



November 8-9th, Iquique, Chile

Specifying Fractal and GCM Components With UML

Solange Ahumada, Ludovic Apvrille, Tomás Barros, Antonio Cansado, Eric Madelaine and Emil Salageanu









Introd uction

 \succ Introduction

> Our contribution

> Fractal

⊳ GCM

> Turtle

> UML 2 and Fractal

> CTTool

> overview

⊳ CSD

> SMD

> tools

> GCM/ProActive components

> Language Extensions

 \succ Conclusion

> References

> Strong emphasis on system specification methods and tools

➤ Component-Based Software Development >UML 2[1] → Component Diagrams

> Specification

> Formal

> Informal non-expert users ③ ambiguity ③

expert user, longer time \otimes precise \rightarrow verification \otimes

> Textual or Graphical

Our contribution:

> Introduction

> Our contribution

State of the art of component models

Fractal

GCM

Turtle

> CTTool

overview

CSD

SMD

tools

> GCM/ProActive components

> Language Extensions

> Conclusion

> References

> A UML-based framework and tool for specifying and model checking software components

> UML 2 and Fractal > A novel UML profile proposal dedicated to distributed and asynchronous software components

>Grid applications

> Fractal[2]

- \succ Introduction
- > Our contribution
- State of the art of component models
 - > Fractal
 - > GCM
 - ≻ Turtle
 - \succ UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - ≻ tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

- > Hierarchical component model
 - Component, controller, content, interface and binding.
- > Behavior protocol between components
- > Graphical editor, but no modelling tool.



Grid Component Model (GCM)[3]

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - ≻ GCM
 - > Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - ≻ tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

> Extension of Fractal to distributed applications

- > Asynchronous method calls
- > Implementation: ProActive[6]
 - > Active object
 - > Future value: rendez-vous



Grid Component Model (GCM)[3]

- > Introduction
- > Our contribution
- - > Fractal
 - > GCM
 - ≻ Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

Extension of Fractal to distributed applications

- > Asynchronous method calls
- > Implementation: ProActive[6]
 - >Active object
 - >Future value: rendez-vous



Grid Component Model (GCM)[3]

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - ≻ GCM
 - > Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - ≻ CSD
 - ≻ SMD
 - ≻ tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

> Extension of Fractal to distributed applications

- > Asynchronous method calls
- > Implementation: ProActive[6]
 - >Active object
 - >Future value: rendez-vous



Grid Component Model (GCM)[3]

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - > GCM
 - ≻ Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- > Conclusion
- > References

> Extension of Fractal to distributed applications

- > Asynchronous method calls
- > Implementation: ProActive[6]
 - >Active object
 - >Future value: rendez-vous



>Turtle_[4] Model

- > Introduction
- > Our contribution
- > State of the art of component models
 - Fractal
 - GCM
 - Turtle
 - UML 2 and Fractal
- > CTTool
 - > overview
 - CSD
 - SMD
 - > tools
- > Language Extensions
- > Conclusion
- > References

> Ludovic Apvrille, ENST, LabSoC Laboratory

- > UML(1.4) profile dedicated to the modelling and formal validation of realtime systems
- >Formal semantics for UML
- >Set of diagrams
- > GCM/ProActive components > Implemented by TTool
 - > Analyze of possible system errors



> UML 2 components and Fractal

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - > GCM
 - > Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

> Vladimir Mencl and Matej Polak, Charles University, Prague Distributed System Research Group

Mapping from Fractal to UML 2 (no behavior)Component:

- > hierarchy / nested components
- > provided and required interfaces
- > attributes
- > Port
 - > has provided and required Interfaces
 - has multiplicity (=> collection interfaces)

> Connectors

- > Introduction > Cannot be linked to interfaces (only to ports) > Our contribution Fractal > Interfaces via Ports: GCM Turtle × \succ Only one interface per port. UML 2 and Fractal A > CTTool > Position of interface client/server. × overview CSD A
 - > Boolean attribute: mandatory or optional.
- > GCM/ProActive components

