

Component-based modelling within GroIMP/XL

– Towards a “construction kit” for interactive,
visual modelling

Michael Henke, Katarína Smoleňová

Department Ecoinformatics, Biometrics and Forest Growth,
University of Göttingen, Germany

Tutorial and Workshop

”Modelling with GroIMP and XL”

combined with the 5th GroIMP user and developer meeting

Göttingen, 2012-02-28



Scientific modelling

Common definition

“Scientific modelling is the process of generating abstract, conceptual, graphical and/or mathematical models. . . . A scientific model can provide a way to read elements easily which have been broken down to a simpler form.”

Scientific modelling

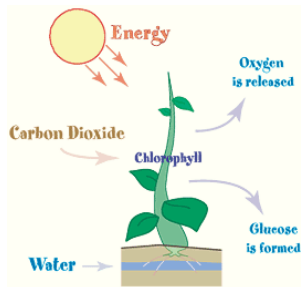
Common definition

“Scientific modelling is the process of generating abstract, conceptual, graphical and/or mathematical models. . . . A scientific model can provide a way to **read elements easily** which have been **broken down to a simpler form.**”

Scientific modelling

Common definition

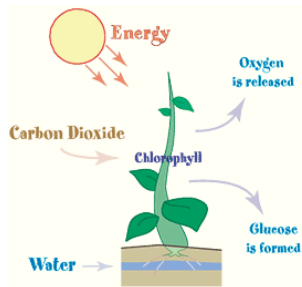
“Scientific modelling is the process of generating abstract, conceptual, graphical and/or mathematical models. . . . A scientific model can provide a way to **read elements easily** which have been **broken down to a simpler form.**”



Scientific modelling

Common definition

“Scientific modelling is the process of generating abstract, conceptual, graphical and/or mathematical models. . . A scientific model **can** provide a way to **read elements easily** which have been **broken down to a simpler form.**”



Messy code

```

1 /* Messy Source Example for Java */
2 package cryptik.examples;import
3 java.io.FileInputStream;import java.io.IOException;import cryptik.security.HDS;
4 /* This is a demo of how to use the HDS or SH0T classes for hashing data */
5 public final class HDSFile{private static final int BUF_LENGTH=1024;public
6 static void main(String argu[]){if(argu.length!=1)System.err.println(
7 "usage java HDSFile filename");byte buf=try{printHash(dofash(argu[0]);}catch(
8 IOException ioe){System.err.println
9 "There has been an IO exception to the file was not hashed.");
10 ioe.printStackTrace();}private static void printHash(byte buf[]){
11 System.out.println("hash of file is:");System.out.print("HDS: ");for(int i=0,j
12 =buf.length;i<+j--){int val=(int)buf[i];System.out.print(Integer.toString(
13 (val>>4)&buf,16));System.out.print(Integer.toString(val&buf,16));}
14 }try{System.out.println(i);}}
15 }
16 }
17 }
18 }
19 }
20 }
21 }
22 }
23 }
24 }
25 }
26 }
27 }
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
42 }
43 }
44 }
45 }
46 }
47 }
48 }
49 }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

Public Function G@Checksum(ByVal sentence As String) As String

- Dim Character As Char
- Dim Checksum As Integer
- For Each Character In sentence
- Select Case Character
 - Case "\$"
 - 'Ignore the dollar sign
 - Case "*"
 - 'Stop processing before the asterisk
 - Exit For
 - Case Else
 - 'Is this the first value for the checksum?
 - If Checksum = 0 Then
 - 'Yes. Set the checksum to the value
 - Checksum = Convert.ToByte(Character)
 - Else
 - Checksum = Checksum Xor Convert.ToByte(Character)
 - End If
- End Select
- Next
- Return Checksum.ToString("X2")
- End Function

