

# Report on How to Use ALF Action Language and fUML execution/debugging with Moka

Coen 6312-Model Driven Software Engineering

SAEED SHOARAYE NEJATI (40044525)

Matin Maleki (40043676)



### Table of Contents

Overview	5
Preparing Steps for Papyrus	6
Installing ALF and ALF Extension for Eclipse Oxygen PAPYRUS:	6
Installing Moka and Nebula updates:	8
Getting Started with Model Execution Import a basic sample of fUML with Moka to start with	
ALF based Model Execution ALF based behavior as part of diagram:	
Conclusion	

### Table of Figures

Figure 1- ALF installation for papyrus	6
Figure 2 - Integrated ALF Editor inside Papyrus additional Components	7
Figure 3 - Enable ALF Support for Founational UML	7
Figure 4 - Get Nebula Updates	8
Figure 5 - Moka core for execution and debugging models	8
Figure 6 - Basic sample of fUML in Moka	9
Figure 7 - Create new project with the name of ALF_Papyrus	10
Figure 8 - Import downloaded sample into the environment	10
Figure 9 - First view of the imported project	11
Figure 10 - Increment Class Diagram in Project	11
Figure 11 - Increment method in the behaviors	. 12
Figure 12 - Increment classifier behavior	12
Figure 13 - Generate factory for Increment Class Behavior	13
Figure 14 - Moka preferences for run and debug	13
Figure 15 - Run Project in debug mode	14
Figure 16 - Debug configuration and add Moka Configs	14
Figure 17 - Steps to initial configurations for Moka	15
Figure 18 – set element for execution of Model	15
Figure 19 - Debugging environment in the eclipse	16
Figure 20 - Debugging a Model in eclipse	16
Figure 21 - Possibility to add breakpoint and see the simulation of the execution for model	17
Figure 22 - Adding Breakpoint in each step to check the object instance values of the class	17
Figure 23 - read values of the attributes inside the class in the variables window	. 18

Figure 24 - All executed with models and nothing textual for actions	18
Figure 25 - Add new operation to the class from palette	19
Figure 26 - Add activity to the existing class	19
Figure 27 – Multiply Method as a behavior added to the class	20
Figure 28 - ALF editor inside the eclipse to create behavior	20
Figure 29 - Compile and Generate behaviors for multiplyMethod Behavior	22
Figure 30 - Drag MultiplyMethod from Model Explorer inside the increment Method as CallBehaviorAction	23
Figure 31 - As you can see the result is our method with two input and one output as we wrote ALF	
Figure 32 - Add value Specification Action to set the inputs of the MultiplyMethod	24
Figure 33 - Add Literal Integer value equal 2 as second input for Behavior	24
Figure 34 - set the value equal 2 to provide the right input for your method	25
Figure 35 - Object Flow edge to connect result of value two to input y	25
Figure 36 - Object flow from the result of the callMultiplyMethod to the Value of set Counter .	26
Figure 37 - Final Modified Diagram with the added ALF part	26
Figure 38 - Add Breakpoints to Debug and see the values inside the model	27
Figure 39 - In the first run value of the counter is zero	27
Figure 40 - Simply hit resume to continue running the diagram	28
Figure 41 - Second run the value will show the result of two	28
Figure 42 - in the third run value will show the 6 as result	29
Figure 43 - Final representation of two different behaviors in the model	30
Figure 44 - Increment method based on the fUML without textual actions	31
Figure 45 - Added ALF based action for Multiply Method	31

### Overview

In this report, we tried to implement part of the executable model in fUML with Action Language **(ALF)**. This report is based on **Eclipse Oxygen** and **Papyrus Modeling environment** and the process of installation and execution of ALF action language in this environment.

First, we introduce how to **Integrate ALF editor** in this environment and then we try to add **Moka** for the purposes of debugging and execution models. At the end, we provide **a new sample** based on the fUML sample of the Moka for execution of the models to show how we can execute ALF based behaviors inside the modeling environment.

Since ALF is supported from the OMG standardization as an action language we started to find out supported tools and modeling environments to use. But the main problem was out dated samples and tutorials to use this important issue in modeling environments. This report is the documentation of how to use ALF language with the help of debugging fUML models in the papyrus environment.

We start this tutorial based on what is existed for running fUML in the Moka without textual action language and the we add our ALF based actions to the diagram.

