A Decade of Software Design and Modeling: A Survey to Uncover Trends of the Practice

Suresh Kumar Mukhiya

Western Norway University of Applied Sciences

PCS 953 / DAT 353
Spring 2019
Bergen
Norway

Høgskulen på Vestlandet

Outline

Introduction
- Software Engineering, SE

Backgrounds

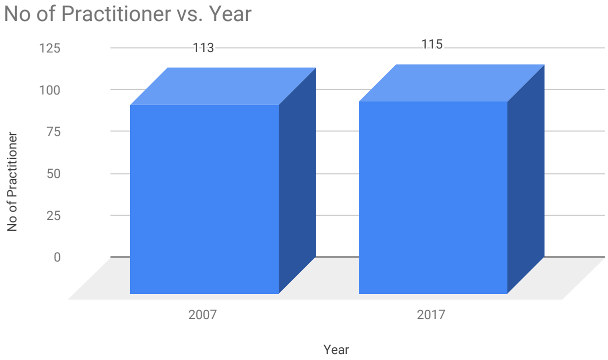
Survey Results

Analysis

Summary

Survey conducted on two phases

- with 228 software paractitioner
- April-December, 2007
- March-November, 2017



Survey Structure

- Topic 1 - **Fundamentals** - Software design and what is software model
- Topic 2 - **Basic Characteristics of practices** - What medium and methods are used for moedling?
- Topic 3 - **Life Cycle** - Activities involved in SDLC
- Topic 4 - **Platforms** - Tools, methodologies, platforms used in SDLC
- Topic 5 - **Efficacy** - Design and development practices
- Topic 6 - **Code VS Model centrism** - Challenges in code-centric vs model centric SD
- Topic 7 - **Open ended and optinal contact info**
- Topic 8 - **Demographics**

Goal of the Survey

- Uncover **trends in the practice** of `software design` and **adaptation pattern** of `modeling language`

Goal of the Survey

- Uncover **trends in the practice** of `software design` and **adaptation pattern** of `modeling language`

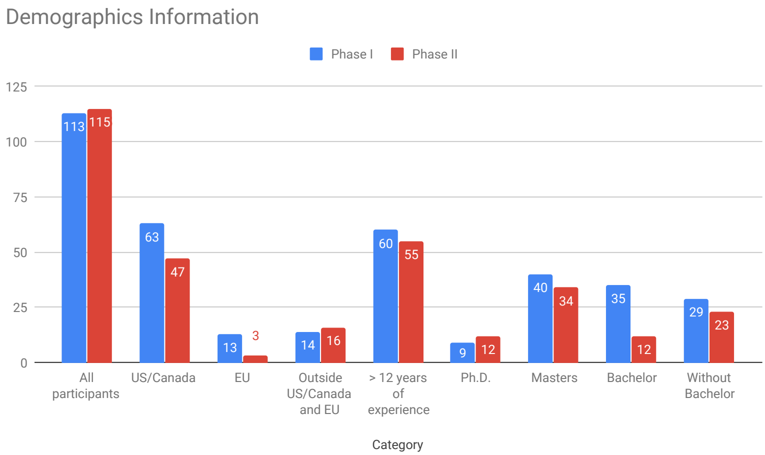
Software Design

- Topic 6 - **Code VS Model centrism** - Challenges in code-centric vs model centric SD
- Topic 7 - **Open ended and optinal contact info**
- Topic 8 - **Demographics**

Modeling languages

- Topic 6 - **Code VS Model centrism** - Challenges in code-centric vs model centric SD
- Topic 7 - **Open ended and optinal contact info**
- Topic 8 - **Demographics**

Demographics



Topic 1: What is a software model?

