Standards for subjectoriented specification of systems

Standardisation Gang

August 2018

Contents

1 Background **1**

2 Classes and Property of the PASS Ontology **3**
2.1 All Classes (95) 3
2.2 Data Properties (27) 19
2.3 Object Properties (42) 28

3 Structure of a PASS Description **37**
3.1 Informal Description 37
3.1.1 Subject 37
3.1.2 Subject-to-Subject Communication 40
3.1.3 Message Exchange 41
3.2 OWL Description 45
3.2.1 Subjects 47
3.2.2 Messages 48
3.2.3 Input Pools 49
3.3 ASM Description 49

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)

- SRN = Subclass Reference Number; Is used for marking the corresponding relations in the following figures. The number identifies the subclass relation to the next level of super class.

- PASSProcessModelElement

  - BehaviorDescribingComponent; SRN: 001
    Group of PASS-Model components that describe aspects of the behavior of subjects
    * Action; SRN: 002
      An Action is a grouping concept that groups a state with all its outgoing valid transitions
    * DataMappingFunction ; SRN: 003
      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 ex-

ecuting a do function, where it is used to define read(get) and write (set) functions for the local data.

- DataMappingIncomingToLocal ; SRN: 004
  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 ; SRN: 005
  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 ; SRN: 006
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 ; SRN: 007
  A super class for specialized FunctionSpecification of communication acts (send and receive)
  - ReceiveFunction ; SRN: 008
    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 ; SRN: 009
    Comments have to be added
- DoFunction ; SRN: 010
  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 ; SRN: 011
  Comments have to be added
* SendType ; SRN: 012
  Comments have to be added
* State ; SRN: 013
  A state in the behavior descriptions of a model
  - ChoiceSegment ; SRN: 014
    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 ; SRN: 015
    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 ; SRN: 016
      Comments have to be added
    - MandatoryToStartChoiceSegmentPath ; SRN: 017
      Comments have to be added

  - OptionalToEndChoiceSegmentPath ; SRN: 018
    Comments have to be added
  - OptionalToStartChoiceSegmentPath ; SRN: 019
    ChoiceSegmentPath and (isOptionalToEndChoiceSegmentPath value false)
- EndState ; SRN: 020
  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 ; SRN: 021
  Comments have to be added
- InitialStateOfBehavior ; SRN: 022
  The initial state of a behavior
- InitialStateOfChoiceSegmentPath ; SRN: 023
  Similar to an initial state of a behavior a choice segment path must have one determined initial state
- MacroState ; SRN: 024
  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 ; SRN: 025
  A super class to the standard PASS states: Do, Receive and Send
  - DoState ; SRN: 026
    The standard state in a PASS subject behavior diagram denoting an action or activity of the subject in itself.
  - ReceiveState ; SRN: 027
    The standard state in a PASS subject behavior diagram denoting an receive action or rather the waiting for a receive possibility.
  - SendState ; SRN: 028
    The standard state in a PASS subject behavior diagram denoting a send action
- StateReference ; SRN: 029
  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 ; SRN: 030
  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 ; SRN: 031
    A super class for the CommunicationTransitions.
    - ReceiveTransition ; SRN: 032
      Comments have to be added
    - SendTransition ; SRN: 033
      Comments have to be added
  - DoTransition ; SRN: 034
    Comments have to be added
  - SendingFailedTransition ; SRN: 035
    Comments have to be added
  - TimeTransition ; SRN: 036
    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 ; SRN: 037
      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 ; SRN: 038
        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 ; SRN: 039
        Comments have to be added
    - TimerTransition ; SRN: 040
      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 ; SRN: 041
  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 ; SRN: 042
  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 ; SRN: 044
  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 ; SRN: 045
  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 ; SRN: 046
  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 transi-

tion 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 ; SRN: 047
  A TransitionCondition for the according DoTransitions and DoStates.
- MessageExchangeCondition ; SRN: 048
  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 ; SRN: 049
    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 ; SRN: 050
    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 ; SRN: 051
  Comments have to be added
- TimeTransitionCondition ; SRN: 052
  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 ; SRN: 053

Comments have to be added

· TimerTransitionCondition ; SRN: 054
Comments have to be added

– DataDescribingComponent ; SRN: 055
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 ; SRN: 056
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 ; SRN: 057
Data definition concept for PASS model build in capabili-

ties of data modeling. Defines a simple list structure.

