

## BOOK IB - INSULIN PUMP (MODULE B)



# Feature Function Package Structures with the Executable MBSE Profile

17 MAY 2024

OPEN LAB TUTORIAL

RHAPSODY V9.0.2

EXECUTABLE MBSE PROFILE V4.3.D.RELEASE

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

[www.mbsetraining.com](http://www.mbsetraining.com)

[www.executablembse.com](http://www.executablembse.com)

[www.mbsetraining.com](http://www.mbsetraining.com)

# Trademarks and the rest

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- IBM, Rhapsody, DOORS, Rational are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.
- Microsoft, Windows, PowerPoint, Excel, MS-DOS are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
- SysML, OMG Systems Modeling Language, OMG, Object Management Group, Unified Modeling Language either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.
- Other company, product, or service names may be trademarks or service marks of others.
- Images used under license from 123RF.com: gunnar3000@123RF.com, serknor@123RF.com, cocoo@123RF.com, lenanet@123RF.com

# Table of Contents

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

Lab #	Lab Title	Page #
Lab IB1	Auto-create use case package structure (Insulin Pump)	p5
Lab IB2	Create a feature function package	p21
Lab IB3	Modeling function chains using an internal block diagram	p35
A	Installing the SysML Helper	p53

# Introduction

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- This lab book is one of two workshop lab books that build the insulin pump system project from scratch using the open-source Executable MBSE profile written by MBSE Training and Consulting Ltd
- It uses version 4.3.d.Release of the SysMLHelper:  
<http://www.executablembse.com/>
- The SysMLHelper is licensed under GPL 3.0 which is a copyleft license, meaning that any copy or modification of the original code must also be released under the GPL v3
- The SysMLHelper is entirely self-contained Java, a conscious decision was made not to use third-party or open-source libraries/.jars other than a standard Java runtime (incl. Swing for GUI) and the Rhapsody Java API .jar
- It may take to 1-2 hours to complete this lab book, assuming everything is correctly installed

# AUTO-CREATE USE CASE PACKAGE STRUCTURE **LAB IB1**

*"INSULIN PUMP" CASE STUDY*

## LAB IB1

*"hear and I forget. I see and I remember. I do  
and I understand" (Confucius 551 BC - 479 BC)*



# Introduction to the lab

- This lab is the same as lab IA4 in previous lab book. If you have done it already then you can skip to next lab.
- There are typically 3 options to perform functional modeling with SysML:
  - Use of operations, owned by blocks (in Rhapsody there are some executable methods that work well with sequence and state machine diagrams)
  - Use of activities and activity diagrams (a classic SysML methods)
  - Use of function blocks (a novel approach possible by extending SysML)
- In this lab we show the function blocks method that makes use of block definition diagrams and internal block diagrams. The following element types are introduced.
  - **Function Block:** A type of block that represents behavioral feature of a subsystem
  - **Feature Block:** A type of function block that represents a behavioral feature of a system, exposed on the boundary of the system (user perceivable)
  - **Feature Function package:** A container for feature and function blocks

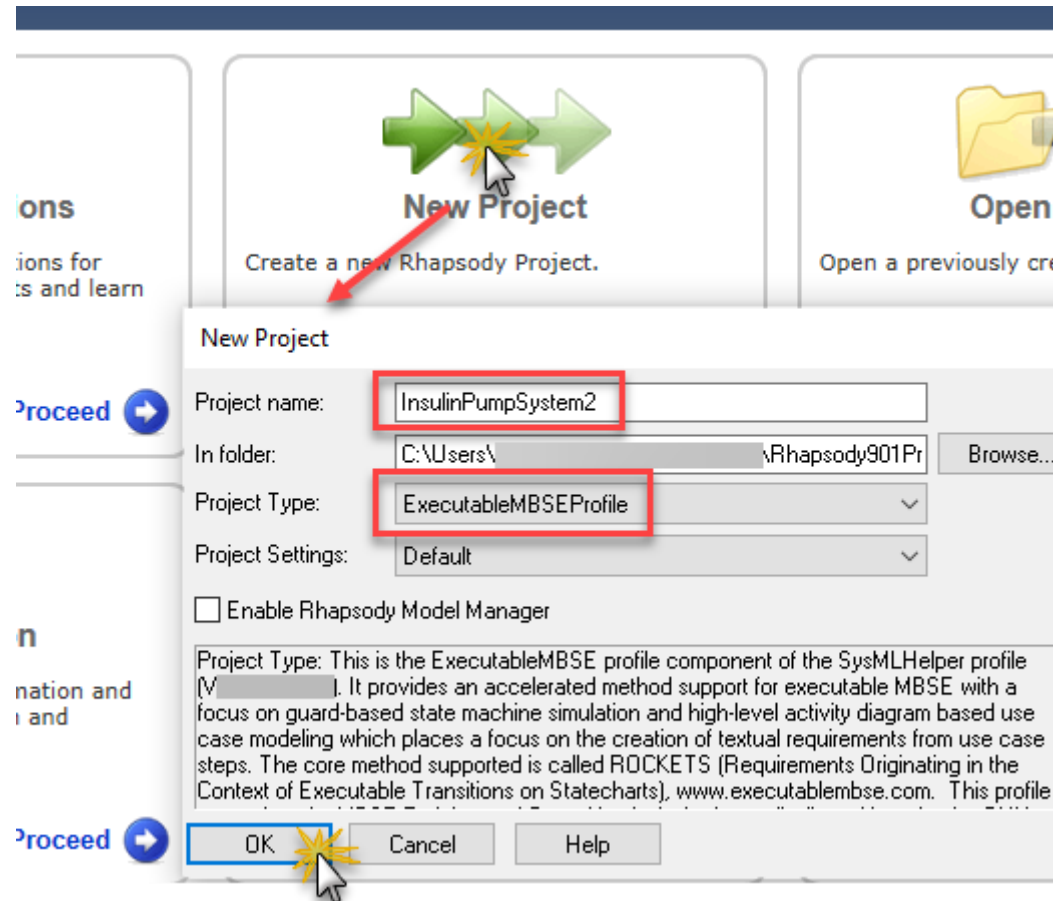
# Benefits of using Function Blocks

- Use of Feature and Function Blocks is a novel approach but does solve some intrinsic issues with both SysML and Rhapsody. It:
  - Enables same diagrams to be used for both structural and functional decomposition, i.e., collapses number of notations to two SysML diagrams, bdd and ibd
  - Enables use of composition to perform allocation, making it easy to allocate same function to different elements of definition
  - The complexity of keeping action pins and activity parameters in sync is considered less work with ports. By using blocks, we can make use of ports for defining interfaces required, provided, or exchanges in/out
  - As blocks are central to SysML, the usability features/display properties of blocks/parts are more powerful than the matching activity diagram counterpart in Rhapsody.

# Create a New Project

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

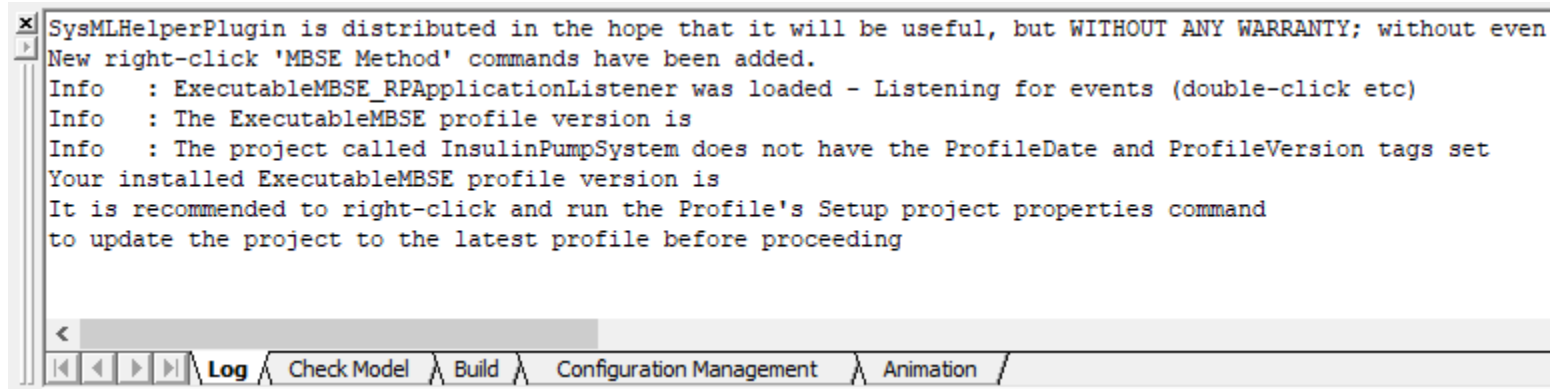
- Create a **New Project** called **InsulinPumpSystem2** in a Read/Write folder using the **ExecutableMBSEProfile** as the **Project Type**





# View the Log window

- View the **Log** window & Ensure the profile loaded



```
SysMLHelperPlugin is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even
New right-click 'MBSE Method' commands have been added.
Info : ExecutableMBSE_RPApplicationListener was loaded - Listening for events (double-click etc)
Info : The ExecutableMBSE profile version is
Info : The project called InsulinPumpSystem does not have the ProfileDate and ProfileVersion tags set
Your installed ExecutableMBSE profile version is
It is recommended to right-click and run the Profile's Setup project properties command
to update the project to the latest profile before proceeding
```

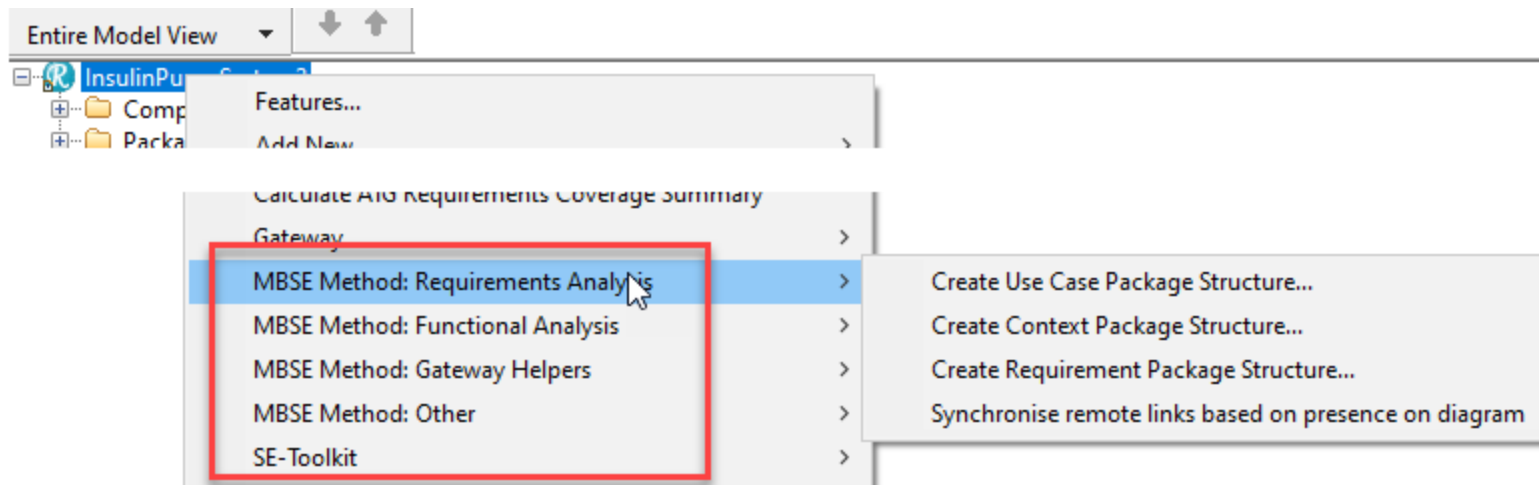


The ExecutableMBSE profile includes a Java plugin that adds new menus to assist with method automation. The Log window contains an indication that this plugin has loaded plus which version it is. If you don't see this, then it's probable your \$OMROOT/Profiles/SysMLProfile folder is not in the correct location! This must be fixed before proceeding

# View the project right-click menu

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

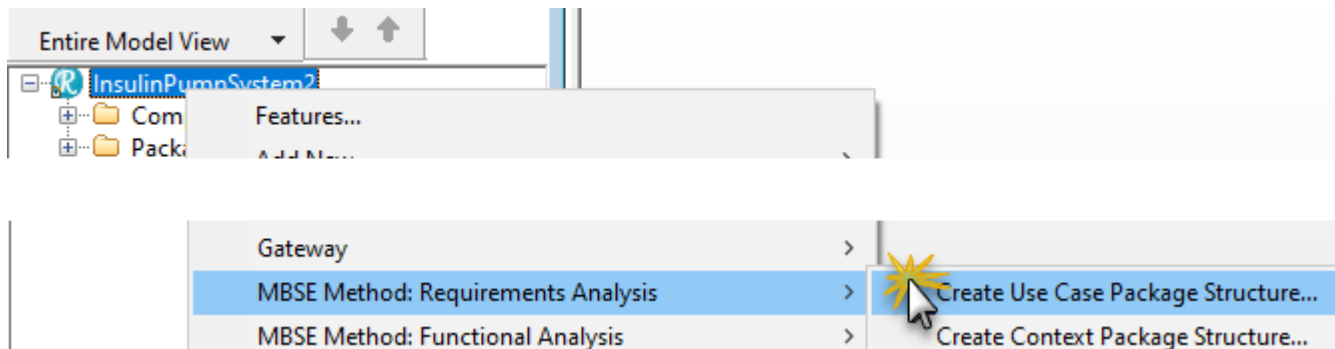
- Right-click the project and view the context menu. Down the bottom of this are menus added by the Executable MBSE profile hep file that begin with MBSE Method: <Group Name>



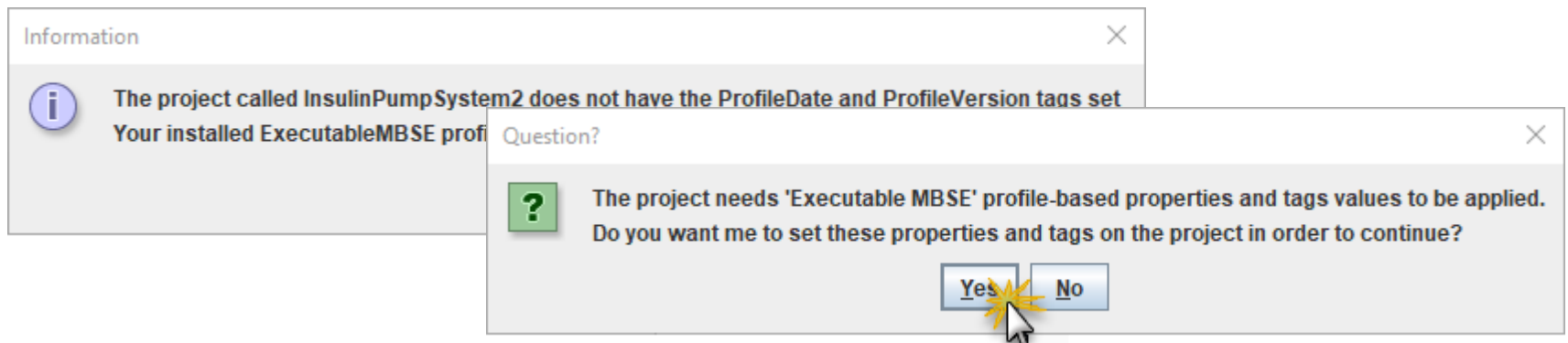
# Create Use Case Package Structure...

