### UMAP: A Universal Layer for Schema Mapping Languages The Implementation

Florin Chertes

Technische Universität Wien, Vienna, Austria Institut für Informationssysteme FlorinChertes@acm.org

#### 1 Introduction

UMAP a new universal layer for schema mapping languages which provides a unified abstraction and middleware for high-level visual mapping languages was presented in [1]. The main Tools used for the UMAP implementation were:

- as operating system (OS), Windows XP (WIN32),
- as UML modeling tool, the well known and appreciated product Sparx Systems' Enterprise Architect 9.1 (EA) and
- as integrated development environment (IDE) for C++, the product Microsoft Visual C++ 2010 Express (MSVC).

No specific WIN32, EA or MSVC features were used making the implementation OS, modeling tool and IDE independent. The implementation is dependent on standard XML, standard UML-OCL and the target programming languages like standard C++11. At the same time the C++ features used are limited to those that are to be found in languages like Java and XQuery.

First we designed with the help of the modeling tool the classes representing the source, the target and the mapping. We augmented them with the functions for reading and writing the data from and to the XML files and the functions for the mapping. The mapping functions were further defined with the help of the OCL expressions. The model was saved in a file of type EAP (Enterprise Architect Project).

Second using the modeling tool we generated the definition and the implementation files of these classes. The programming language C++ uses for each class two files: a header file defining the class and an implementation file for the member functions of the class. The mapping function were implemented using the OCL expressions as definition.

Third, using the IDE we created a project, a MSVC specific artifact with a main function, a C++ specific artifact. The project was saved in a file of type SLN (Microsoft Visual Studio Solution). We added the generated files to the project. The generated files are OS (Windows, Linux) and IDE (MSVC, Eclipse) independent, this means that we could as well choose on Linux the Eclipse as IDE. The MSVC IDE could immediately create an executable. The main program was extended to use the topmost classes from the model: the source, the target and the mapping class.

Last, the implementation was executed. In this way the input file, a XML file defining the source, used in the CLIP presentation, was transform in another XML file, the target, defined by the mapping.

### 2 Implementation of the CLIP feature: Mapping with Context Propagation

All the files, implementing the classes from the diagram Fig. 1 (presented in [1, Fig. 2]) are stored in the archive file ex205\_reg\_emp.zip, Fig. 2. Each class from the class diagram hat its implementation files.



Fig. 1: Mapping with Context Propagation, the class diagram

ex205_reg_emp				
Datei Bearbeiten Ansicht Eavoriten Extras	2			<u></u>
🌀 Zurück 🝷 🕥 - 🎓 🖉 Suchen 💫 O	rdner 🛄 🔻			
Adresse 🛅 E:\users\chfl\ms100_app\div\ex205_reg	_emp\ex205_reg_emp			💌 🔁 Wechseln zu
Ordner	× Name A	Größe	Тур	Geändert am
🗉 🚞 ex141_yami_cs_lib	Builder	1 KB	C++ Source	21.12.2011 21:07
🗉 🧰 ex142 yami calculator	- <u>M</u> Builder	1 KB	C/C++ Header	14.12.2011 20:59
🗷 🦳 ex143. winsork	epartament	1 KB	C++ Source	21.12.2011 20:53
ex152 varia subscription	in departament	1 KB	C/C++ Header	21.12.2011 20:46
a autre dana d	departamentSet	I KB	C++ Source	21.12.2011 20:48
E C extoo_uleau	in departamentset	1 KB	C/C++ Header	21.12.2011 20:03
🗉 🧰 ex202_tirma	epartaments-ile	1 KB	XML-Dokument	18.03.2012 17:15
🗷 🗀 ex203_reg_emp	endept	1 KB	C++ Source	19.12.2011 22:24
📧 🚞 ex204_reg_emp	in dept	1 KB	C/C++ Header	19.12.2011 22:24
🖃 🚞 ex205_reg_emp	aeptsuider	1 KB	C++ Source	18.12.2011 14:33
😂 ex205_req_emp	in deptsuider	1 KB	C/C++ Header	18.12.2011 13:55
Ŧ 🧰 ex206 reg emp	Совремар	1 KB	C++ Source	19.12.2011 22:32
= <u>ex207</u> reg.emp	in deptiviap	1 KB	C/C++ Header	18.12.2011 13:08
Dobug	deptSet	2 KB	C++ Source	19.12.2011 21:10
	in deptset	I KB	VAL Deleverent	10.02.2011 22:38
ereg_emp	i depts-rie	1 KB	XML-Dokument	16.03.2012 10:53
Debug	employee	1 KB	C++ Source	16.11.2011 23:12
🗄 🧰 ipch	in employee	I KB	C/C++ Header	16.11.2011 22:51
🖃 🚞 ex208_reg_emp	employeeset	1 KB	C++ Source	21.12.2011 21:03
🚞 Debug	m employeeset	I KB	C/C++ Header	21.12.2011 20:43
🖃 🚞 ex208_req_emp	ex205_reg_emp	1 KB	C++ SUURCE	18.03.2012 18:52
Con Debug	ex205_reg_emp	2.220 KB	Enterprise Architect Repusitory	10.03.2012 17:23
🗉 🤭 inch	ex205_reg_emp	J KB	Vieual Ctudio Project Llear Ontione file	10.03.2012 10.39
	ReadMo	vcxproj IKB	Tostdokumont	13 11 2011 20:59
a constant	E Reduine	2 KB	Cht Course	14 12 2011 21:37
⊞ ex≥to_reg_emp	M regEmp	1 KB	C/C++ Header	14.12.2011.21.37
I in factory	Collegemp CollegempCot	I ND	Ctt Fourse	10 12 2011 21:37
I CATTCP	h rogEmpSet	1 / 8	C/C++ Hoodor	10 12 2011 21:16
🗉 🚞 loki_app	CHValueMan	1 KB	C++ Source	14 12 2011 20:49
🗉 🚞 tri_logger	NalueMap	1 /0	C/C++ Hooder	14 12 2011 20:40
🗉 🧰 yami4	- Valacinap	110	c)ci i i ilidaddi	11.12.2011 20.10
🗉 🚞 yami4-gpl-1.2.1				
🖃 🧰 visio				
ex205 reg emp				
ex207 reg emp				
av209 reg emp				
- excuolieg_emp	~			