```

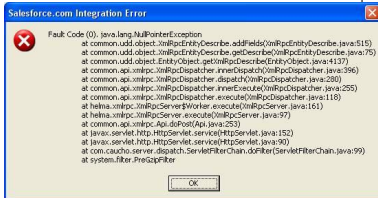
void CMyafc30bView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    CMyafc30bDoc* pDoc = GetDocument();
    if(m_tracker.HitTest(point) == CRectTracker::hitMiddle) {
        CFileDataSource* pSource = SaveDib();
        if(pSource) {
            // DoDragDrop returns only after drop is complete
            CClientDC dc(this);
            OnPrepareDC(&dc);
            CPoint topleft = m_rectTracker.TopLeft();
            dc.LPtoDP(&topleft);
            // 'point' here is not the same as the point parameter in
            // OnDragEnter, so we use this one to compute the offset
            m_dragOffset = point - topleft; // device coordinates
            pDoc->m_bDragHere = TRUE;
            DROPEFFECT dropEffect = pSource->DoDragDrop(
                DROPEFFECT_MOVE|DROPEFFECT_COPY, CRect(0, 0, 0, 0));
            TRACE("after DoDragDrop -- dropEffect = %ld\n", dropEffect);
            if (dropEffect == DROPEFFECT_MOVE && pDoc->m_bDragHere) {
                pDoc->OnEditClearAll();
            }
            pDoc->m_bDragHere = FALSE;
            delete pSource;
        }
    }
}

```

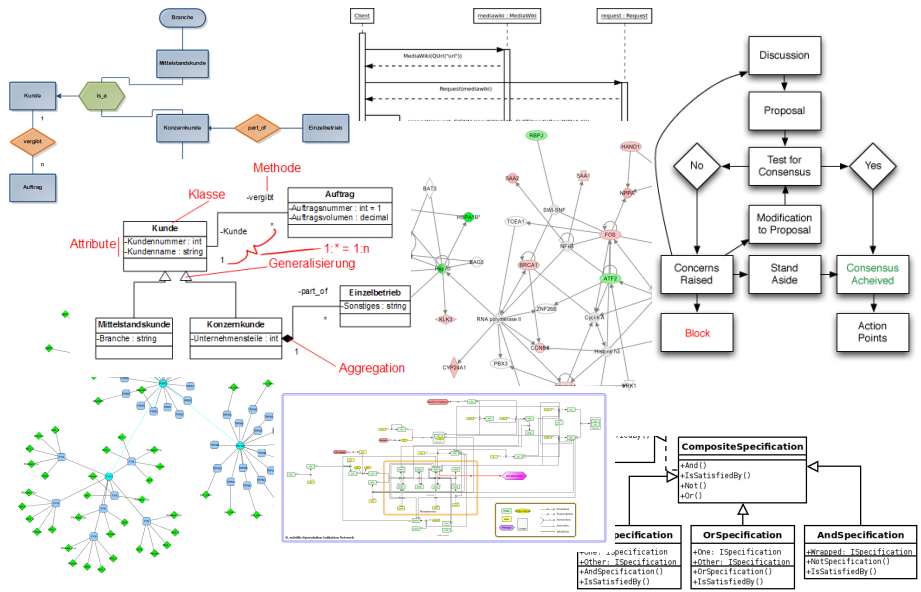
```