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- The most common method to start a project with is **MBSE Method: Requirements Analysis > Create Use Case Package Structure...**
- Try this now



- Click **OK** to proceed and **Yes** to setup the project properties



# View the dialog

- A dialog pops up that has been provided by the profile

**Create use case package structure**

Choose a unique name:  (package post-fixed with \_UseCasePkg will created under the project)

This helper will create a package hierarchy for simple activity-based use case analysis underneath the project. It creates a nested package structure and use case diagram, imports the appropriate profiles if not present, and sets default display and other options to appropriate values for this using Rhapsody profile and property settings.

Create new shared actor package using default actor names	<input type="text" value="InsulinPumpSystem2_ActorPkg"/>
Create new «Stakeholder Requirement» requirements package under project	<input type="text" value="FeatureA_RequirementPkg"/>
Skip creation of a context package	<input type="text" value="&lt;None&gt;"/>
Skip creation of a shared external signals package	<input type="text" value="&lt;None&gt;"/>

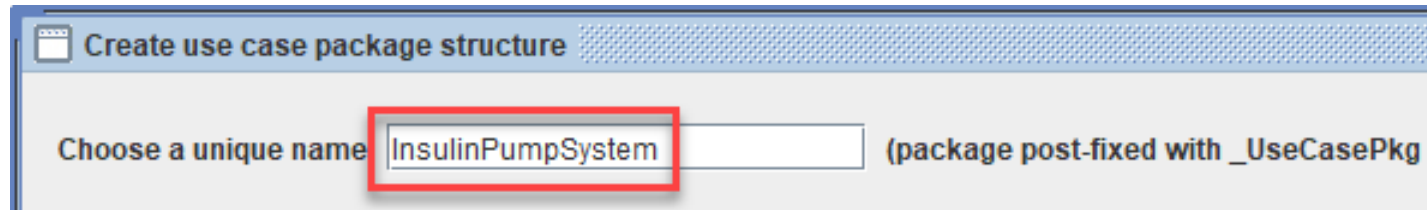


The options here are setup to create a use case package, requirement package and actor package at the same time

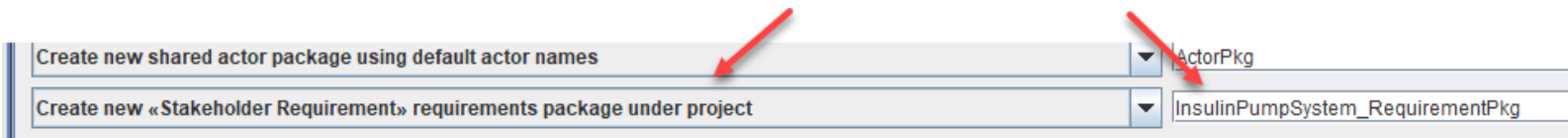
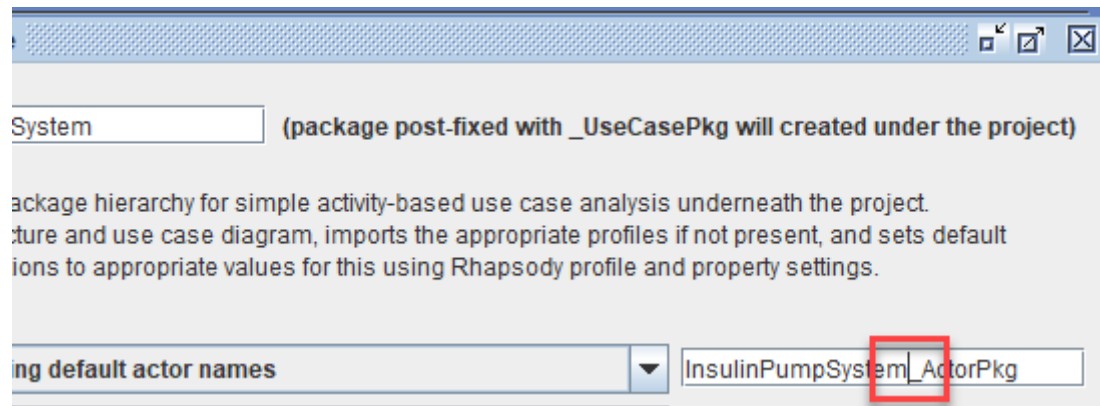
# Choose the unique name InsulinPumpSystem

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Choose the unique name **InsulinPumpSystem**



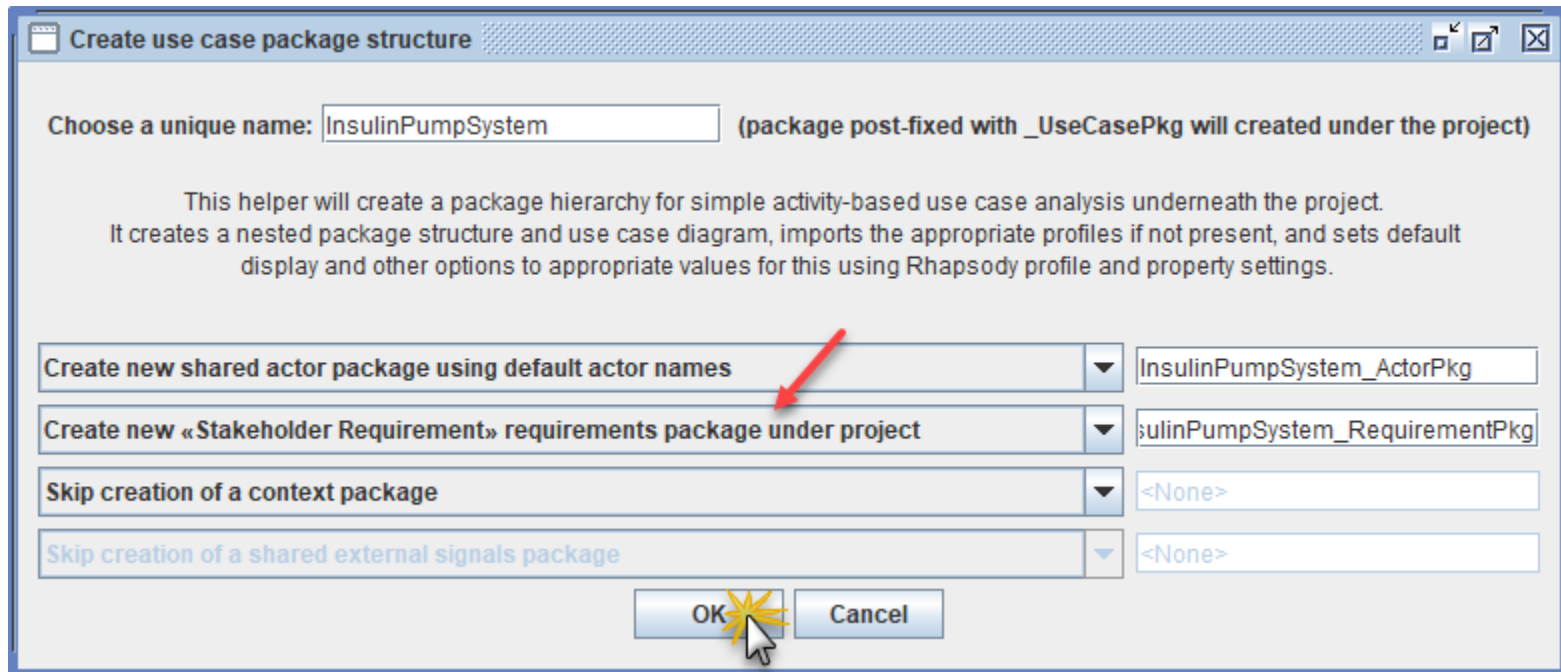
- Remove the 2 from the Actor package name:



# Check settings and click OK

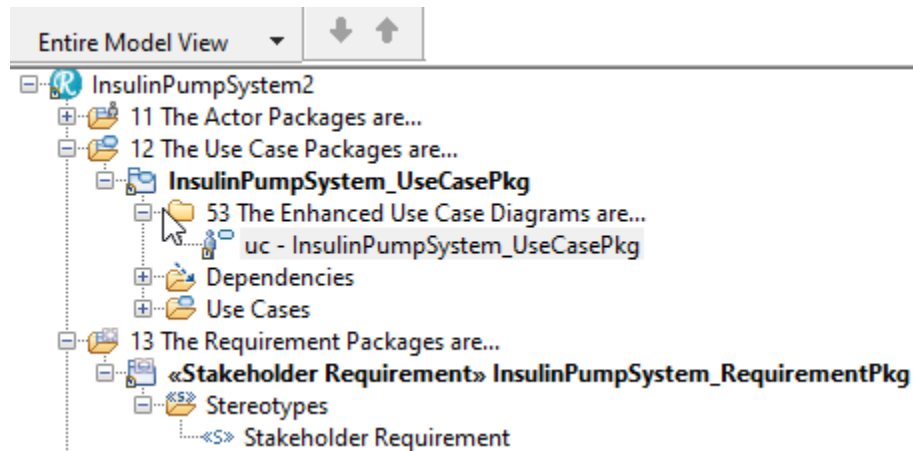
Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- The option to create a stereotyped requirements package has been selected by default. Check the settings below
- Click **OK** to create the model

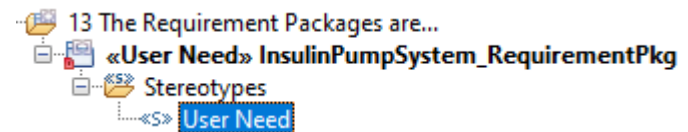
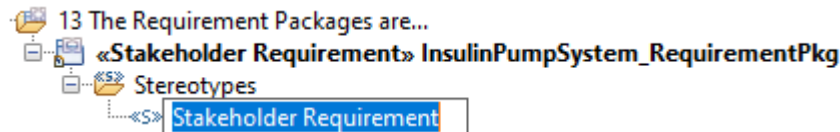


# View the created structure

- The packages are automatically created including the dependency from the use case package to the requirements package

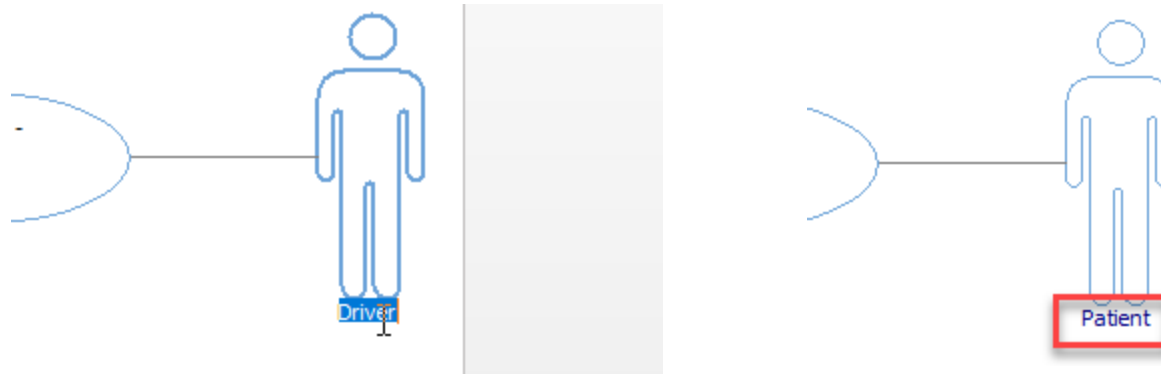


- Click the **Stakeholder Requirement** stereotype and rename it to **User Need**

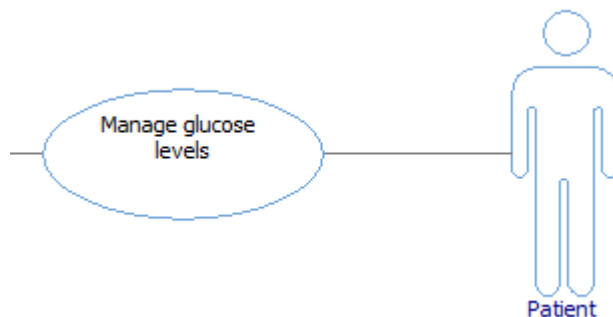


# Rename the Driver actor to Patient

- Double-click the **Driver** actor's name and change it to **Patient**



- Single-click the text to rename the use case to: **Manage glucose levels**



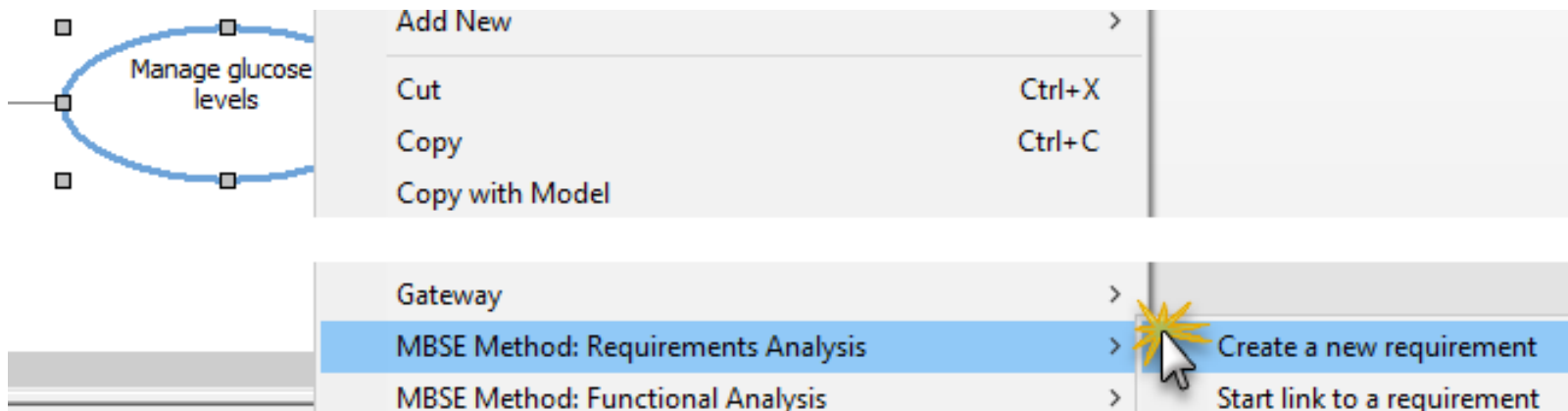
Double-click functionality is overridden on a use case. If a dialog pops up asking to create an activity diagram, then click No.