Responses for Topic 1: What is a software model							
Entity that might be a model	**Phase I**			**Phase II**			**Mean Gap**
	% SA+A	**% SD+D**	**Mean**	**%SA+A**	**%SD+D**	**Mean**	
Class Diagram	88.4	2.7	4.3	87	4.9	4	-0.3
UML Deployment Diagram	77.5	5.4	4.1	72	17.5	3.8	-0.2
Use Case Diagram	82.1	9.8	4	80	13.5	3.8	-0.3
Picture By Drawing Tool	**85.6**	**7.2**	**4**	**62**	**20.3**	**3.5**	**-0.5**
Textual Use Case	**78.8**	**10.6**	**4**	**59**	**18.4**	**3.5**	**-0.5**
Whiteboard Drawing	**78.8**	**8.8**	**3.9**	**63**	**20**	**3.6**	**-0.4**
Picture By Hand	**57.1**	**9.8**	**3.9**	**61**	**13.4**	**3.5**	**-0.4**
Source Code	46.8	38.7	3.2	47	38.7	3.1	-0.1
Source Code Comment	33.9	41.1	2.9	44	39.9	3	0.1

Topic 2: Characterization of Practices 1/4

- Medium and Methods used for modeling
- What models are used for?
- Reference Materials
- Participants daily activities

Topic 2: Medium and methods of modeling							
Medium or methods used to model	**Phase I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**% Never& Sometimes**	**% Very Often**	**Mean**	
Whiteboard drawing	33.3	45.0	3.2	40	57.9	2.9	-0.3
Diagramming tool (e.g. Visio)	42.3	36.9	2.9	43	43.2	2.8	-0.1
Word processor / text	45.5	26.8	2.8	42	55.3	2.7	-0.1
Word of mouth	**42.3**	**27.0**	**2.8**	**54**	**46.1**	**2.4**	**-0.4**
Handwritten material	51.4	22.5	2.6	49	**51.3**	2.6	0.0
Comments in source code	51.4	21.6	2.5	49	37.8	2.6	0.1
Modeling tool/CASE	58.9	29.5	2.4	55	29.0	2.5	0.1
Drawing software	72.1	12.6	2.1	68	29.0	2.3	0.2

Høgskulen på Vestlandet

Characterization of Practices 2/4

Topic 2: What models are used for?							
Activity	**Phase I**			**Phase II**			**Mean Gap**
	% Never & Sometimes	**% Very Often**	**Mean**	**% Never & Sometimes**	**% Very Often**	**Mean**	
Developing a design	26.6	48.4	3.3	28	55.1	3.2	-0.1
Transcribing a design into digital format	32.8	39.1	3.1	41	51.7	2.9	-0.2
Prototyping a design	**53.1**	**32.8**	**2.7**	**24**	**32.2**	**2.2**	**-0.5**
Brainstorming possible designs	**54.7**	**23.4**	**2.6**	**34**	**44.8**	**3**	**0.4**
Generating code (code editable)	65.1	17.5	2.2	66	34.4	2.2	0
Generating all code	76.6	14.1	1.8	66	31	2.1	0.3

Characterization of Practices 3/4

Responses for Topic 2: Reference materials							
Refer to material created by/as	Phase I			Phase II			Mean Gap
	% Never and Sometimes	% Very Often	Mean	% Never and sometimes	% Very Often	Mean	
Word of mouth	22.3	54.5	3.4	40	60.5	3.1	-0.3
Word processor / text	30	48.2	3.3	29	54	2.9	-0.4
Diagramming tool	32.4	42.3	3.1	70	36.9	2.7	-0.4
Whiteboard drawing	34.5	41.8	3	37	48.6	2.7	-0.3
Comments in source code	42	30.4	2.9	55	47.3	2.7	-0.2
Drawing software	57.8	13.8	2.6	32	39.5	2.4	-0.2
Modeling tool/CASE	55.9	31.5	2.5	85	28.9	2.3	-0.2
Handwritten material	56	20.2	2.4	27	29.7	2.3	-0.1

