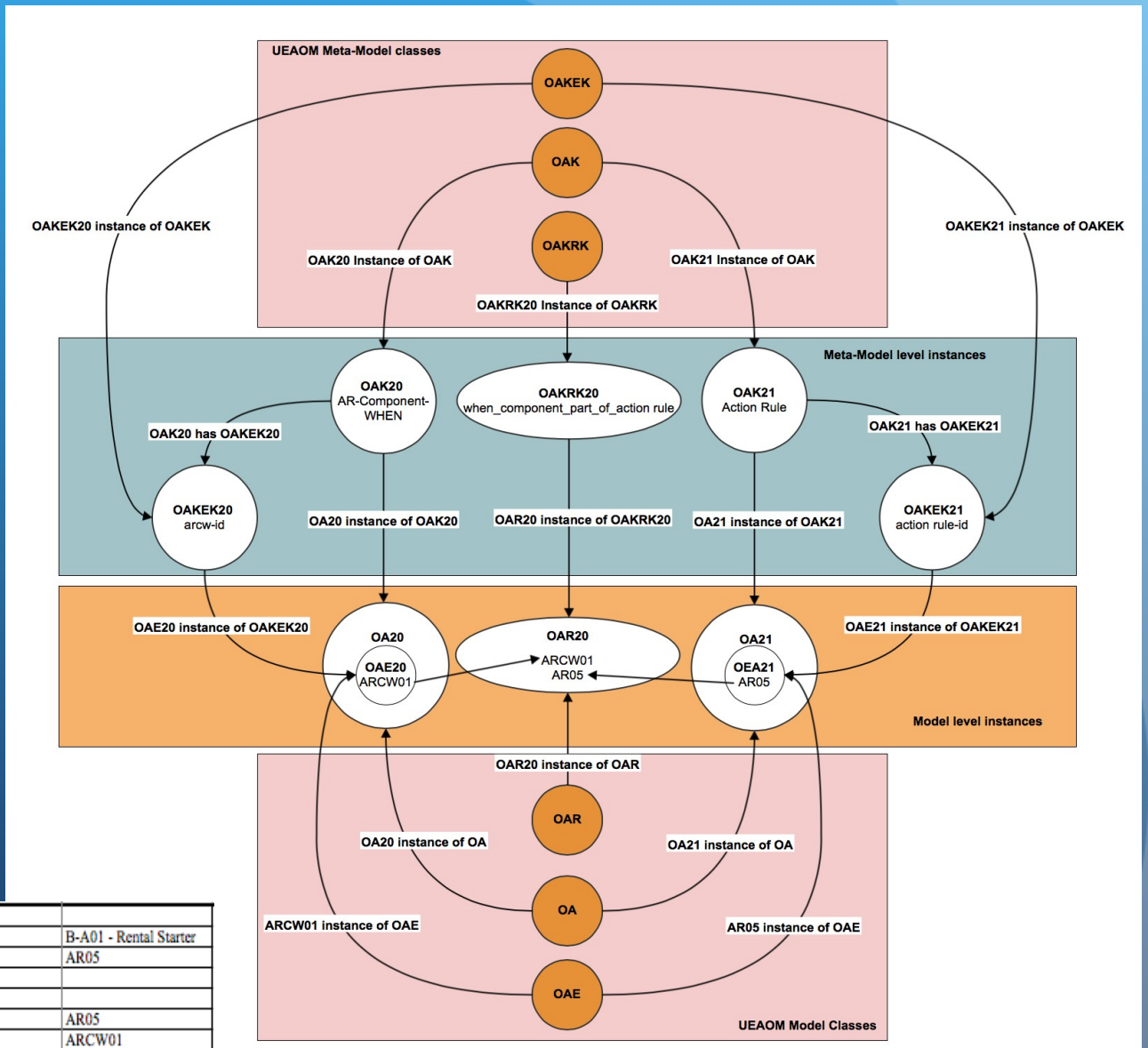
Solution

- UEAOM instantiation

Organization Artifact Kind	**Action Rule**	
OAK Relation Kind	**executing actor**	
OAK Element Kind	action rule id	<string>
Organization Artifact Kind	**AR-Component-WHEN**	
OAK Relation Kind	**when component part of action rule**	
OAK Element Kind	arcw-id	<string>
Organization Artifact Kind	**AR-Component-ENACTING TRANSACTION**	
OAK Relation Kind	**enacting transaction component part of when component**	
OAK Relation Kind	**enacting transaction**	
OAK Element Kind	enacting transaction C-fact	<requested, promised, stated, accepted, revoke request requested, revoke resquet allowed, revoke request refused, revoke promise requested, revoke promise allowed, revoke promise refused, revoke statement requested, revoke statement allowed, revoke statement refused, revoke acceptance requested, revoke acceptance allowed, revoke acceptance refused, rejected, declined>
OAK Element Kind	arcet-id	<string>
Organization Artifact Kind	**AR-Component-CONDITION**	
OAK Relation Kind	**subcondition of condition**	
OAK Element Kind	condition type	<AND, OR, NOT, EXPRESSION>
OAK Element Kind	expression	<string>
OAK Element Kind	arcc-id	<string>
Organization Artifact Kind	**AR-Component-IF**	
OAK Relation Kind	**flow component of action**	
OAK Relation Kind	**evaluated condition**	
OAK Element Kind	arci-id	<string>
Organization Artifact Kind	**AR-Component-FOREACH**	
OAK Relation Kind	**flow component of action**	
OAK Relation Kind	**type of element**	
OAK Element Kind	arcf-id	<string>
Organization Artifact Kind	**AR-Component-WHILE**	
OAK Relation Kind	**flow component of action**	
OAK Relation Kind	**evaluated condition**	
OAK Element Kind	arcwhile-id	<string>
Organization Artifact Kind	**AR-Component-THEN**	
OAK Relation Kind	**then component part of if component**	
OAK Element Kind	ar-ct-id	<string>
Organization Artifact Kind	**AR-Component-ELSE**	
OAK Relation Kind	**else component part of if component**	
OAK Element Kind	arce-id	<string>
Organization Artifact Kind	**AR-Component-ACTION**	
OAK Relation Kind	**action component part of component**	
OAK Relation Kind	**previous action**	
OAK Relation Kind	**fact creation**	
OAK Element Kind	precedence type	<FIRST, AFTER>
OAK Element Kind	action type	<ATOMIC, BLOCK, FLOW>
OAK Element Kind	atomic action type	<WRITE_DATA, READ_DATA, TRANSMIT_DATA, SPECIFY_DATA, PRODUCE_DATA, PRODUCE_DOC, COPY_DOC, STORE_DOC, GET_DOC, READ_DOC, C-ACT, P-ACT>
OAK Element Kind	action description	<string>
OAK Element Kind	generic requirement	<string>
OAK Element Kind	implementation requirement	<string>
OAK Element Kind	arca-id	<string>

