

Studies into interactive didactic approaches for learning software design using UML

Stikkolorum, D.R.

Citation

Stikkolorum, D. R. (2022, December 14). Studies into interactive didactic approaches for learning software design using UML. Retrieved from https://hdl.handle.net/1887/3497615

Version: Publisher's Version

Licence agreement concerning inclusion of doctoral

License: thesis in the Institutional Repository of the University

of Leiden

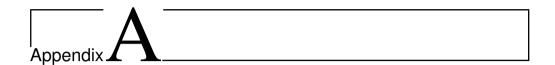
Downloaded from: https://hdl.handle.net/1887/3497615

Note: To cite this publication please use the final published version (if applicable).

About the Author

Dave Stikkolorum was born (1976) and raised in the The Hague-region of The Netherlands. After high school he studied electrical engineering with 'technical computer science' as major. After his studies, in 2001, he decided to start as a part-time software developer and as a secondary school teacher in mathematics. He traded the job of software developer for a job as teacher at The Hague University of Applied Sciences (HHS). From 2005 Dave became a full-time member of the 'technical informatics' team at HHS. Because of an interest in the education of software design he started his PhD research at the Leiden Institute of Advanced Computer Science (LIACS, Leiden University). The research was conducted under the supervision of Michel Chaudron and later accompanied by Peter van der Putten. Currently Dave is involved as program director of HBO-ICT at The Hague University of Applied Sciences and teaches in the Game Development and Simulation program.





Overview Recommendations for Teaching Software Modelling

 Table A.1: Overview Recommendations for Teaching Software Modelling

Student	(Education of) Theory	Tool	Skills		Process Skills
Chal-					
lenges					
			Abstraction	Bloom levels	
Analysis	Problem 1: Students	Problem 2: because	Problem 3: Students	Problem 4: Software	Problem 5: Students
	have difficulties with	UML uses the same	have difficulties to	analysis reaches the	have difficulties in
	the analysis of text	notation for Analy-	identify the relevant	higher level bloom	reflecting to their
	based assignments.	sis models as well as	concepts from a prob-	tasks: analyse (4) eval-	own analysis models.
	Because of incom-	design models, most	lem (domain) in or-	uate (5) and create	Recommendation 6:
	plete instructions	tools do not support	der to abstract them	(6). Recommendation	students should be
	students tend to loose	analysis steps by hav-	and present them in	5: Students should	trained using an ap-
	the overall overview.	ing a special analysis	a (UML) diagram.	practice with exer-	proach where analy-
	Recommendation	mode. Recommen-	Recommendation 4:	cises that are not too	sis models iteratively
	1: keep instruction	dation 3: emphasise	use the noun iden-	complex. Using a do-	evolve. Recommen-
	text simple when stu-	the analysis phase	tification technique	main that students	dation 7: organise
	dents start to learn	in which a student is	to train the identi-	are familiar with can	process support were
	and practice software	modelling and use	fication of relevant	be helpful.	continuously feed-
	analysis. Recommen-	a simple feature set	concepts.		back articulated by
	dation 2: When it	for making domain			the lecturer and/or
	is need for practice	models. In best case a			teaching assistants.
	to have incomplete	teacher can tailor the			
	or poor assignment	tool that is used.			
	texts, provide enough				
	feedback moments				
	and/or peer review				
	sessions to overcome				
	students wandering.				

Design	Problem 6: Students	Problem 8: because	Problem 9: Students	idem*	Problem 11: Students
	have difficulties with	UML uses the same	have difficulties to		have difficulties in
	determining how de-	notation for analy-	include relevant		reflecting to their
	tailed a design model	sis models as well as	abstractions of the		own design models.
	should be. Recom-	design models, most	domain concepts (pre-		Recommendation 12:
	mendation 8: Offer	tools do not support	sented in a domain		students should be
	assignment that make	design steps by hav-	model) in their soft-		trained using an ap-
	clear which level of	ing a special design	ware design. Problem		proach where designs
	detail is expected	mode. Recommen-	10: Students have		iteratively evolve.
	from the student.	dation 11: emphasise	difficulties deciding		Recommendation
	Problem 7: Because	the design phase in	between classes and		13: organise process
	of the open character	which a student is	attributes (abstrac-		support were con-
	of most assignments	modelling and use an	tion).		tinuously feedback
	students have diffi-	increasing complex	To be explored further.		articulated by the lec-
	culties in deciding	feature set for making			turer and/or teaching
	of their design are	domain models. In			assistants.
	"done". Recommen-	best case a teacher			
	dation 9: Include	can tailor the tool that			
	feedback moments	is used.			
	and design discus-				
	sions during a lecture.				
	Recommendation 10:				
	Offer follow-up as-				
	signments that enable				
	students to discover				
	missing elements in				
	their initial design.				