.Track(this, point, FALSE, NULL)) {
    C dc(this);
    eDC(&dc);
    d have some way to prevent it going out of bounds
    acker = m_tracker.m_rect;
    F(m_rectTracker); // Update logical coords
}

```



Software engineering tools



Reasons

- Knowledge is all on the Web
 - distributed
 - very time consuming
- Justifiably sceptical
 - several “great innovations” - without noticeable improvements
- Frustrated by the complexity
 - stick to lowest common denominators

Reasons

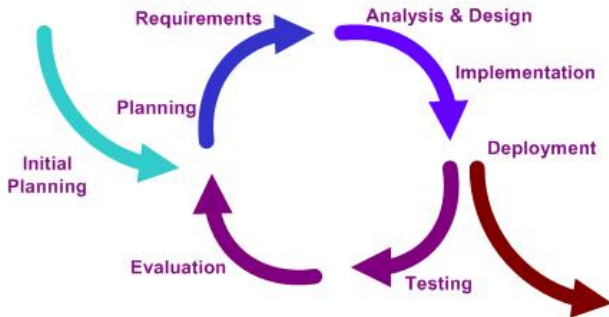
- Knowledge is all on the Web
 - distributed
 - very time consuming
- Justifiably sceptical
 - several “great innovations” - without noticeable improvements
- Frustrated by the complexity
 - stick to lowest common denominators

Reasons

- Knowledge is all on the Web
 - distributed
 - very time consuming
- Justifiably sceptical
 - several “great innovations” - without noticeable improvements
- Frustrated by the complexity
 - stick to lowest common denominators

Modelling process

... or “Modeller’s treadmill”



⇒ the **real limit** on what computational scientists can accomplish is **how quickly and reliably** they can **translate their ideas into working code**

Summarize the initial situation

- Modelling is often based on programming
 - Requires programming knowledge / skills
 - → Concentration is divided: modelling vs. programming
- Developer with:
 - No or little programming experience → challenge of programming
 - Little knowledge of biological systems → challenge of modelling
- Common way of modelling:
 - *Ad hoc*
 - Usually starting "from scratch"
 - (initially) no (clear) concept/design
 - Unsystematic
 - Extend/change existing models
 - Getting overview of code more difficult
- No reuse \implies "Reinventing the wheel"

Aim

Goals:

- Low entry threshold: make access simpler
- Reuse of software (parts)
 - independently developed
 - development by experts