# Create a new requirement

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

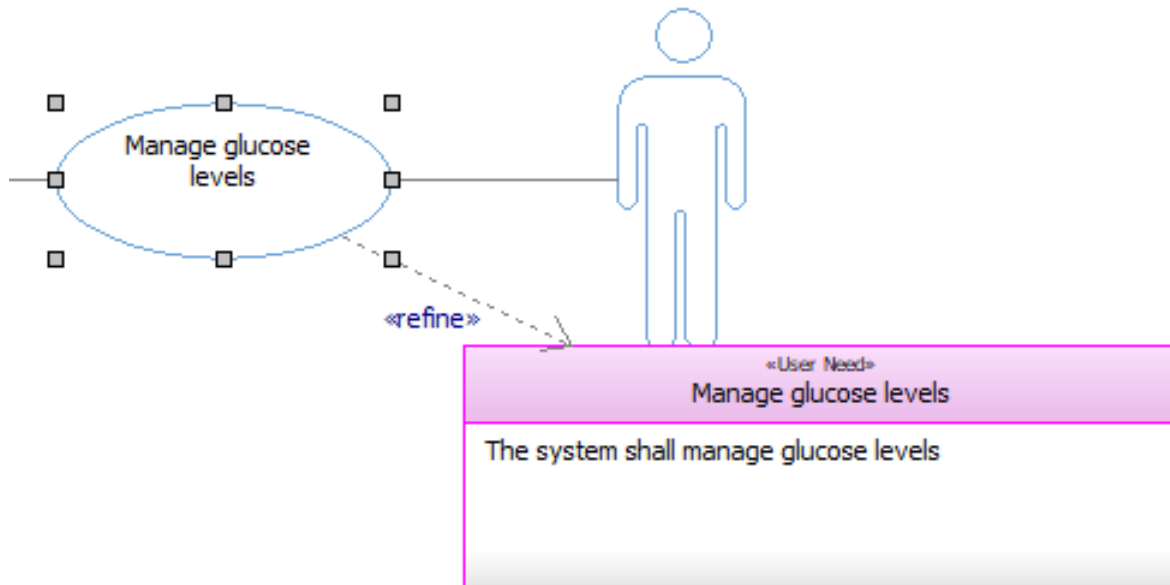
- A simple helper can be used to create a requirement and associated traceability at the same time
- Right-click **Manage glucose levels** and choose **MBSE Method: Requirements Analysis > Create a new requirement**



# Create a new requirement

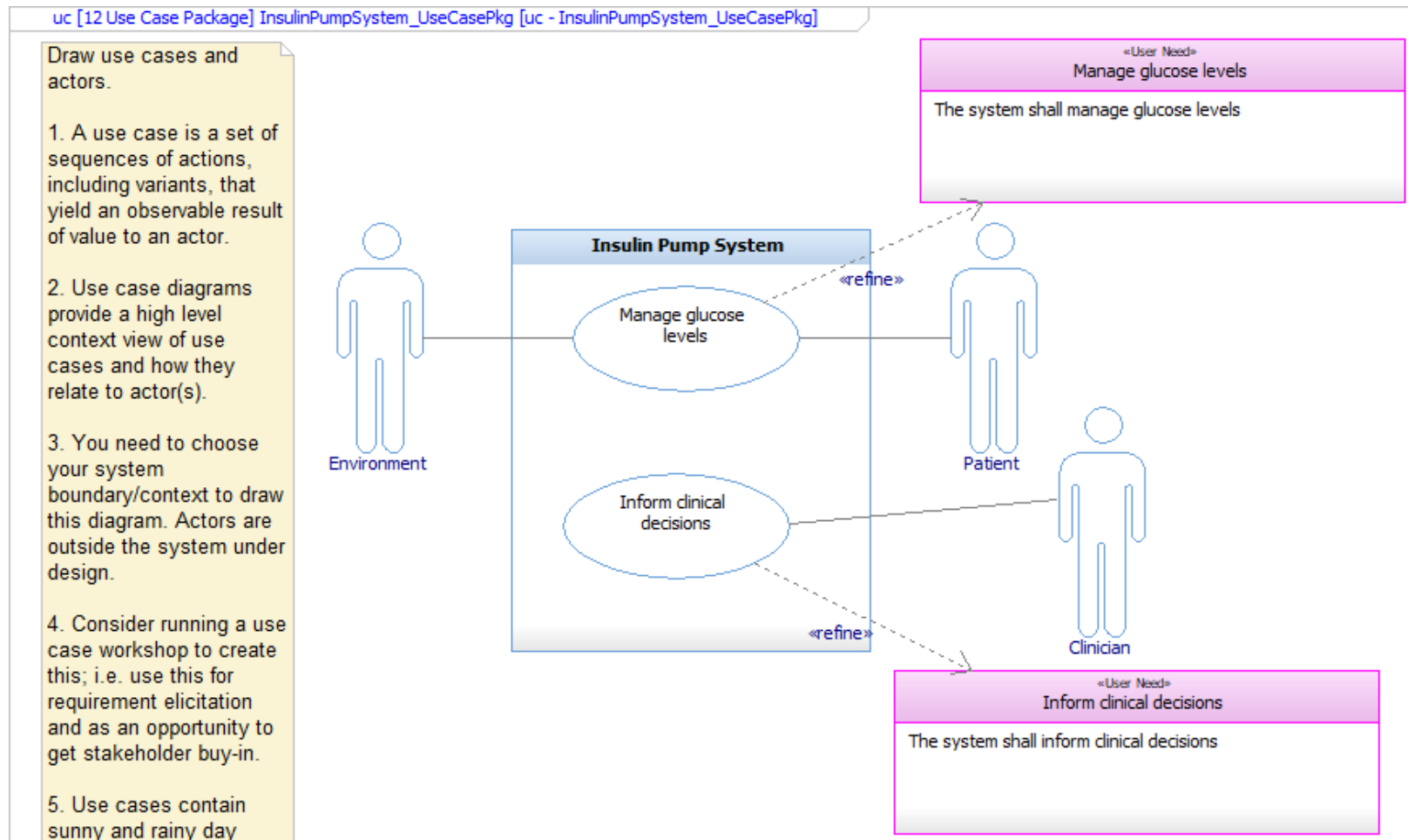
Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- The helper has created the requirement, established a refinement relationship to it, and populated the requirement on the diagram. Some initial text has been added based on the source element



# Finish the diagram

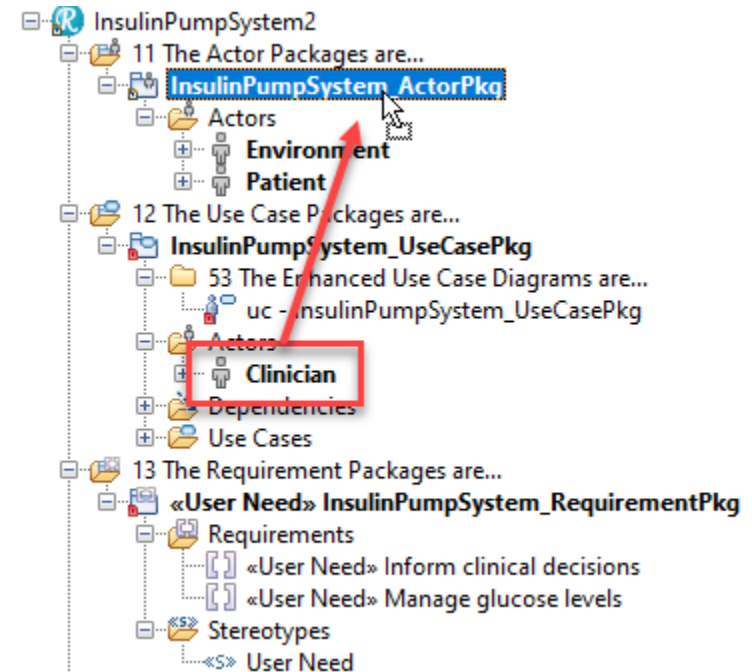
- Add the **Inform clinical decisions** use case and **Clinician** actor to finish the diagram below



# Move the Clinician into the Actor package

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

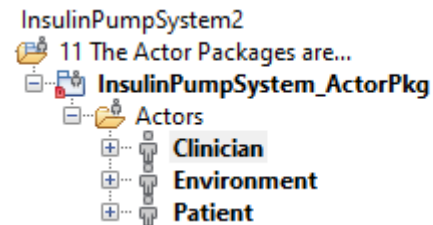
- Move the **Clinician** into the **\_Actor** package



- Check the package structure looks the same as this:



It was much quicker it was to create this model structure using the helper. You can get straight into use case and requirement analysis



# CREATE A FEATURE FUNCTION PACKAGE LAB IB2

"INSULIN PUMP" CASE STUDY

## LAB IB2

*"hear and I forget. I see and I remember. I do  
and I understand" (Confucius 551 BC - 479 BC)*



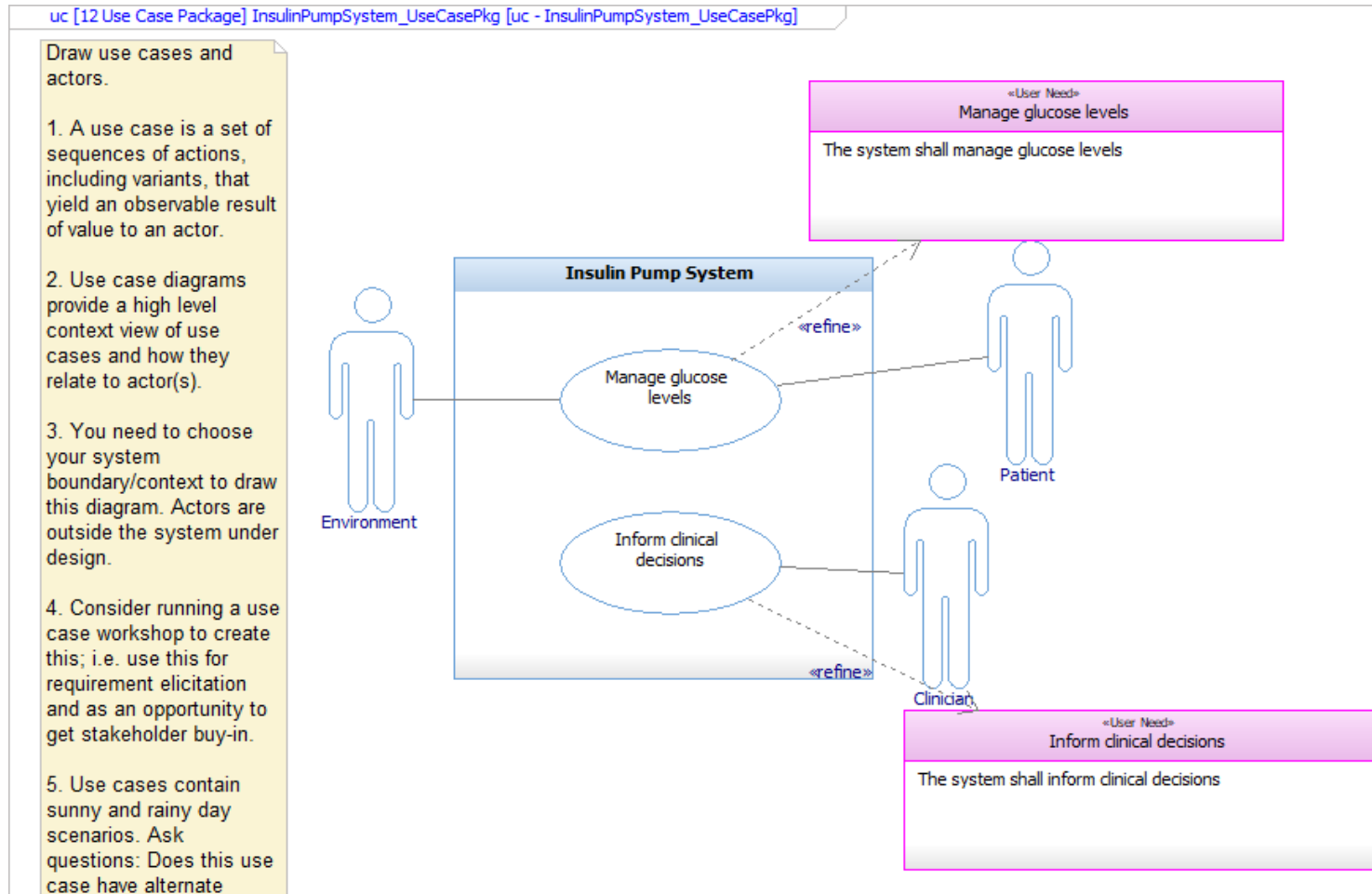
# Functional Analysis

- **Requirements** are fantastic things to have in a model, because they can be linked to any model element. They provide a link outside the model into other worlds such as a requirements management tool
- They are limited as a model element hence we may choose to link the requirement to a model element that represents it instead. For example, an operation or a block
- The profile allows for both possibilities. However, in this example we will focus on use of specialized blocks to represent behaviours that we want our system or subsystems to do. By representing a function as a block, we can interweave them with blocks that represent structural elements on bdds and ibds
- A special type of package called a **Feature Function Package** is introduced to perform functional analysis (the identification of functions) using function blocks owned by a feature

# Open the uc – Insulin Pump System

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Locate the use case diagram and open it up if necessary



# Features and Functions

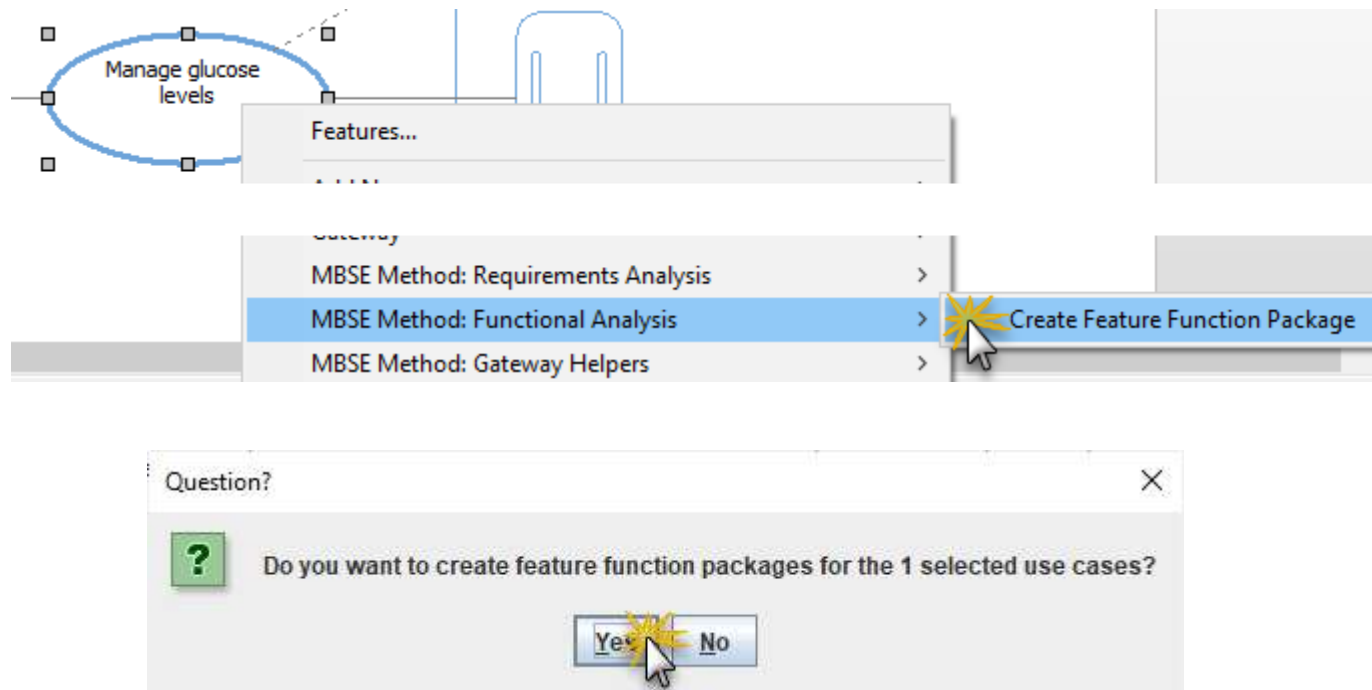
- In this lab we explore the use of function blocks and function usages to describe use case behavior.
  - **Function Block:** A specialisation of a Block introduced by the Executable MBSE profile that describes behavior rather than structure
  - **Function Usage:** A specialisation of a Part introduced by the Executable MBSE profile that represents a use of the Function Block either under a structural block or under another function or feature block
  - **Feature Block:** A specialisation of a Block introduced by the Executable MBSE profile that describes a top-level system function (one visible to actors on the system boundary)
- Features and Functions are kept in a separate part of the model to enable functional decomposition to be performed without touching the use case packages. This means that the two activities can be performed independently by different people