### Preparing Steps for Papyrus

#### Installing ALF Extension for Eclipse Oxygen:

- 1. Click on "help", "install new software" then add the link below and install all ALF plug-ins on your Eclipse. The address for the update the sources.
  - web address: Oxygen http://download.eclipse.org/modeling/mdt/papyrus/updates/releases/oxygen/

• • •	Install	
Available Software Check the items that you wish to install.		(s)=
Work with: <sup>(8)</sup> Oxygen - http://download.eclipse.org/modeling/mdt/papyrus/upp	ates/releases/oxygen/	Add Manage
ALF		8
Name  Papyrus ALF  Papyrus ALF  Papyrus ALF  Papyrus ALF  Papyrus ALF Developer Resources  Papyrus ALF Developer Resources  Papyrus ALF Developer Resources  Papyrus ALF Developer Resources  Papyrus ALF Developer Resources	Version 3.0.0.201712060842 3.0.0.201709130748 3.0.0.201709140736 3.0.0.201710160842 3.0.0.201709130748 3.0.0.201706140736	
Select All Deselect All Details		
Show only the latest versions of available software Group items by category Show only software applicable to target environment Contact all update sites during install to find required software	Ide items that are already installed What is <u>already installed</u> ?	
0		Back Next > Cancel Finish

Figure 1- ALF installation for papyrus

2. From top toolbar select "help" then "Eclipse MarketPlace", search ALF and install "Integrated ALF Editor".

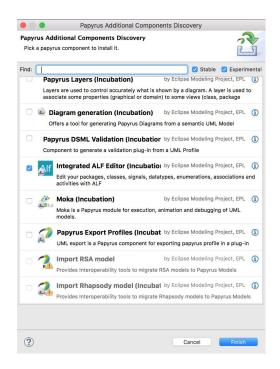


Figure 2 - Integrated ALF Editor inside Papyrus additional Components

3. From top toolbar on "preference" search ALF and active all Supports for it.

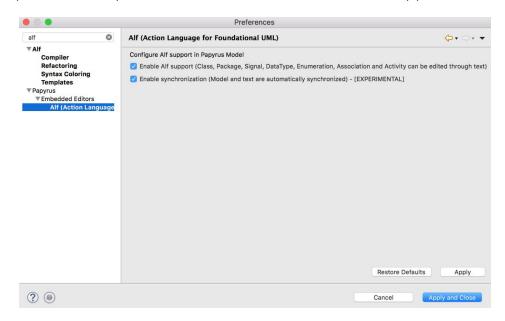


Figure 3 - Enable ALF Support for Founational UML

Installing Moka and Nebula updates:

1. On "help", "install new software" add this link below and install all Nebula.

• • •	Install	
Available Software Check the items that you wish to install.		
Work with:  Nebula - http://download.eclipse.org/nebula/releases	/1.0.0	Add Manage
Name Name Name Name Name Name Name Name	Version 1.0.0.201605312033 1.0.0.201605312033 1.0.0.201605312033 1.0.4.201605312033	
Show only the latest versions of available software Group items by category Group items by category Group items by category Crownly software applicable to target environment Crownly software Crown	Hide Items that are already installed What is <u>already installed</u> ?	
0	< Back	Next > Cancel Finish

#### Figure 4 - Get Nebula Updates

2. On "help", "install new software" add this link below and install all Moka cores.

• •	Install	
Available Software		
Check the items that you wish to install.		
Work with: Moka - http://download.eclipse.org/modeling/mdt/papyrus/comp	onents/moka/oxygen	Add Manage
type filter text		
Name	Version	
Moka Core (Incubation)	3.1.0.201710171318	
Moka Core (Incubation)     Roka For Cosimulation (Incubation)	2.0.100.201705162304 3.0.0.201710171318	
Moka For Cosimulation (Incubation)	3.0.0.201710171318	
Moka For Cosimulation (Incubation)	2.1.0.201705162304	
Select All Deselect All Details		
Show only the latest versions of available software	Hide items that are already installed	
Group items by category	What is already installed?	
Show only software applicable to target environment		
Contact all update sites during install to find required software		
0	< Back	Next > Cancel Finish

Figure 5 - Moka core for execution and debugging models

3. download the basic sample of project use the link below and save it on your system.
- web address: https://wiki.eclipse.org/File:BasicActiveObjectExample.zip

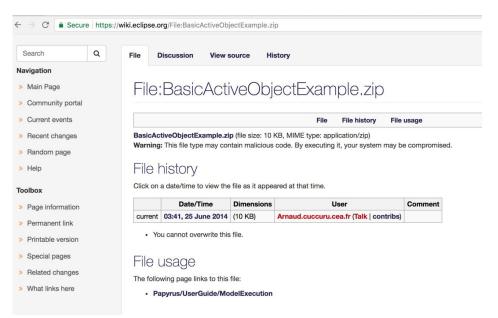


Figure 6 - Basic sample of fUML in Moka

# Getting Started with Model Execution

#### Import a basic sample of fUML with Moka to start with

1. we will make a new papyrus project then import the sample project (that we download it before). We called the project "ALF\_PAPYRUS".

