MVE Analysis

Kamil Ježek

26^{th} April 2009

Basic information

- data-flow driven component framework
- used for transforming data mainly for graphical experiments [Frank et al., 2006]
- components are called 'modules'
- fully programmed in *C#*
- the modules are managed by the core
- the user controls the modules through the command-line or graphics client
- a set of the modules creates a module map
- each run of the module map transfers data among the modules
- the result of the computation are modified data

Meta-model of framework

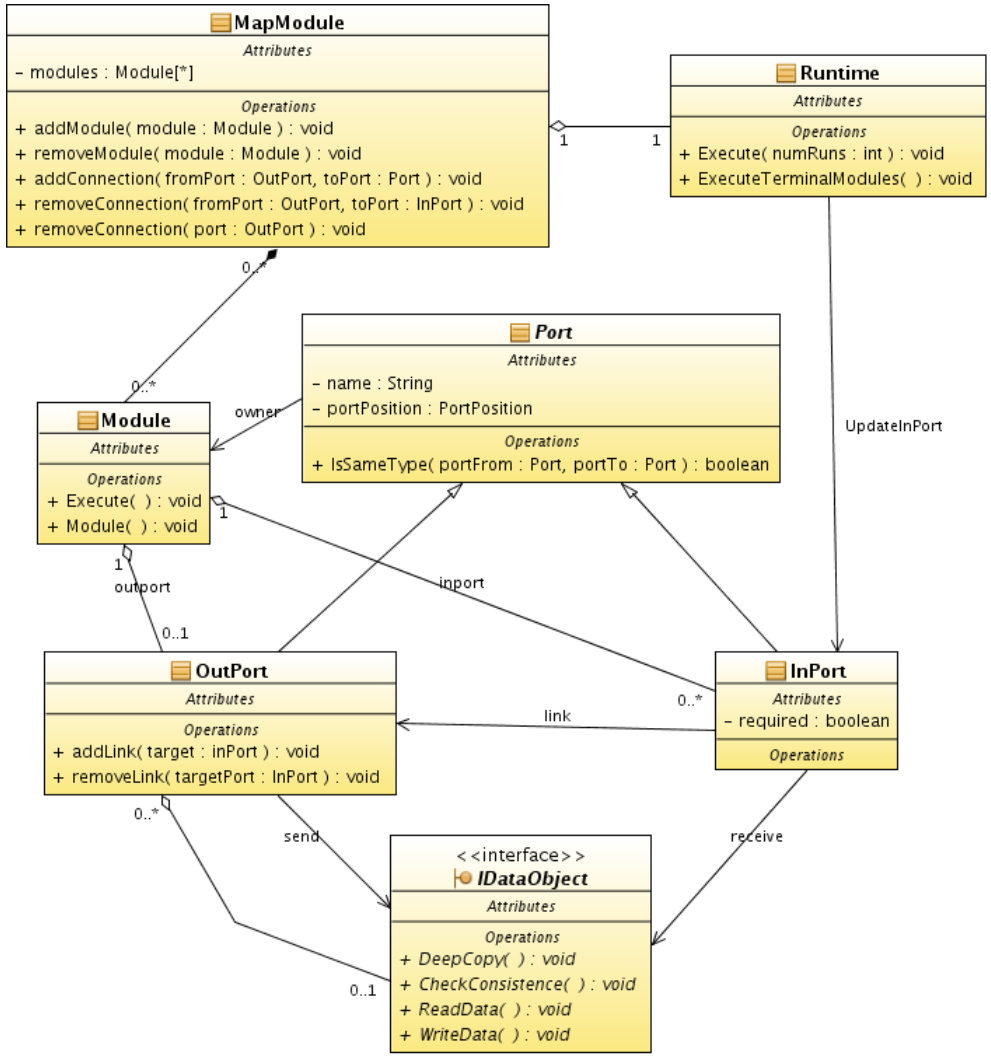

Figure 1: The meta-model of the framework core

Description of Meta-model

- a module is a class inherited from the *Module*
- a module contains a set of input and output ports
- data transferred between the ports are of the type of *IDataObject*
- each computation loop is controlled by the *Runtime* which can repeat the computation loop
- *MapModule* allows the change of the structure of a module map