  · PayloadDataObjectDefinition ; SRN: 058
    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 ; SRN: 059
    Comments have to be added

* DataTypeDefinition ; SRN: 060
  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 ; SRN: 061
    Using this class, tool vendors can include their own custom data definitions in the model.

    · JSONDataTypeDefinition ; SRN: 062
      Comments have to be added

    · OWLDataTypeDefinition ; SRN: 63
      Comments have to be added

    · XSD-DataTypeDefinition ; SRN: 064
      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 ; SRN: 065
    Comments have to be added

* PayloadDescription ; SRN: 066
  Comments have to be added

  · PayloadDataObjectDefinition ; SRN: 067
    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 ; SRN: 068
    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 ; SRN: 069
  This class is the super class of all model elements used to define or specify the interaction means within a process model
  * InputPoolConstraint ; SRN: 070
    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 ; SRN: 071
      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 ; SRN: 072
      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 ; SRN: 073
      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 ; SRN: 074
  Should an InputPoolConstraint 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.
  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 ; SRN: 075
  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 ; SRN: 076
  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 ; SRN: 077
  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 ; SRN: 078
  The subject is the core model element of a subject-oriented

PASS process model.

- FullySpecifiedSubject ; SRN: 079
  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 ; SRN: 080
  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 ; SRN: 081
  The Multi-Subject is term for a subject that "has a maximum subject instantiation restriction" within a process context larger than 1.
- SingleSubject ; SRN: 082
  Single Subject are subject with a maximumInstanceRestriction of 1
- StartSubject ; SRN: 083
  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 ; SRN: 084
  The main class that contains all relevant process elements

– SubjectBehavior ; SRN: 085
  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 ; SRN: 086
    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 ; SRN: 087
    A macro behavior is a specialized behavior that may be entered

    and exited from a function state in another behavior.

    * SubjectBaseBehavior ; SRN: 088
      The standard behavior model type

- SimplePASSElement ; SRN: 089
  Comments have to be added

  - CommunicationTransition ; SRN: 090
    A super class for the CommunicationTransitions.

    * ReceiveTransition ; SRN: 091
      Comments have to be added
    * SendTransition ; SRN: 092
      Comments have to be added

  - DataMappingFunction ; SRN: 093
    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 ; SRN: 094
      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 ; SRN: 095
      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 ; SRN: 096
    Comments have to be added

  - DoTransitionCondition ; SRN: 097
    A TransitionCondition for the according DoTransitions and DoStates.

- EndState ; SRN: 098
  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 ; SRN: 099
  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 ; SRN: 100
    A super class for specialized FunctionSpecification of communication acts (send and receive)
    · ReceiveFunction ; SRN: 101
      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 ; SRN: 102
      Comments have to be added
  * DoFunction ; SRN: 103
    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 ; SRN: 104
  The initial state of a behavior
- MessageExchange ; SRN: 105
  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 ; SRN: 106
  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 ; SRN: 107
    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 ; SRN: 108
    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 ; SRN: 109
  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 ; SRN: 110
  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 ; SRN: 111
  Comments have to be added
- PayloadDataObjectDefinition ; SRN: 112
  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 ; SRN: 113
  A super class to the standard PASS states: Do, Receive and Send
  * DoState ; SRN: 114
    The standard state in a PASS subject behavior diagram denoting an action or activity of the subject in itself.
  * ReceiveState ; SRN: 115
    The standard state in a PASS subject behavior diagram denoting an receive action or rather the waiting for a receive possibility.
  * SendState ; SRN: 116
    The standard state in a PASS subject behavior diagram denoting a send action
- Subject ; SRN: 117
  The subject is the core model element of a subject-oriented PASS process model.
  * FullySpecifiedSubject ; SRN: 118
    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 ; SRN: 119
    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 ; SRN: 120
      The Multi-Subject is term for a subject that "has a maximum subject instantiation restriction" within a process context larger than 1.
    * SingleSubject ; SRN: 121
      Single Subject are subject with a maximumInstanceRestriction of 1
    * StartSubject ; SRN: 122
      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 ; SRN: 123
    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:		See https://www.omg.org/spec/DMN for specification of Feel-Statement-Strings The idea of these expression is to map data fields from and to the internal Data storage of a subject	

