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
    - Action
    - DataMappingFunction
      - DataMappingIncomingToLocal
      - DataMappingLocalToOutgoing
    - FunctionSpecification
      - CommunicationAct
        - ReceiveFunction
        - SendFunction
      - DoFunction
    - ReceiveType
    - SendType
    - State
      - ChoiceSegment
      - ChoiceSegmentPath
        - MandatoryToEndChoiceSegmentPath
        - MandatoryToStartChoiceSegmentPath
        - OptionalToEndChoiceSegmentPath
        - OptionalToStartChoiceSegmentPath

- EndState
- GenericReturnToOriginReference
- InitialStateOfBehavior
- InitialStateOfChoiceSegmentPath
- MacroState
- StandardPASSState
  - DoState
  - ReceiveState
  - SendState
- StateReference

* Transition
  - CommunicationTransition
    - ReceiveTransition
    - SendTransition
  - DoTransition
  - SendingFailedTransition
  - TimeTransition
    - ReminderTransition
      - CalendarBasedReminderTransition
      - TimeBasedReminderTransition
    - TimerTransition
      - BusinessDayTimerTransition
      - DayTimeTimerTransition
      - YearMonthTimerTransition
    - UserCancelTransition
  - TransitionCondition
    - DoTransitionCondition
    - MessageExchangeCondition
      - ReceiveTransitionCondition
      - SendTransitionCondition
    - SendingFailedCondition
    - TimeTransitionCondition
      - ReminderEventTransitionCondition
      - TimerTransitionCondition

– DataDescribingComponent

        * DataObjectDefinition
            · DataObjectListDefintion
            · PayloadDataObjectDefinition
            · SubjectDataDefinition
        * DataTypeDefinition
            · CustomOrExternalDataTypeDefinition
                · JSONDataTypeDefinition
                · OWLDataTypeDefinition
                · XSD-DataTypeDefinition
            · ModelBuiltInDataTypes
        * PayloadDescription
            · PayloadDataObjectDefinition
            · PayloadPhysicalObjectDescription
    – InteractionDescribingComponent
        * InputPoolConstraint
            · MessageSenderTypeConstraint
            · MessageTypeConstraint
            · SenderTypeConstraint
        * InputPoolContstraintHandlingStrategy
        * MessageExchange
        * MessageExchangeList
        * MessageSpecification
        * Subject
            · FullySpecifiedSubject
            · InterfaceSubject
            · MultiSubject
            · SingleSubject
            · StartSubject
    – PASSProcessModel
    – SubjectBehavior
        * GuardBehavior
        * MacroBehavior
        * SubjectBaseBehavior

- SimplePASSElement

- CommunicationTransition
  - ReceiveTransition
  - SendTransition
- DataMappingFunction
  - DataMappingIncomingToLocal
  - DataMappingLocalToOutgoing
- DoTransition
- DoTransitionCondition
- EndState
- FunctionSpecification
  - CommunicationAct
    - ReceiveFunction
    - SendFunction
  - DoFunction
- InitialStateOfBehavior
- MessageExchange
- MessageExchangeCondition
  - ReceiveTransitionCondition
  - SendTransitionCondition
- MessageExchangeList
- MessageSpecification
- ModelBuiltInDataTypes
- PayloadDataObjectDefinition
- StandardPASSState
  - DoState
  - ReceiveState
  - SendState
- Subject
  - FullySpecifiedSubject
  - InterfaceSubject
  - MultiSubject
  - SingleSubject
  - StartSubject
- SubjectBaseBehavior