Solution

- UEAOM EU-Rent case instantiation example

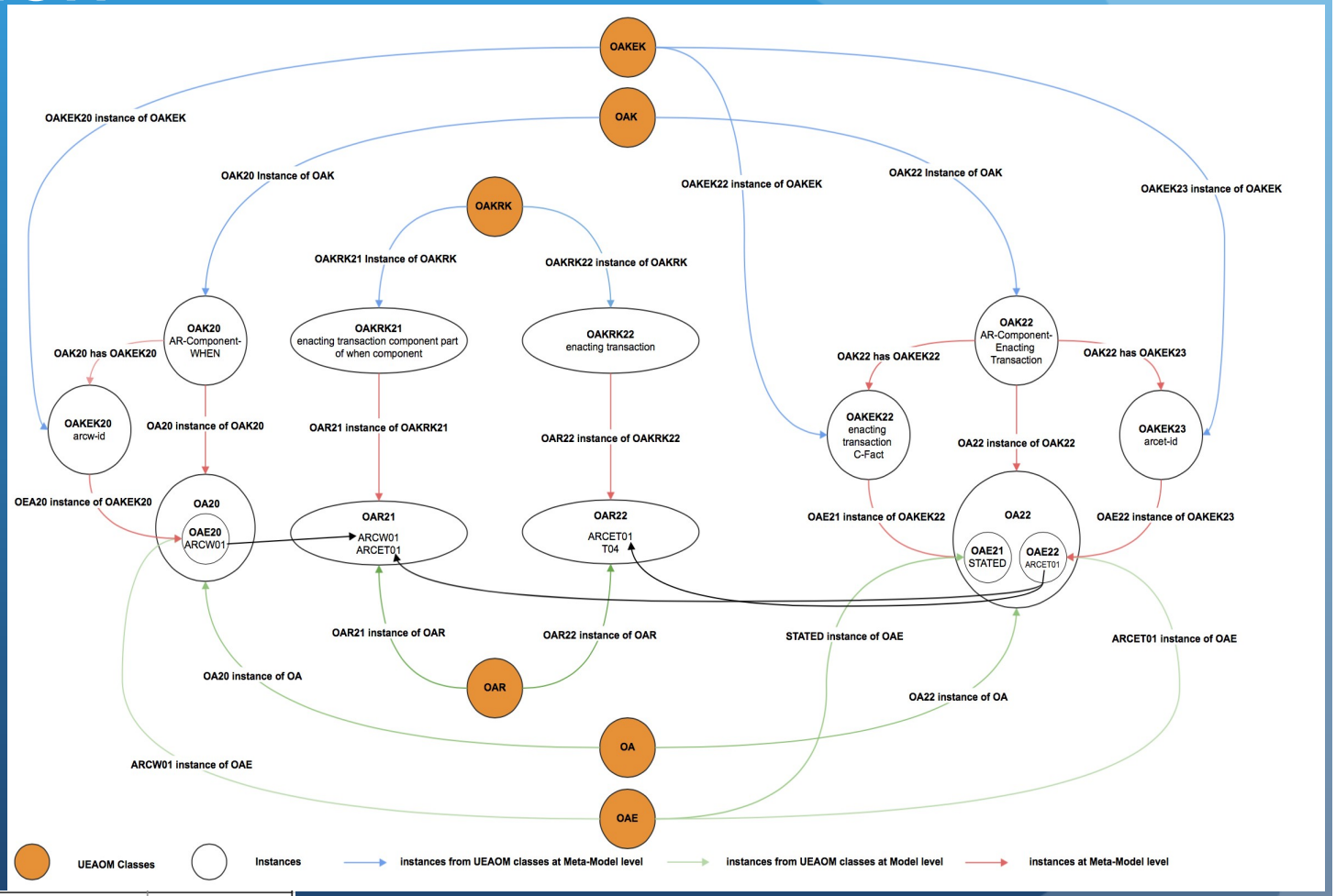


Organization Artifact Kind	**Action Rule**	
OAK Relation Kind	**executing actor**	B-A01 - Rental Starter
OAK Element Kind	action rule id	AR05
Organization Artifact Kind	**AR-Component-WHEN**	
OAK Relation Kind	**when component part of action rule**	AR05
OAK Element Kind	arcw-id	ARCW01