Property name		Domain-Range	Comments	Reference
hasGraphicalRepresentation	Domain: Range:		The process models are in principle abstract graph structures. Yet the visualization of process models is very important since many process models are initially created in a graphical form using a graphical graph editor (e.g. MS Visio, yEd, etc.) that was created to foster human comprehensibility. If available any process element may have a graphical representation attached to it	
hasKey	Domain: Range:			
hasLimit	Domain: Range:			
hasMaximumSubjectInstanceRestriction	Domain: Range:			

Property name		Domain-Range	Comments	Reference
hasMetaData	Domain: Range:			
hasModelComponentComment	Domain: Range:		equivalent to rdfs:comment	
hasModelComponentID	Domain: Range:		The unique ID of a PASSProcessModel-Component	
hasModelComponentLabel	Domain: Range:		The human legible label or description of a model element.	

Property name		Domain-Range	Comments	Reference
hasPriorityNumber	Domain: Range:		Transitions or Behaviors have numbers that denote their execution priority in situations where two or more options could be executed. This is important for automated execution. E.g. when two messages are in the inbox and could be followed, the message denoted on the transition with the higher priority (lower priority number) is taken out and processed. Similarly, SubjectBehaviors with higher priority (lower priority number) are to be executed before Behaviors with lower priority.	

Property name		Domain-Range	Comments	Reference
hasReoccuranceFrequenyOrDate	Domain: Range:		A data field meant for the two classes ReoccuranceTime-OutTransition and ReoccuranceTimeOutExitCondition. ToDo: Define the according data format for describing the iteration frequencies or reoccurring dates. Opinion: rather complex: expressive capabilities should cover expressions like: "every 2nd Monday of Month at 7:30 in Morning." Every 29th of July" or "Every Hour", "ever 25 Minuets" , "once each day", "twice each week" etc	

Property name		Domain-Range	Comments	Reference
hasSVGRepresentation	Domain: Range:		The Scalable Vector Graphic (SVG) XML format is a text based standard to describe vector graphics. Adding according image information as XML literals is therefor a suitable, yet not necessarily easily changeable option to include the graphical representation of model elements in the an OWL file.	
hasTimeBasedReoccuranceFrequencyOrDate	Domain: Range:			
hasTimeValue	Domain: Range:		Generic super class for all data properties of time based transitions.	

Property name		Domain-Range	Comments	Reference
hasToolSpecificDefinition	Domain: Range:		This is a placeholder DataProperty meant as a tie in point for tool vendors to include tool specific data values/properties into models. By denoting their own data properties as subclasses to this one the according data fields can easily be recognized as such. However, this is only an option and a place holder to remind that something like this is possible.	
hasValue	Domain: Range:			
hasYearMonthDurationTimeOutTime	Domain: Range:			
isOptionalToEndChoiceSegmentPath	Domain: Range:			

Property name		Domain-Range	Comments	Reference
isOptionalToStartChoiceSegmentPath	Domain: Range:			
owl:topDataProperty	Domain: Range:			
PASSModelDataProperty	Domain: Range:		Generic super class of all DataProperties that PASS process model elements may have.	
SimplePASSDataProperties	Domain: Range:		Every element/sub-class of SimplePASS-DataProperties is also a Child of PASSMod-elDataPropertiy. This is simply a surro-gate class to group all simple elements together	

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).	200
contains	Domain: Range:	PASSProcessModelElement PASSProcessModelElement	Generic ObjectProperty that links two model elements where one contains another (possible multiple)	201
containsBaseBehavior	Domain: Range:	Subject SubjectBehavior		202
containsBehavior	Domain: Range:	Subject SubjectBehavior		203
containsPayload-Description	Domain: Range:	MessageSpecification PayloadDescription		204
guardedBy	Domain: Range:	State, Action GuardBehavior		205

