MONASH University

MONASH
INFORMATION
TECHNOLOGY

Human-centric (Issues in) Software Engineering

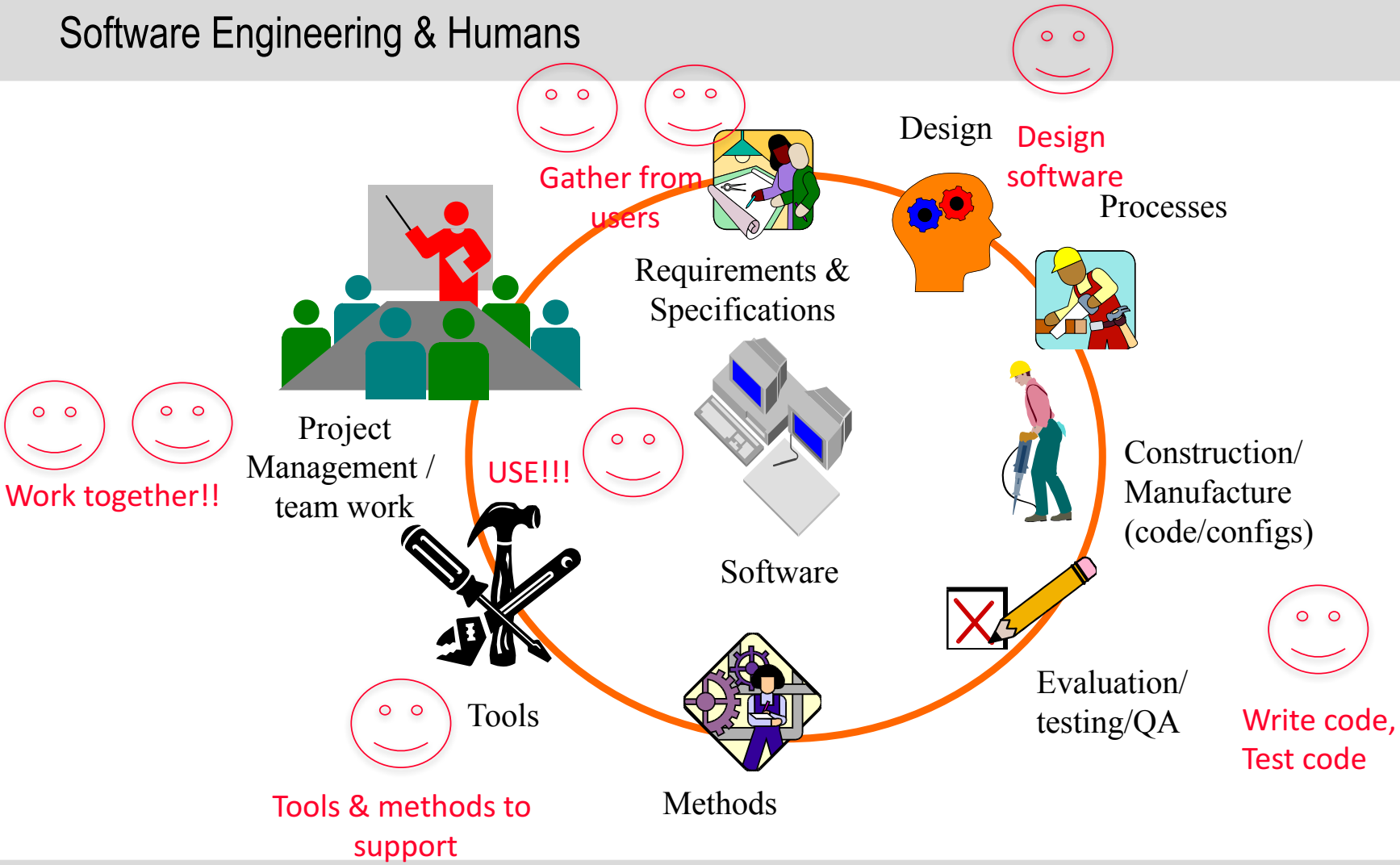
Prof John Grundy

GROUP OF EIGHT AUSTRALIA

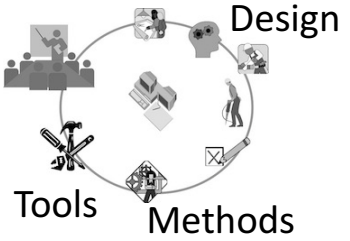
Outline

- Software Engineering & humans
- Examples from our work
  - Human-centric, domain-specific visual models for non-technical experts to specify and generate systems
  - Personality impact on aspects of software development
  - Proactive design critics in software tools to augment human decision making
  - Reporting usability defects
  - Incorporating end user emotions into software requirements engineering
  - Understanding interpersonal issues in agile practices
- Challenges, issues and future directions