Fig. 2: Mapping with Context Propagation, the implementation files

The main function, Fig. 3 triggers the mapping. It uses:

- an object of the class *deptSet* named aDeptSet representing the source,
- an object of the class deptBuilder named builder representing the mapping and
- an object of the class *departmentSet* named aDepartmentSet representing the target.

The lines 11-12 create an instance of the source and read the input from the XML input file, Fig. 4. The lines 14-15 create an instance of the mapping object and use it to map the source to the target. The line 15 executes the mapping. The mapping object, using the function *build* takes as argument the source-object and produces the target-object. The line 17 writes the target-object to an XML output file, Fig. 5.



Fig. 3: Mapping with Context Propagation, the main function



Fig. 4: Mapping with Context Propagation, the input



Fig. 5: Mapping with Context Propagation, the output

# 3 Implementation of the CLIP feature: Mapping with Join

All the files, implementing the classes from the diagram Fig. 6 (presented in [1, Fig. 4]) are stored in the archive file ex207\_reg\_emp.zip, Fig. 7. Each class from the class diagram hat its implementation files.



Fig. 6: Mapping with Join, the class diagram

ex207_reg_emp					
Datei Bearbeiten Ansicht Eavoriten Extras	2				
🌀 Zurück 👻 🛞 - 🎓 🖉 Suchen 🜔 C	Ordn	er 🛄 🕶			
Adresse 🗁 E:\users\chfl\ms100_app\div\ex207_reg	g_er	np\ex207_reg_emp			👻 ラ Wechseln zu
Ordner	×	Name 🔺	Größe	Тур	Geändert am
a 🕞 mc90, ann		eddept	1 KB	C++ Source	26.12.2011 19:02
🖬 🛄 misoo_app	-	h dept	1 KB	C/C++ Header	26.12.2011 19:02
		edeptMap	2 KB	C++ Source	23.01.2012 21:39
		b) deptMap	1 KB	C/C++ Header	23.01.2012 20:49
🗉 🧰 cf		edeptSet	2 KB	C++ Source	26.12.2011 21:13
🗉 🧰 date_time		b deptSet	1 KB	C/C++ Header	26.12.2011 13:33
🖃 🧰 div		📺 deptsFile	2 KB	XML-Dokument	23.01.2012 21:12
🗷 🚞 ex87_gregorian_date		ex207_reg_emp	2 KB	C++ Source	18.03.2012 14:24
🗉 🚞 ex138_logger		🕺 ex207_reg_emp	2.242 KB	Enterprise Architect Repository	18.03.2012 14:42
🗑 🦳 ex138 logger t		@ex207_reg_emp	5 KB	VC++ Project	18.03.2012 14:03
		eteroj	1 KB	C++ Source	05.01.2012 21:09
P Co ev141 vami cs		h Proj	1 KB	C/C++ Header	05.01.2012 21:09
a autit uppi se lie		proj_emp_tile	1 KB	XML-Dokument	18.03.2012 14:24
		emproject_emp	1 KB	C++ Source	06.01.2012 19:59
ex142_yami_calculator		h project_emp	1 KB	C/C++ Header	06.01.2012 19:35
🗷 🧰 ex143_winsock		∰project_empSet	2 KB	C++ Source	23.01.2012 21:07
🗷 🚞 ex153_yami_subscription		http://project_empSet	1 KB	C/C++ Header	23.01.2012 21:07
표 🚞 ex155_thread		mproject_empSetBuilder	1 KB	C++ Source	18.03.2012 13:58
😠 🚞 ex202_firma		h project_empSetBuilder	1 KB	C/C++ Header	18.03.2012 14:00
🗑 🦳 ex203 reg. emp		ProjSet	2 KB	C++ Source	05.01.2012 21:11
🕀 🦳 ex204 reg. emp		h ProjSet	1 KB	C/C++ Header	26.12.2011 12:46
P C ev205 reg emp		e projsFile	1 KB	XML-Dokument	05.01.2012 21:43