see Figure 1

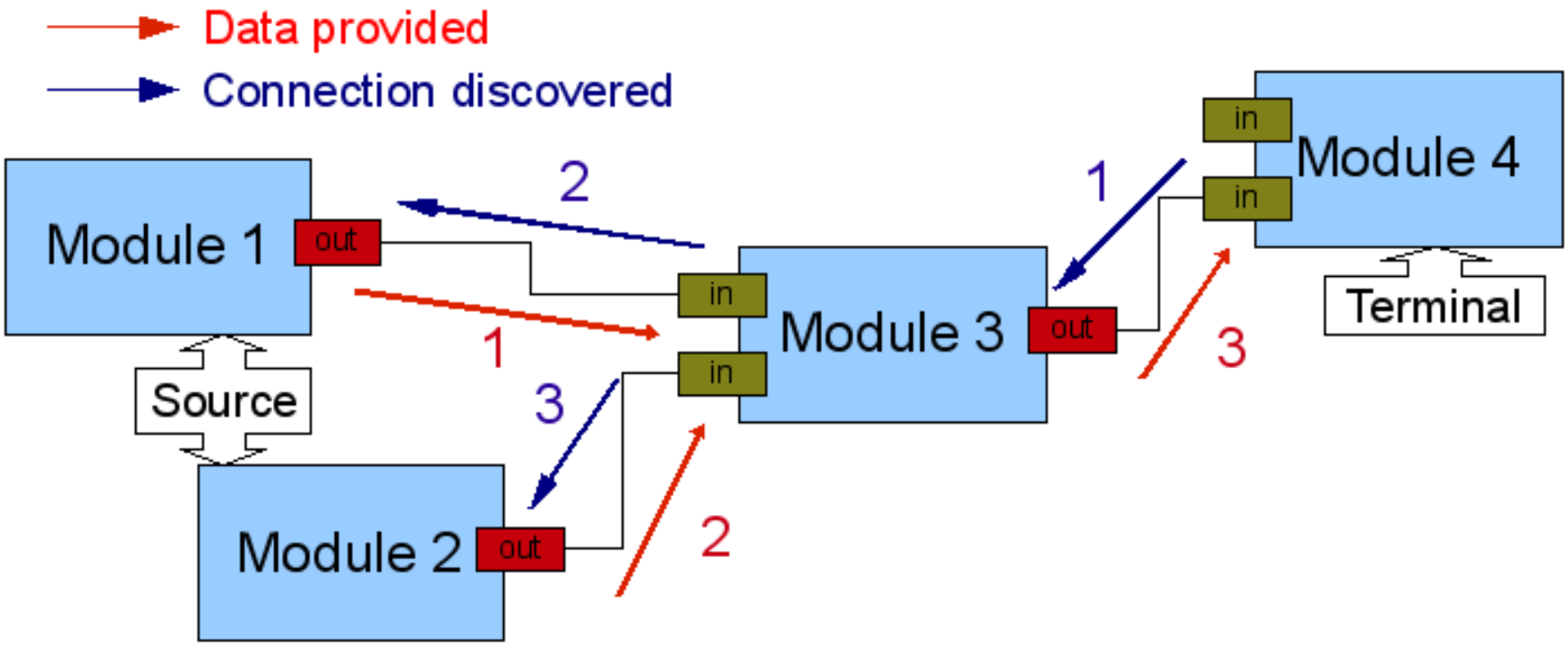

Figure 2: The connection of modules

Data processing

- there is a depth-first search algorithm used (see Figure 2)
- a module is the sink (terminal), source or filter one

Algorithm

- a procession starts from terminal modules
- all connected modules are detected
- the detection continues until all source modules are reached
- the data are sent from the source modules
- the computation ends when the data reach terminal modules

Comparison of Strengths and Weaknesses

The comparison has been done along definitions from [Szyperski et al., 2002], [Bachman et al., 2000].

Characteristic	Supported	Strength	Weakness
Observable state	no	√	
Re-entrance	yes	√	
Unique identity of components	yes	–	–
Versioning	no		√
Black box	yes	√	
Callbacks	no	√	
Third-party composition	yes	√	
Uniform component types	yes	√	
Extra-functional properties	no		√

Figure 3: General characteristics supported in the framework

Description of the table

The description of Figure 3

- observable state - only one re-entrant interface method *Execute*
- re-entrance - each computation loop is processed in the *Execute* method, which is re-entrant in essence
- unique identity - a module map often contains more instances of the same component
- versioning - no mechanism distinguishing between versions of components is provided
- black-box - information about modules may be obtained from the GUI editor

Description of the table

- callbacks - a data object contains the method *DeepCopy* which **may** (not necessarily) create a copy of the data
- third-party composition - the GUI editor allows an easy composition of components
- uniform component types - there is only one super-type for modules and for data
- extra-functionl properties - is not supported

Thank you for you attention


Bachman, F., Bass, L., Buhman, C., Comella-Dorda, S., Long, F., Rober, J., Seacord, R., and Wallnau, K. (2000).
Volume ii: Technical concepts of component-based software engineering, 2nd edition.
Technical report, SEI Joint Program Office.


Frank, M., Váša, L., and Skala, V. (2006).
V.: Pipeline approach used for recognition of dynamic meshes.
In *Proceedings of 3IA 2006*.


Szyperski, C., (with Dominik Gruntz, and Murer), S. (2002).
Component Software - Beyond Object-Oriented Programming – Second Edition.
Addison-Wesley / ACM Press.