 - periodic maintenance
- Visual support

Side effects:

- Reduction of time for a model developer to get familiar with a system / language
- Transparent and flexible modelling process and models
- Models that can be evaluated / combined
- Models become comparable
- Enhancement of quality
- Faster model development
- Communication between modeller and experimentator is facilitated

Aim

Goals:

- Low entry threshold: make access simpler
- Reuse of software (parts)
 - independently developed
 - development by experts
 - periodic maintenance
- Visual support

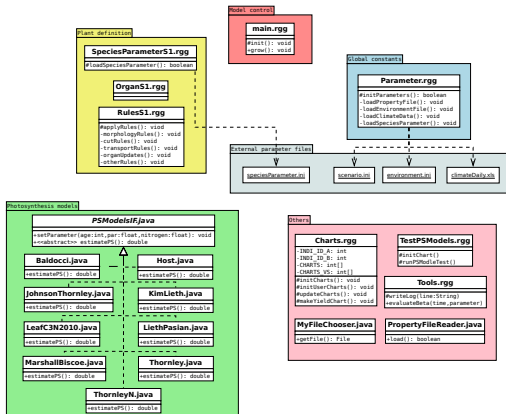
Side effects:

- Reduction of time for a model developer to get familiar with a system / language
- Transparent and flexible modelling process and models
- Models that can be evaluated / combined
- Models become comparable
- Enhancement of quality
- Faster model development
- Communication between modeller and experimentator is facilitated

Current approaches

- Modularization
- Object-orientation

Structure of the FSPM-Prototype^a:



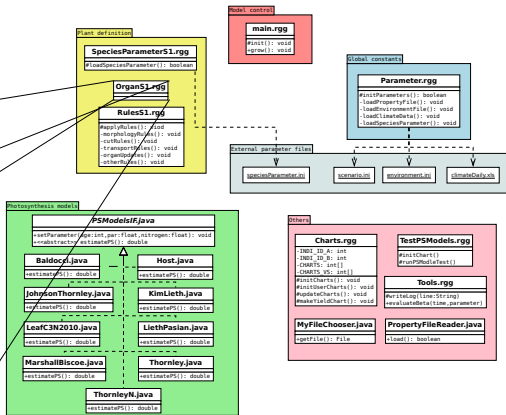
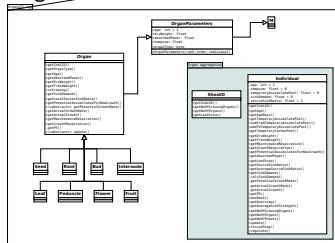
^aM.Henke and GH. Buck-Sorlin unpubl., 2010

Current approaches

- Modularization
- Object-orientation

Structure of the FSPM-Prototype^a:

Organ structure:



^aM.Henke and GH. Buck-Sorlin unpubl., 2010

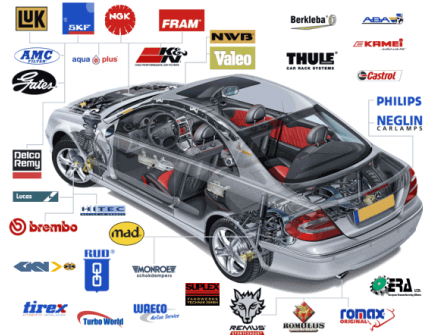
Component-based engineering

- Well-known engineering technique

Pre-assembled components



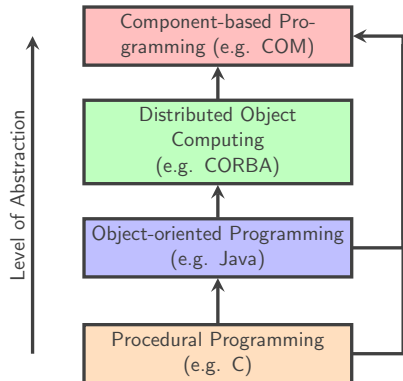
Different, independent subcontractors



Component-based software engineering

- Starts in 1968
- Early 1990s: IBM - System Object Model
- Microsoft: OLE, COM
- Today: many successful software component models exist
 - Microsoft (.NET, COM, DCOM, OLE, ActiveX, COM+)
 - Object Management Group (CORBA)
 - Sun Microsystems (JavaBeans, Servlets, Applets, Enterprise JavaBeans)