# Create Feature Function Package

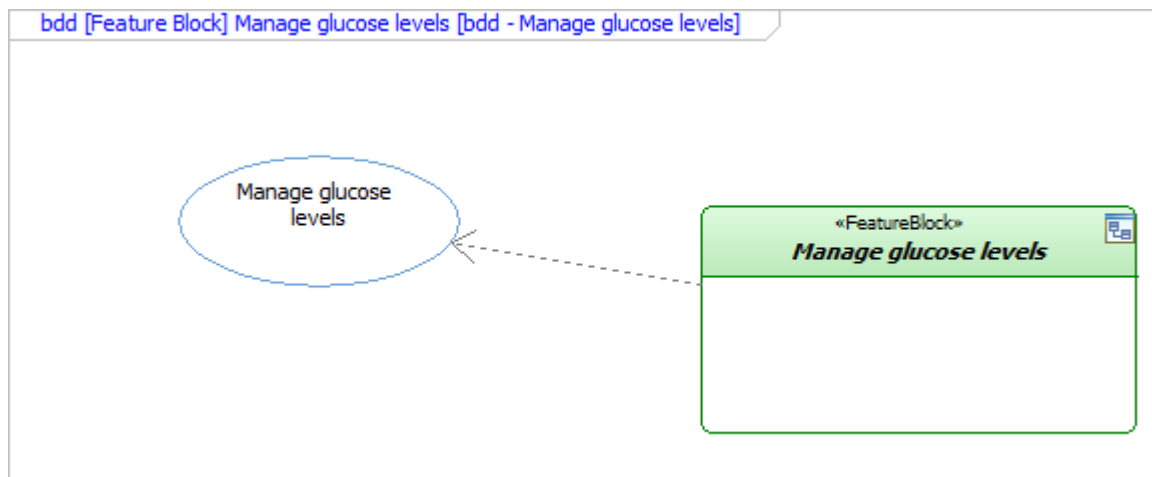
Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Although it's possible to hand build, we can use a helper create a Feature Function Package for a Feature that represents a Use Case
- Locate the **Manage glucose levels** use case and choose **MBSE Method: Functional Analysis > Create Feature Function Package**



# A Feature Block is created


- A **Feature Block** with the same name as the use case is created. This can be thought of as a system-level function. It may have a one-to-one correspondence to the use case, although many-to-one or using them without Use Case diagrams is possible



To show the Feature represents the behaviour of a use case, a dependency is established. The dependency is from the Feature to the use case (which may be in a read-only part of the model). No stereotype is applied so it is easier to draw

# Block Definition Diagram

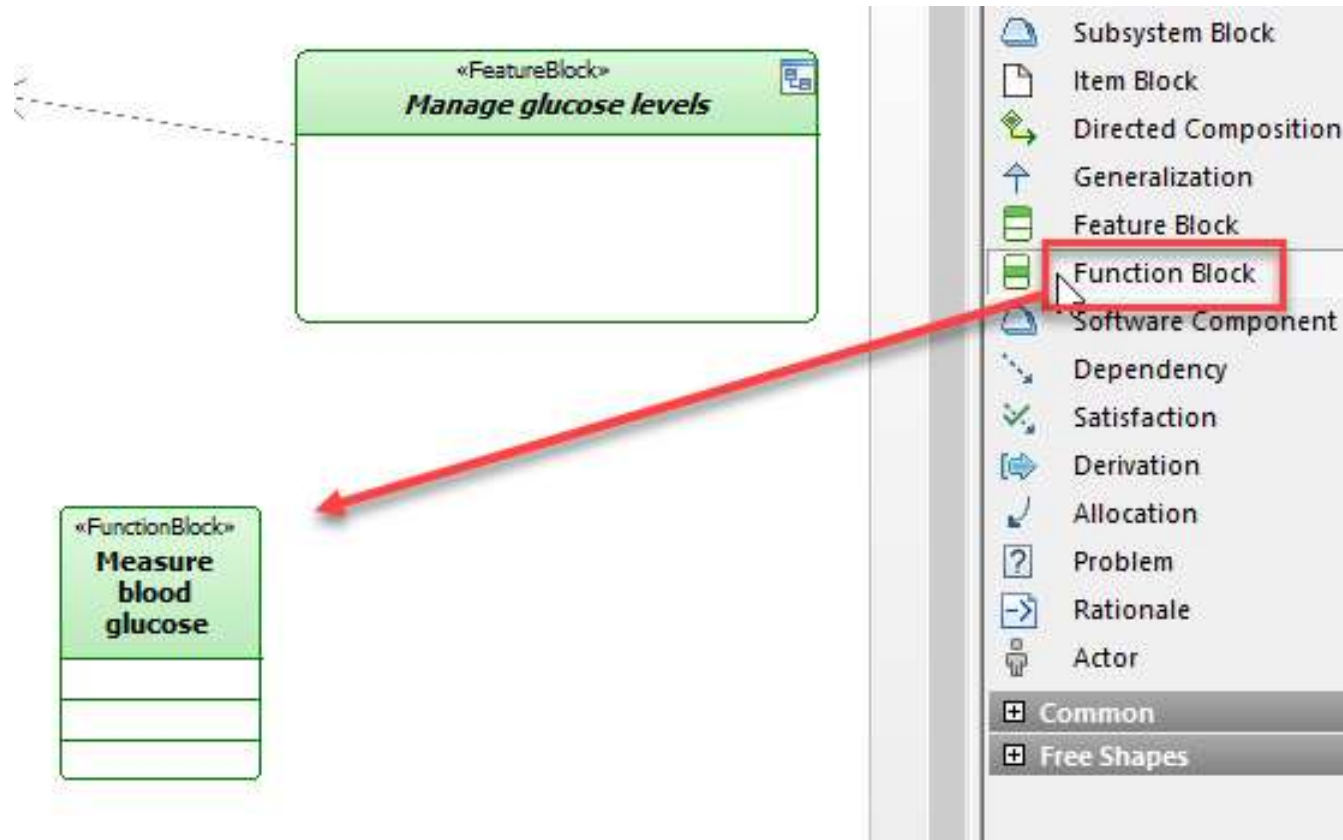
Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- The diagram that is opened is known as a **block definition diagram** (bdd)
- Blocks are elements of definition. They represent types of things that may exist, rather than their usage, hence the term block definition for the diagram
- We can use a bdd to describe a hierarchical breakdown structure by establishing a **Directed Composition**  relation between blocks. The level of behavior below the feature block will be modeling by breaking down the feature into a set of function blocks

# Add a Function Block: Measure blood glucose

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

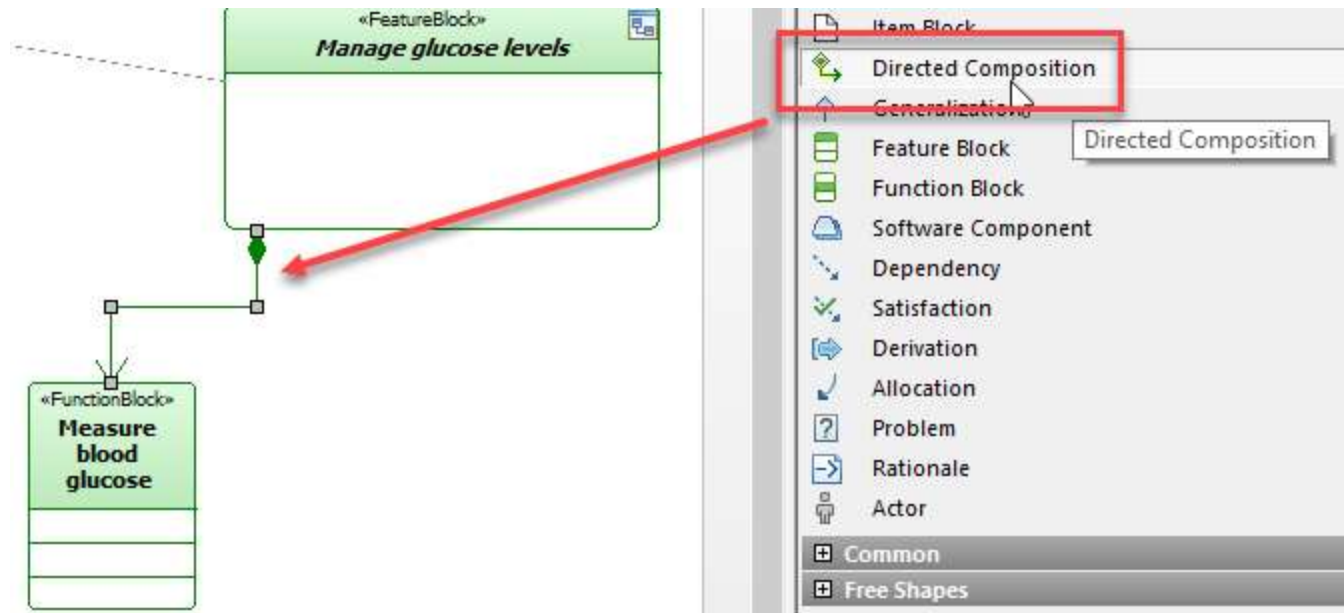
- Add a **Function Block** called **Measure blood glucose** below the feature



# Draw a Directed Composition

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

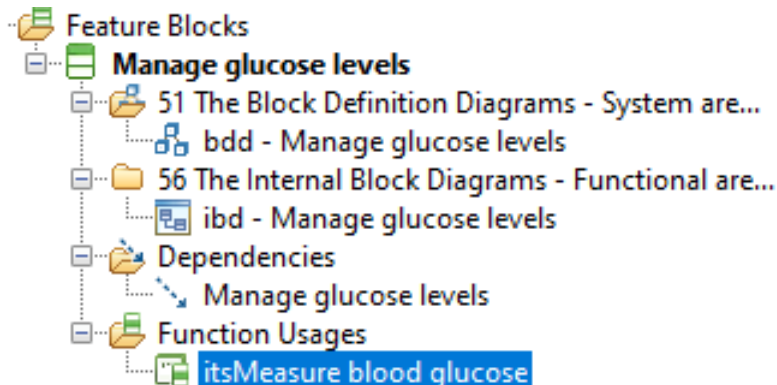
- Draw a **Directed Composition** from **Manage glucose levels** feature to **Measure blood glucose** function



- Select the Directed Composition on the diagram and choose **Navigate > Locate in browser**

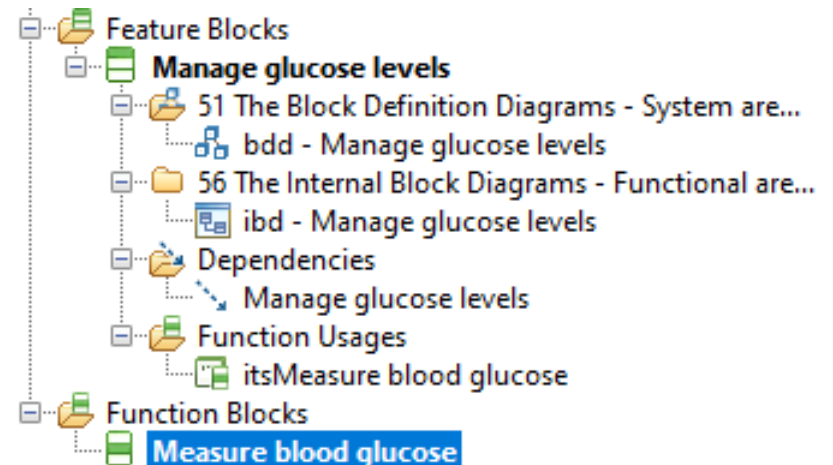
# Locate the function block

- The **Function Usage** (aka Part) is created underneath the owning feature block



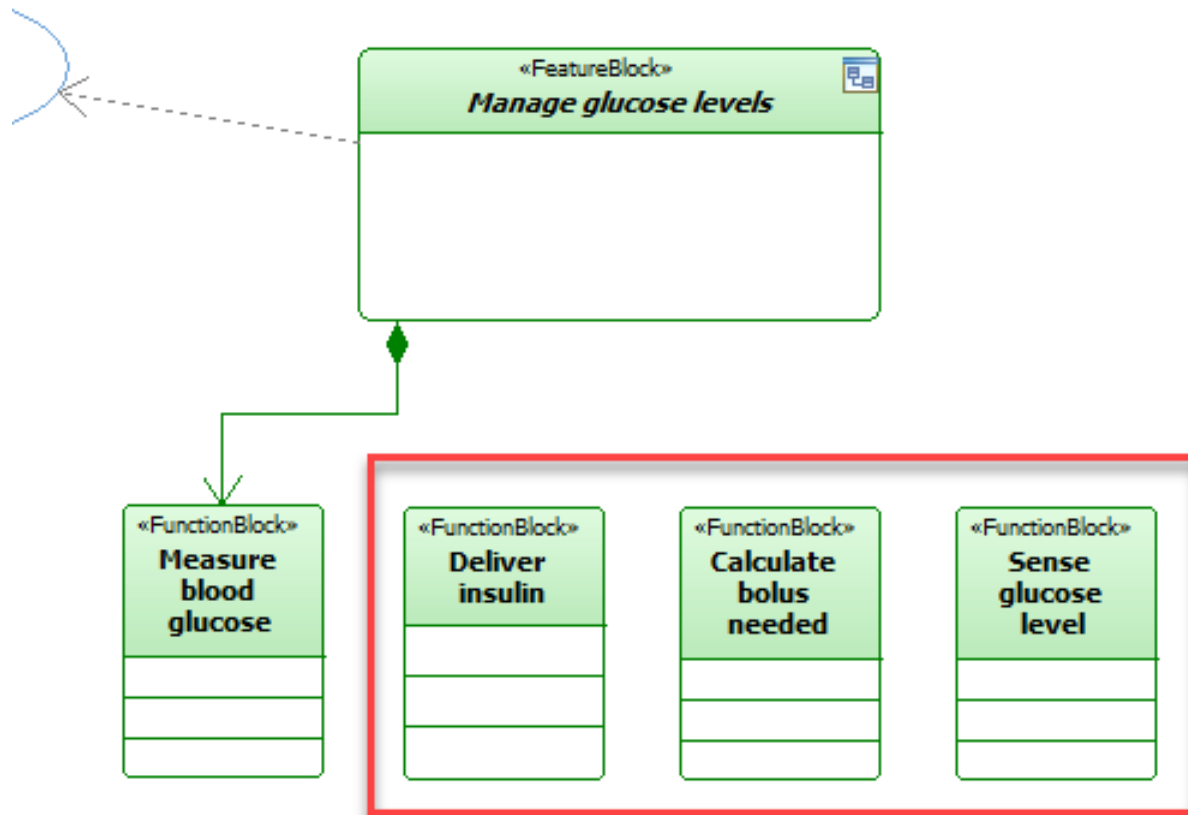
The same function to be reused in many different contexts. The element that defines the properties is the function block. A function usage is the use of the behavior properties defined by the function block

