Requirement definition

State Machine



Neutorstraße 13  
5020 Salzburg

AUSTRIA

Tel: +43 (662) 276198-11

Fax: +43 (662) 276198-98

Mail: [office@breanos.com](mailto:office@breanos.com)

|  |  |
| --- | --- |
| File | Requirements definition State Machine.docx |
| Date | 24.08.2018 |

Table of contents

[1 Change history 3](#_Toc522199292)

[2 Requirement definition 4](#_Toc522199293)

[2.1 Introduction 4](#_Toc522199294)

[2.2 Scope 4](#_Toc522199295)

[2.3 Exclusion from scope 4](#_Toc522199296)

[2.4 Action plan 4](#_Toc522199297)

[2.5 Effort estimation 4](#_Toc522199298)

[2.5.1 Annotations 4](#_Toc522199299)

[2.6 Risk assessment 5](#_Toc522199300)

[2.6.1 Risk short description 5](#_Toc522199301)

[2.6.1.1 Description 5](#_Toc522199302)

[2.6.1.2 Consequences 5](#_Toc522199303)

[2.6.1.3 Avoidance 5](#_Toc522199304)

[2.6.1.4 Probability 5](#_Toc522199305)

[2.7 Prerequisites 5](#_Toc522199306)

[2.8 Acceptance 5](#_Toc522199307)

# Change history

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
| 16.08.2018 | 0.1 | EBE | Initial version |
| 23.8.2018 | 0.2 | FUL, FKE | Real initial version |

Abbreviations / Terms

|  |  |
| --- | --- |
| BIF | BREANOS Industrial work*Flow* |
| KPU | Knowledge Processing Unit |
| DAIPAN | BREANOS’s proprietary framework for its suite of software |

# Requirement definition

## Introduction

The Wexflow Workflow has to be extended to our needs. Wexflow has been taken as a working base for the BIF. The runtime environment of Wexflow has thus been dubbed “BIF Engine”.

In this document the State Machine is described in more detail especially in relation to the current implementation of Wexflow and how to expand upon it.

Wexflow doesn’t intrinsically support the creation of state machines. Wexflow is a framework for sequential workflows and flowcharts with direct, conditional and iterative flow control via a workflow’s definition in XML.

For BIF Engine we need a more flexible way to define machine and KPU Workflows.

Historically, Petri-Nets have been defined by places and transitions. Additionally, the current state of the net is presented as one or several tokens that is/are being passed around by the places and transitions. If and only if a transition has the token, it can fire. If and only if a state has the token, it is active. The token-semantic is not currently built into Wexflow/BIF.

This state/token can/should contain all the necessary information e.g. for restarting/resuming the workflow after breakdown/shutdown. This can then be used for consistent passing of information between activities. Such a system would contrast the current file-based communication model used by Wexflow itself.

## Scope

The scope of this project is the part of the modification of Wexflow into BIF regarding the state machine that a workflow will be represented by in the BIF Engine.

## Exclusion from scope

* All parts of the BIF/BIF Engine software not regarding the implementation of the state machine functionality
* All other parts of DAIPAN

## Action plan

* The token-semantic as described in 2.1 is currently not built into Wexflow/BIF
* Wexflow itself doesn’t offer a ‘state’ for making it a state-machine.
* Therefore, Wexflow needs to be expanded upon by making it compatible with the node-edge duality of petri-nets’ “places” and transitions
* Additionally, the current state of the flow should be presented, i.e. able to be persisted as a state-representing token being passed around the by the graph’s flow

## Effort estimation

The implementation effort is estimated as follows.

|  |  |
| --- | --- |
| Task | Hours |
| Analysis: extension of the Workflow-XML-definition and serializer/de-serializer to support token/state properties-definition | 30 |
| Workflow-XML’s token/state implementation | 20 |
| Analysis: extension for BIF Activities to use the XML-defined tokens to pass/persist state information and to pass the tokens as the activation mechanism of the activities | 30 |
| Activities’ token passing implementation | 30 |
| Analysis: creating a functionality similar to DataBindings of WPF/Xamarain markup for flow control conditionals/iterative steps to use actual variables instead of static literals. Note: estimate is not included in total. | 999 |
| DataBinding parser implementation; Note: estimate is not included in total. | 999 |
| Total | **84** |

## Risk assessment

### Wexflow engine might be too inflexible to adapt to our needs (with reasonable/feasible effort)

#### Description

Wexflow is a complex software which might have restrictions in its implementation that cannot, without massive cost in man-hours, be changed.

**Probability:**

It’s a C# library of which we have the source code. It’s got an XML de-serializer which is plugged into the execution engine. It shouldn’t be expected that the change will go completely “as planned” but as long as there are no boundaries beyond which no source code is available, the cost shouldn’t get too high to change the code.

**Damage:**

If such a proprietary-wall exists or if the implementation turns out to be so complex that in order to make the changes, an improbable amount of areas of the code need to be considered, Wexflow itself might become infeasible to use for the endeavor and thus an alternative will have to be found / developed.

### Design might contain flaws that become only apparent after implementation

#### Description

Infusing the implementation of workflows by Wexflow with the concept of petri-nets carries with it an inherent increase of complexity. Should the resulting system become too cumbersome to use for implementation of actual projects, its acceptance by customers and other developers will suffer greatly.

**Probability:**

At the time of writing, it’s not ultimately apparent how the final product will look and what the set of all the features will contain. While measures will be taken in form of a thorough Design phase to ensure this probability to be low, there might be unknown unknowns that will become emergent only during or after development with this complex of a software

**Damage:**

Redesign would have to take place, iteratively improving the design to a point where the designed quality of the software has sufficiently been ameliorated.

### Performance might be insufficient compared to an equivalent (or even sub-par) system developed without the modular flexibility to such an extent that usage of the BIF becomes cost-ineffective.

#### Description

See 2.6.2.1

With great complexity comes great overhead. As such, it needs to be exceptionally well-defined which parts will be able to be defined via XML configuration and which parts wont (and in turn will have to be defined in coded, binary/CIL modules).

**Probability:**

<Your guess is as good as mine>

**Damage:**

As this doesn’t directly affect the functionality of the software, possible damage would be felt only after release and during the cycles of continuous improvement.

#### Consequences

Forget the promise of progress and understanding, for in the grim dark future there is only war. There is no peace amongst the stars, only an eternity of carnage and slaughter, and the laughter of thirsting gods.

#### Probability

## Prerequisites

## Acceptance