Standards for subjectoriented specification of systems

Standardisation Gang

August 2018

Contents

Chapter 1

Background

Structure of PASS descriptions and ts relation to the execution semantics defined as Abstract State Machines (ASM).

- Start Event
- Intermediate Event
- End Event

Structure of each chapter docuement

- Informal description of PASS aspects
- OWL Description of these aspects
- ASM Sematic

Chapter 2

Classes and Property of the PASS Ontology

2.1 All Classes (95)

- PASSProcessModelElement
  - BehaviorDescribingComponent
    Group of PASS-Model components that describe aspects of the behavior of subjects
    * Action
      An Action is a grouping concept that groups a state with all its outgoing valid transitions
    * DataMappingFunction
      Standard Format for DataMappingFunctions must be define: XML? OWL? JSON? Definitions of the ability/need to write or read data to and from a subject's personal data storage. DataMappingFunctions are behavior describing components since they define what the subject is supposed to do (mapping and translating data) Mapping may be done during reception of message, where data is taken from the message/Business Object (BO) and mapped/put into the local data field. It may be done during sending of a message where data is taken from the local vault and put into a BO. Or it may occur during executing a do function, where it is used to define read(get) and write (set) functions for the local data.
      · DataMappingIncomingToLocal
        A DataMapping that specifies how data is mapped from

an an external source (message, function call etc.) to a subject's private defined data space.

- DataMappingLocalToOutgoing
  A DataMapping that specifies how data is mapped from a subject's private data space to an an external destination (message, function call etc.)

* FunctionSpecification
  A function specification for state denotes
  Concept: Definitions of calls of (mostly technical) functions (e.g. Web-service, Scripts, Database access,) that are not part of the process model.
  Function Specifications are more than "Data Properties"? –¿ - If special function types (e.g. Defaults) are supposed to be reused, having them as explicit entities is a the better OWL-modeling choice.
  - CommunicationAct
    A super class for specialized FunctionSpecification of communication acts (send and receive)
    - ReceiveFunction
      Specifications/descriptions for Receive-Functions describe in detail what the subject carrier is supposed to do in a state.
      DefaultFunctionReceive1_EnvoironmentChoice : present the surrounding execution environment with the given exit choices/conditions currently available depending on the current state of the subjects in-box. Waiting and not executing the receive action is an option.
      DefaultFunctionReceive2_AutoReceiveEarliest: automatically execute the according activity with the highest priority as soon as possible. In contrast to DefaultFunctionReceive1, it is not an option to prolong the reception and wait e.g. for another message.
    - SendFunction
      Comments have to be added
  - DoFunction
    Specifications or descriptions for Do-Functions describe in detail what the subject carrier is supposed to do in an according state. The default DoFunction
    1: present the surrounding execution environment with the given exit choices/conditions and receive choice of one exit

option –¿ define its Condition to be fulfilled in order to go to the next according state. The default DoFunction
2: execute automatic rule evaluation (see DoTransition-Condition - ToDo) More specialized Do-Function Specifications may contain Data mappings denoting what of a subjects internal local Data can and should be:
a) read: in order to simply see it or in order to send it of to an external function (e.g. a web service)
b) write: in order to write incoming Data from e.g. a web Service or user input, to the local data fault

* ReceiveType
  Comments have to be added
* SendType
  Comments have to be added
* State
  A state in the behavior descriptions of a model
  - ChoiceSegment
    ChoiceSegments are groups of defined ChoiceSegement-Paths. The paths may contain any amount of states. However, those states may not reach out of the bounds of the ChoiceSegmentPath.
  - ChoiceSegmentPath
    ChoiceSegments are groups of defined ChoiceSegement-Paths. The paths may contain any amount of states. However, those states may not reach out of the bounds of the ChoiceSegmentPath.The path may contain any amount of states but may those states may not reach out of the bounds of the choice segment path. Similar to an initial state of a behavior a choice segment path must have one determined initial state. A transition within a choice segment path must not have a target state that is not inside the same choice segment path.
    - MandatoryToEndChoiceSegmentPath
      Comments have to be added
    - MandatoryToStartChoiceSegmentPath
      Comments have to be added
    - OptionalToEndChoiceSegmentPath
      Comments have to be added
    - OptionalToStartChoiceSegmentPath

ChoiceSegmentPath and (isOptionalToEndChoiceSegment-Path value false)

- EndState
  An end state a behavior. A subject behavior may have one or more end states. Only Do and Receive states may be end states. Send States cannot be end states.There are no individual end states that are not Do, Send, or Receive States at the same time.
- GenericReturnToOriginReference
  Comments have to be added
- InitialStateOfBehavior
  The initial state of a behavior
- InitialStateOfChoiceSegmentPath
  Similar to an initial state of a behavior a choice segment path must have one determined initial state
- MacroState
  A state that references a macro behavior that is executed upon entering this state. Only after executing the macro behavior this state is finished also.
- StandardPASSState
  A super class to the standard PASS states: Do, Receive and Send
  - DoState
    The standard state in a PASS subject behavior diagram denoting an action or activity of the subject in itself.
  - ReceiveState
    The standard state in a PASS subject behavior diagram denoting an receive action or rather the waiting for a receive possibility.
  - SendState
    The standard state in a PASS subject behavior diagram denoting a send action
- StateReference
  A state reference is a model component that is a reference to a state in another behavior. For most modeling aspects it is a normal state.