- Right-click the function usage to **Navigate to Class**



# Complete the diagram

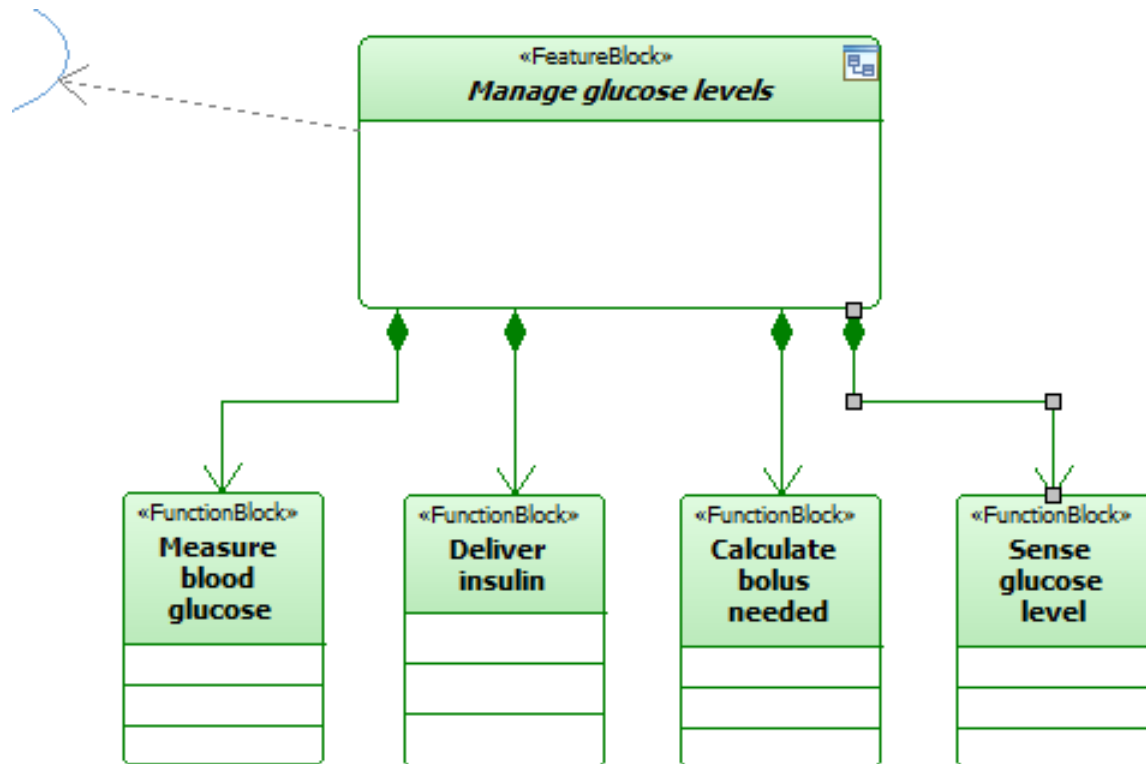
- Add three model function blocks: **Deliver insulin**, **Calculate bolus needed**, **Sense glucose level**



# Draw directed composition relations

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

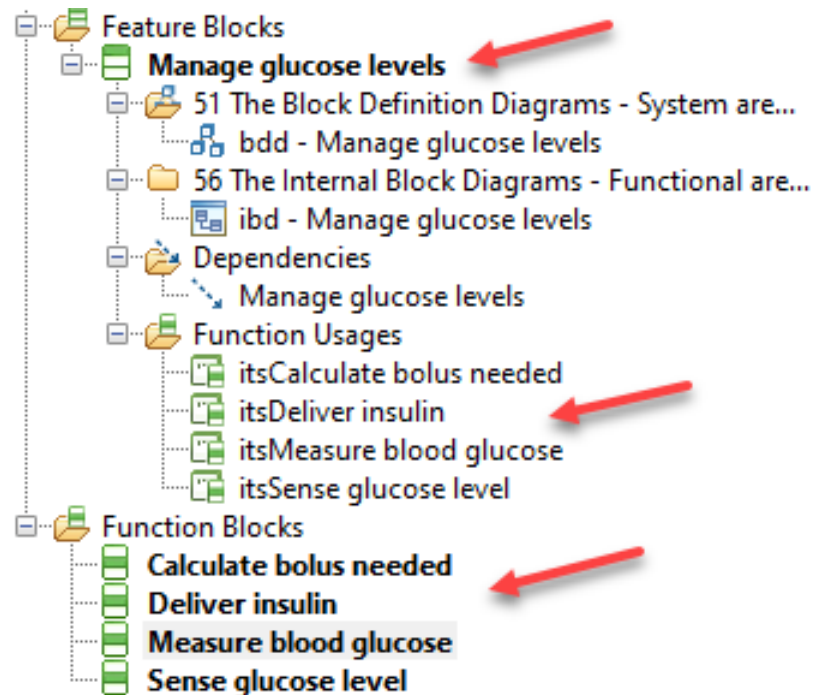
- Draw **Directed Compositions** to complete the diagram and create the associated function usages underneath the feature





# Review the browser

- Expand the browser. The **Feature Block** has nested **Function Usages**, and the package contains the **Function Blocks** (which are the blocks - units of definition that detail their requirements, properties and ports)





# FUNCTION CHAINS WITH IBDS LAB IB3

"INSULIN PUMP" CASE STUDY

## LAB IB3

*"hear and I forget. I see and I remember. I do  
and I understand" (Confucius 551 BC - 479 BC)*



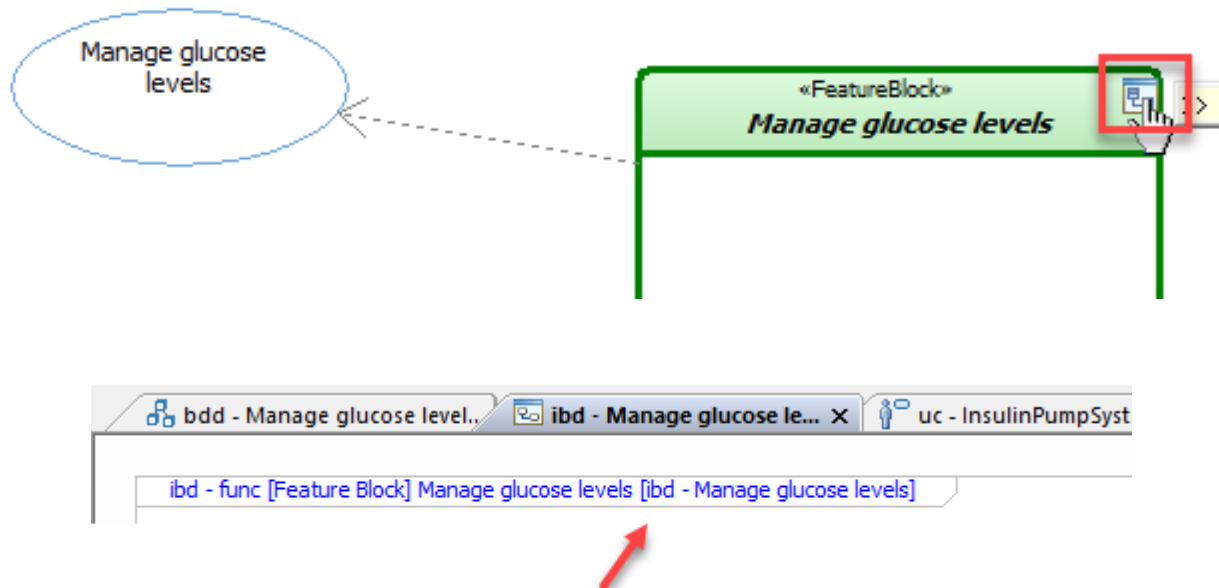
# Using Internal Block Diagrams

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Using convention SysML methodologies we might model functional flow using SysML activities and activity diagrams
- The Functional Method supported by the Executable MBSE profile does not do this. Instead, it allows the behavioural aspects of how Function Usages are chained together to be described using Internal Block Diagrams. This can be done in either:
  - A behavioural context, irrespective of the allocation of behavior to subsystems
  - A structural context, where the subsystem boundaries are present
- The key benefit of using function blocks is that it allows a closely relationship between behavior and structural aspects of the system, something not easy to achieve with separate structure and behavior diagrams (something that the Arcadia/Capella method is better at than SysML).

# Open the IBD

- An internal block diagram was created automatically for the Feature Block. Navigate to it by clicking on the quick navigation icon in top-right of the graphical node

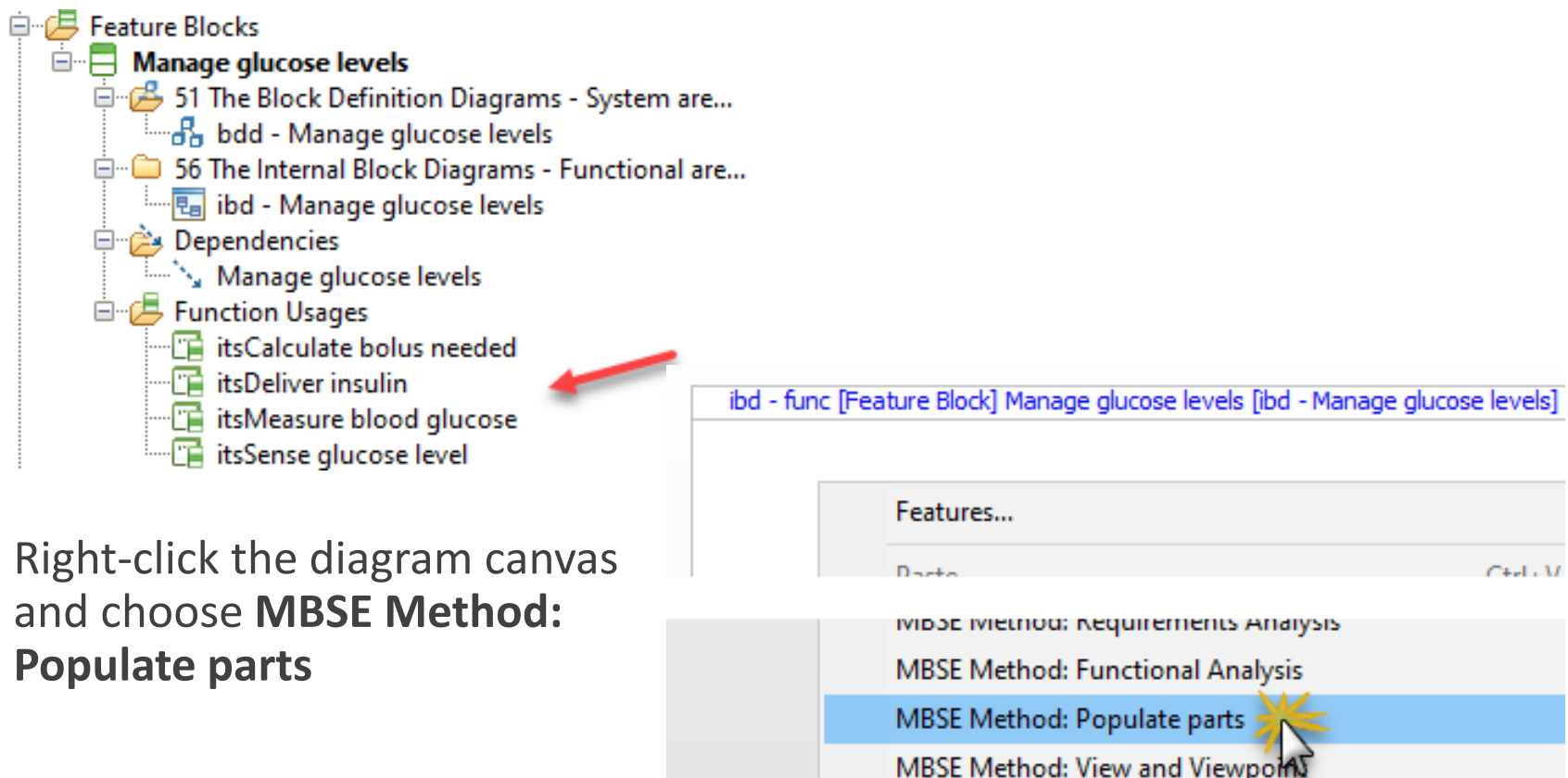


When using the methods supported by the profile, diagrams are always named with a prefix which is the SysML diagram type. This makes it easier to identify in the diagram tabs and pull-down lists

# MBSE Method: Populate parts

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

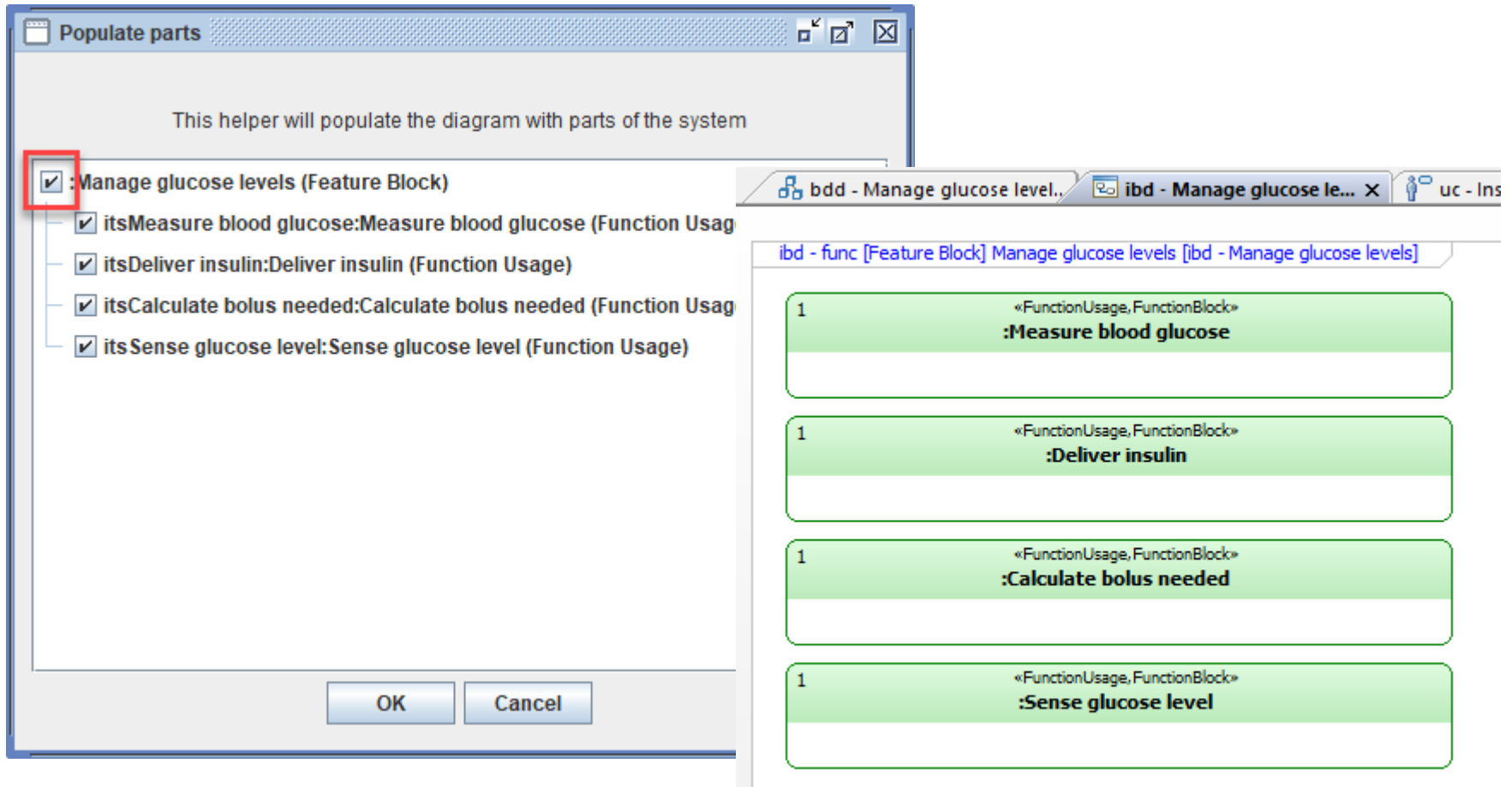
- An **internal block diagram** is a diagram of usages (aka parts) rather than blocks. We already have created these using directed compositions on the bdd. We can populate these using a helper provided by the profile