SMD > tools

- > Language Extensions
- > Conclusion

×

> References

CTTool Overview

≻ Based on UML 2

> Fractal component model

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - ≻ GCM
 - > Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

Editor + verification environment using TTool code base

- > generation of Lotos code
- > bridges to CADP toolset

Producer-Consumer Case-Study



> References

CTTool: Composite Structure Diagrams



CTTool: State Machine Diagrams



- ≻ CSD
- ≻ SMD
- > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References



CTTool: use of CADP toolbox

- > Introduction
- > Our contribution
- > State of the art of component models
 - ≻ Fractal
 - > GCM
 - > Turtle
 - > UML 2 and Fractal

> CTTool

- > overview
- > CSD
- > SMD
- > tools
- > GCM/ProActive components
- > Language Extensions
- > Conclusion

> References

Anal	ysis on the last RG (AUT format)	0
General info. Statistics Deadlock	s 🕲 Shortest Paths	
States (origin, action)	1 put> [1] b1 SEP> [4] bd1 put> [7] b1 SE<1>> [13] bd1	
45 (37, bq1_put), (38, cb1_get) [0] bq	$1_put \rightarrow [1] \rightarrow b1_str \rightarrow [4] \rightarrow bq1_put \rightarrow [7] \rightarrow b1_str(1) \rightarrow [13] \rightarrow bq1_1$ $1_put \rightarrow [1] \rightarrow b1_str \rightarrow [4] \rightarrow bq1_put \rightarrow [7] \rightarrow b1_str(1) \rightarrow [13] \rightarrow bq1_1$	
	•	
Checking LOTOS specification with CAESAR	DOTTY	
Analysis options		
Show warnings		
caesar.adt: - variables binding		
caesar.adt: - operations binding	(ichl seri) (hal nut)	
caesar.adt: - Tunctionality analysis caesar.adt: interface of ``sner"	And the second	
caesar.adt: verification of ``spec"		
caesar.adt: type survey of ``spec"	of of a	
caesar.adc compliation of "spec" caesar.adt: optimization of "spec"	i(ba2 aGet)(ba1 put) i(cb1 set)(b1 SFP) i(ba1 put)	
caesar.adt: C translation of ``spec"		
caesar.adt: indentation of ``spec"		
Analysing LOTOS specification	∇ ∇ ∇ ∇ ∇	
caesar 7.0 Hubert Garavel (INRIA Rhone-Alpes)	i(bal put) (ba2 aGet)i(b1 SFP) i(ba1 put) (cb1 set) i(ba1 put) (cb1 set) i(ba1 set) i(b1 SFP)	
raecar syntax analysis of ``sner"		
caesar: semantic analysis of ``spec"		
caesar: - processes binding		
caesar: - yates omuny caesar: - tynes hinding		$\overline{\}$
caesar: - signature analysis 😑	hor and hor an	
caesar: - sorts binding		\ \
caesar. – operations binding		
caesar: - functionality analysis		
caesar: restriction of 'spec'	i(bq1_put) i(bq2_qGet) ((b1_SF<1>) Xi(cb1_get)	
caesar. type survey of ``spec"		
caesar: generation of ``spec"		
Analysis done		
	(I(D1_SH<1>) I(DQ2_QGet)	
Start Stop Close		

Specifying GCM/ProActive Components

> Limitations

- > Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - ≻ GCM
 - > Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

- > Asynchronous method calls: queue, proxy.
- > Serving Policy
- > Multiplicity
- > Multicast / Gathercast interfaces



> A GCM/ProActive component provides:

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - > GCM
 - > Turtle
 - > UML 2 and Fractal

> CTTool

- > overview
- > CSD
- > SMD
- > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