	New Papyrus Project		New Papyrus Project
Select Architecture Context Select the architecture context(s)	and viewpoints to apply to the Papyrus model	Choose your project	path and the model name
Architecture Contexts:	Executable UML	Project name: ALF	
Vectoriable one			ion hatin/eclipse-workspace/ALF_Papyrus Browse
		Model file name: ALF_Papyrus	
Architecture Viewpoints:			
Ill Structure			
?	< Back Next > Canc	el Finish ?	< Back Next > Cancel Finish

Figure 7 - Create new project with the name of ALF\_Papyrus

File system Import resources from the local file system.  From directory: /Users/matin/Downloads/BasicActiveObjectExample  Form directory: /Users/matin/Downloads/BasicActiveObjectExample.  Files Types Select All Deselect All  Filter Types Select All Deselect All  Options Options Options Create top-level folder Advanced >>		Import	
Into folder:       ALF_Papyrus         Options       Overwrite existing resources without warning         Create top-level folder			
Overwrite existing resources without warning Create top-level folder	BasicActiveObjectExample      Filter Types     Select All     Desele      Into folder: ALF_Papyrus	<ul> <li>BasicActiveObjectExample.di</li> <li>BasicActiveObjectExample.notation</li> <li>BasicActiveObjectExample.uml</li> </ul>	
Cancel Finish	<ul> <li>Overwrite existing resources without warning</li> <li>Create top-level folder</li> </ul>	< Back Next > Cancel	Finish

Figure 8 - Import downloaded sample into the environment

• • •	eclipse-workspace - ALF_Papyrus/BasicActiveObjectExample.di - Eclipse
📑 • 🗟 🕼 🗣 📏 🧭 🛅 • 🛄 •	隨 • 원 →• 團 凝• 애• 남• ਛ• 양• ⊕• •• 팀 四 원• 100%
	Guick Access 🔡 🖑 🔁
🍐 Project Explorer 😒 📃 🗖	🔿 BasicActiveObjectExample.dl 🕴 🗖
C ⇒ Q ∈ S > ▼ ALF_Papyrus ↑ BasicActiveObjectExample	<u>م</u>
Model Explorer 🗱 📟 🗖	
Image: Section 2         Image: Section 2         Image: Section 2         Image: Section 2           Image: Section 2 <t< td=""><td>The backdarwood gent barrier  The searce control ( The searce control (</td></t<>	The backdarwood gent barrier  The searce control (
E Outline 🛛 🕞 🗄 💕 🔻 🗖	
	Increment ALP ARRESPACE BasicActiveObjectExample;
	ALF Comments Profile © Commit

2. when you open the project, you should have all these diagrams attached to it.



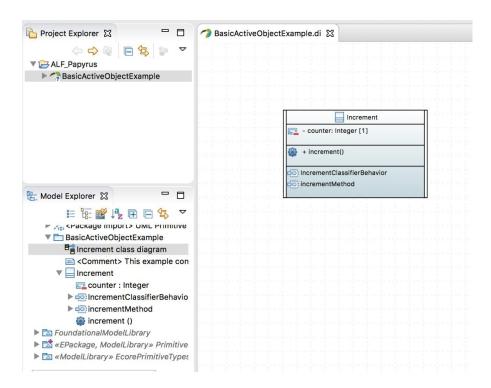


Figure 10 - Increment Class Diagram in Project

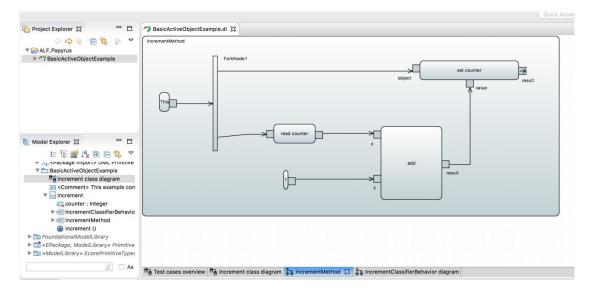
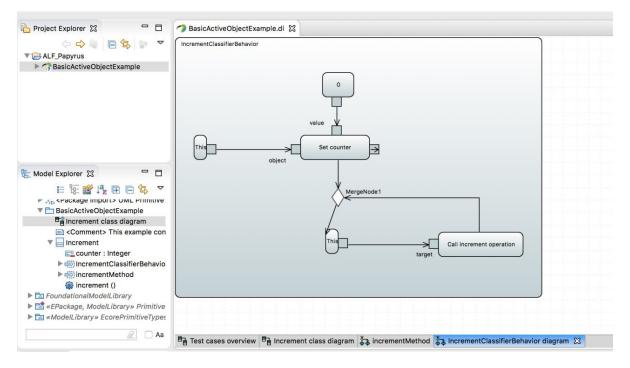


Figure 11 - Increment method in the behaviors





3. The behaviors associated with this class (i.e., IncrementClassifierBehavior, which is the classifier behavior, and incrementMethod, are the implementations of operation increment) are specified by activity diagrams. Corresponding activities are executable, according to the semantics given in OMG standards fUML and PSCS. Anyway, in fUML and PSCS, the execution of a model usually starts by executing a kind of "main" activity, which is responsible for instantiating objects, and stimulate them if needed (through signals or operation calls). Moka provides some facilities to generate this kind of activities. Just right click on class Increment, then go to Moka - Modeling Utils - Generate Factory.