 - OSGi Alliance (OSGi)

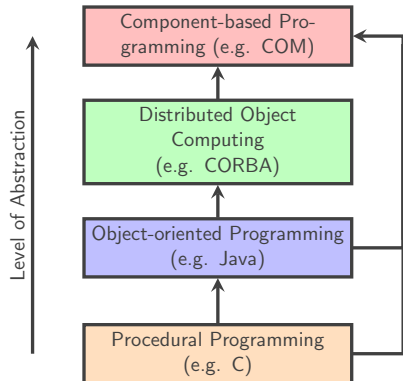
⇒ dividing SW into re-usable parts



Component-based software engineering

- Starts in 1968
- Early 1990s: IBM - System Object Model
- Microsoft: OLE, COM
- Today: many successful software component models exist
 - Microsoft (.NET, COM, DCOM, OLE, ActiveX, COM+)
 - Object Management Group (CORBA)
 - Sun Microsystems (JavaBeans, Servlets, Applets, Enterprise JavaBeans)
 - OSGi Alliance (OSGi)

⇒ dividing SW into re-usable parts



Component - Definition

- “A software component is a **unit of composition** with contractually specified **interfaces** and **explicit context dependencies** only. A software component can be **deployed independently** and is subject to **compositions by third parties.**” (C. Szyperski)
- “A software component is a **software element**, that can be **connected, used and executed** without changes at any other components according to a **component model** and corresponding to **composition standard.**” (W.T. Councill)
- “A small **binary object** or **program** that performs a **specific function** and is designed in such a way to easily **operate with other components** and applications.” www.webopedia.com
- “A component consists of diverse (**software-**) **artefacts**. It is **re-usable, closed** and **marketable**, provides **services** via **well-defined interfaces, covert her implementation** details and be used in **combination** with other components, ...” (Gesellschaft für Informatik)

Component - Requirements

Basic requirements:

- useful functionality
- strict encapsulation
- re-usability
- a component can provide a self description
- distributed in binary form
- configurable, no persistent state

Additional requirements:

- platform-independent
- location-independent (distributed)
- independent of programming languages
- ready for integration and communication (plug-and-play)