UNIVERSIDADE da MADEIRA

Solution

- UEAOM EU-Rent case instantiation example

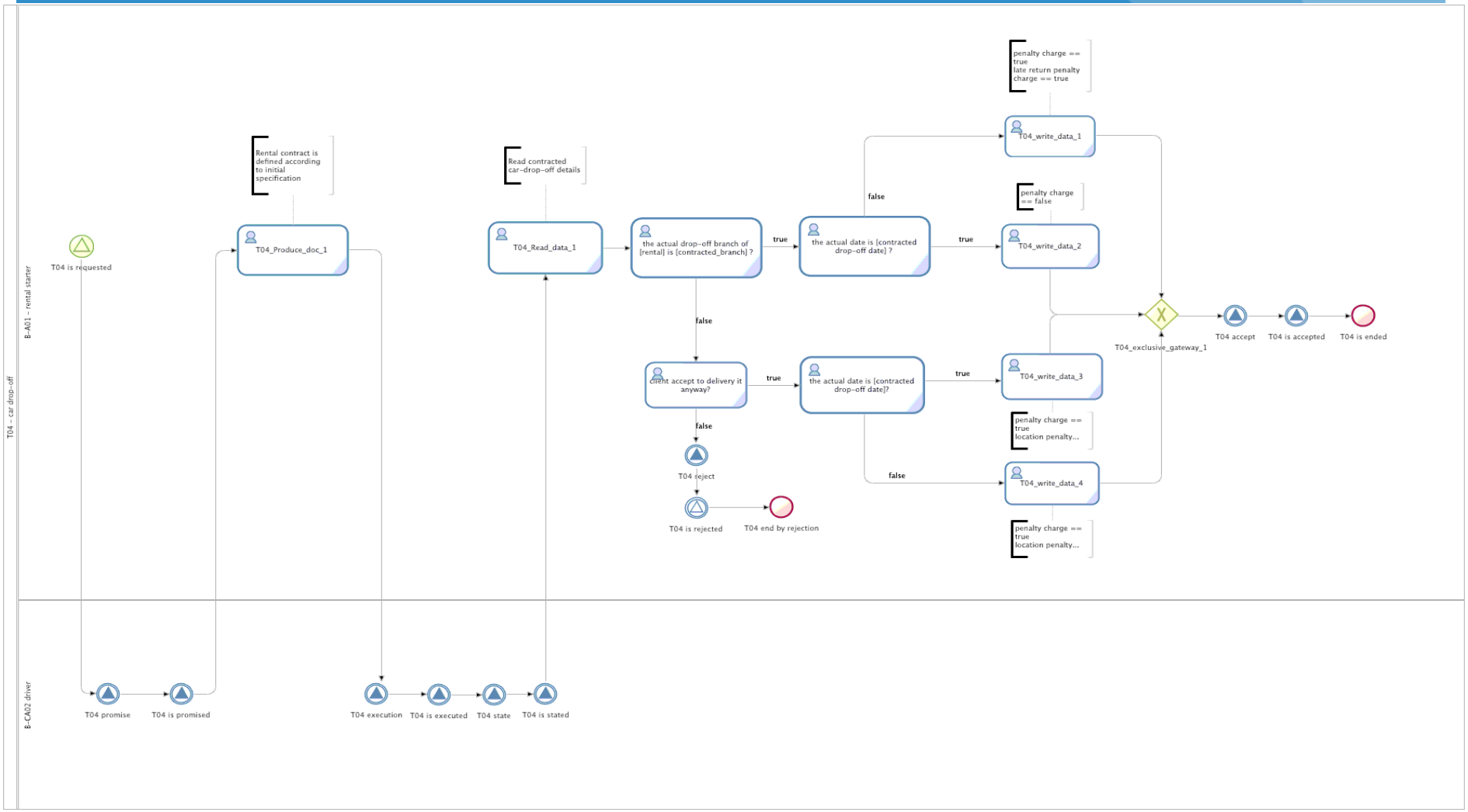


Organization Artifact Kind	**AR-Component-WHEN**	
OAK Relation Kind	**when component part of action rule**	AR05
OAK Element Kind	arcw-id	ARCW01
Organization Artifact Kind	**AR-Component-ENACTING TRANSACTION**	
OAK Relation Kind	**enacting transaction component part of when component**	ARCW01
OAK Relation Kind	**enacting transaction**	T04 - Car drop-off
OAK Element Kind	enacting transaction C-fact	stated
OAK Element Kind	arcet-id	ARCET01

UNIVERSIDADE da MADEIRA

Solution

- DEMO model based BPMN process:



Solution

- Syntax proposal using Backus-Naur-Form

<When> : : =	<Transaction kind name> "of" <Object Identifier> "is" <C-Fact> | <P-Fact> <Action> | <Flow>
<Transaction kind name> : : =	<String>
<Object Identifier> : : =	"[" <String> "]"
<P-Fact> : : =	"produced"
<C-Fact> : : =	"requested" | "promised" | "stated" | "accepted" | "revoke request requested" | "revoke resquet allowed" | "revoke request refused" | "revoke promise requested" | "revoke promise allowed" | "revoke promise refused" | "revoke statement requested" | "revoke statement allowed" | "revoke statement refused" | "revoke acceptance requested" | "revoke acceptance allowed" | "revoke acceptance refused" | "rejected" | "declined"
<C-Act> : : =	<Transaction kind name> "of" <Object Identifier> "must be" <C-Fact>