Property name		Domain-Range	Comments	Reference
guardsBehavior	Domain:	GuardBehavior	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.	206
	Range:	SubjectBehavior		
guardsState	Domain:	State, Action		207
	Range:	guardedBy		
hasAdditionalAttribute	Domain:	PASSProcessModelElement		208
	Range:	AdditionalAttribute		
hasCorrespondent	Domain:		Generic super class for the ObjectProperties that link a Subject with a MessageExchange either in the role of Sender or Receiver.	209
	Range:	Subject		
hasDataDefinition	Domain:			210
	Range:	DataObjectDefinition		

Property name		Domain-Range	Comments	Reference
hasDataMapping-Function	Domain: Range:	state, SendTransition, ReceiveTransition DataMappingFunction		211
hasDataType	Domain: Range:	PayloadDescription or DataObjectDefinition DataTypeDefinition		212
hasEndState	Domain: Range:	SubjectBehavior or Choice-SegmentPath State, not SendState		213
hasFunction-Specification	Domain: Range:	State FunctionSpecification		214
hasHandlingStrategy	Domain: Range:	InputPoolConstraint InputPoolContstraint-HandlingStrategy		215
hasIncomingMessage-Exchange	Domain: Range:	Subject MessageExchange		216
hasIncomingTransition	Domain: Range:	State Transition		217
hasInitialState	Domain: Range:	SubjectBehavior or Choice-SegmentPath State		218

Property name		Domain-Range	Comments	Reference
hasInputPoolConstraint	Domain: Range:	Subject InputPoolConstraint		219
hasKeyValuePair	Domain: Range:			220
hasMessageExchange	Domain: Range:	Subject	Generic super class for the ObjectProperties linking a subject with either incoming or outgoing MessageExchanges.	221
hasMessageType	Domain: Range:	MessageTypeConstraint or MessageSenderTypeConstraint or MessageExchange MessageSpecification		222
hasOutgoingMessageExchange	Domain: Range:	Subject MessageExchange		223
hasOutgoingTransition	Domain: Range:	State Transition		224
hasReceiver	Domain: Range:	MessageExchange Subject		225

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.	226
hasSender	Domain: Range:	MessageExchange Subject		227
hasSourceState	Domain: Range:	Transition State		228
hasStartSubject	Domain: Range:	PASSProcessModel StartSubject		229
hasTargetState	Domain Range	Transition State		230
hasTransitionCondition	Domain Range	Transition TransitionCondition		231
isBaseBehaviorOf	Domain: Range:	SubjectBaseBehavior	A specialized version of the ”belongsTo” Object-Property to denote that a -SubjectBehavior belongs to a Subject as its BaseBehavior	232
isEndStateOf	Domain: Range:	State and not SendState SubjectBehavior or Choice-SegmentPath		233

Property name		Domain-Range	Comments	Reference
isInitialStateOf	Domain: Range:	State SubjectBehavior or Choice-SegmentPath		234
isReferencedBy	Domain: Range:			235
references	Domain: Range:			236
referencesMacroBehavior	Domain: Range:	MacroState MacroBehavior		237
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.	238
requiresActiveReception-OfMessage	Domain: Range:	ReceiveTransitionCondition MessageSpecification		239
requiresPerformed-MessageExchange	Domain: Range:	MessageExchangeCondition MessageExchange		240

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	241

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 represent the behavior of an active entity. A specification of a subject does not say anything about the technology used to execute the described behavior. Subjects communicate with each other by exchanging messages. Messages have a name and a payload. The name should express the meaning of a message informally and the payloads are the data (business objects) transported. Internally subjects execute local activities such as calculating a price, storing an address etc. A subject sends messages to other subjects, expects messages from other subjects, and executes internal actions. All these activities are done in sequences which are defined in a subject's behavior specification.

In the following we use an 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.

There are the following types of subjects:

- Fully specified subjects

- Multisubjects
- Single subject
- Interface subjects

Fully specified Subjects

This is the standard subject type. A subject communicates with other subjects by exchangeing messages. Fully specified subjects consists of following components:

- Business Objects
  Each subjects has some business objects. A basic structure of business objects consists of an identifier, data structures, and data elements. The identifier of a business object is derived from the business environment in which it is used. Examples are business trip requests, purchase orders, packing lists, invoices, etc. Business objects are composed of data structures. Their components can be simple data elements of a certain type (e.g., string or number) or even data structures themselves.