Characterization of Practices 4/4

Responses for Topic 2: Daily activities of participants							
Available tasks	**Phas I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**% Never& Sometimes**	**% Very Often**	**Mean**	
Think about s/w system	9.4	77.1	4.1	12	41.2	4.1	0
Run / attend meetings	19.8	60.4	3.6	14	68.6	3.5	-0.1
Explain s/w design to others	15.8	51.6	3.5	26	65.7	3.2	-0.3
Design a s/w system	18.8	57.3	3.5	34	54.3	3.3	-0.2
Lead software project	29.2	53.1	3.3	23	65.7	3.2	-0.1
Search about s/w system	31.2	46.2	3.2	31	51.4	3.2	0
Model a s/w system	30.2	45.8	3.2	37	45.8	3.1	-0.1
Write new code	37.5	49	3.1	29	54.3	3.3	0.1
Maintain existing code	37.5	40.6	3	26	60	3.3	0.3
Fix bugs	**39.4**	**39.4**	**3**	**23**	**48.6**	**3.5**	**0.5**
Perform manual testing	35.1	34	2.9	37	51.4	3.1	0.2
Write / maintain requirements	41.1	40	2.9	34	48.6	3.1	0.2
General administration	40.4	29.8	2.8	43	54.3	2.8	0
Write / maintain test scripts	**58.3**	**17.7**	**2.4**	**47**	**44.1**	**2.8**	**0.4**

Topic 3: Life Cycle - 1/2

Activivties invovled in various development phases of Software Development Life Cycle (SDLC)

Topic 3: When do you perform the following tasks?					
Available tasks	**Phase I**		**Phase II**		**% Gap**
	Mode	**%**	**Mode**	**%**	
Searching	Constantly	64.5	Constantly	36.1	-28.4
Requirements	Start	60	Start	72.2	12
Design	Start	53.8	Start	44.4	-9.4
Modeling	Start	46.5	Start	66.7	20.2
Perform testing	Constantly	44.1	Constantly	42.9	-1.2
Coding	Constantly	41.7	Constantly	31.4	-10.3
Knowledge transfer	Constantly	41.7	Constantly	30.6	-11.1
Develop tests	Constantly	40.2	Constantly	34.3	-5.9
Documentation	End	38.7	End	27.8	-10.9

Life Cycle 2/2

Topic 3: When is modeling performed?							
Timeline	**Phase I**			**Phase II**			**Mean Gap**
	% Never&Sometimes	**% Very Often**	**Mean**	**%Never & Sometimes**	**% Very Often**	**Mean**	
Before coding	18.8	59.8	3.7	16	54	3.7	0
During coding	33.3	36	3.1	41	51.3	2.8	-0.3
After coding	60.4	19.8	2.5	54	37.8	2.5	0
Only on request	**78.5**	**10.3**	**1.9**	**59**	**32.4**	**2.3**	**0.4**

Topic 4: Platforms

Topic 4: Modeling notations and tools							
Modeling notations	Phase I			Phase II			Mean Gap
	% Never&Sometimes	% Very Often	Mean	% Never & Sometimes	% Very Often	Mean	
UML (any version)	30.9	51.8	3.3	46	33.4	2.9	-0.4
UML 2.*	52.1	34.4	2.6	53	34.4	2.5	-0.1
SQL	55.6	29.6	2.5	49	34.3	2.7	0.2
Structured Design models	58.8	21.6	2.5	50	38.2	2.7	0.2
UML 1.*	**54.8**	**28**	**2.4**	**73**	**26.7**	**1.9**	**-0.5**
ERD	**63.2**	**20.8**	**2.3**	**46**	**40**	**2.9**	**0.6**
Well-defined DSL	**78.8**	**5.8**	**1.7**	**62**	**32.3**	**2.4**	**0.7**
ROOM / RT for UML	85.9	7.1	1.5	79	15.2	1.8	0.3
SDL	**89.2**	**3.2**	**1.3**	**68**	**25.8**	**2.2**	**0.9**
Formal (e.g. Z, OCL)	**93.9**	**2**	**1.3**	**75**	**18.8**	**1.9**	**0.6**
BPEL	92.8	3.1	1.3	87	13	1.6	0.3

7

Technology options	Phase I			Phase II			Mean Gap
	% Never& Sometimes	% Very Often	Mean	%Never & Sometimes	% Very Often	Mean	
Java	**46.3**	**31.6**	**2.4**	**80**	**11.5**	**1.8**	**-0.6**
PHP / Perl	74.2	19.4	2	74	14.3	2.2	0.2
ASP.Net	79.4	14.4	1.8	74	14.3	2	0.2
Ruby / Python	**88.3**	**8.5**	**1.6**	**77**	**17.2**	**1.9**	**0.3**
C / C++*	60	30	2.4	65	25	2.3	-0.1