Project Explorer		*BasicActiveObjectE					
🔻 📂 ALF_Papyrus	veObjectExample						
				Increment	]		
	Moka		- cour	Modeling Utils		Generate Factory	Generate Factor
	Navigate		Þ	Export CSV		Generate Constructor Using Fields	
Model Explore	New Child New Relationship		* * *	<ul> <li>Run</li> <li>Debug</li> <li>Breakpoints</li> </ul>	•		
En Test ca	🗙 Delete		∞				
► Packa ► Packa ► Packa ► Packa ■ BasicAc ■ Incr	<b>⇔</b> Undo SRedo	Û	<b>%Z</b>				
Co < <ti>&lt;<ti>Incr &lt;&lt;&lt;&lt;</ti></ti>	<ul> <li>✓ Cut</li> <li>i Copy</li> <li>i Paste</li> </ul>		жх жс жv				
C C	Profiles Model refactor		•				

Figure 13 - Generate factory for Increment Class Behavior

4. You should also check the execution engine from "preferences".

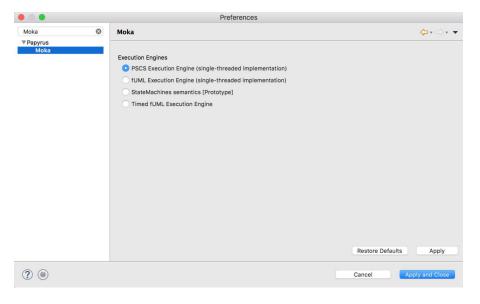


Figure 14 - Moka preferences for run and debug

5. After checking the engine, hit "**Debug Configurations**..." and add new configuration as below.

ÌÌÀv º? ▾ ¦¦ ▾ ≣ ▾ 𝔅 ▾ ↔		:	☆ • ○ • • • • ○ (no launch his)	- 14-1 - 15
ctExample.di 🔀		Debug As Debug Configurations Organize Favorites		
Increment	<b>_</b>			
- counter: Integer [1]     + increment()				
incrementClassifierBehavior				

Figure 15 - Run Project in debug mode

reate, manage, and run cor		
	figurations	Ť
type filter text      Celipse Application     Java Applet     Java Applet     Java Applet     Java Applet     Java Applet     Java Applet     Jourd Durit     Jur JUnit     Jur JUnit     Jur JUnit     Jur JUnit     Generate Group      Moka Banch     OCL Express     OCL Express     OUL     Remote Java     Soci Framey     Jur Task Context Test	plicate etc.	

Figure 16 - Debug configuration and add Moka Configs

	Debug Configurations		Please select an fUML model
Create, manage, and run confi	gurations	Ť.	Tree Flat
🗅 🖹 🗶 🖻 🏇	Name: New_configuration		🖉 🗆 Aa 🖻 🖻
type filter text	Moka Main Common		T 🔁 ALF_Papyrus
Eclipse Application Java Applet	UML Model		BasicActiveObjectExample.uml
Juava Application		Browse	
Ju JUnit Plug-in Test ☐ Launch Group ✓ Moka launch configuration ✓ New_configuration Ø OCL Expression	Element to be executed		
⊕ 05Gi Framework	Execution Engine (if no selection, the default engine is used)		
			UML (*.uml)
Filter matched 13 of 13 items	Revert	Apply	
?	Close	Debug	Cancel OK

Figure 17 - Steps to initial configurations for Moka

	Debug Configurations	
Create, manage, and run config	urations	Ť.
Y       Image: Second Se	Name: New_configuration  Moka Main  Common  UML Model  platform:/resource/ALF_Papyrus/BasicActiveObjectExample.uml  Element to be executed  [Activity] model::BasicActiveObjectExample::Increment:Increment_Factory [Activity] model::BasicActiveObjectExample::Increment:Increment_Factory [Activity] model::BasicActiveObjectExample::Increment:Increment_Factory [Activity] model::BasicActiveObjectExample::Increment:IncrementMethod [Class] model::BasicActiveObjectExample::Increment	Browse
Filter matched 13 of 13 items	Revert	Apply
?	Close	Debug