Analysis	Problem 12 : there is	Problem 13: Going	idem*	Problem 14: Students
vs Design,	no mature modelling	from analysis to de-		don't have experience
Analysis	tool that supports	sign covers going		with the development
and De-	the transition from	from a concrete prob-		process and therefore
sign	analysis to design	lem domain to an		find it difficult to see
	Recommendation 14:	abstraction followed		the relation between
	develop tools that	by creating a design		analysis and design.
	supports the transi-	from 1) the gathered		Recommendation
	tion from analysis to	domain concept ab-		17: students should
	design Recommen-	stractions 2) the in-		be trained using an
	dation 15: integrate	troduced software		approach where anal-
	modelling tools with	concepts from the so-		ysis and design are
	software that sup-	lution domain. This		part of iterative cycles
	ports (agile) software	combining of abstrac-		(agile UML work-
	development process	tions of two sources		shop).
	tools.	is difficult to grasp		
		for students. Recom-		
		mendation 16: Offer		
		standard abstrac-		
		tions (responsibility		
		driven design, Wirfs-		
		Brock) and standard		
		architectures (layered		
		architecture) as a first		
		start.		

Design	Problem 15: students	Problem 16 : there is	Problem 17: UML	idem*	Problem 18: students
decisions	have problems ac-	no mature modelling	offers a very high		have problems decid-
	cepting that there are	tool that supports	abstraction for gener-		ing when a design is
	multiple solutions to	design decision mak-	alisation of (software		better than another.
	a problem. Recom-	ing. Most tools check	elements): class. Rec-		Recommendation 22:
	mendation 18: use	on syntax level. Rec-	ommendation 21:		use software design
	software design prin-	ommendation 19:	Make use of stereo-		principles as a guide
	ciples as a guide for	develop tools that	types / roles when		for decision making.
	decision making.	use feedback based	designing (responsi-		Principles should be
		on design principles.	bility driven design,		used when discussing
		Recommendation 20:	Wirfs-Brock)		(pair/collaborated
		put a low emphasis			modelling)
		on syntax.			

Design vs	Problem 19: Students	Problem 20: Students	Problem 22: there is	idem*	Problem 23: Students
Implemen-	do not see the effect	don't update their	a gap between the		don't update their
tation	of their design (deci-	design from the mo-	completeness of an		design from the mo-
	sions) on the imple-	ment they start im-	'abstract' design and		ment they start im-
	mentation. There is	plementing in code.	the desired imple-		plementing in code.
	a difference in time	Problem 21: Most	mentation. Students		Problem 24: Students
	in the development	tools only offer one	don't feel comfortable		neglect their models
	process. Recommen-	way code generation	with the degree of		when implementing
	dation 23: Teach stu-	/ import or only par-	freedom when they		the software. Rec-
	dents an agile ap-	tial code generation /	are novice software		ommendation 26:
	proach in modelling	import. Recommen-	developers.		students should be
	for having smaller	dation 24: Develop	To be explored further.		trained using an ap-
	steps in the devel-	tools that support			proach where design
	opment process that	complete round-trip			and implementation
	makes the effect of de-	code generation -			are part of iterative
	sign decisions clearer.	code import (model			cycles (agile UML
		generation). Recom-			workshop).
		mendation 25: Find			
		ways to keep track			
		of code changes and			
		updating students'			
		designs.			

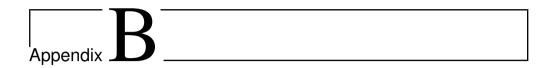
Implemen-	Problem 25: Because	Problem 26: students	Problem 27: Students
tation	of the direct effect	have difficulties not	apply implementa-
Thinking	of implementation,	to think in implemen-	tion knowledge too
	students are not mo-	tation when making	soon in the develop-
	tivated to use mod-	initial designs. It dis-	ment process.
	elling as a first step	tracts them from us-	To be explored further.
	during analysis and	ing their abstraction	
	design. Recommen-	skills.	
	dation 27: Use moti-	To be explored further.	
	vating environments		
	such as gaming ("The		
	Art of Software De-		
	sign") to engage stu-		
	dents into modelling.		

Model	Problem 28: Students	Problem 29: Tools	Problem 30: as model
Complex-	have difficulties in	offer often more infor-	complexity grows,
ity	understanding a de-	mation than needed.	the cognitive load
-	sign when the design	Recommendation	increases on every
	consists of a lot of	30: When, available	level. Often models
	information. Rec-	use auto layout and	are presented that
	ommendation 28:	change level of detail	consist of too many
	Consider the layout		elements (rule of 7
	when preparing as-		+/-2). Recommenda-
	signments. Recom-		tion 31: offer students
	mendation 29: Teach		assignments with in-
	according to a layout		creasing complexity
	style (Scott Ambler)		of the models. Rec-
			ommendation 32:
			keep models simple
			and keep the amount
			of elements in a dia-
			gram low (7+/-2)

General	Problem 31: Students	Problem 32: Students
	find tools complex.	have difficulties in
	They have a long and	understanding the
	steep learning curve	role of the different
	and most of the times	UML diagrams in a
	difficult to install.	development process.
	Recommendation	Problem 33: Students
	33: Develop easy to	find it difficult to
	use tools that don't	articulate their dif-
	require installation,	ficulties during an
	like WebUML	assignment. (they
		don't know what they
		don't know). Prob-
		lem 34: Students do
		not automatically re-
		consider their designs.
		Recommendation 34:
		Train students to use
		their modelling ac-
		tivities with an agile
		approach (agile UML
		workshop), using pair
		modelling or peer
		reflection moments,
		in order to discover
		diagram relevancy
		within development
		cycles and to enable
		learning discussions.
		Discussion enables
		the re-thinking of the
		delivered design.