* Transition
  An edge defines the transition between two states. A transition can be traversed if the outcome of the action of the state

it originates from satisfies a certain exit condition specified by it's "Alternative

- CommunicationTransition
  A super class for the CommunicationTransitions.
  - ReceiveTransition
    Comments have to be added
  - SendTransition
    Comments have to be added
- DoTransition
  Comments have to be added
- SendingFailedTransition
  Comments have to be added
- TimeTransition
  Generic super calls for all TimeTransitions, transitions with conditions based on time events. E.g.passing of a certain time duration or the (reoccurring) calendar event.
  - ReminderTransition
    Reminder transitions are transitions that can be traverses if a certain time based event or frequency has been reached. E.g. a number of months since the last traversal of this transition or the event of a certain preset calendar date etc.
    - CalendarBasedReminderTransition
      A reminder transition, for defining exit conditions measured in calendar years or months
      Conditions are e.g.: reaching of (in model) preset calendar date (e.g. 1st of July) or the reoccurrence of a a long running frequency ("every Month", "2 times a year")"
    - TimeBasedReminderTransition
      Comments have to be added
  - TimerTransition
    Generic super calls for all TimeTransitions, transitions with conditions based on time events. E.g.passing of a certain time duration or the (reoccurring) calendar event.
    - BusinessDayTimerTransition
      imer transitions, denote time outs for the state they originate from. The condition for a timer transition

is that a certain amount of time has passed since the state it originates from has been entered.
The time unit for this timer transition is measured in business days. The definition of a business day depends on a subject's relevant or legal location

- DayTimeTimerTransition
  Timer Transitions, denoting time outs for the state they originate from. The condition for a timer transition is that a certain amount of time has passed since the state it originates from has been entered.
  Day or Time Timers are measured in normal 24 hour days. Following the XML standard for time and day duration. They are to be differed from the timers that are timeout in units of years or months.
- YearMonthTimerTransition
  Timer transitions, denote time outs for the state they originate from. The condition for a timer transition is that a certain amount of time has passed since the state it originates from has been entered.
  Year or Month timers measure time in calendar years or months. The exact definitions for years and months depends on relevant or legal geographical location of the subject.
- UserCancelTransition
  A user cancel transition denotes the possibility to exit a receive state without the reception of a specific message.
  The user cancel allows for an arbitrary decision by a subject carrier/processor to abort a waiting process.

- TransitionCondition
  natives which in turn is given for a state. An alternative (to leave the state) is only a real alternative if the exit condition is fulfilled (technically: if that according function returns "true").
  Note: Technically and during execution exit conditions belong to states. They define when it is allowed to leave that state. However, in PASS models exit conditions for states are defined and connected to the according transition edges. Therefore transition conditions are individual entities and not DataProperties.

The according matching must be done by the model execution environment.
By its existence, an edge/transition defines one possible follow up "state" for its state of origin. It is coupled with an "Exit Condition" that must be fulfilled in the originating state in order to leave the state.

- DoTransitionCondition
  A TransitionCondition for the according DoTransitions and DoStates.
- MessageExchangeCondition
  MessageExchangeConditon is the super class for Send End Receive Transition Conditions the both require either the sending or receiving (exchange) of a message to be fulfilled.
  - ReceiveTransitionCondition
    ReceiveTransitionConditions are conditions that state that a certain message must have been taken out of a subjects in-box to be fulfilled.
    These are the typical conditions defined by Receive Transitions.
  - SendTransitionCondition
    SendTransitionConditions are conditions that state that a certain message must have been successfully passed to another subjects in-box to be fulfilled.
    These are the typical conditions defined by Send transitions.
- SendingFailedCondition
  Comments have to be added
- TimeTransitionCondition
  A condition that is deemed 'true' and thus the according edge is gone, if: a surrounding execution system has deemed the time since entering the state and starting with the execution of the according action as too long (predefined by the outgoing edge)
  A condition that is true if a certain time defined has passed since the state this condition belongs to has been entered. (This is the standard TimeOut Exit condition)
  - ReminderEventTransitionCondition
    Comments have to be added
  - TimerTransitionCondition

Comments have to be added

- DataDescribingComponent
  Subject-Oriented PASS Process Models are in general about describing the activities and interaction of active entities. Yet these interactions are rarely done without data that is being generated by activities and transported via messages. While not considered by Börger's PASS interpreter, the community agreed on adding the ability to integrate the means to describe data objects or data structures to the model and enabling their connection to the process model. It may be defined that messages or subject have their individual DataObjectDefinition in form of a SubjectDataDefinition in the case of FullySpecifiedSubjects and PayloadDataObjectDesfinition in the case of MessageSpecifications In general, it expected that these DataObjectDefinition list on or more data fields for the message or subject with an internal data type that is described via a DataTypeDefinition. There is a rudimentary concept for a simple build-in data type definition closely oriented at the concept of ActNConnect. Otherwise, the principle idea of the OWL standard is to allow and employ existing or custom technologies for the serialized definition of data structures (CustomOrExternalDataTypeDefinition) such as XML-Schemata (XSD), according elements with JSON or directly the powerful expressiveness of OWL itself.
  * DataObjectDefinition
    Data Object Definitions are model elements used to describe that certain other model elements may posses or carrier Data Objects.
    E.G. a message may carrier/include a Business Objects. Or the private Data Space of a Subject may contain several Data Objects.
    A Data Objects should refer to a DataTypeDefinition denoting its DataType and structure.
    DataObject: states that a data item does exist (similar to a variable in programming)DataType: the definition of an Data Object's structure.
    - DataObjectListDefintion
      Data definition concept for PASS model build in capabilities of data modeling. Defines a simple list structure.
    - PayloadDataObjectDefinition

Messages may have a description regarding their payload (what is transported with them).
This can either be a description of a physical (real) object or a description of a (digital) data object

· SubjectDataDefinition
Comments have to be added

* DataTypeDefinition
Data Type Definitions are complex descriptions of the supposed structure of Data Objects.
DataObject: states that a data item does exist (similar to a variable in programming).
DataType: the definition of an Data Object's structure.