Figure 18 – set element for execution of Model

6. Start debugging by click on Debug icon on up-right side of screen.



Figure 19 - Debugging environment in the eclipse

7. You will see the debug start when the action diagram becomes green

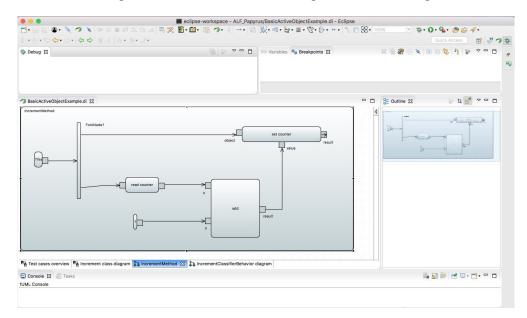


Figure 20 - Debugging a Model in eclipse

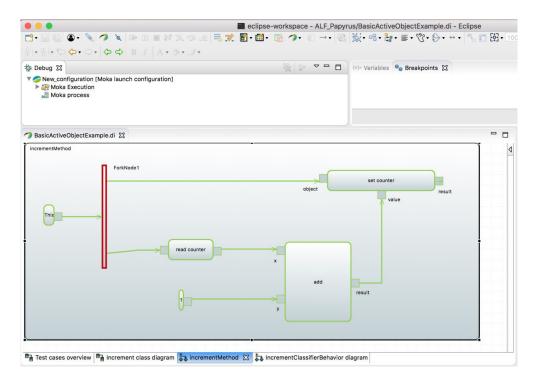


Figure 21 - Possibility to add breakpoint and see the simulation of the execution for model

- 🤿 BasicActiveObjectExample.di 🔀 🗖 🗖 📑 Outline 🖾 incrementMethod 4 Navigate . ForkNode1 File . Load resource... Enable write This X Delete Selected Element 🖉 Delete From diagram Format read counter . Filters . x J Validation . 7 Edit . 1 Edit OutputPin . у Modeling Utils Moka ۲ Profiles 🔘 Run Show EClass information 🎋 Debug Breakpo Toggle breakpoint activation 🖥 Test cases overview 📴 Increment class diagram 🗦 incrementMethod 🛛 🛟 IncrementClassifierBehavior diag Show Properties View 🔄 Console 🔀 🧔 Tasks R . . **Provide States** Show References View fUML Console Show Documentation View Properties . Remove from Context \\\\C\0 \\\$ ↓ Moka Execution Eng
- 8. By Adding breakpoints in different steps, you can monitor your debugging completely.

Figure 22 - Adding Breakpoint in each step to check the object instance values of the class

9. As you see in the pictures, counter will be increase each time the cycle finishes.

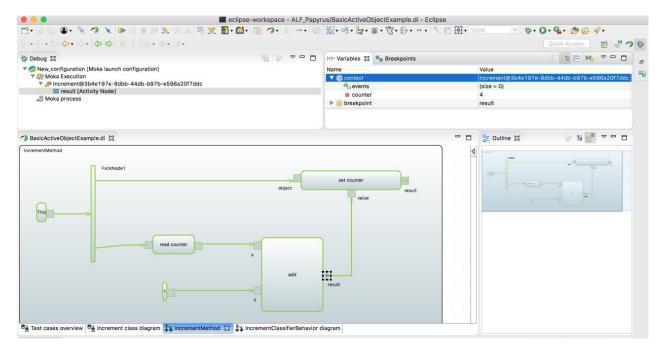


Figure 23 - read values of the attributes inside the class in the variables window

10. As you see in this photo, there is no command in ALF part, because here we only use Moka to execute this project. Next step we will add another method and write the ALF codes in it.