- independent deployment
- GUI aided design

Component - Requirements

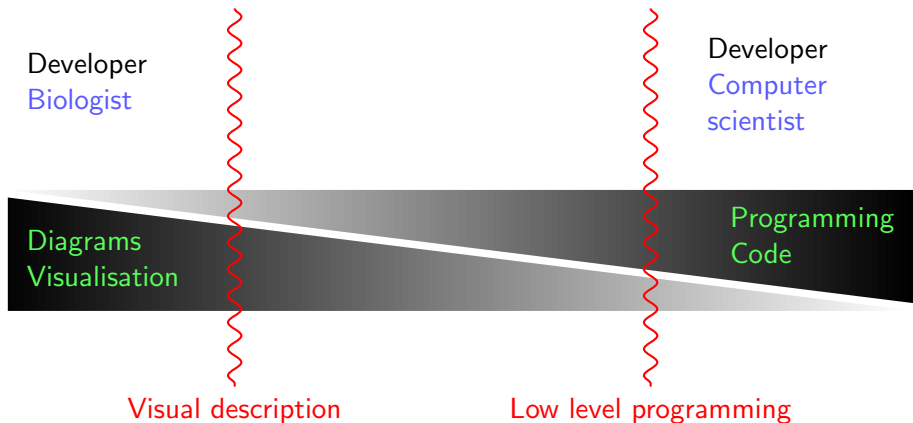
Basic requirements:

- useful functionality
- strict encapsulation
- re-usability
- a component can provide a self description
- distributed in binary form
- configurable, no persistent state

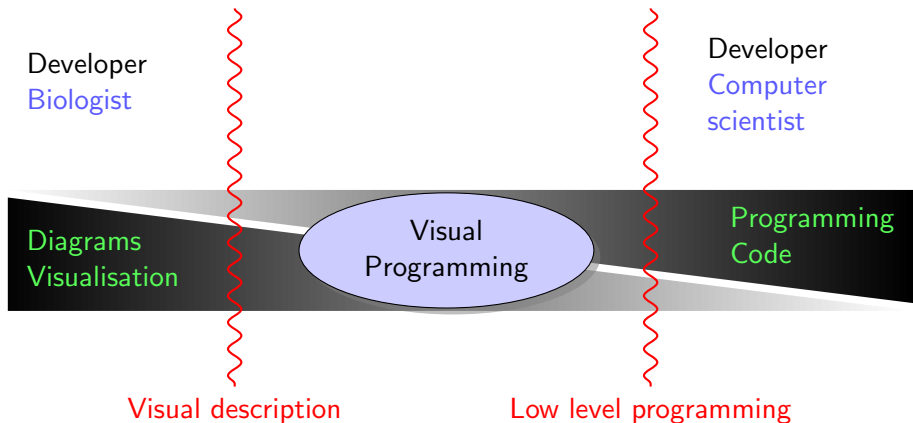
Additional requirements:

- platform-independent
- location-independent (distributed)
- independent of programming languages
- ready for integration and communication (plug-and-play)
- independent deployment
- GUI aided design

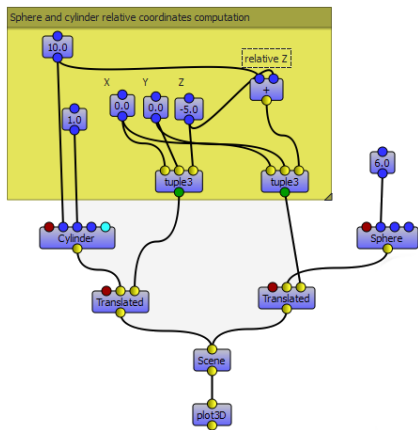
Actors - Tools - Level of Abstraction



Actors - Tools - Level of Abstraction



Visual programming – OpenAlea-like

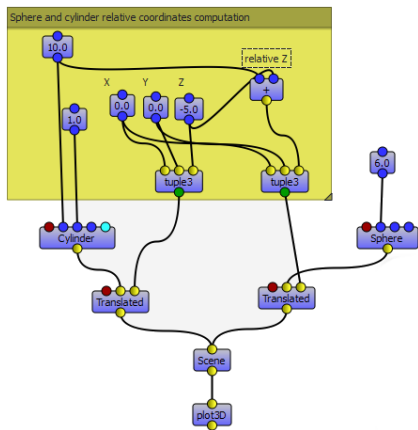


Mathematical calculations:

- atomic elements for numbers and operations
- visual low level programming with a “quasi” one-to-one translation of code to visual objects
- would only shift problems (and add some we did not have previously?!)

⇒ NO equivalent within VisualGroIMP

Visual programming – OpenAlea-like



Mathematical calculations:

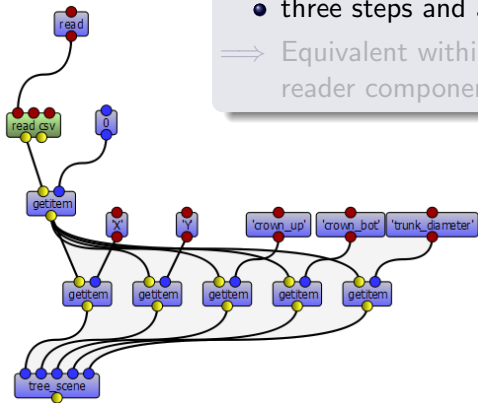
- atomic elements for numbers and operations
- visual low level programming with a “quasi” one-to-one translation of code to visual objects
- would only shift problems (and add some we did not have previously?!)

⇒ NO equivalent within VisualGroIMP

Visual programming – OpenAlea-like

File input:

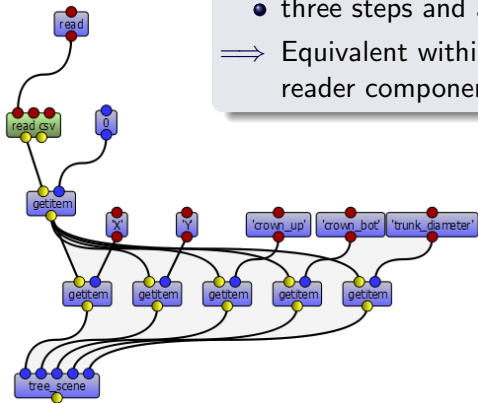
- three steps and a loop are needed to read a file
- ⇒ Equivalent within VisualGroIMP: generic file reader component



Visual programming – OpenAlea-like

File input:

- three steps and a loop are needed to read a file
 ⇒ Equivalent within VisualGroiMP: generic file reader component



World

TreeScene

<<file>>
 FileReader

VisualGroIMP - Project

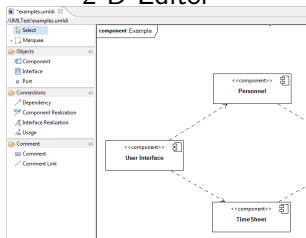