- Right-click the diagram canvas and choose **MBSE Method: Populate parts**

# Select usages and click ok

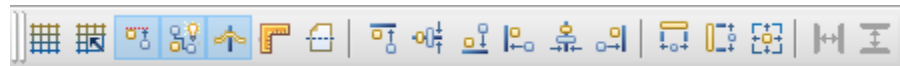
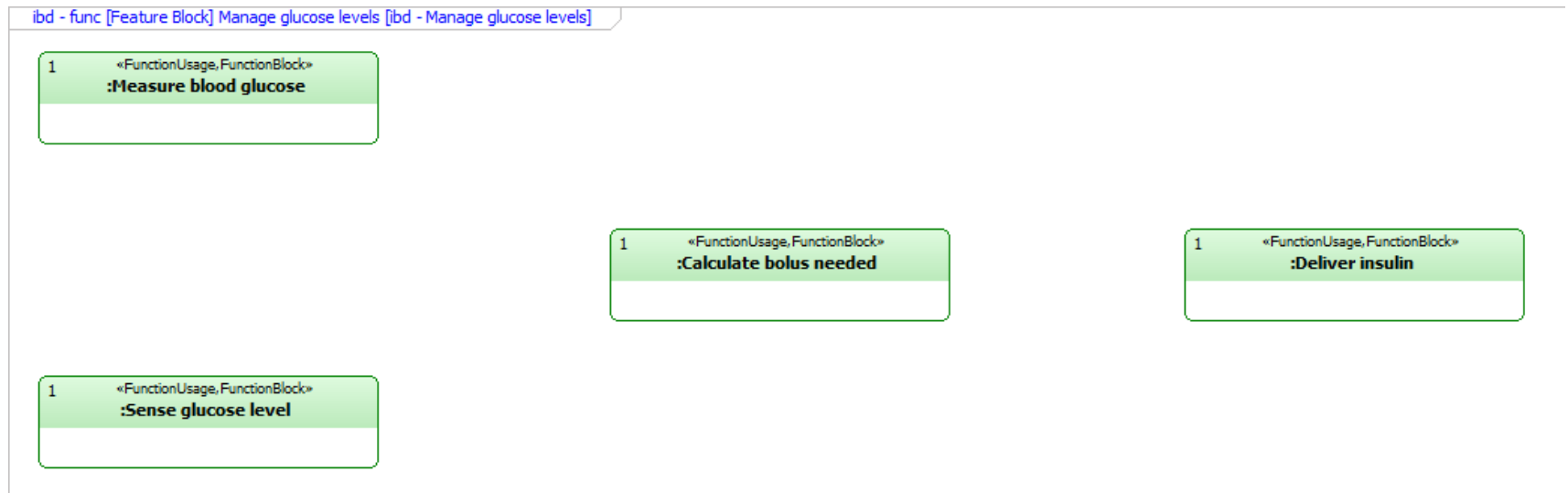
- Check the box to select all the usages and click **OK** to populate on the diagram



# Reorder and resize the function usages on the ibd

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Reorder and size the function usages like the below



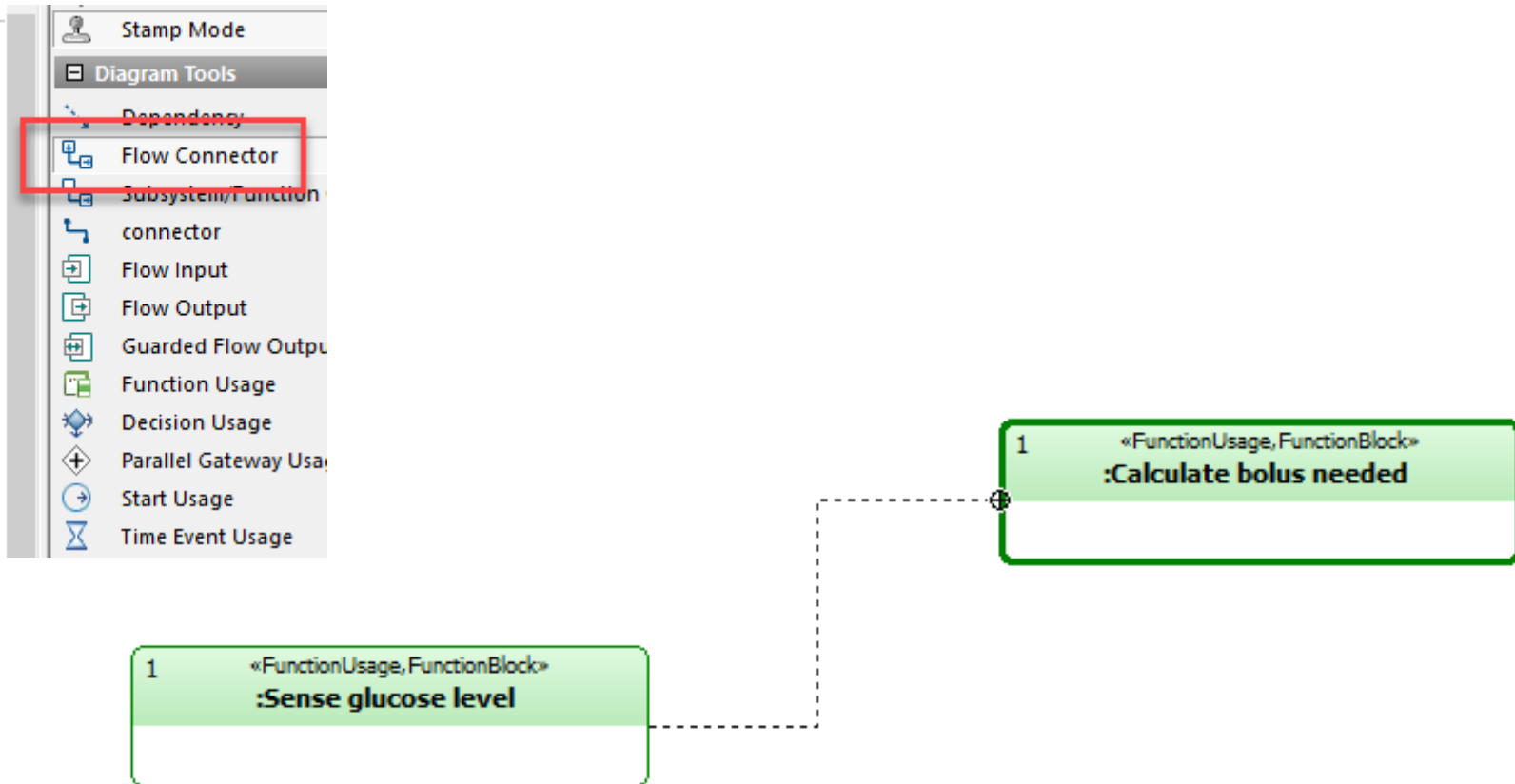
The layout toolbar can prove handy. Holding down Ctrl key can be used to multi-selected items



# Reorder and resize the function usages on the ibd

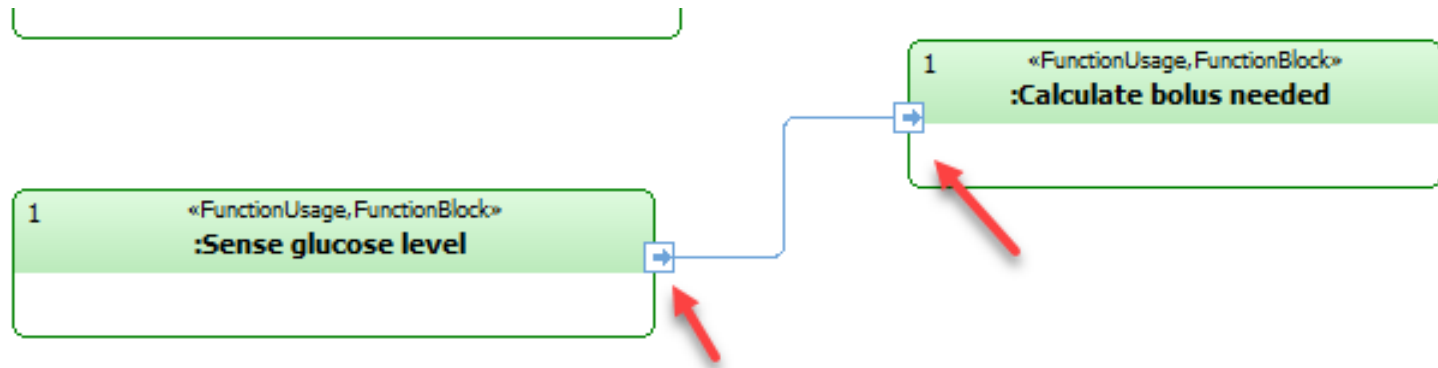
Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Add a **Flow Connector** from **Sense glucose level** to the **Calculate bolus needed** function usage



# View created flow ports

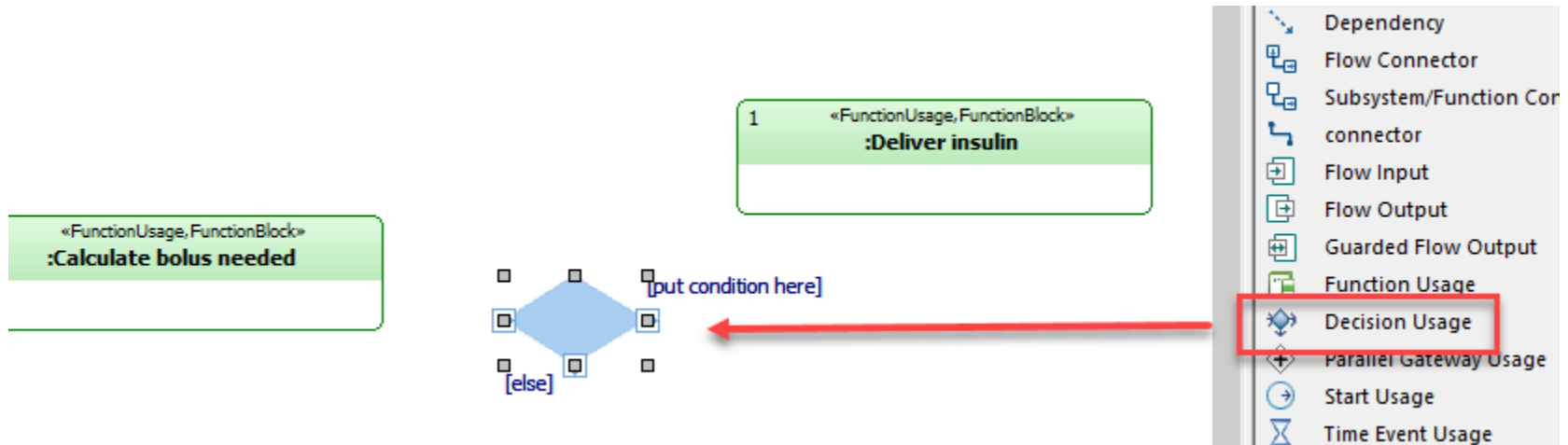
- The helper automatically creates flow ports with directionality on the boundary of the function usage (these are properties of the function block)



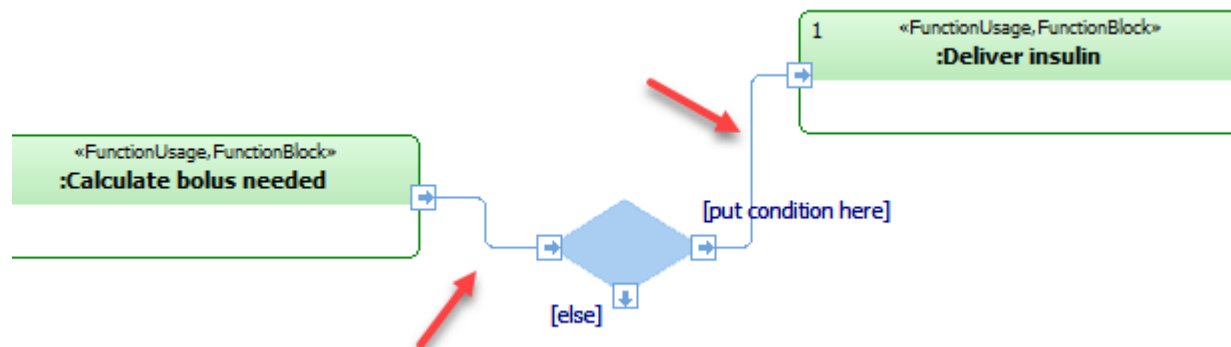
The display options are currently set-up to hide the type of these flow ports, hence we would read this as control flow. For info: we could type the flow-ports to show what flows, if that is more appropriate for the behavior, we are modeling

# Add a Decision Usage

- The profile adds concepts to allow parallel and decision logic to be added to the ibd. Add a **Decision Usage**



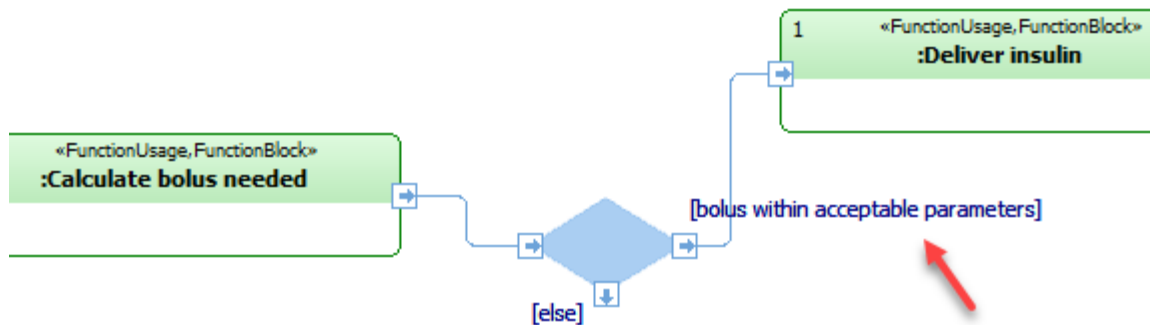
- Draw the flow connectors to wire the decision usage to the function usages