2.2 Data Properties (27)

hasBusinessDayDurationTimeOutTime hasCalendarBasedFrequencyOrDate hasDataMappingString hasDayTimeDurationTimeOutTime hasDurationTimeOutTime hasFeelExpressionAsDataMapping hasGraphicalRepresentation hasKey hasLimit hasMaximumSubjectInstanceRestriction hasMetaData hasModelComponentComment hasModelComponentID hasModelComponentLabel hasPriorityNumber hasReoccuranceFrequenyOrDate hasSVGRepresentation hasTimeBasedReoccuranceFrequencyOrDate hasTimeValue hasToolSpecificDefinition hasValue hasYearMonthDurationTimeOutTime isOptionalToEndChoiceSegmentPath isOptionalToStartChoiceSegmentPath owl:topDataProperty PASSModelDataProperty SimplePASSDataProperties

2.3 Object Properties (42)

Property name	Domain	Range	Reference
belongsTo			
contains			
belongsTo	PASSProcessModelElement	PASSProcessModelElement	
contains	PASSProcessModelElement	PASSProcessModelElement	
containsBaseBehavior	Subject	SubjectBehavior	
containsBehavior	Subject	SubjectBehavior	
containsPayloadDescription	MessageSpecification	PayloadDescription	
guardedBy	State, Action	GuardBehavior	
guardsBehavior	GuardBehavior	SubjectBehavior	
guardsState	State, Action	guardedBy	
hasAdditionalAttribute	PASSProcessModelElement	AdditionalAttribute	
hasCorrespondent		Subject	
hasDataDefinition		DataObjectDefinition	
hasDataMappingFunction	state, SendTransition, ReceiveTransition	DataMappingFunction	
hasDataType	PayloadDescription or DataObjectDefinition	DataTypeDefinition	
hasEndState	SubjectBehavior or ChoiceSegmentPath	State, not SendState	
hasFunctionSpecification	State	FunctionSpecification	
hasHandlingStrategy	InputPoolConstraint	InputPoolContstraintHandlingStrategy	
hasIncomingMessageExchange	Subject	MessageExchange	
hasIncomingTransition	State	Transition	

Property name	Domain	Range	Reference
hasInitialState	SubjectBehavior or ChoiceSegmentPath	State	
hasInputPoolConstraint	Subject	InputPoolConstraint	
hasKeyValuePair			
hasMessageExchange	Subject		
hasMessageType	MessageTypeConstraint or MessageSender-TypeConstraint or MessageExchange	MessageSpecification	
hasOutgoingMessageExchange	Subject	MessageExchange	
hasOutgoingTransition	State	Transition	
hasReceiver	MessageExchange	Subject	
hasRelationToModelComponent	PASSProcessModelElement	PASSProcessModelElement	
hasSender	MessageExchange	Subject	
hasSourceState	Transition	State	
hasStartSubject	PASSProcessModel	StartSubject	
hasTargetState	Transition	State	
hasTransitionCondition	Transition	TransitionCondition	
isBaseBehaviorOf	SubjectBaseBehavior		
isEndStateOf	State and not Send-State	SubjectBehavior or ChoiceSegmentPath	
isInitialStateOf	State	SubjectBehavior or ChoiceSegmentPath	
isReferencedBy			
references			
referencesMacroBehavior	MacroState	MacroBehavior	

Property name	Domain	Range	Reference
refersTo	CommunicationTransitionConditionMessageExchange		
requiresActiveReceptionOfMessage	ReceiveTransitionConditionMessageSpecification		
requiresPerformedMessageExchange	MessageExchangeConditionMessageExchange		
SimplePASSObjectPropertie			

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

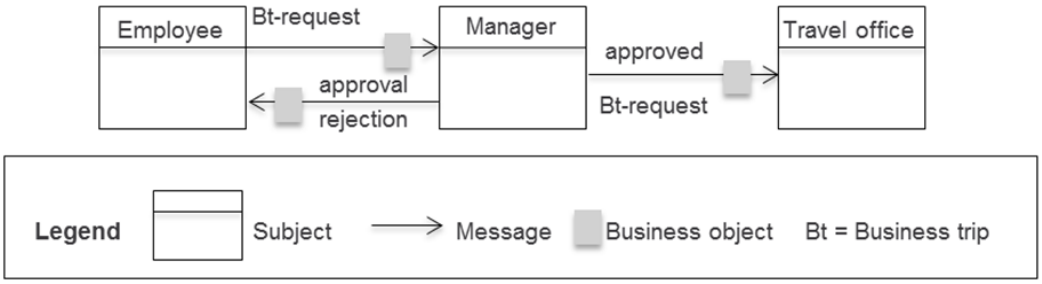

Figure 3.1: Subject interaction diagram for the process 'business trip application'

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.