· CustomOrExternalDataTypeDefinition
Using this class, tool vendors can include their own custom data definitions in the model.

· JSONDataTypeDefinition
Comments have to be added

· OWLDataTypeDefinition
Comments have to be added

· XSD-DataTypeDefinition
XML Schemata Description (XSD) is an established technology for describing structure of Data Objects (XML documents) with many tools available that can verify a document against the standard definition

· ModelBuiltInDataTypes
Comments have to be added

* PayloadDescription
Comments have to be added

· PayloadDataObjectDefinition
Messages may have a description regarding their payload (what is transported with them).
This can either be a description of a physical (real) object or a description of a (digital) data object

· PayloadPhysicalObjectDescription
Messages may have a description regarding their payload (what is transported with them).
This can either be a description of a physical (real) object or a description of a (digital) data object

– InteractionDescribingComponent

This class is the super class of all model elements used to define or specify the interaction means within a process model

* InputPoolConstraint
  Subjects do implicitly posses input pools.
  During automatic execution of a PASS model in a work-flow engine this message box is filled with messages.
  Without any constraints models this message in-box is assumed to be able to store an infinite amount of messages.
  For some modeling concepts though it may be of importance to restrict the size of the input pool for certain messages or senders.
  This is done using several different Type of InputPoolConstraints that are attached to a fully specified subject.
  Should a constraint be applicable, an "InputPoolConstraintHandlingStrategy" will be executed by a work-flow engine to determine what to do with the message that does not fit in the pool.
  Limiting the input pool for certain reasons to size 0 together with the InputPoolConstraintStrategy-Blocking is effectively modeling that a communication must happen synchronously instead of the standard asynchronous mode. The sender can send his message only if the receiver is in an according receive state, so the message can be handled directly without being stored in the in-box.
  - MessageSenderTypeConstraint
    An InputPool constraint that limits the number of message of a certain type and from a certain sender in the input pool.
    E.g. "Only one order from the same customer" (during happy hour at the bar)
  - MessageTypeConstraint
    An InputPool constraint that limits the number of message of a certain type in the input pool.
    E.g. You can accept only "three request at once
  - SenderTypeConstraint
    An InputPool constraint that limits the number of message from a certain Sender subject in the input pool.
    E.g. as long as a customer has non non-fulfilled request of any type he may not place messages
* InputPoolContstraintHandlingStrategy
  Should an InputPoolConstraint be applicable, an "InputPool-

ConstraintHandlingStrategy" will be executed by a work-flow engine to determine what to do with the message that does not fit in the pool.
There are types of HandlingStrategies.
InputPoolConstraintStrategy-Blocking - No new message will be adding will need to be repeated until successful
InputPoolConstraintStrategy-DeleteLatest - The new message will be added, but the last message to arrive before that applicable to the same constraint will be overwritten with the new one. (LIFO deleting concept)
InputPoolConstraintStrategy-DeleteOldest - The message will be added, but the earliest message in the input pool applicable to the same constraint will be deleted (FIFO deleting concept)
InputPoolConstraintStrategy-Drop - Sending of the message succeeds. However the new message will not be added to the in-box. Rather it will be deleted directly.

* MessageExchange
  A message exchange is an element in the interaction description section that specifies exactly one possibility of exchanging messages in the given process context of the model.
  A message exchange is a triple of, a sender, a receiver, and the specification of the message that may be exchanged.
  While message exchanges are singular occurrences, they may be grouped in MessageExchangeLists
* MessageExchangeList
  While MessageExchanges are singular occurrences, they may be grouped in MessageExchangeLists.
  In graphical PASS modeling that is usually the case when one arrow between two subjects contains more than one message and thereby specifies more than one possible message exchange channel between the two subjects.
* MessageSpecification
  MessageSpecification are model elements that specify the existence of a message. At minimum its name and id.
  It may contain additional specification for its payload (contained Data, exact form etc.)
* Subject
  The subject is the core model element of a subject-oriented PASS process model.
  - FullySpecifiedSubject

Fully specified Subjects in a PASS graph are entities that, in contrast to interface subjects, linked to one ore more Behaviors (they posses a behavior).

· InterfaceSubject
Interface Subjects are Subjects that are not linked to a behavior. In contrast, they may refer to FullySpecified-Subjects that are described in other process models.

· MultiSubject
The Multi-Subject is term for a subject that "has a maximum subject instantiation restriction" within a process context larger than 1.

· SingleSubject
Single Subject are subject with a maximumInstanceRestriction of 1

· StartSubject
Subjects that start their behavior with a Do or Send state are active in a process context from the beginning instead of requiring a message from another subject.
Usually there should be only one Start subject in a process context.

– PASSProcessModel
The main class that contains all relevant process elements

– SubjectBehavior
Additional to the subject interaction a PASS Model consist of multiple descriptions of subject's behaviors. These are graphs described with the means of BehaviorDescribingComponents
A subject in a model may be linked to more than one behavior.

* GuardBehavior
A guard behavior is a special usually additional behavior that guards the Base Behavior of a subject.ßß It starts with a (guard) receive state denoting a special interrupting message. Upon reception of that message the subject will execute the according receive transition and the follow up states until it is either redirected to a state on the base behavior or terminates in an end-state within the guard behavior

* MacroBehavior
A macro behavior is a specialized behavior that may be entered and exited from a function state in another behavior.

    * SubjectBaseBehavior
      The standard behavior model type

- SimplePASSElement
  Comments have to be added

  – CommunicationTransition
    A super class for the CommunicationTransitions.