Table A.1 summarises the problems identified in the research of this dissertation. Most of the problems have been investigated and are accompanied by a recommendation.

^{*} same kind of problem / solution



Coffee Machine Assignment

a Coffee Machine A coffee machine can make different types of coffee (one at a time): black, with sugar, with sugar and milk. The different types of coffee are poured into a cup. The front of the machine contains a pane with buttons. The buttons are used for selecting the desired drink. One can pay with a chip card or coins. The chipcard is read by a chip sensor and the coins by a coin sensor.



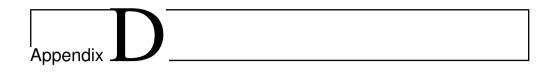
Tank Assignment

The Modelling Task – a Tank Game¹ In this task you will design a game with use of the UML class diagram. You do not need to use packages in this assignment. The description of the game is as follows: A player (user) controls a certain tank. This tank is a Panzer Tank, a Centurion Tank or a Sherman Tank. They fire bullets and Tank shells. Bullets can be Metal, Silver or Gold bullets.

A tank moves around a world (level). The aim is to destroy all other tanks in the world. After a world has been completed the tank advances to the next world. A list of all the worlds visited is kept.

An entire game consists of 8 levels. A world contains a maximum of 20 tanks that compete for victory. Each tank remembers which tanks it has destroyed in the past. The score for each level is kept by a scoreboard that gets notified by the individual tanks each time an opponent is shot. The players control their tanks through an interface allowing for steering, driving (reverse / forward), switching ammo and firing.

¹text: B. Karasneh



Grading Rubric

 Table D.1: Class Diagram Rubric for Grading Design Modelling

Grade	Judgement, criteria description
1	The student does not succeed to produce a UML diagram related to the task. He/she is not
	able to identify the important concepts from the problem domain (or only a small number of
	them) and name them in the solution/diagram. The diagram is poor and not/poorly related
	to problem description with a lot of errors: high number of wrong uses of UML elements
	mostly no detail in the form of attributes or operations.
2	The student is not able to capture the majority of the task using the UML notation. Most
	of the concepts from the problem domain are not identified. The detail, in the form of at-
	tributes or operations, linked to the problem domain is low. Some elements of the diagram
	link to the assignment, but too much errors are made: misplaced operation / attributes non
	cohesive classes few operation or attributes are used.
3	The student is able to understand the assignment task and to use UML notions to partly
	solve the problem. The student does not succeed to identify the most important concepts. A
	number of logical mistakes could have been made.
	Most of the problem is captured (not completely clear) with some errors: missing labels on
	associations missing a couple important classes / operations / attributes Logical mistakes
	that could have been made: wrong use of different types of relationships wrong (non logical)
	association of classes.
4	The student captures the assignment requirements well and is able to use UML notations in
	order to solve the problem. Almost all important concepts from the problem are identified.
	Some (trivial) mistakes have been made: Just one or two important classes / operations /
	attributes are missing Design could have been somewhat better (e.g. structure, detail) if the
	richness of the UML (e.g. inheritance) was used.
5	Student efficiently and effectively used the richness of UML to solve the assignment. The
	problem is clearly captured from the description. concepts from the domain / task are
	identified and properly named The elements of the problem are represented by cohesive,
	separate classes (supports modularity) with a single responsibility In the problem domain
	needed attributes and operations are present Multiplicity is used when appropriate Naming
	is well done (consistent and according to UML standard) Aggregation / Composition /
	Inheritance is well used No unnecessary relationships (high coupling) are included.



Class Diagram Evaluation Criteria

- □ Syntax does the diagram follow the formal notation rules of the modelling language? At the time of writing UML 2.4 for example.
- □ Layout Is the diagram readable? Does it not mix styles? For example: is an inheritance relationship drawn from top to bottom across the whole diagram?
- □ Consistency is the diagram consistent with the other diagrams in the model? For example: does a class name corresponds with a lifeline in a sequence diagram?
- □ Semantics Does the diagram represent what it should? For example: a '1 .. *' multiplicity is applied. Which means '1 to many'. Was this meant?
- □ Requirements satisfaction does the diagram support the (functional) requirements? For example: Does the class diagram have a set of classes that cover the responsibilities of the system that is designed?
- □ Design principles are common design principles applied? For example: did the student take coupling and cohesion into account?
- □ Design decisions were the right design decisions made and applied? For example: a pattern is applied to support a quality requirement.