Topic 5: Efficacy

- Questions related to suitability of the modeling tools
- Participants' perceptions of key characteristics of modeling tools

How good are modeling tools for ?							
Available activities	**Phase I**			**Phase II**			**Mean Gap**
	% Poor	**% Good**	**Mean**	**% Poor**	**% Good**	**Mean**	
Developing a design	**16.9**	**47.9**	**3.4**	**11**	**53.6**	**2.9**	**-0.5**
Transcribing a design into digital format	24.6	42	3.2	25	60.7	3.3	0.1
Generating code (code is editable)	39.1	29	2.9	32	64.3	3	0.1
Prototyping a design	41.2	29.4	2.9	25	71.4	3.1	0.2
Brainstorming possible designs	45.1	32.4	2.8	18	74.7	3.1	0.3
Generating all code (no manual coding)	**79.7**	**8.7**	**1.9**	**50**	**42.9**	**2.5**	**0.6**

Topic 6: Code VS Model centralism - 1/2

Topic 6: Available activities	Phase I			Phase II			Mean Gap
	% Easier in Models	% Easier in Code	Mean	% Easier in Models	% Easier in Code	Mean	
Fixing a bug	28.9	43.3	3.2	19	40.6	3.2	0
Creating efficient software	35.9	43.5	3.1	27	50	3.2	0.1
Creating a system as quickly as possible	46.7	42.4	3	31	56.2	3.2	0.2
Creating a prototype	43	32.6	2.9	44	37.5	2.7	-0.2
Creating a usable system for end users	42.4	22.8	2.7	49	27.3	2.4	-0.3
Modifying a system when requirements change	54.9	24.2	2.5	41	37.5	2.8	0.3
Creating a system that most accurately meets requirements	67	19.8	2.2	56	26.4	2.3	0.1
Creating a re-usable system	**63**	**15.2**	**2.2**	**42**	**30.4**	**2.6**	**0.4**
Creating a new system overall	68.5	20.7	2.2	64	24.2	2.3	0.1
Comprehending a system's behaviour	71.9	15.7	2	75	15.7	1.9	-0.1
Explaining a system to others	81.8	7.6	1.7	66	15.6	1.9	0.2

- Results show it is easier to create a prototypes, modify the system, create a reusable system and explain system to others in the form of model
- It is easier to debug, create effecient software system, create a system as soon as possible in code.

Topic 6: Code VS Model centralism - 2/2

Topic 6: Problems with Model-Centric Approaches	Phase I			Phase II			Mean Gap
	% Slight Problem	% Bad Problem	Mean	% Slight Problem	% Bad Problem	Mean	
Models become out of date and inconsistent with code	**16.3**	**68.5**	**3.8**	**25**	**40.6**	**3.2**	**-0.6**
Models can not be easily exchanged between tools	26.4	51.6	3.3	19	40.7	3.3	0
Modeling tools are 'heavyweight'(install,learn,configure,use)	31.5	39.1	3.1	41	37.6	3	-0.1
Code generated from modeling tool not of the kind kind I would like	39.6	38.5	3	44	31.3	2.7	-0.3
Cannot model in enough detail-must write code	43.8	36	2.8	47	28.1	2.6	-0.2
Creating and editing model is slow	43.5	22.8	2.7	38	34.4	3	0.3
Modeling tools change, models become obsolete	44.6	32.6	2.7	31	34.4	3	0.3
Modeling tools lack features I need or want	44.9	21.3	2.6	44	18.8	2.6	0
Modeling tools hide too many details(fully visible in source)	44.6	23.9	2.6	34	31.3	2.9	0.3
Modeling tools are too expensive	46.7	26.7	2.6	38	15.7	2.7	0.1
Modeling tools cannot be analyzed as intended	51.1	25.6	2.5	56	21.9	2.5	0
Semantics of models different from prog. language	56.7	23.3	2.4	48	16.2	2.5	0.1
Modeling languages are not expressive enough	54.9	17.6	2.4	50	15.7	2.5	0.1
Modeling languages are hard to understand	62.6	9.9	2.2	58	15.2	2.3	0.1
Have had bad experience with modeling	63.7	16.5	2.2	61	16.2	2.2	0
Do not trust companies will continue to support their tools	**67.4**	**10.1**	**2**	**41**	**15.7**	**2.6**	**0.6**