    * ReceiveTransition
      Comments have to be added
    * SendTransition
      Comments have to be added

  – DataMappingFunction
    Definitions of the ability/need to write or read data to and from a subject's personal data storage.
    DataMappingFunctions are behavior describing components since they define what the subject is supposed to do (mapping and translating data)
    Mapping may be done during reception of message, where data is taken from the message/Business Object (BO) and mapped/put into the local data field.
    It may be done during sending of a message where data is taken from the local vault and put into a BO.
    Or it may occur during executing a do function, where it is used to define read(get) and write (set) functions for the local data.

    * DataMappingIncomingToLocal
      A DataMapping that specifies how data is mapped from an an external source (message, function call etc.) to a subject's private defined data space.
    * DataMappingLocalToOutgoing
      A DataMapping that specifies how data is mapped from a subject's private data space to an an external destination (message, function call etc.)"

  – DoTransition
    Comments have to be added

  – DoTransitionCondition
    A TransitionCondition for the according DoTransitions and DoStates.

  – EndState
    An end state a behavior. A subject behavior may have one or

more end states. Only Do and Receive states may be end states. Send States cannot be end states.
There are no individual end states that are not Do, Send, or Receive States at the same time.

- FunctionSpecification
  A function specification for state denotes
  Concept: Definitions of calls of (mostly technical) functions (e.g. Web-service, Scripts, Database access,) that are not part of the process model.
  Function Specifications are more than "Data Properties"? –¿ - If special function types (e.g. Defaults) are supposed to be reused, having them as explicit entities is a the better OWL-modeling choice.
  * CommunicationAct
    A super class for specialized FunctionSpecification of communication acts (send and receive)
    · ReceiveFunction
      Specifications/descriptions for Receive-Functions describe in detail what the subject carrier is supposed to do in a state.
      DefaultFunctionReceive1_EnvoironmentChoice : present the surrounding execution environment with the given exit choices/conditions currently available depending on the current state of the subjects in-box. Waiting and not executing the receive action is an option.
      DefaultFunctionReceive2_AutoReceiveEarliest: automatically execute the according activity with the highest priority as soon as possible. In contrast to DefaultFunctionReceive1, it is not an option to prolong the reception and wait e.g. for another message.
    · SendFunction
      Comments have to be added
  * DoFunction
    Specifications or descriptions for Do-Functions describe in detail what the subject carrier is supposed to do in an according state.
    The default DoFunction 1: present the surrounding execution environment with the given exit choices/conditions and receive choice of one exit option –¿ define its Condition to be fulfilled in order to go to the next according state.

The default DoFunction 2: execute automatic rule evaluation (see DoTransitionCondition).
More specialized Do-Function Specifications may contain Data mappings denoting what of a subjects internal local Data can and should be:
a) read: in order to simply see it or in order to send it of to an external function (e.g. a web service)
b) write: in order to write incoming Data from e.g. a web Service or user input, to the local data fault

- InitialStateOfBehavior
  The initial state of a behavior
- MessageExchange
  A message exchange is an element in the interaction description section that specifies exactly one possibility of exchanging messages in the given process context of the model.
  A message exchange is a triple of, a sender, a receiver, and the specification of the message that may be exchanged.
  While message exchanges are singular occurrences, they may be grouped in MessageExchangeLists
- MessageExchangeCondition
  MessageExchangeConditon is the super class for Send End Receive Transition Conditions the both require either the sending or receiving (exchange) of a message to be fulfilled.
  * ReceiveTransitionCondition
    ReceiveTransitionConditions are conditions that state that a certain message must have been taken out of a subjects in-box to be fulfilled.
    These are the typical conditions defined by Receive Transitions.
  * SendTransitionCondition
    SendTransitionConditions are conditions that state that a certain message must have been successfully passed to another subjects in-box to be fulfilled.
    These are the typical conditions defined by Send transitions.
- MessageExchangeList
  While MessageExchanges are singular occurrences, they may be grouped in MessageExchangeLists.
  In graphical PASS modeling that is usually the case when one arrow between two subjects contains more than one message and

thereby specifies more than one possible message exchange channel between the two subjects.

- MessageSpecification
  MessageSpecification are model elements that specify the existence of a message. At minimum its name and id.
  It may contain additional specification for its payload (contained Data, exact form etc.)
- ModelBuiltInDataTypes
  Comments have to be added
- PayloadDataObjectDefinition
  Messages may have a description regarding their payload (what is transported with them).
  This can either be a description of a physical (real) object or a description of a (digital) data object
- StandardPASSState
  A super class to the standard PASS states: Do, Receive and Send
  * DoState
    The standard state in a PASS subject behavior diagram denoting an action or activity of the subject in itself.
  * ReceiveState
    The standard state in a PASS subject behavior diagram denoting an receive action or rather the waiting for a receive possibility.
  * SendState
    The standard state in a PASS subject behavior diagram denoting a send action
- Subject
  The subject is the core model element of a subject-oriented PASS process model.
  * FullySpecifiedSubject
    Fully specified Subjects in a PASS graph are entities that, in contrast to interface subjects, linked to one ore more Behaviors (they posses a behavior).
  * InterfaceSubject
    Interface Subjects are Subjects that are not linked to a behavior. In contrast, they may refer to FullySpecifiedSubjects that are described in other process models.

    * MultiSubject
      The Multi-Subject is term for a subject that "has a maximum subject instantiation restriction" within a process context larger than 1.
    * SingleSubject
      Single Subject are subject with a maximumInstanceRestriction of 1
    * StartSubject
      Subjects that start their behavior with a Do or Send state are active in a process context from the beginning instead of requiring a message from another subject.
      Usually there should be only one Start subject in a process context.