a checker of comp		E ReadMe	2 KB	Textdokument	13.11.2011 20:58
🗄 📴 ex200_reg_emp		egemp	1 KB	C++ Source	18.03.2012 12:39
e cave and the second s		h regEmp	1 KB	C/C++ Header	06.01.2012 19:58
🖨 🖾 ex207_reg_emp		gregEmpSet	3 KB	C++ Source	18.03.2012 12:39
🖃 🚞 ex208_reg_emp		h regEmpSet	1 KB	C/C++ Header	18.03.2012 12:39
🚞 ex208_reg_emp			1 KB	XML-Dokument	05.01.2012 20:34
🗷 🚞 ex209_reg_emp					
🗷 🚞 ex210_req_emp	-				
😠 🦳 factory					
F CATTCP					
🖼 🛄 loki ann					
a 👝 ioki_app					
🖽 🛄 tri_logger					
🖬 🥅 yami4					
🕀 🚞 yami4-gpi-1.2.1	~				

Fig. 7: Mapping with Join, the implementation files

The main function, Fig. 8 triggers the mapping. It uses:

- an object of the class *deptSet* named aDeptSet representing the source,
- an object of the class  $project\_empBuilder$  named builder representing the mapping and
- an object of the class  $project\_empSet$  named a Project\\_empSet representing the target.

The lines 10-11 create an instance of the source and read the input from the XML input file, Fig. 9. The lines 13-14 create an instance of the mapping object and use it to map the source to the target. The line 14 executes the mapping. The mapping object, using the function *build* takes as argument the source-object and produces the target-object. The line 16 writes the target-object to an XML output file, Fig. 10.



Fig. 8: Mapping with Join, the main function

🖬 E:	lusers	\chfl\ms100_app\div\ex207_r 🔳	
Eile	Edit Sea	arch ⊻iew Encoding Language Settings Macro F	Run
Plugin	s <u>W</u> indo	2 wc	Х
		à 🕞 🕞 😹 🖌 🕞 🛅 🗦 C 📾 🧏 🤏 😪	🖪 »
📄 dep	otsFile.xml	🖶 proj_emp_file.xml 📄 ex207_reg_emp.cpp	
1	E <so< td=""><td>urce&gt;</td><td>^</td></so<>	urce>	^
2	¢.	<dept></dept>	
3		<name>ICT</name>	
4	白	<proj></proj>	
5		<pid>001</pid>	
6		<pre><pname>Appliances</pname></pre>	
7	-		
8	白	<proj></proj>	
9		<pid>002</pid>	
10		<pre><pname>Robotics</pname></pre>	
11	-		-
12	白	<regemp></regemp>	
13		<pid>001</pid>	
14		<ename>John Smith</ename>	
15		<sal>10000</sal>	
16		<age>32</age>	
17	-		
18	白	<regemp></regemp>	
19		<pid>001</pid>	
20		<ename>Andrew Clarance</ename>	
21		<sal>12000</sal>	
22		<age>56</age>	
23	-		
24	₽	<regemp></regemp>	
25		<pid>002</pid>	
26		<ename>Mark Tane</ename>	
27		<sal>10500</sal>	
28		<age>60</age>	
29	-		
30	P	<regemp></regemp>	
31		<pid>002</pid>	
32		<ename><b>Jim Belish</b></ename>	
33		<sal>11000</sal>	
34		<age>34</age>	
3.5	-		
36	1		
3.7	F	<dept></dept>	100
38		<name>Marketing</name>	
Ln : 1	Col : 1	Sel : 0 Dos\Windows ANSI	INS

Fig. 9: Mapping with Join, the input



Fig. 10: Mapping with Join, the output

# 4 Implementation of the CLIP feature: Mapping with Join and Grouping

All the files, implementing the classes from the diagram Fig. 11 (presented in [1, Fig. 5]) are stored in the archive file ex208\_reg\_emp.zip, Fig. 12. Each class from the class diagram hat its implementation files.