- Sent messages
  Messages which a subject sends to other subjects. Each message has a name and may transport some data objects as a payload. The values of these payload data objects are copied from internal business objects of a subject.

- Received messages
  Messages received by a subject. The values of the payload objects are copied to business objects of the receiving subject.

- Input Pool
  Messages sent to a subjects are deposited in the input pool of the receiving subject.

- Behavior
  The behavior of each subject describes in which order it sends messages, expects (receives) and performs internal functions. Messages transport data from the sending to the receiving subject, and internal functions operate on internal data of a subject.

Multsubjects and Multiprocesses

Multisubjects are simular to Fully specified subjects. If in a process model several identical subjects are required e.g. in order to increse the through put these subjects can be modelled by a multi subject. If several communicating subjects in a process modell are multi subjects they can be combined to a multi process.

In a business process, there may be several identical sub-processes that perform certain similar tasks in parallel and independently. This is often the case in a procurement process, when bids from multiple providers are solicited. A process or sub-process is therefore executed simultaneously or sequentially multiple times during overall process execution. A set of type-identical, independently running processes or sub-processes are termed multi-process. The actual number of these independent sub-processes is determined at runtime. Multi-processes simplify process execution, since a specific sequence of actions can be used by different processes. They are recommended for recurring structures and similar process flows. An example of a multi-process can be illustrated as a variation of the current booking process. The travel agent should simultaneously solicit up to five bids before making a reservation. Once three offers have been received, one is selected and a room is booked. The process of obtaining offers from the hotels is identical for each hotel and is therefore modeled as a multi-process.

Single subjects

Single subjects can be instantiated only once. They are used if for the execution of a subject a resource is required which is only available once.

Interface Subjects

Interface subjects are used as interfaces to other process systems. If a subject of a process system sends or receives messages from a subject which belongs to an other process system. These so called interface subjects represent fully described subjects which belong to that other process system. This means to each interface subject belongs a fully described subject in an other process system. Interface subjects specifications contain the sent messages, received messages and the refernce to the fully described subject which they represent.

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).

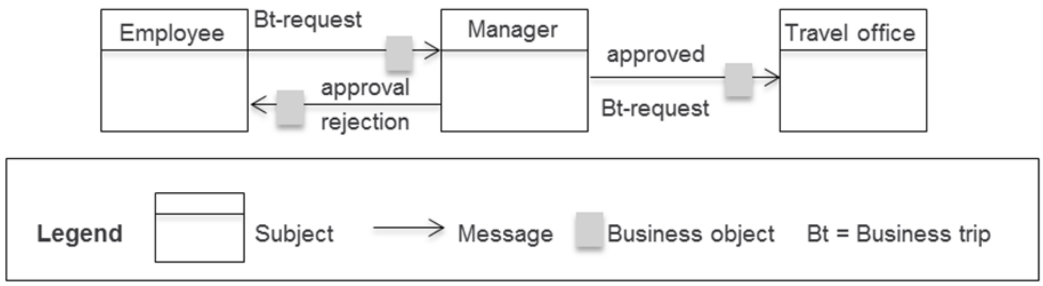

Figure 3.1: Subject interaction diagram for the process 'business trip application'

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.

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

The various building blocks of a PASS description and their relations are defined in a ontology.

3.2.1 Process Model

The following figure 3.3 shows athe subset of the classes and properties of the ontology. This subset allows to express the subject interaction diagram.

The central classes are Subject and MessageExchange. Between these classes are defined the properties hasIncomingTransition (in figure 3.3 number 217) and hasOutgoingTransition (in figure 3.3 number 224). This properties defines that subjects have incoming and outgoing messages. Each message has a sender and a receiver (in figure 3.3 number 227 and number 225). Messages have a type. This is expressed by the property hasMessageType (in figure 3.3 number 222). Instead of the property 222 a message exchange may have the property 201 if a list of messages is used instead of a single message.