  – SubjectBaseBehavior
    The standard behavior model type

2.2 Data Properties (27)

Property name		Domain-Range	Comments	Reference
hasBusinessDayDurationTimeOutTime	Domain: Range:			
hasCalendarBasedFrequencyOrDate	Domain: Range:			
hasDataMappingString	Domain: Range:			
hasDayTimeDurationTimeOutTime	Domain: Range:			
hasDurationTimeOutTime	Domain: Range:			
hasFeelExpressionAsDataMapping	Domain: Range:			
hasGraphicalRepresentation	Domain: Range:			
hasKey	Domain: Range:			
hasLimit	Domain: Range:			
hasMaximumSubjectInstanceRestriction	Domain: Range:			
hasMetaData	Domain: Range:			
hasModelComponentComment	Domain: Range:			
hasModelComponentID	Domain:			

Property name		Domain-Range	Comments	Reference
	Range:			
hasModelComponentLabel	Domain: Range:			
hasPriorityNumber	Domain: Range:			
hasReoccuranceFrequenyOrDate	Domain: Range:			
hasSVGRepresentation	Domain: Range:			
hasTimeBasedReoccuranceFrequencyOrDate	Domain: Range:			
hasTimeValue	Domain: Range:			
hasToolSpecificDefinition	Domain: Range:			
hasValue	Domain: Range:			
hasYearMonthDurationTimeOutTime	Domain: Range:			
isOptionalToEndChoiceSegmentPath	Domain: Range:			
isOptionalToStartChoiceSegmentPath	Domain: Range:			
owl:topDataProperty	Domain: Range:			

Property name		Domain-Range	Comments	Reference
PASSModelDataProperty	Domain: Range:			
SimplePASSDataProperties	Domain: Range:			

2.3 Object Properties (42)

Property name		Domain-Range	Comments	Reference
belongsTo	Domain: Range:	PASSProcessModelElement PASSProcessModelElement	Generic ObjectProperty that links two process elements, where one is contained in the other (inverse of contains).	
contains	Domain: Range:	PASSProcessModelElement PASSProcessModelElement	Generic ObjectProperty that links two model elements where one contains another (possible multiple)	
containsBaseBehavior	Domain: Range:	Subject SubjectBehavior		
containsBehavior	Domain: Range:	Subject SubjectBehavior		
containsPayload-Description	Domain: Range:	MessageSpecification PayloadDescription		
guardedBy	Domain: Range:	State, Action GuardBehavior		

Property name		Domain-Range	Comments	Reference
guardsBehavior	Domain: Range:	GuardBehavior SubjectBehavior	Links a GuardBehavior to another SubjectBehavior. Automatically all individual states in the guarded behavior are guarded by the guard behavior. There is an SWRL Rule in the ontology for that purpose.	
guardsState	Domain: Range:	State, Action guardedBy		
hasAdditionalAttribute	Domain: Range:	PASSProcessModelElement AdditionalAttribute		
hasCorrespondent	Domain: Range:	 Subject	Generic super class for the ObjectProperties that link a Subject with a MessageExchange either in the role of Sender or Receiver.	
hasDataDefinition	Domain: Range:	 DataObjectDefinition		

Property name		Domain-Range	Comments	Reference
hasDataMapping-Function	Domain: Range:	state, SendTransition, ReceiveTransition DataMappingFunction		
hasDataType	Domain: Range:	PayloadDescription or DataObjectDefinition DataTypeDefinition		
hasEndState	Domain: Range:	SubjectBehavior or Choice-SegmentPath State, not SendState		
hasFunction-Specification	Domain: Range:	State FunctionSpecification		
hasHandlingStrategy	Domain: Range:	InputPoolConstraint InputPoolContstraint-HandlingStrategy		
hasIncomingMessage-Exchange	Domain: Range:	Subject MessageExchange		
hasIncomingTransition	Domain: Range:	State Transition		
hasInitialState	Domain: Range:	SubjectBehavior or Choice-SegmentPath State		

Property name		Domain-Range	Comments	Reference
hasInputPoolConstraint	Domain: Range:	Subject InputPoolConstraint		
hasKeyValuePair	Domain: Range:			
hasMessageExchange	Domain: Range:	Subject	Generic super class for the ObjectProperties linking a subject with either incoming or outgoing MessageExchanges.	
hasMessageType	Domain: Range:	MessageTypeConstraint or MessageSenderTypeConstraint or MessageExchange MessageSpecification		
hasOutgoingMessageExchange	Domain: Range:	Subject MessageExchange		
hasOutgoingTransition	Domain: Range:	State Transition		
hasReceiver	Domain: Range:	MessageExchange Subject		

Property name		Domain-Range	Comments	Reference
hasRelationToModel-Component	Domain: Range:	PASSProcessModelElement PASSProcessModelElement	Generic super class of all object properties in the standard-pass-ont that are used to link model elements with one-another.	
hasSender	Domain: Range:	MessageExchange Subject		
hasSourceState	Domain: Range:	Transition State		
hasStartSubject	Domain: Range:	PASSProcessModel StartSubject		
hasTargetState	Domain Range	Transition State		
hasTransitionCondition	Domain Range	Transition TransitionCondition		
isBaseBehaviorOf	Domain: Range:	SubjectBaseBehavior	A specialized version of the "belongsTo" Object-Property to denote that a -SubjectBehavior belongs to a Subject as its BaseBehavior	
isEndStateOf	Domain: Range:	State and not SendState SubjectBehavior or Choice-SegmentPath		