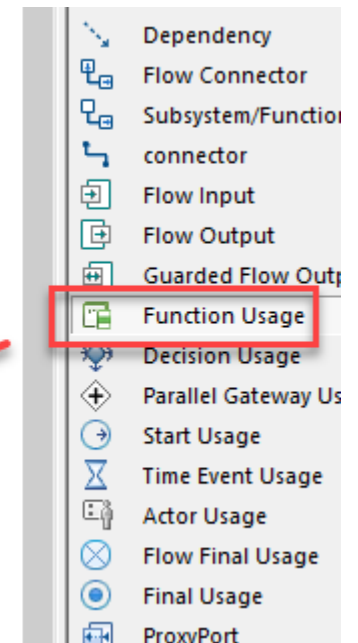
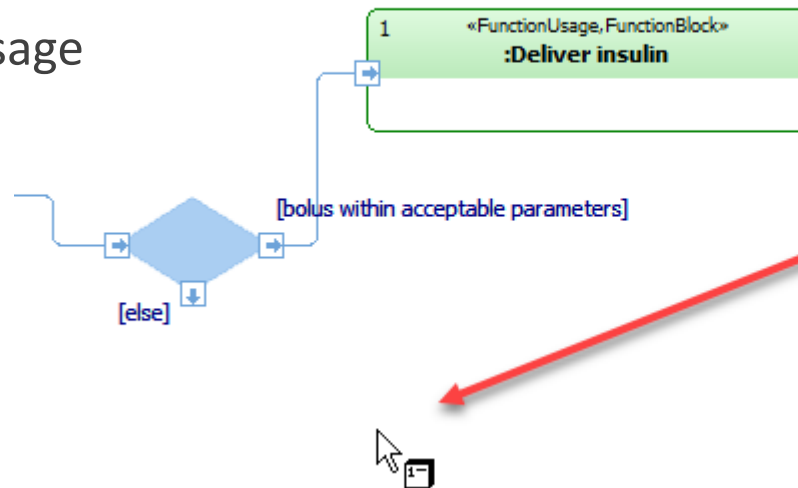
# Add text to the guard condition

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Add text to the guard condition: **Bolus within acceptable parameters**



- Add a new function usage



# Use dialog to create a new Function Block

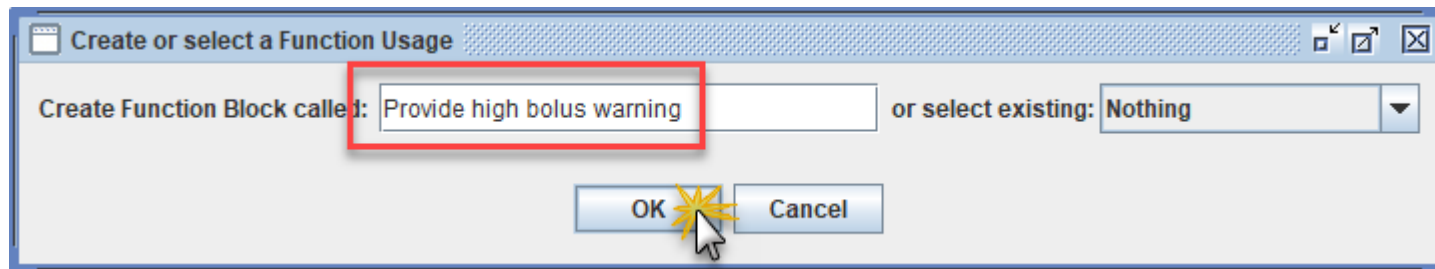
Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- The helper provides a dialog to create a new function block or select an



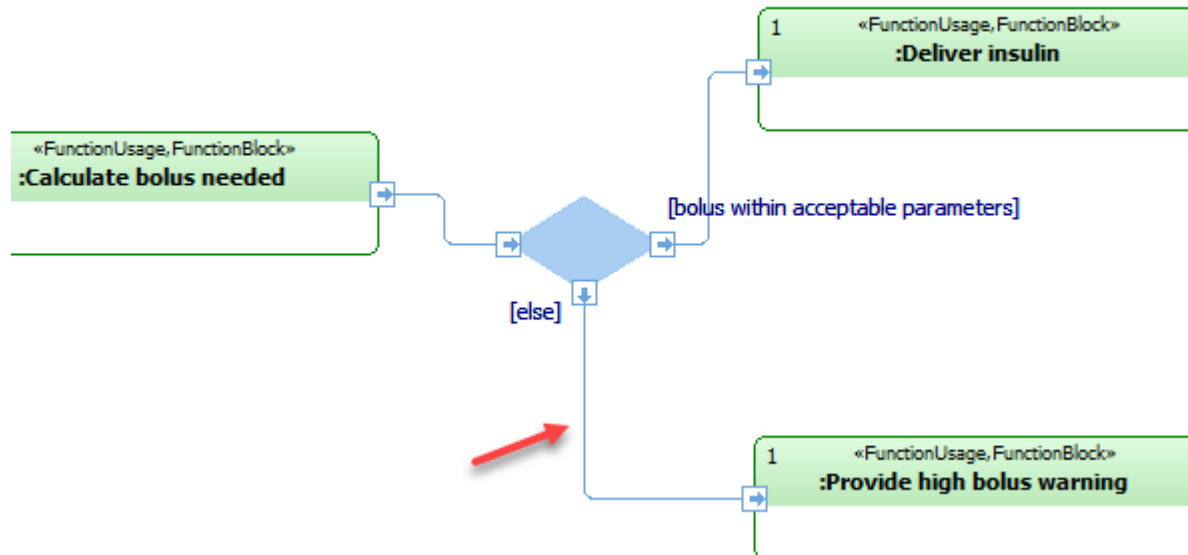
If you can't see this dialog, then it's possible it was launched behind Rhapsody or on a different screen

- Call the function block: **Provide high bolus warning** and click **OK**

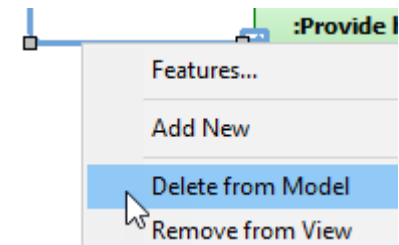


# Add a flow connector

- Add a **Flow Connector** to the new function usage

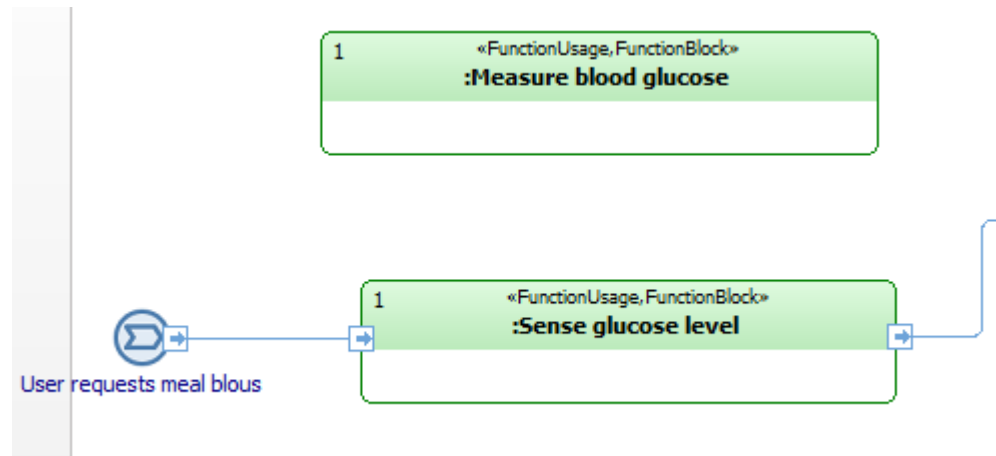


If you draw the wrong relationship or add a port you don't want, avoid pressing just the Delete key as the relationship and ports are not deleted from the model if you do this, just the diagram. Always Delete from Model rather than Remove from view, when drawing this diagram



# Add an accept event usage

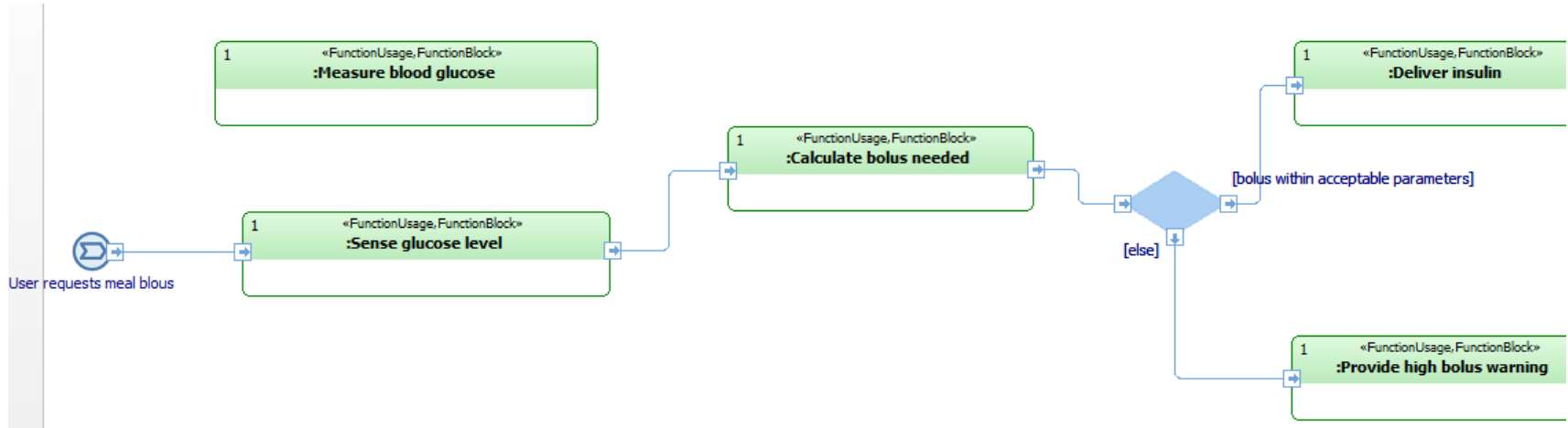
- Add an **Accept Event Usage** to show when the flow starts: **User requests meal bolus**



You may need to press Ctrl + A to highlight and move other elements to the right to make space on the left for this

# Review the diagram

- Review the diagram



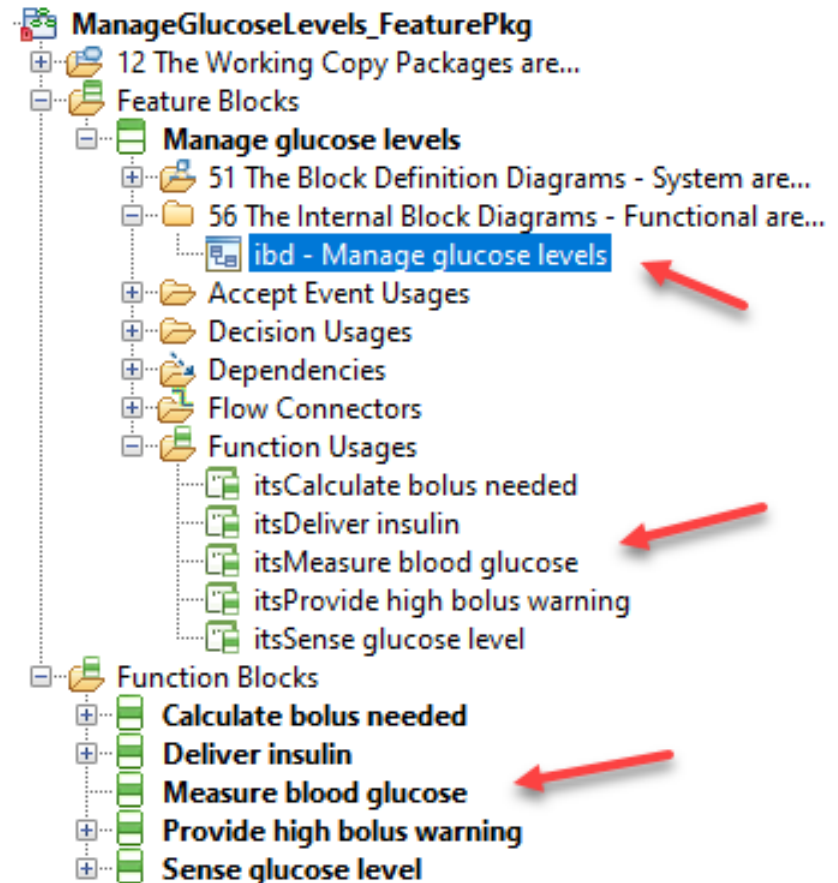
Using the diagram toolbar provided by the profile we can build function chains using an internal block diagram. We can also decompose functions into their own internal block diagrams, similar to how call behavior actions in the activity diagram model work



# Summary

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

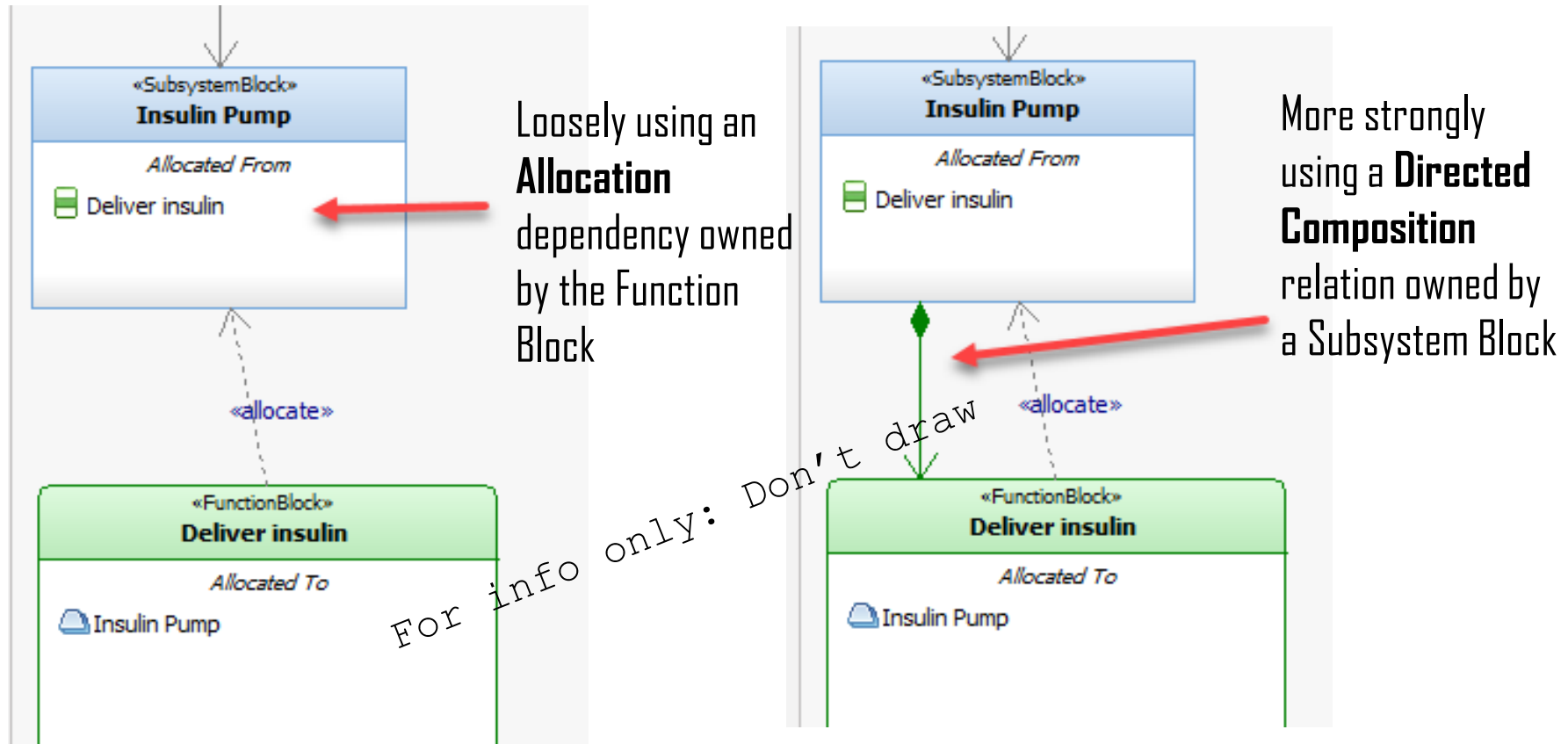
- Internal Block Diagrams are created underneath the element that defines them
- This diagram was created under a **Feature Block** and is therefore purely functional in nature, we've created a function chain of what the system may do, irrespective of how the system is decomposed
- The function chain is created between **Function Usages** of **Function Blocks**. The same function blocks can be re-used in other features. The same function blocks can also be used to type function usages in the logical architectural model