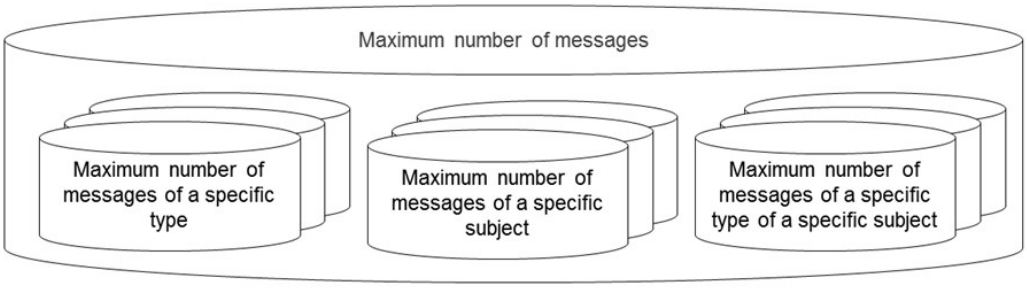

Figure 3.2: Configuration of Input Pool Parameters

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

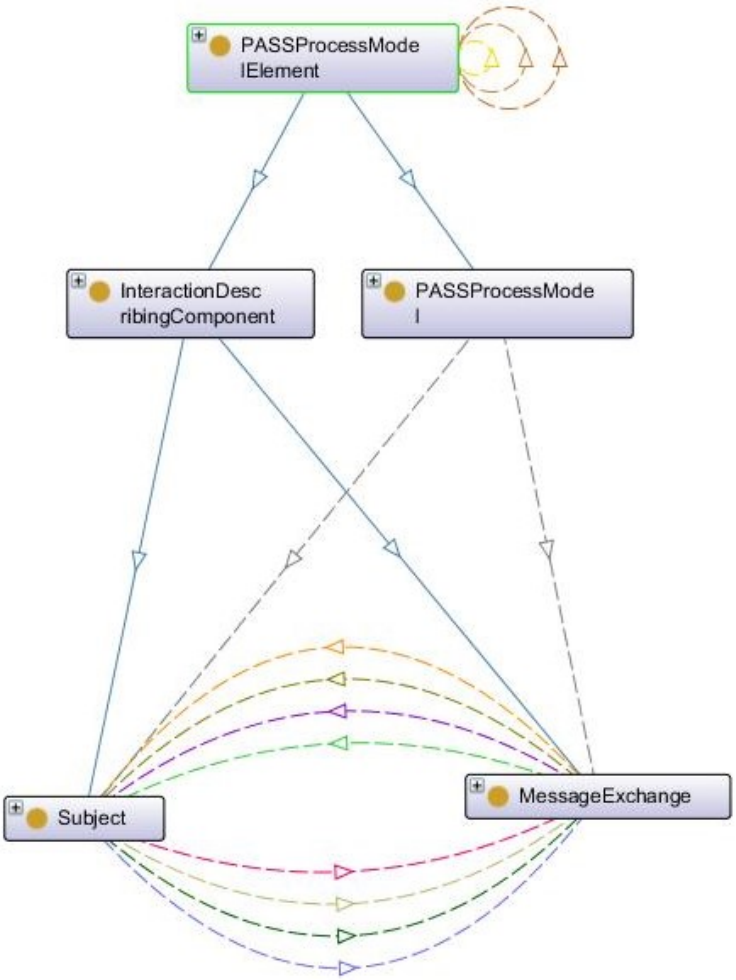

Figure 3.3: Subject Interaction

3.2.2 Subjects

Different types of subjects 3.4

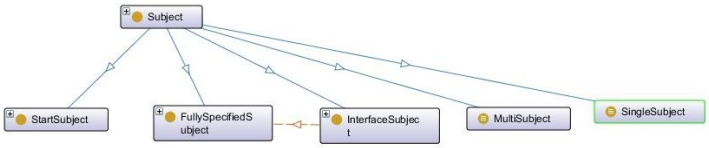

Figure 3.4: Different Types of Subjects