Property name		Domain-Range	Comments	Reference
isInitialStateOf	Domain: Range:	State SubjectBehavior or Choice-SegmentPath		
isReferencedBy	Domain: Range:			
references	Domain: Range:			
referencesMacroBehavior	Domain: Range:	MacroState MacroBehavior		
refersTo	Domain: Range:	CommunicationTransition MessageExchange	Communication transitions (send and receive) should refer to a message exchange that is defined on the interaction layer of a model.	
requiresActiveReception-OfMessage	Domain: Range:	ReceiveTransitionCondition MessageSpecification		
requiresPerformed-MessageExchange	Domain: Range:	MessageExchangeCondition MessageExchange		

Property name		Domain-Range	Comments	Reference
SimplePASSObject-Propertie	Domain: Range:		Every element/sub-class of SimplePASSObjectProperties is also a Child of PASSModelObjectPropertiy. This is simply a surrogate class to group all simple elements together	

Chapter 3

Structure of a PASS Description

In this chapter we describe the structure of a PASS specification. The structure of a PASS descritption consists of the subjects and the messages they exchange.

3.1 Informal Description

3.1.1 Subject

Subjects are the active entities in system described in PASS in dependent how they are implemented. Subjects can be relized either by human, machines, software all combinitions of them. Subject descriptions do not contain any information about their implementation. If implementaion information is added to a subject it becomes an actor. In the following we use aan example for the informal definiton of subjects. In the simple scenario of the business trip application, we can identify three subjects, namely the employee as applicant, the manager as the approver, and the travel office as the travel arranger.

The definition of which subjects should be part of a process is a leadership decision. On the one hand, the necessary subjects result from the actual (as-is) situation, as it has for example already been described in the process analysis. On the other hand, the subject scoping, i.e., the question of what subjects there are and what tasks they roughly perform, can be adjusted to the envisioned or desired (to-be) situation.

Depending on the required or desired division of labor in a process, a corresponding number of subjects is necessary. This division is a design

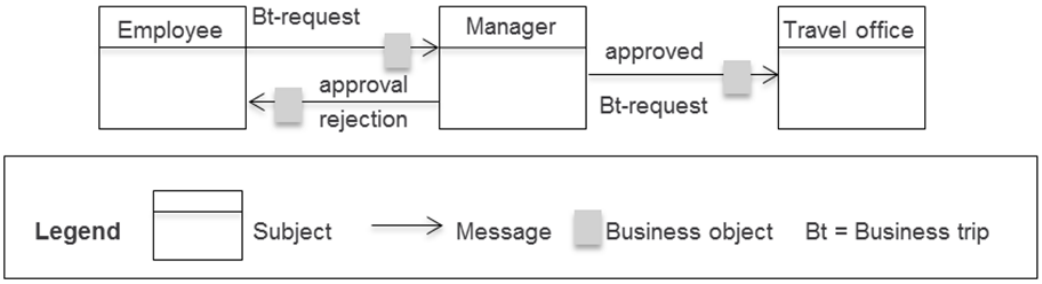

Figure 3.1: Subject interaction diagram for the process 'business trip application'

decision that must be taken in accordance with business needs. It influences the necessary granularity of a process model (see Section XXXXX).

In case there are many specialized subjects involved in a process, it may lead to many potentially complex interactions between the subjects. This can be a problem, since the communication between process participants always carries the risk of delays and misunderstandings. In case of few subjects, however, the subject carriers often cover a too wide a range of activities, which puts high demands on the participants. The decision with respect to subject scoping therefore has far-reaching consequences. It is complex, represents a major challenge, and requires extensive experience and care.

3.1.2 Subject-to-Subject Communication

After the identification of subjects involved in the process (as process-specific roles), their interaction relationships need to be represented. These are the messages exchanged between the subjects. Such messages might contain structured information—so-called business objects (see Section xxxxxxx).

The result is a model structured according to subjects with explicit communication relationships, which is referred to as a Subject Interaction Diagram (SID) or, synonymously, as a Communication Structure Diagram (CSD) (see Figurefig:beispiel-subject-interaction).

Messages represent the interactions of the subjects during the execution of the process. We recommend naming these messages in such a way that they can be immediately understood and also reflect the meaning of each particular message for the process. In the sample 'business trip application', therefore, the messages are referred to as 'business trip request', 'rejection', and 'approval'.

Messages serve as a container for the information transmitted from a sending to a receiving subject. There are two options for the message content:

- Simple data types: Simple data types are string, integer, character, etc. In the business trip application example, the message 'business trip request' can contain several data elements of type string (e.g., destination, reason for traveling, etc.), and of type number (e.g., duration of trip in days).

- Business Objects: Business Objects in their general form are physical and logical 'things' that are required to process business transactions., We consider data structures composed of elementary data types, or even other data structures, as logical business objects in business processes. For instance, the business object 'business trip request' could consist of the data structures 'data on applicants', 'travel data', and 'approval data'—with each of these in turn containing multiple data elements.

3.1.3 Message Exchange

In the previous subsection, we have stated that messages are transferred between subjects and have described the nature of these messages. What is still missing is a detailed description of how messages can be exchanged, how the information they carry can be transmitted, and how subjects can be synchronized. These issues are addressed in the following sub-sections.

Synchronous and Asynchronous Exchange of Messages

In the case of synchronous exchange of messages, sender and receiver wait for each other until a message can be passed on. If a subject wants to send a message and the receiver (subject) is not yet in a corresponding receive state, the sender waits until the receiver is able to accept this message. Conversely, a recipient has to wait for a desired message until it is made available by the sender.