9

- Models become out of date and inconsistent with code
- Models can not be easily exchanged between tools

Analysis

- A meta-model is a model of a modelling language
- Meta-models are used to define modelling languages
- E.g. in OO modelling a person is an instance of class
- Meta-modelling is used to create Domain Specific Modelling Languages DSMLs, i.e. one create language constructs for important domain concepts, e.g. a student and a teacher is instances of persons

Meta-model example

- Models: first class entities

Meta-model example

- Models: specified by means of a modelling language

Meta-model example

- Modelling language: corresponding meta-model + semi-formal semantics

OMG Meta-modelling Levels

OMG levels	OMG Standards/examples
M_3: Meta-meta-model	MOF
M_2: Meta-model	UML language
M_1: Model	A UML model: Class "Person" with attributes "name" and "address"
M_0: Instance	An instance of "Person": "Ola Nordmann" living in "Sotraveien 1, Bergen"

MOF based modelling languages

UML System on a Chip for microchip/hardware/firmware/software definition

SoaML for service-oriented architecture

Business Process Modelling Notation (BPMN, together with it's XML form BPML and executable form BPEL) examples of a Process Modelling language

SysML for modelling large, complex systems of software, hardware, facilities, people and processes

UPDM for modelling enterprise architectures

CWM for data warehouse

Benefits of MDE

- Engineers can reason about the system at different abstraction levels
- Platform independent models without concern of implementation details
- Less errors and faster development speed by automatic software generation
- Software adoption by (automatic) model transformations

Chalenges in MDE

- Modelling languages need to haver the right abstractions, i.e. one need domain specific modelling languages
- Specification of constraints integrated in the meta-modelling approach, i.e. graphical modelling formalisms with well defined semantics
- MDE traditionally concerned by software architecture and behaviour, need to have technologies for:
  - Model management (version control, meta-model evolution)
  - Model based security engineering
  - Model based testing, software dependencies, ...

State of the art in MDE

Modeling UML or EMF used as modelling language

Model transformations Rule based (e.g. Atlas) or ad hoc transformations are used

Meta modelling Only tool support for 2 levels of meta-modelling

Tool support Eclipse based (EMF, GMF) tools

Software constraints Specified in text based language (OCL)

Links to resources

- mde-model-driven-engineering-reference-guide
- model-driven-engineering
- mda-model-driven-architecture-basic-concepts
- Diagram-Predicate-Framework
- Eclipse-Modeling-Technologies

Litterateur

- Conceptual data modelling from a categorical perspective. ter Hofstede, A. H., Lippe, E. and Frederiks, P. J. M. (1996), The Computer Journal, 39(3), 215-231.
- View updates in a semantic data modelling paradigm. Johnson, M., Rosebrugh, R. and Dampney C.N. G. In: Proceedings of the 12th Australasian database conference. IEEE Computer Society, 2001. p. 29-36
- Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Fensel D. Berlin: Spring-Verlag
- Symbolic graphs for attributed graph constraints. Journal of Symbolic Computation, 46(3), 294-315. Orejas, F. (2011)
- Diagram predicate framework: a formal approach to MDE. Rutle, A. (2010)

More Litterateur

- Domain-Specific Languages, Martin Fowler (With Rebecca Parsons), Addison Wesley
- Prolog-based infrastructure for RDF: performance and scalability. Wielemaker, J., Schreiber, A.T., Wielinga, B.J. In: (Ed.), The Semantic Web - Proceedings ISWC'03, Sanibel Island, Florida (pp. 644-658). Springer Verlag
- Alloy: a lightweight object modelling notation. Jackson, D. (2002). ACM Transactions on Software Engineering and Methodology (TOSEM), 11(2), 256-290 [LEV80]

References I

EDWARD L. LEVINE, *Introductory Remarks for the Symposium "Organizational Application of self-appraisal and self-assessment: Another look"*, Personnel Psychology **33** (1980), no. 2, 259–262.