Fig. 11: Mapping with Join and Grouping, the class diagram

ex208_reg_emp					
Datei Bearbeiten Ansicht Eavoriten Extras	2				<b></b>
🔇 Zurück 🔹 🕥 - 🎓 🔎 Suchen 📂 O	dner 🛄 🔹				
Adresse Ethuserstabilities 100 ann/divtey209 reg	empley208 reg emp				Wacheala 7
Ordnor	Name A	Größe	Tvn	Geändert am	
		1 KB	C++ Source	26.12.2011 19:02	
🕀 🧰 ex141_yami_cs_lib	indept	1 KB	C/C++ Header	26.12.2011 19:02	
🗄 🚞 ex142_yami_calculator	e-1 deptMap	2 KB	C++ Source	23.01.2012 21:39	
🗟 🚞 ex143_winsock	b deptMap	1 KB	C/C++ Header	08.01.2012 15:23	
🗷 🚞 ex153_yami_subscription	endeptSet	2 KB	C++ Source	26.12.2011 21:13	
🗉 🚞 ex155_thread	b deptSet	1 KB	C/C++ Header	26.12.2011 13:33	
🗉 🦳 ex202 firma	e deptsFile	2 KB	XML-Dokument	08.01.2012 17:57	
E Carlos emp	employee	1 KB	C++ Source	16.11.2011 23:12	
a a cx205_reg_emp	m employee	1 KB	C/C++ Header	16.11.2011 22:51	
a association and a second	employeeSet	1 KB	C++ Source	08.01.2012 00:47	
🗷 🧰 ex2u5_reg_emp	n employeeSet	1 KB	C/C++ Header	08.01.2012 00:46	
🗄 🧰 ex206_reg_emp	enex208 reg emp	2 KB	C++ Source	18.03.2012 13:02	
🖃 🧰 ex207_reg_emp	@lex208 reg emp	2.242 KB	Enterprise Archit	18.03.2012 14:40	
🚞 Debug	mex208 reg emp	5 KB	VC++ Project	18.03.2012 13:24	
🖃 🚞 ex207_req_emp	C+1Proj	1 KB	C++ Source	05.01.2012 21:09	
🦰 Debug	Proi	1 KB	C/C++ Header	05.01.2012 21:09	
III 🚞 inch	e-project	1 KB	C++ Source	08.01.2012 17:49	
	b project	1 KB	C/C++ Header	08.01.2012 17:49	
e excoolleg_emp	project empSet	1 KB	C/C++ Header	06.01.2012 19:59	
ex208_reg_emp	project group by na	me 1 KB	XML-Dokument	18.03.2012 14:25	
	en projectSet	3 KB	C++ Source	08.01.2012 16:35	
🕀 🧰 ex210_reg_emp	h projectSet	1 KB	C/C++ Header	08.01.2012 15:32	
🗉 🧰 factory	- projectSetBuilder	1 KB	C++ Source	08.01.2012 14:41	
🗷 🚞 PCATTCP	b projectSetBuilder	1 KB	C/C++ Header	08.01.2012 01:55	
🗉 🚞 loki app	ProiSet	2 KB	C++ Source	05.01.2012 21:11	
🗊 🗁 tri logger	ProjSet	1 KB	C/C++ Header	26.12.2011 12:46	
🗏 🧰 vami4	ProjsFile	1 KB	XML-Dokument	05.01.2012 21:43	
a Carrierant-1 2 1	🗊 ReadMe	2 KB	Textdokument	13.11.2011 20:58	
a janier gpri.2.1	en]reaEmp	1 KB	C++ Source	05.01.2012 20:51	
	h regEmp	1 KB	C/C++ Header	06.01.2012 19:58	
ex205_reg_emp	edreqEmpSet	3 KB	C++ Source	05.01.2012 20:52	
🚞 ex207_reg_emp	h reqEmpSet	1 KB	C/C++ Header	26.12.2011 21:10	
ex208_reg_emp	reqEmpsFile	1 KB	XML-Dokument	05.01.2012 20:34	
🗄 🚞 Programme					
🗉 🧰 temp					
🕫 🔂 Systemsteuerung					
Gemeinsame Dokumente					
	~				

Fig. 12: Mapping with Join and Grouping, the implementation files

The main function, Fig. 13 triggers the mapping. It uses:

- an object of the class *deptSet* named aDeptSet representing the source,
- an object of the class projectSetBuilder named builder representing the mapping and
- an object of the class *projectSet* named aProjectSet representing the target.

