Requirement definition

General KPU design

BREANOS GmbH

Neutorstraße 13
5020 Salzburg

AUSTRIA

Tel: +43 (662) 276198-11

Fax: +43 (662) 276198-98

Mail: office@breanos.com

File RD_GeneralKpuStructure.docx

Date 27.04.2018

Table of contents

1 Change history

Date	Version	Author	Description
25.04.2018	1	Dominik Hutterer	Initial version
26. 04.2018	1	Florian Krisch	Fleshing out

2 Requirement definition

2.1 Introduction

A *KPU* is basic functional element within the BREANOS system. It is realized as microservice and hosted in a *SFA-Cluster*.

A *KPU* acts a bit similar to a neuron in a neural network. Multiple inputs (inhibitory, excitatory) are combined within the neuron (OR, XOR, AND …) to trigger the *KPU's* specific action in its body to produce a defined output. This can be realized via several *Activities* in a superordinate *Workflow* or for very primitive *KPUs* without *Workflow*.

Each KPU is instanced either by a *Worker* (which represents a production unit, like a production line) or directly by the *Personal Assistant AI (ADAM)*. The lifetime of the KPU is managed by itself, or rather by its workflow. So, if the workflow terminates, the resources of the KPU have to be set free. Several workers can also be organized in so called *Worker Groups*.

The functionality of *KPUs* is wide, and ranges from analysis and converters to machine, transport or logistic units, handling the production. If a *KPU* produces or requires data from a database, this data is stored in *Data Marts* which keep synchronized with the data warehouse.

2.2 Structure

2.2.1 Input

Each *KPU* has to implement an interface to handover an input object (an input transition) `[void SetInput(object inputValue)]`. Since the blackboard system is agnostic of the objects it distributes, recipients are chosen by the sender. This means that a receiving KPU must always accept Input objects. If an Input object is of unknown type or purpose to the KPU, this must be communicated towards the controlling systems (ADAM) as an error.

It also falls within the responsibility of each KPU to allow for constant receiving (as an asynchronous activity, independently of workload execution) and buffering of Input objects. Additionally, the KPU is responsible for persisting all received objects, and handle status recovery after errors.

2.2.2 Body

The inputs form a set of requirements for the execution of work. Therefore, all Input objects (or Input object types) can be freely combined with logical operators (AND, OR, XOR). Once the requirements are met, i.e. evaluation of requirements equals `true`, the workload execution is activated. This workload execution code (workflow, functional code) forms the so-called Body of the KPU.

The body of the KPU also implements all external communication, e.g. machine communication.

2.2.3 Output

Execution and termination of one workload results in exactly zero or one output transitions. These are forwarded to the blackboard, or other KPUs, and serve as input transitions for those.

2.2.4 Error Handling

If an error occurs during execution of the body, the output transition by default becomes an error object to be forwarded to the blackboard / ADAM. In such a case, the KPU has to decide what happens with the input objects handled during this body execution.

Also, the KPU is required to check at boot time if there are any unfinished / unhandled input or output objects from a past workload execution, if there are input objects inside its persistent buffer, and handle those accordingly.