Each subject has an input pool. Input pools have three types of constraints (see section 3.1.3). This is expressed by the property references (in figure 3.3 number 236) and InputPoolConstraints (in figure 3.3 number 219). Constraints which are related to certain messages have references to the class MessageSpecification.

There are four subclasses of the class subject (in figure 3.3 number 079, 080, 081 and 082). The specialties of these subclasses are described in section 3.1.1. A class StartSubject (in figure 3.3 number 83) which is a subclass of class subject denotes the subject in which a process instance is started.

All other relarions are subclass relations. The class PASSProcessModelElement is the central PASS class. From this class all the other classes are derived (see next sections). From class InteractionDescribingCOmponent all the classes required for describing the structure of a process system are derived.

3.2.2 Subjects

Different types of subjects 3.4

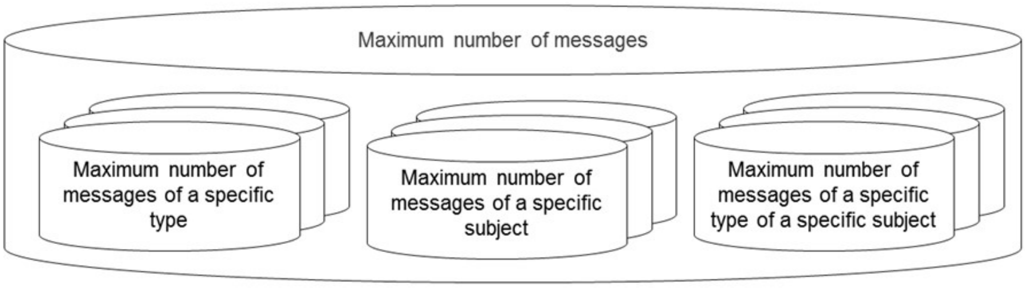

Figure 3.2: Configuration of Input Pool Parameters

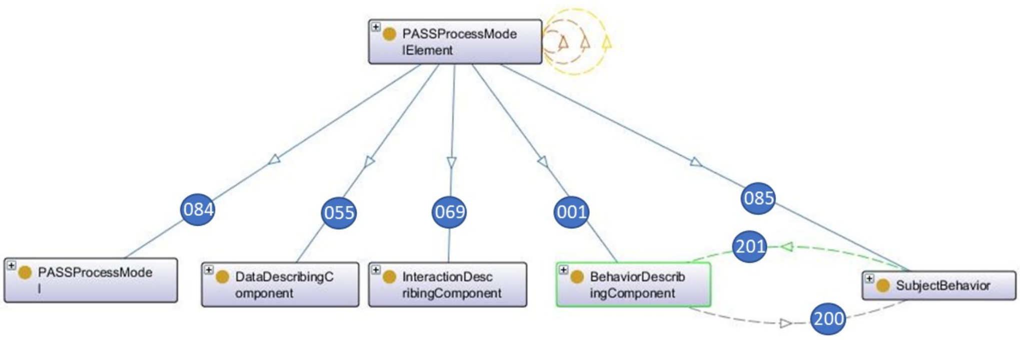

Figure 3.3: Elements of PASS Process Models