The lines 11-12 create an instance of the source and read the input from the XML input file, Fig. 14. The lines 14-15 create an instance of the mapping object and use it to map the source to the target. The line 15 executes the mapping. The mapping object, using the function *build* takes as argument the source-object and produces the target-object. The line 18 writes the target-object to an XML output file, Fig. 15.



Fig. 13: Mapping with Join and Grouping, the main function

🖬 E:	lusers	\chfl\ms100_app\div\ex208_r	
Eile I	<u>E</u> dit <u>S</u> ea	arch ⊻iew Encoding Language Se <u>t</u> tings Mac	ro Run
Plugin:	s <u>W</u> indo	w 2	Х
		) 🕞 🕞 🚽 🖓 🕩 👘 ⊃ C 👘 🍢 🤏	🔫 »
😑 dep	otsFile.xml	📄 project_group_by_name.xml 📔 ex208_reg_emp.cpp	
1	Ekso	urce>	^
2	E	<dept></dept>	
3	T	<name>ICT</name>	
4	ė.	<proj></proj>	
5		<pid>001</pid>	
6		<pre><pname>Appliances</pname></pre>	
7	-		
8	白	<proj></proj>	
9		<pid>002</pid>	
10		<pre><pname>Robotics</pname></pre>	
11	-		
12	₽	<regemp></regemp>	
13		<pid>001</pid>	
14		<ename>John Smith</ename>	
15		<sal>10000</sal>	
16		<age>32</age>	
17	_		
18	F	<regemp></regemp>	
19		<pid>001</pid>	
20		<ename>Andrew Clarance<td>e&gt;</td></ename>	e>
21		<sal>12000</sal>	
22		<age>56</age>	_
23	1		
24	F	<regump></regump>	
25			
20			
28			
20	-		
30	L	<regemp></regemp>	
31	T	<pre><nid>002</nid></pre>	
32		<pre><pre><pre><pre><pre>dim Belish</pre></pre></pre></pre></pre>	
33		<sal>11000</sal>	
34		<age>34</age>	
35	-		
36	-		
37	E .	<dept></dept>	
38	T	<name>Marketing</name>	~
1 00	Colut.		
Ln : 1	Col : 1	Sel : U Dos/Windows ANSI	INS

Fig. 14: Mapping with Join and Grouping, the input

🖬 E:	users	\chfl\ms100_app\div\ex208_r 📰 🗖	
Eile I	Edit <u>S</u> ea	arch <u>V</u> iew Encoding <u>L</u> anguage Se <u>t</u> tings Macro Ri	un
Plugin	s <u>W</u> indo	2 wc	Х
I Dan			>>
dep	tsFile.xml	project_group_by_name.xml	
1	Ę	<project></project>	~
2		<name>Appliances</name>	
3	白	<employee></employee>	
4		<name>John Smith</name>	
5	-		
6	白	<employee></employee>	
7		<name>Andrew Clarance</name>	
8	F		
9	¢.	<employee></employee>	
10		<name>Mark Tane</name>	
11	-		
12	L		
13	E	<project></project>	
14		<name>Robotics</name>	
15	d d	<employee></employee>	
16		<name>Mark Tane</name>	
17	-		
18	E E	<employee></employee>	
19		<name>Jim Belish</name>	
20	-		
21	Ļ		
22	Ξ	<project></project>	
23	T	<name>Brand Promotions</name>	
24	E -	<employee></employee>	
25	T	<name>Richard Dowson</name>	
26	-		
27	La la	<employee></employee>	
28	T	<name>Steven Aikin</name>	
29	-		
30	L		
31			*
Ln : 1	Col:1	Sel : 0 Dos\Windows ANSI II	NS

Fig. 15: Mapping with Join and Grouping, the output

### 5 Conclusion

In this document we presented the UMAP implementation. The complete implementation translates all seven Clip [2] main use cases to UMAP. We presented here only the most important three. As future work we plan to implement interfaces and semi-automatic compilation to several targets for additional schema mapping languages and tools.

### References

- 1. Florin Chertes. DBAI-TR-2012-76. Technical report, DBAI, Institute of Information Systems, Vienna University of Technology, 2012.
- Alessandro Raffio, Daniele Braga, Stefano Ceri, Paolo Papotti, and Mauricio A. Hernández. Clip: a visual language for explicit schema mappings. In *ICDE 2008*, pages 30–39. IEEE, 2008.