User manual



Library



2-D Editor



VisualGroIMP - Current state

GroIMP Screenshot

The screenshot displays the VisualGroIMP software interface. The main workspace shows a diagram with two yellow rectangular boxes: "LeafOrgan" at the top left and "Kim Lieth" at the bottom right. A green dashed arrow points from "LeafOrgan" to "Kim Lieth".

On the right side, the "Component Explorer" panel is visible, showing a tree view of components:

- components
 - Objects
 - Processes
 - PhotosynthesisModel
 - Thornley [2011.06.07]
 - Kim Lieth [2011.06.07]
 - LeafCN [2011.06.07]
 - Lieth Pasion [2011.06.07]
 - Respiration
 - Respiration [2011.06.07]
 - ReimplementedModels

Below the tree view, the "Component Description" panel provides details for the selected component:

Name	: Kim Lieth
Version	: 2011.06.07
Author	: me
Description	: Implementation of Kim Lieth
Input slots	: int time [h]
Output slots	: no
Conditions	: no rain

The path for the component is shown as `/Processes/PhotosynthesisModel/KimLieth.zip`.

VisualGroIMP - Current state

GroIMP Screenshot

The screenshot displays the VisualGroIMP software interface. The main workspace shows a component diagram with two components: "LeafOrgan" (a white rounded rectangle) and "KimLieth" (a yellow rounded rectangle). The "LeafOrgan" component is positioned above and to the left of the "KimLieth" component. The "KimLieth" component is highlighted with a yellow background.

The Component Explorer on the right side of the interface shows the following tree structure:

- components
 - Objects
 - Processes
 - PhotosynthesisModel
 - Thornley [2011.06.07]
 - Kim Lieth [2011.06.07]
 - LeafC3N [2011.06.07]
 - Lieth Pasiian [2011.06.07]
 - Respiration
 - Respiration [2011.06.07]
 - ReimplementedModels

The Component Description panel at the bottom right provides the following details for the selected component:

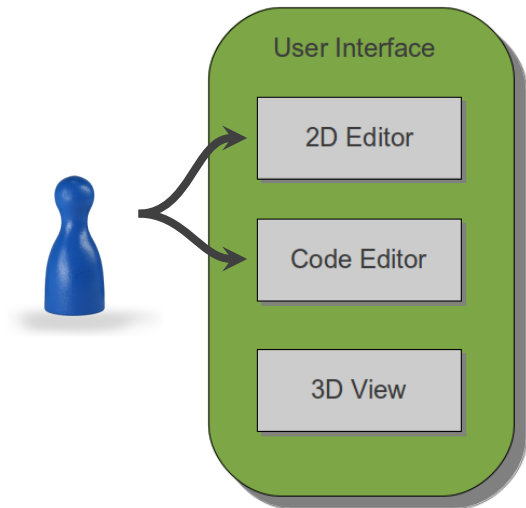