The disadvantage of the synchronous method is a close temporal coupling between sender and receiver. This raises problems in the implementation of business processes in the form of workflows, especially across organizational borders. As a rule, these also represent system boundaries across which a tight coupling between sender and receiver is usually very costly. For long-running processes, sender and receiver may wait for days, or even weeks, for each other.

Using asynchronous messaging, a sender is able to send anytime. The subject puts a message into a message buffer from which it is picked up by the receiver. However, the recipient sees, for example, only the oldest message in the buffer and can only accept this particular one. If it is not the desired

message, the receiver is blocked, even though the message may already be in the buffer, but in a buffer space that is not visible to the receiver. To avoid this, the recipient has the alternative to take all of the messages from the buffer and manage them by himself. In this way, the receiver can identify the appropriate message and process it as soon as he needs it. In asynchronous messaging, sender and receiver are only loosely coupled. Practical problems can arise due to the in reality limited physical size of the receive buffer, which does not allow an unlimited number of messages to be recorded. Once the physical boundary of the buffer has been reached due to high occupancy, this may lead to unpredictable behavior of workflows derived from a business process specification. To avoid this, the input-pool concept has been introduced in PASS.

Exchange of Messages via the Input Pool

To solve the problems outlined in asynchronous message exchange, the input pool concept has been developed. Communication via the input pool is considerably more complex than previously shown; however, it allows transmitting an unlimited number of messages simultaneously. Due to its high practical importance, it is considered as a basic construct of PASS. Consider the input pool as a mail box of work performers, the operation of which is specified in detail. Each subject has its own input pool. It serves as a message buffer to temporarily store messages received by the subject, independent of the sending communication partner. The input pools are therefore inboxes for flexible configuration of the message exchange between the subjects. In contrast to the buffer in which only the front message can be seen and accepted, the pool solution enables picking up (= removing from the buffer) any message. For a subject, all messages in its input pool are visible.

The input pool has the following configuration parameters (see Figure-Figure 5.2):

- Input-pool size: The input-pool size specifies how many messages can be stored in an input pool, regardless of the number and complexity of the message parameters transmitted with a message. If the input pool size is set to zero, messages can only be exchanged synchronously.

- Maximum number of messages from specific subjects: For an input pool, it can be determined how many messages received from a particular subject may be stored simultaneously in the input pool. Again, a value of zero means that messages can only be accepted synchronously.

- Maximum number of messages with specific identifiers: For an input

pool, it can be determined how many messages of a specifically identified message type (e.g., invoice) may be stored simultaneously in the input pool, regardless of what subject they originate from. A specified size of zero allows only for synchronous message reception.

- Maximum number of messages with specific identifiers of certain subjects: For an input pool, it can be determined how many messages of a specific identifier of a particular subject may be stored simultaneously in the input pool. The meaning of the zero value is analogous to the other cases.

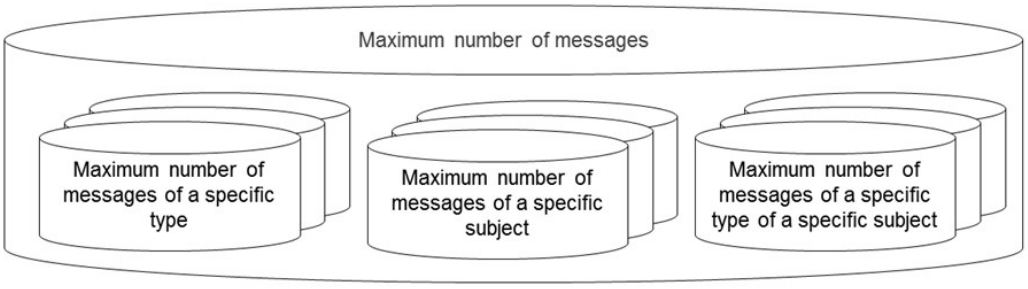

Figure 3.2: Configuration of Input Pool Parameters

By limiting the size of the input pool, its ability to store messages may be blocked at a certain point in time during process runtime. Hence, messaging synchronization mechanisms need to control the assignment of messages to the input pool. Essentially, there are three strategies to handle the access to input pools:

- Blocking the sender until the input pool's ability to store messages has been reinstated: Once all slots are occupied in an input pool, the sender is blocked until the receiving subject picks up a message (i.e. a message is removed from the input pool). This creates space for a new message. In case several subjects want to put a message into a fully occupied input pool, the subject that has been waiting longest for an empty slot is allowed to send. The procedure is analogous if corresponding input pool parameters do not allow storing the message in the input pool, i.e., if the corresponding number of messages of the same name or from the same subject has been put into the input pool.

- Delete and release of the oldest message: In case all the slots are already occupied in the input pool of the subject addressed, the oldest message is overwritten with the new message.

- Delete and release of the latest message: The latest message is deleted from the input pool to allow depositing of the newly incoming message. If all the positions in the input pool of the addressed subject are taken, the latest message in the input pool is overwritten with the new message. This strategy applies analogously when the maximum number of messages in the input pool has been reached, either with respect to sender or message type.