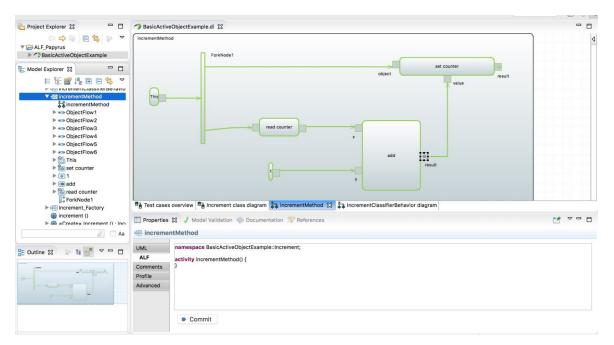


Figure 24 - All executed with models and nothing textual for actions

# ALF based Model Execution

#### ALF based behavior as part of diagram:

1. In the Class diagram add an operation and call it Multiply

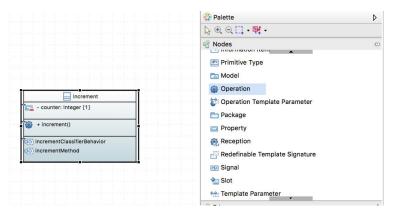


Figure 25 - Add new operation to the class from palette

2. In Model Explorer, right click on Increment and add new activity behavior call it "MultiplyMethod"

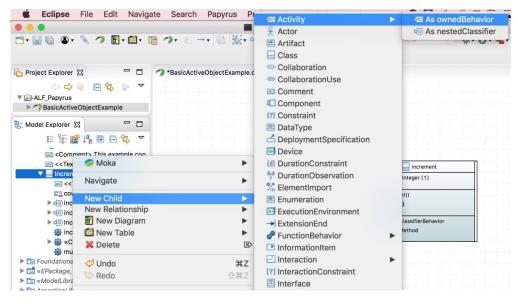


Figure 26 - Add activity to the existing class

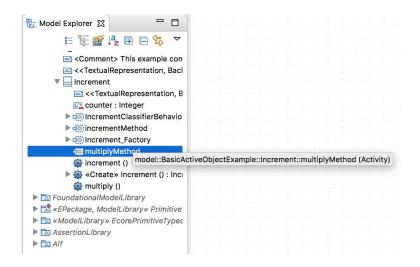


Figure 27 – Multiply Method as a behavior added to the class

3. As you see, you will have this class diagram and model Explorer. If you click on MultiplyMethod, it is ready to add your behavior with ALF language.

Model Explorer     S       □     □       □     □       □     <		overview 📴 Increment class diagram 😫 🚑	incrementMethod	Behavior diagram	Products Primitive Type Primitive T			*
Alf	Properties	🔀 🥑 Model Validation 📀 Documentation 🖇	P References			1	7 0	
	UML ALF Comments Profile Advanced	namespace BasicActiveObjectExample::Increm activity multiplyMethod() { }	ient;					
		Commit						

Figure 28 - ALF editor inside the eclipse to create behavior

4. Write ALF codes same as below in "ALF properties" for "MultiplyMethod"

}

Sample ALF method for to input x and y and prepare the result by returning x multiply by y.

Sample Code: namespace BasicActiveObjectExample::Increment; activity multiplyMethod(in x:Integer, in y:Integer):Integer { return x\*y; Increment - counter: Integer [1] + increment() 🛞 + multiply() IncrementClassifierBehavior incrementMethod 📑 Test cases overview 📑 Increment class diagram 🕱 💺 incrementMethod 💲 IncrementClassifierBehavior diagram 🔲 Properties 🕱 🤳 Model Validation 🧇 Documentation 💖 References i multiplyMethod UML namespace BasicActiveObjectExample::Increment; ALF activity multiplyMethod(in x:Integer, in y:Integer):Integer { return x\*y; Comments Profile Advanced Commit

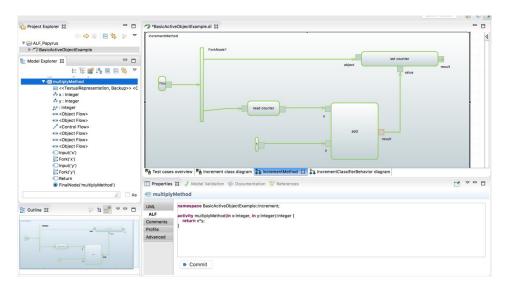
5. You will have these codes under the "multiplymethod" same as this. Hit Commit. With the commit button inside the ALF editor these commands will be executed and compiled as behavioral parts which you can see the next picture. The beauty of this process is all the generated parts can be used as an action inside another diagram.

We have added the ALF codes but still we do not have the multiplymethod in our Action diagram. It is possible to Drag and drop the "MultiplyMethod" from Model Explorer to increment class diagram to add it even inside to diagram.

Model Explorer 😵			
Contractions of the second	thod')	ases overview 타 Increment class diagram 없 ties 없 J Model Validation 🍥 Documentat	counter: Integer (1)     forment()     forment()     forment()     formentClassifierBehavior     formentMethod     formentMethod
E Outline 23	UML ALF Comme Profile Advance	}	

Figure 29 - Compile and Generate behaviors for multiplyMethod Behavior

6. Now we need to execute our ALF part in the previous fUML we tested and debugged in the previous section.



7. Choose "Activity as a CallBehaviorAction" second item from the menu.

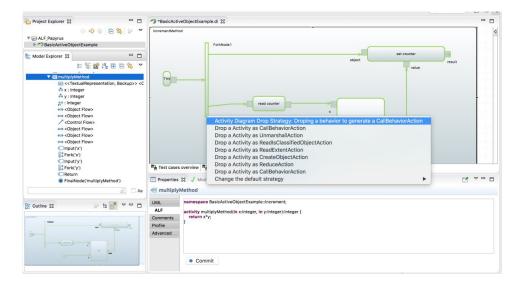


Figure 30 - Drag MultiplyMethod from Model Explorer inside the increment Method as CallBehaviorAction

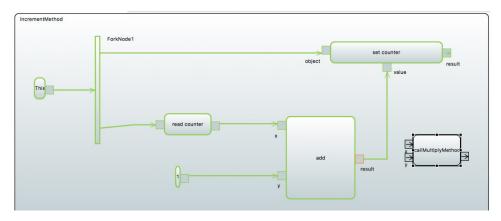


Figure 31 - As you can see the result is our method with two input and one output as we wrote in ALF

8. To test the "MultiplyMethod" we will add a constant value (equal to 2) and get the output of incrementing. So, each time we will multiply the increment output.