<P-Act> : : =	<Transaction kind name> "of" <Object Identifier> "must be" <P-Fact>
<Flow> : : =	<Foreach> | <If> | <While>
<Action> : : =	<Atomic action> | <Block> | <Flow>
<Atomic action> : : =	<Action type> <Action description> | <Generic requirement> | <Implementation requirement>
<Action description> : : =	<String>
<Generic requirement> : : =	<String>
<Implementation requirement> : : =	<String>
<Action type> : : =	"Write_Data" | "Read_Data" | "Transmit_Data" | "Specify_Data" | "Produce_Data" | "Produce_Doc" | "Copy_Doc" | "Store_Doc" | "Get_Doc" | "Read_Doc" | <C-Act> | <P-Act>
<Block> : : =	<Action> { <Action> }
<Condition> : : =	<Condition type> { <Condition> } | <Expression>
<Condition type> : : =	"And" | "Or" | "Not" | <Expression>
<Expression> : : =	<String>
<If> : : =	"If" <Condition> "then" <Action> | <Flow> "else" <Action> | <Flow>
<While> : : =	"While" <Condition> <Action> | <Flow>
<Foreach> : : =	"Foreach" <type of element> <Action> | <Flow>

Conclusions / Contributions

- Solved part of the problem of the lack of concise semantics in BPMN
- Specified a much more complete and comprehensive syntax for the specification of action rules still keeping implementation independence, but allowing specification of implementation details
- A formal base was created that will allow the development of a parser for the generation of a DEMO based BPMN model more “ready” to be executed compared to options currentely available

Thank you for your time!