3.2 OWL Description

3.2.1 Subject Interaction

Overview Subject InterAction

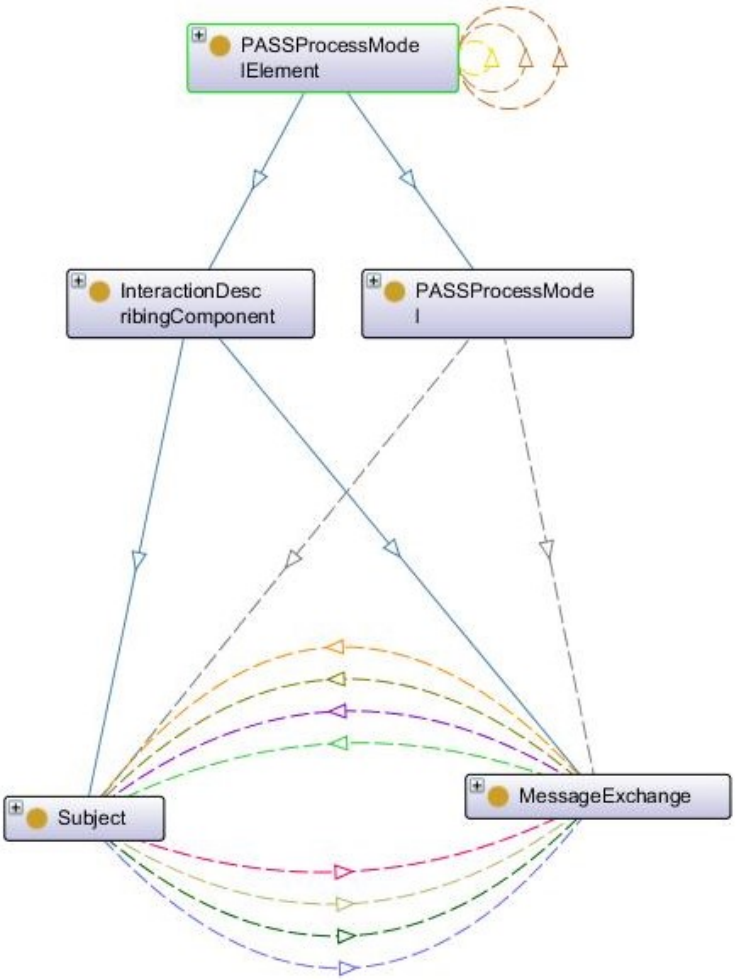

Figure 3.3: Subject Interaction

3.2.2 Subjects

Different types of subjects 3.4

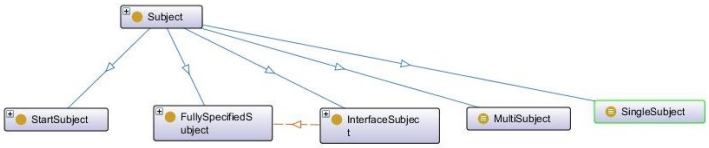

Figure 3.4: Different Types of Subjects

3.2.3 Messages

SDescription of messages 3.5

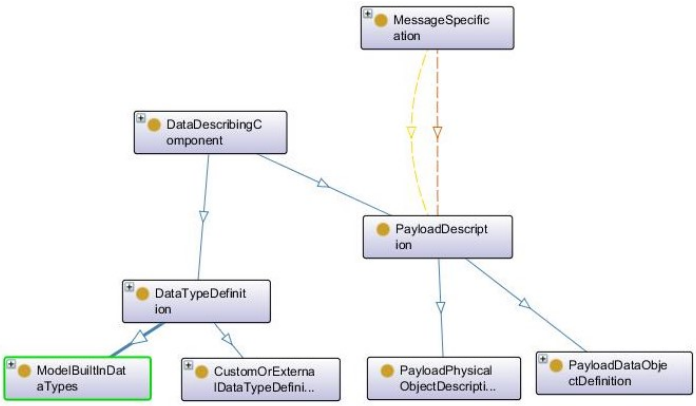

Figure 3.5: Message Specification with Payload

3.2.4 Input Pools

Description of input pools 3.7

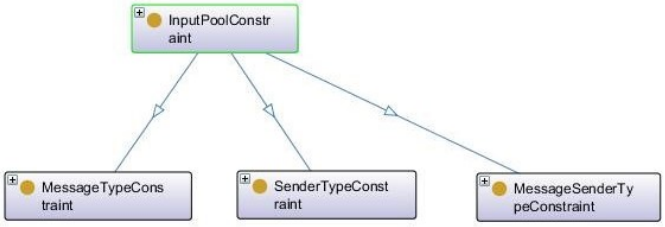

Figure 3.6: Input Pool description

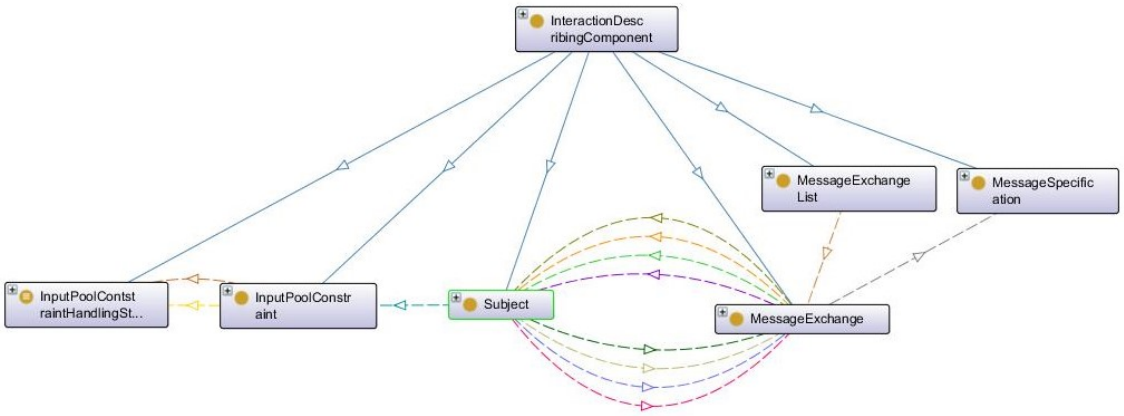

Figure 3.7: Message Exchange and Input Pools

3.3 ASM Description