Component Description	
Name	: Kim Lieth
Version	: 2011.06.07
Author	: me
Description	: Implementation of Kim Lieth
Input slots	: int time [h]
Output slots	: no
Conditions	: no rain

The file path for the component is shown as `/Processes/PhotosynthesisModel/KimLieth.zip`.

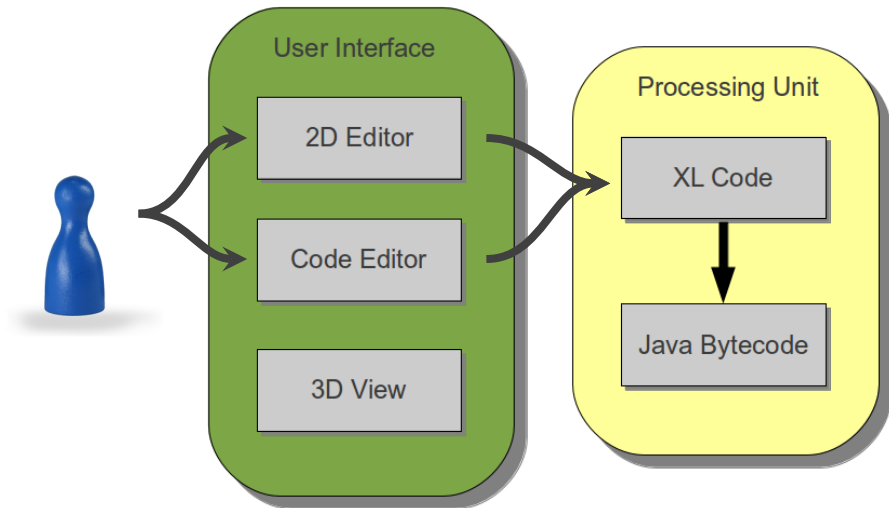
VisualGrolMP - Workflow



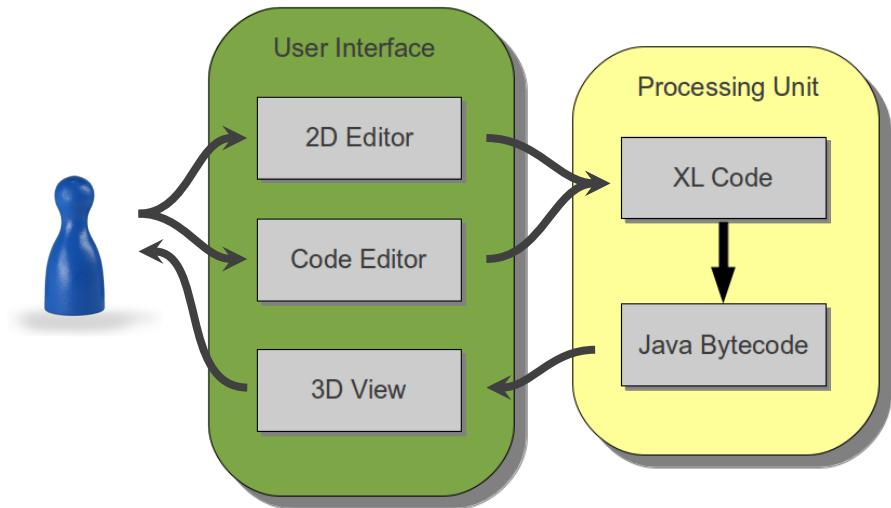
VisualGrolMP - Workflow



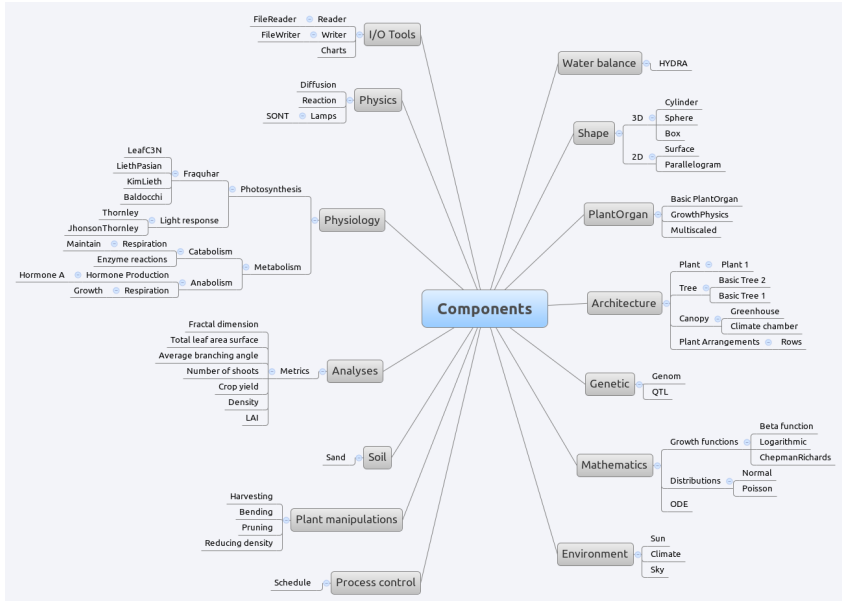
VisualGroIMP - Workflow




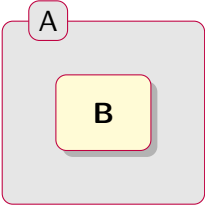
VisualGroIMP - Workflow



Component library



Connectors

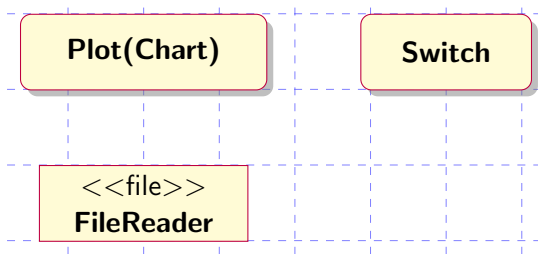
Relation	Edge type	Graph
Usage	$\sim >$	
Composition	$/>$	

Connectors

Relation	Edge type	Graph
Slot	-\$->	
Send	-s->	

Tool components

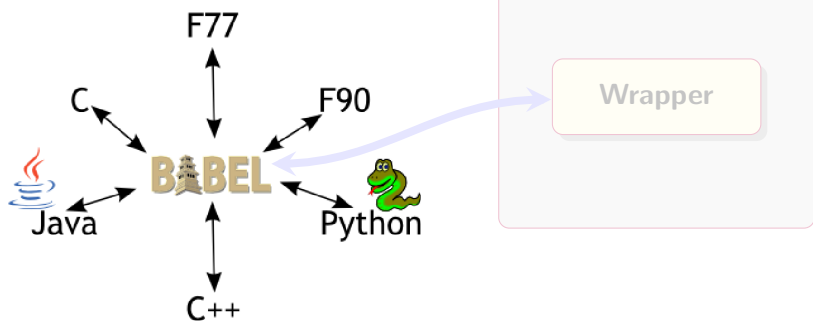
“Re-useable” components



Wrapper Components for Multi-Language Scientific Software

Babel tool:

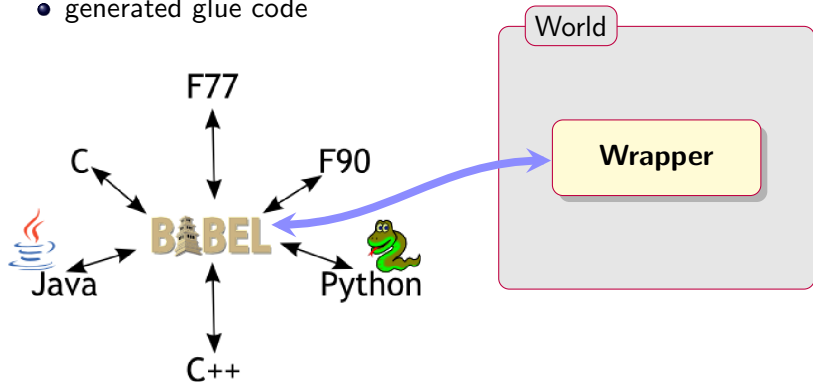
- to make scientific software libraries equally accessible from all of the standard languages
- generated glue code



Wrapper Components for Multi-Language Scientific Software

Babel tool:

- to make scientific software libraries equally accessible from all of the standard languages
- generated glue code



Team

Project leader : W. Kurth

Project member : K. Smoleňová, M. Henke

Co-worker /

Student assistant : Wu Shining, Ding Cong, Ong Yongzhi

Consulting : R. Hemmerling

Thank you for your attention!



References



William T. Councill, George T. Heineman

Component-Based Software Engineering: *Addison-Wesley*, 2001, ISBN 0-201-70485-4



Clemens Szyperski

Component Software, Beyond Object-Oriented Programming: *Addison-Wesley, London*, 2002, ISBN 0-201-74572-0



Greg Wilson

Where's the Real Bottleneck in Scientific Computing?: *American Scientist*, January-February 2006, Volume 94, Number 1, Page: 5, DOI: 10.1511/2006.1.5