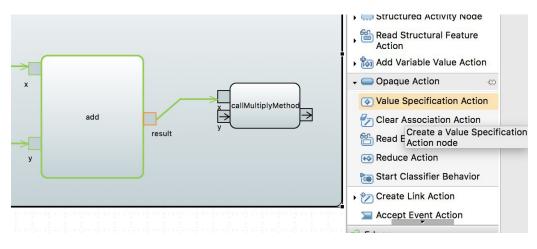


Figure 32 - Add value Specification Action to set the inputs of the MultiplyMethod

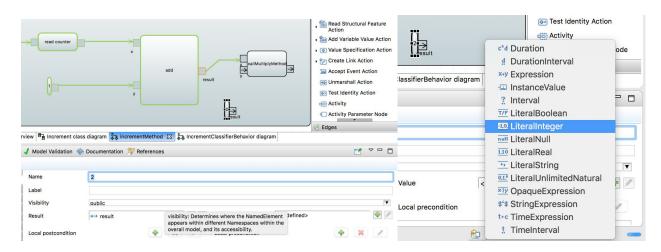


Figure 33 - Add Literal Integer value equal 2 as second input for Behavior

	Create a new LiteralInteger	
Name	2	
Label		
Value	2	
UML Comments		
?		Cancel OK



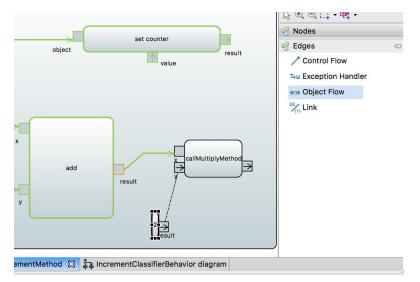


Figure 35 - Object Flow edge to connect result of value two to input y

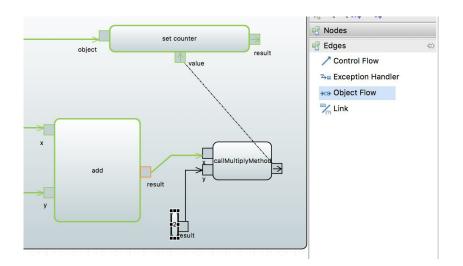


Figure 36 - Object flow from the result of the callMultiplyMethod to the Value of set Counter

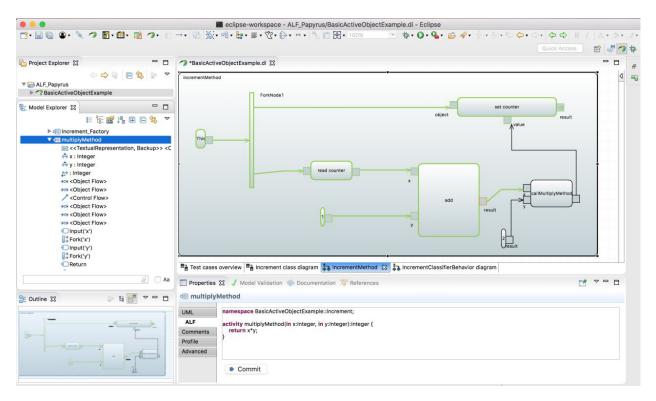


Figure 37 - Final Modified Diagram with the added ALF part

eclipse-workspace - ALF_Papy	rus/BasicActiveObjectExample.di - Eclipse
🖆 • 📓 🕲 • 🌂 🥠 🔌 🕨 💷 = M 🌫 🖘 🗷 🧠 📅 • 🛍 • 🏙 • 🦓 🔶 🕂 🧐	◎※・・・ 💱 • 章 • 🗇 • 🐂 🖏 🛍 • 100% 🛛 📓 🕸 • 🔘 • 💁 🖉 •
	Quick Access 😰 💕 🧭
🗱 Debug 🕅 🛞 😒 🗢 🗖	🗱 🗱 🚱 🗣 🗛 🖓 😨 😒 🗮 🖻 🧐 🗸 😨
	BssicActiveObjectExample.uml
	No details to display for the current selection.
→ *BasicActiveObjectExample.dl X	B: Outline 🔀 🔷 🗄 📑 🗢 -
Increment/Lethod	set counter result result gagement

Figure 38 - Add Breakpoints to Debug and see the values inside the model

9. When you run the debug, counter is zero and each time it will first increment by 1 and multiply by 2. First step counter is zero.