3.2.3 Messages

SDescription of messages 3.5

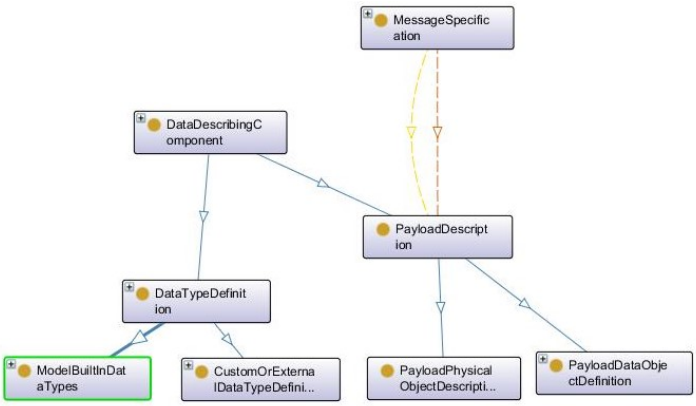

Figure 3.5: Message Specification with Payload

3.2.4 Input Pools

Description of input pools 3.7

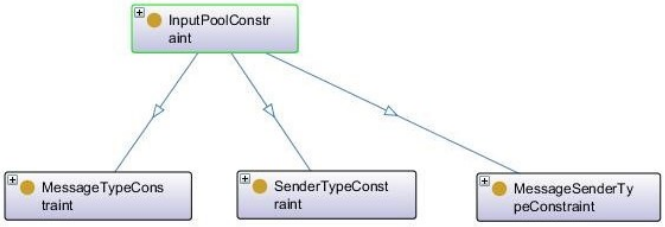

Figure 3.6: Input Pool description

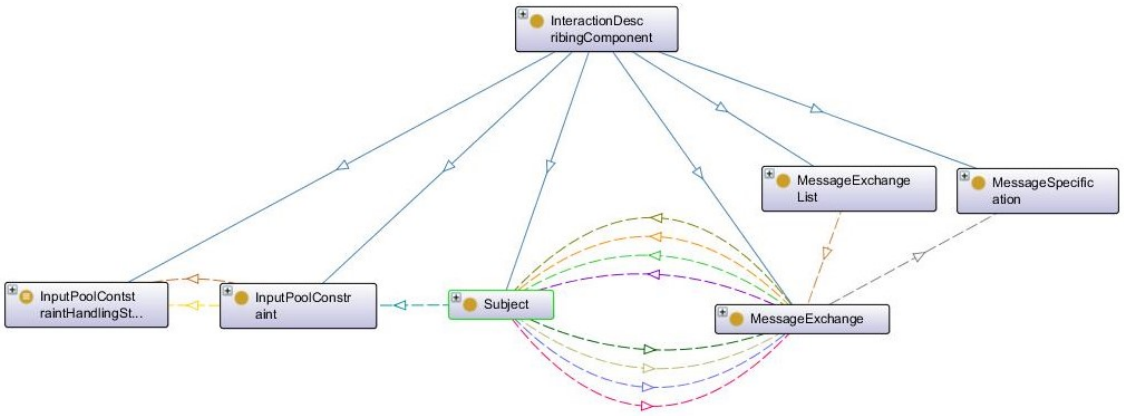

Figure 3.7: Message Exchange and Input Pools

3.3 ASM Description