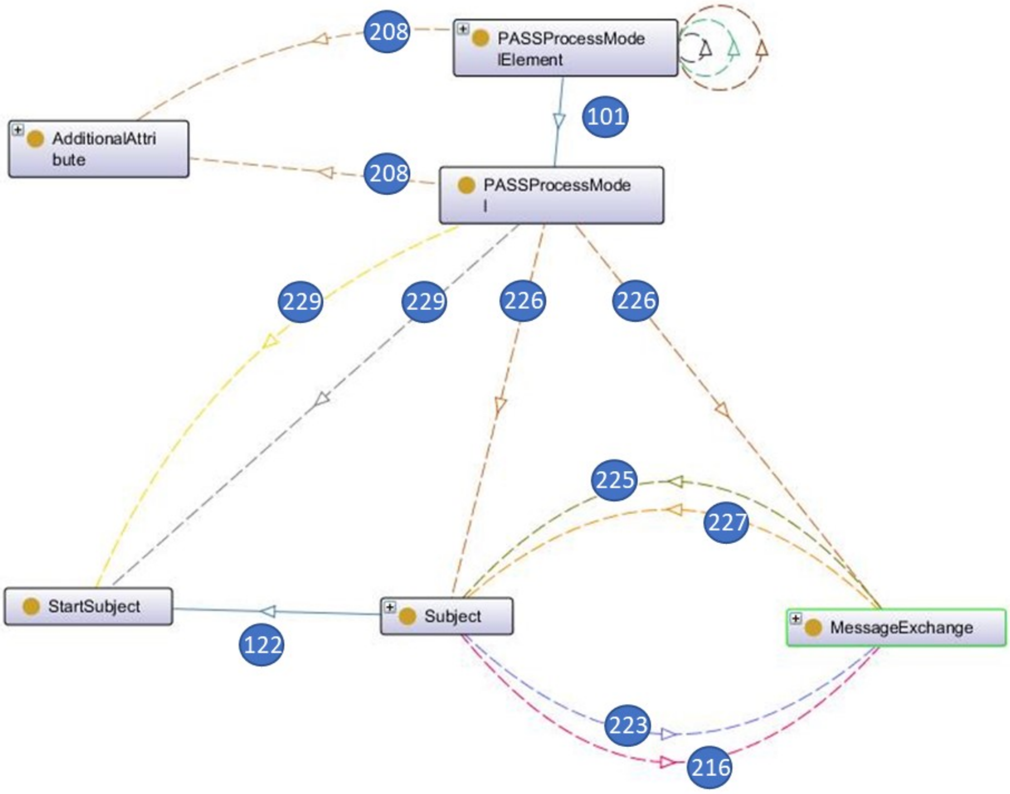

Figure 3.4: PASS Process Modell

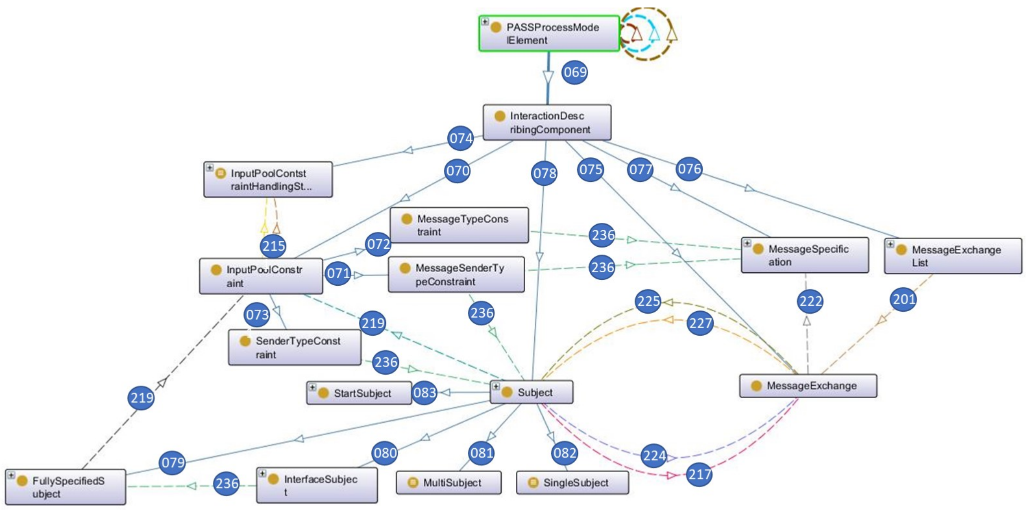

Figure 3.5: Subject Interaction Diagram

3.2.3 Messages

SDescription of messages 3.5

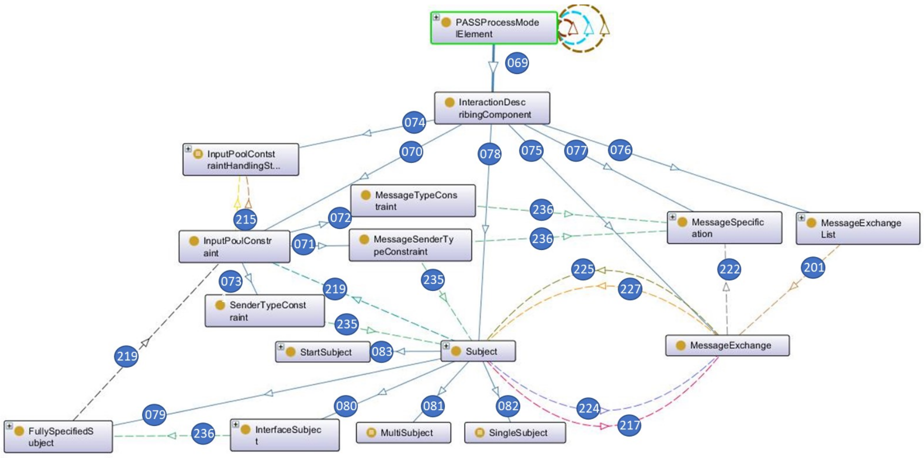

Figure 3.6: Different Types of Subjects

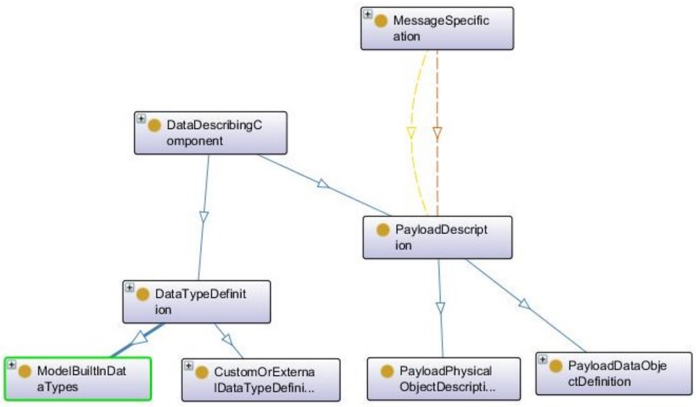

Figure 3.7: Message Specification with Payload

3.2.4 Input Pools

Description of input pools 3.7

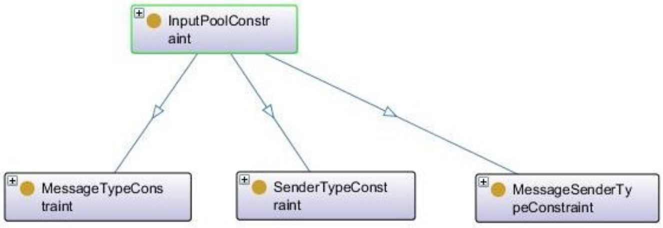

Figure 3.8: Input Pool description

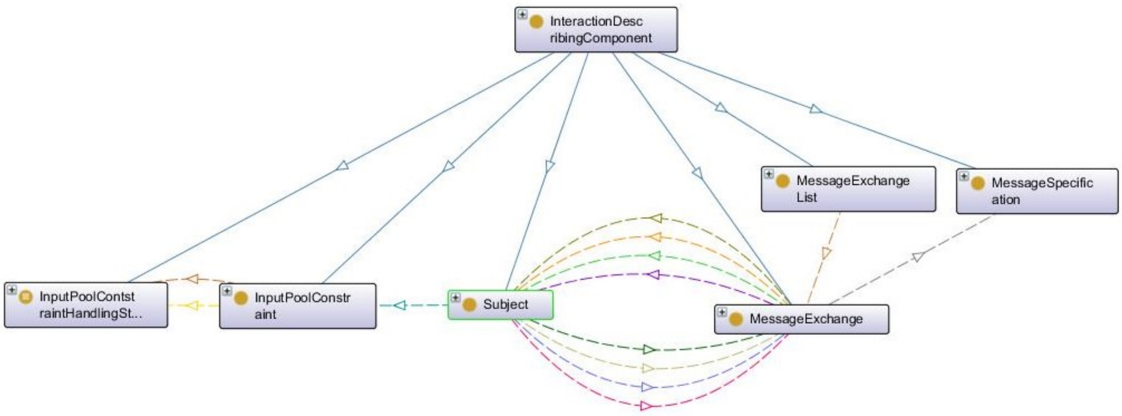

Figure 3.9: Message Exchange and Input Pools

3.3 ASM Description