pebug 🕱 🛛 🖓 🐨 🗖 🗖	(x)= Variables 🔀 💁 Breakpoints	1 😽 🖻 🙌 🔻 🗖 E
New_configuration [Moka launch configuration]     Wex Execution     Moka Execution     wenteente@cc00b/77-1aeb-4b8b-bdaf-0b5c2eda1970     wenteente@cc00b/77-1aeb-4b8b-bdaf-0b5c2eda1970     wetaitee[Activity]Node]     Moka process	Name ▼	Value Increment@9c00bf77-1aeb-4b8b-bdaf-0b5c2eda197 (size = 0) O value
BasicActiveObjectExample.di 23	set counter result	E Outline S
read counter x add	result	

Figure 39 - In the first run value of the counter is zero

2				14	2	P	-R
\$	R	esu	me (	F8)	191 -		•

#### Figure 40 - Simply hit resume to continue running the diagram

10. Second step 0 will increment by1 then multiply by 2. counter: (0+1) \*2=2

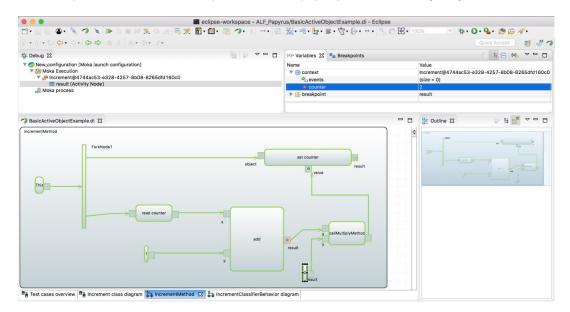


Figure 41 - Second run the value will show the result of two

11. Third step. Counter: (2+1) \* 2=6

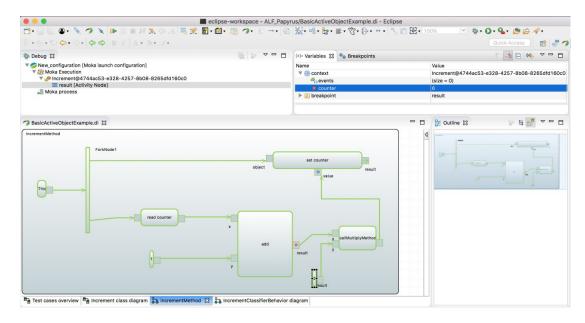


Figure 42 - in the third run value will show the 6 as result

## Conclusion

In textual representation of the model, we can see, we have the multiply method as public which is written by ALF and there is no difference in the execution between ALF parts and fUML with Moka.

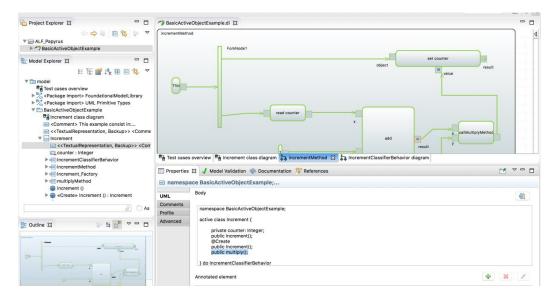


Figure 43 - Final representation of two different behaviors in the model

Next two pictures clearly show that "incrementMethod" which is in Moka has not the ALF commands and "MultiplyMethod" which is written in ALF has the commands and they are working together properly.

• "incrementMethod" which is implemented with fUML and actions inside model.

Model Explorer IS       □         □       □    <	ealMultiplyMethod Test cases overview B Test cases overview B Increment class diagram B Increment ClassifierBehavior diagram
▼ do) multiplyMethod	ment::incrementMethod (Activity) <sup>dation</sup> 🛞 Documentation 💖 References 📑 💌 🗖 🗖
Image: Second secon	UML namespace BasicActiveObjectExample::Increment;
🖉 🗆 Aa	ALF activity incrementMethod() { Comments )
문 Outline 없 😥 1월 🔐 🍷 🗖	Profile Advanced
	Commit

Figure 44 - Increment method based on the fUML without textual actions

• "MultiplyMethod" which is added with ALF action language and compiled as a part of the model execution in the behavior.

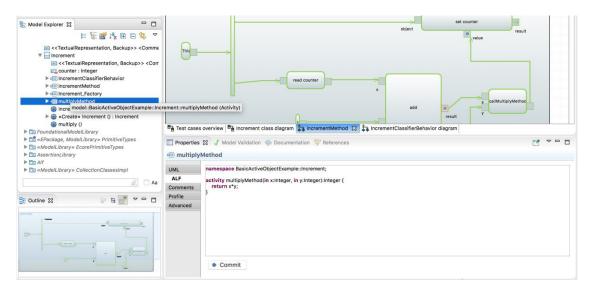


Figure 45 - Added ALF based action for Multiply Method