# For info: Allocation of function blocks to subsystems

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

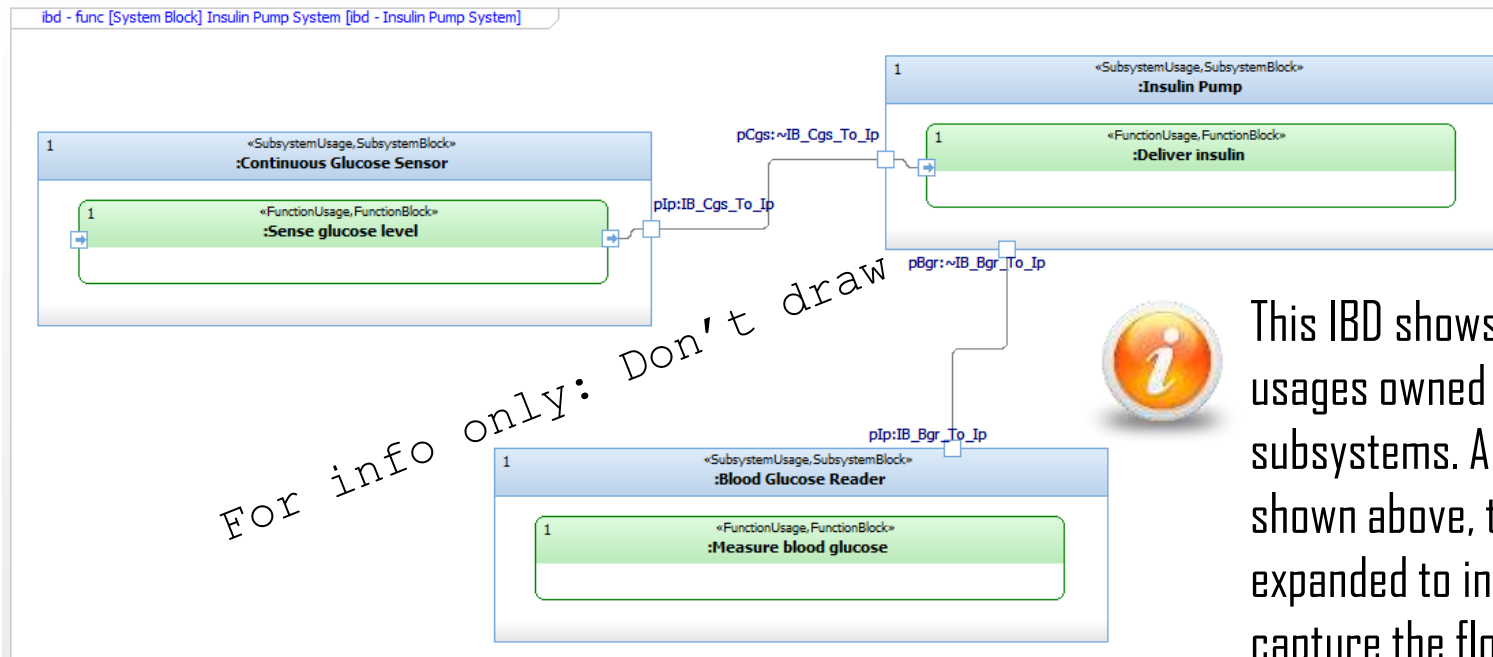
- There are two ways we can allocate these functions to structural definition elements. These two approaches are explored in different training module



# For info: Mixing behavior and structure on same diagram

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Use of Function Blocks enables the following notation to be achieved; a much closer relationship between behavioural and structural modeling than can be achieved with the activity model. E.g. You could show function chains inside a subsystem, if you wished



This IBD shows the function usages owned by the subsystems. Although not shown above, the model can be expanded to include Events that capture the flows between system boundaries, i.e. as exchange items



# INSTALLING THE SYSML HELPER

## Appendix A

"INSULIN PUMP" CASE STUDY

### APPENDIX A

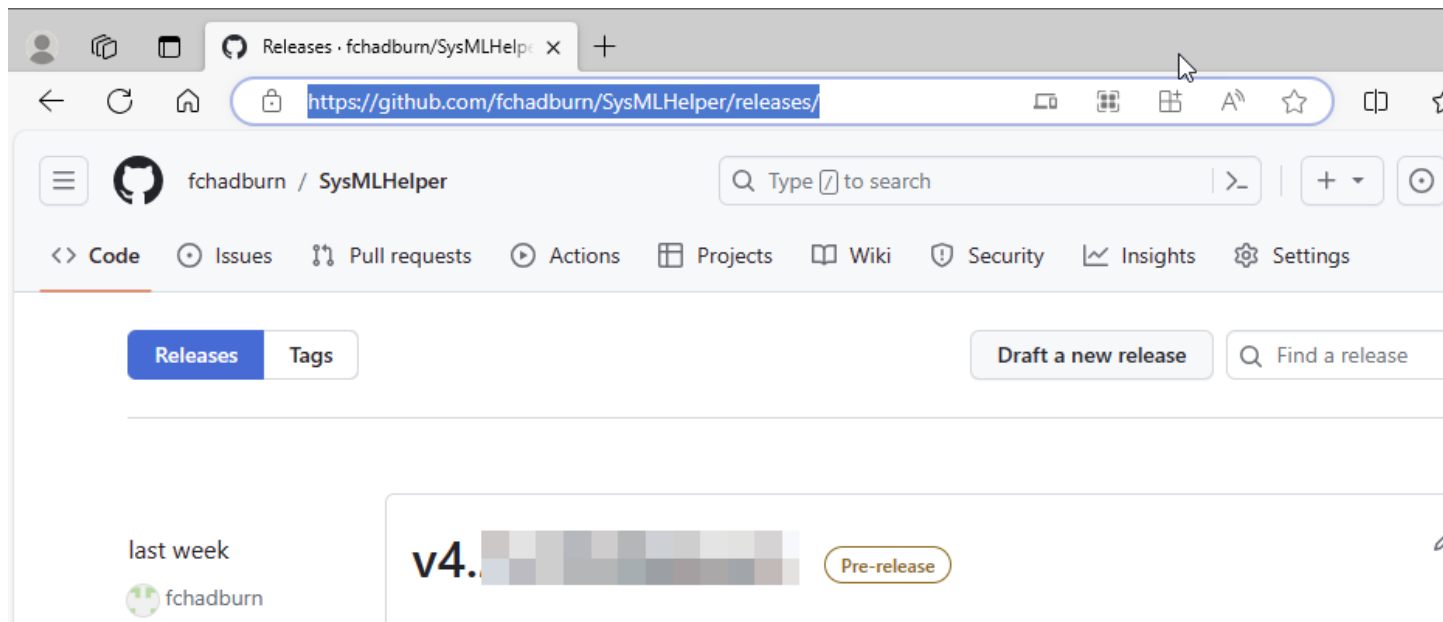
*"I hear and I forget. I see and I remember. I do  
and I understand" (Confucius 551 BC - 479 BC)*



# Obtain the profile from Github

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Go to <https://github.com/fchadburn/SysMLHelper/releases/> to find the releases. Locate the version needed



You can also visit my site at:  
<http://www.executablembse.com/> which contains  
some further info on latest changes

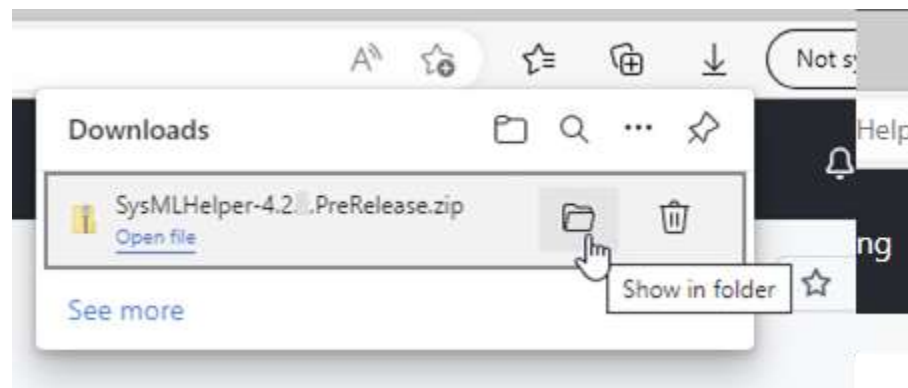
# Obtain profile from Github

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- Click to download Source code (zip)

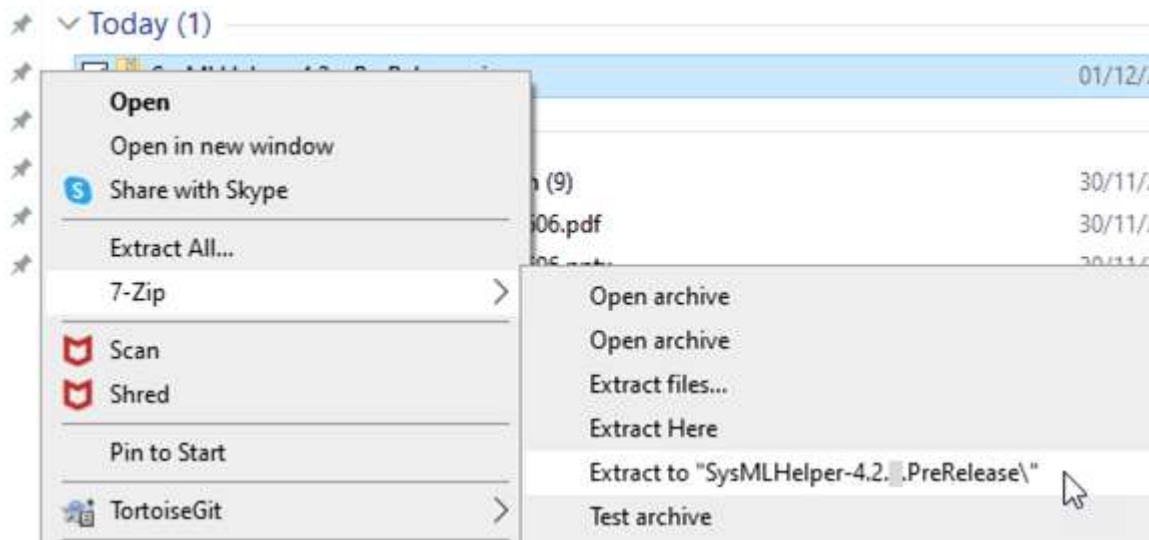


- Locate file

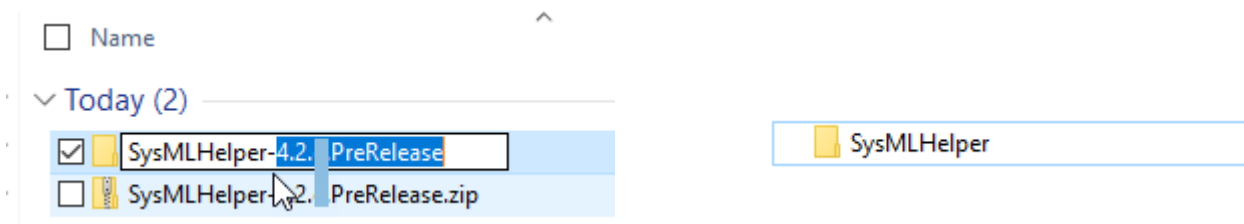


# Extract to folder of same name

- Extract to folder of same name



- Remove the release part of name so top-level is called SysMLHelper

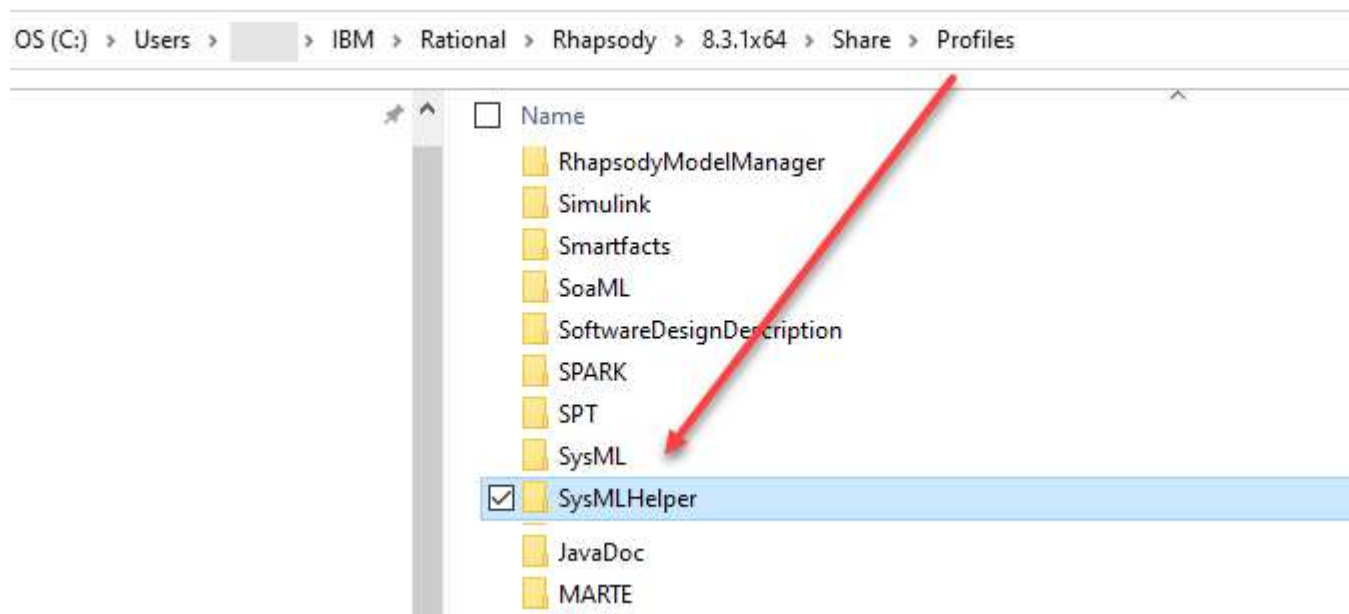




# Locate your Rhapsody share/profiles folder

Copyright © 2015-2024 - MBSE Training And Consulting Ltd

- This is where Rhapsody's factory (out-of-the-box) profiles are located. E.g. in 9.0.1. this is C:\Program Files\IBM\Rhapsody\9.0.1\Share\Profiles
- This is where the SysMLHelper profile needs to go, e.g., 8.3.1x64 may be



- Delete, or move out, rather than rename the existing profile you have or overwrite over the top (this is very important as Rhapsody assumes only one .sbs/.sbsx file of a particular name exists under its Profiles folder)