- > Request queue
- > Service thread



Graphical Element	Representation	
Components	· · · · · · · · · · · · · · · · · · ·	
(Parameterized) Active Component	Producer p:int mo f	
Multiple interfaces	₩ <u>*</u> € ₩ * ●	
Multicast and Gathercast	<u>∎{•</u> <u>∎</u> {<	
State Machines	in Silanda Sila	
Active Component Behaviour (with local variables)	Active component Buffer Buffer (safe policy runActivity (service get() via B2(Int c)) output	
Regions, Forks and Joins		
Request on a required interface	10(R)((((0+1)4)))	
Wait for a Future Update	us+(val)	

TABLE III NEW GRAPHICAL ELEMENTS FOR ACTIVE COMPONENTS

> Introduction	behaviour of an active component
> Our contribution	
> State of the art of component models	> Region diagrams
> Fractal	
> GCM	> Sub-regions contains state machines diagrams
> Turtle	
> UML 2 and Fractal	\succ Service policy of the component
≻ CTTool	> FIFO by default
> overview > CSD	> States of the lifecycle
> SMD	> InitActivity
> tools	> RunActivity
> GCM/ProActive components	> EndActivity

the

- > Language Extensions
- > Conclusion
- > References

≻ Sub-machines

> Service methods offered by the component



Fig. 4. State machines for the Buffer component

> Multicast client interface

- \succ Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - > GCM
 - ≻ Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

- > A client interface connected to N server interfaces.
- > Gathercast server interface
 - > N client interfaces connected to a single server interface.



Conclusion

- > Introduction
- > Our contribution
- > State of the art of component models
 - ≻ Fractal
 - ≻ GCM
 - ≻ Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - ≻ CSD
 - > SMD
 - ≻ tools
- > GCM/ProActive components
- > Language Extensions
- > Conclusion
- > References

Fractal and GCM components can be specified using UML 2 diagrams for specifications of architecture and behaviour.

> The graphical specification language is formal enough to be model-checked

> CTTool tested in a large scale casestudy

- > Common Component Modelling Example
 (CoCoME)[5]
 - >16 components, 5 of them being composites
 - > 5 layers of hierarchy

Conclusion

- > Introduction
- > Our contribution
- > State of the art of component models
 - > Fractal
 - ≻ GCM
 - > Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - ≻ CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- > Conclusion
- > References

> Common Component Modelling Example (CoCoME)[5]

- > Generation of LOTOS model for modelcheking in CADP:
 - > 81 distinc transition labels
 - > Before reduction: 1.25 million states / 3
 million transitions
 - > After reduction: 9800 states / 33000
 transitions

> Basis for addressing distributed components specification

To create a new UML profile for dealing with distributed active components.

References

> Introduction

- > Our contribution
- > State of the art of component models
 - > Fractal
 - ≻ GCM
 - ≻ Turtle
 - > UML 2 and Fractal
- > CTTool
 - > overview
 - > CSD
 - > SMD
 - > tools
- > GCM/ProActive components
- > Language Extensions
- \succ Conclusion
- > References

[1] UML 2.0 Superstructure Specification, http://www.omg.org/cgibin/doc?ptc/2004-10-02, omg, Oct. 2004.

[2] E. Bruneton, T. Coupaye, M. Leclercp, V. Quema, and J. Stefani, "An open component model and its support in java." in 7th Int. Symp. On Component-Based Software Engineering (CBSE-7), ser. LNCS 3054, may 2004.

[3] OASIS team and other partners in the CoreGRID Programming Model Virtual Institute, "Basic features of the grid component model (assessed)," 2006, deliverable D.PM.04, CoreGRID, Programming Model Institute.

[4] L. Apvrille, J.-P. Courtiat, C. Lohr, and P. de Saqui-Sannes, "TURTLE: A Real-Time UML Profile Supported by a Formal Validation Toolkit," IEEE transactions on software Engineering, vol. 30, no. 7, jul 2004.

[5] "Common component modelling example (cocome)." [Online]. Available: http://agrausch.informatik.uni-kl.de/CoCoME

[6] D. Caromel, C. Delb'e, A. di Costanzo, and M. Leyton, "Proactive: an integrated platform for programming and running applications on grids and p2p systems," Computational Methods in Science and Technology, vol. 12, no. 1, pp. 69-77, 2006.