Software Engineering & Humans



Human-centric, domain-specific visual models

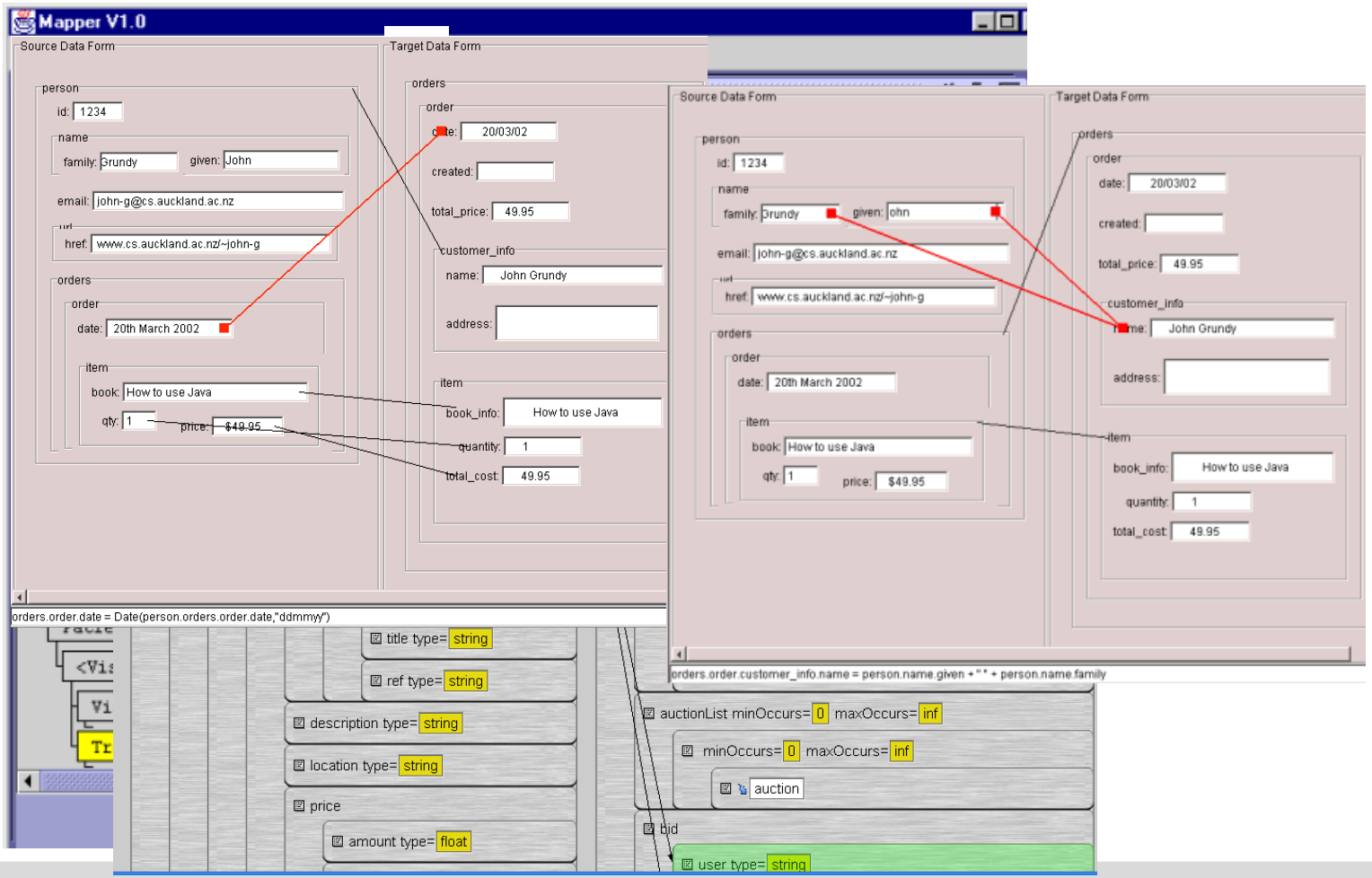


- Idea: complex models hard to work with for developers
  - And non-develpers!!
- Represent using more ”human-centric” way – visual metaphors, visual constructs – “like what sketch on a napkin in a café…” ☺
- (very) Large body of work on this (200+ papers):
  - Platforms – MViews, JViews, Pounamu, Marama, Horus, …
  - Software Engineering uses – Design tool generators, software architecture, performance engineering, user interfaces, requirements, testing, software visualisations, traceability, …
  - “End-user” Application modelling and generation – Statistical Design Language, Report Generation Language, Mobile Health App generation, Business processes, Music, Games, Visual Wikis, …

Example #1: Data integration

- Scenario: complex XML or EDI message format; want to translate into a different format; then process e.g. data wrangling, harmonization ☺
- Traditionally: write QVT/ATL/XSLT/code to do
- Alternative: model transformation visually and generate these transformation implementations
- Meta-model = source/target and mappings
- Visual models might include forms, trees, concrete data visualisations
- Model-drive Engineering = generate XSLT, ATL, Code (C++, Java),…
- Done various with Orion Health Ltd, XSOL Ltd, NICTA/Data61

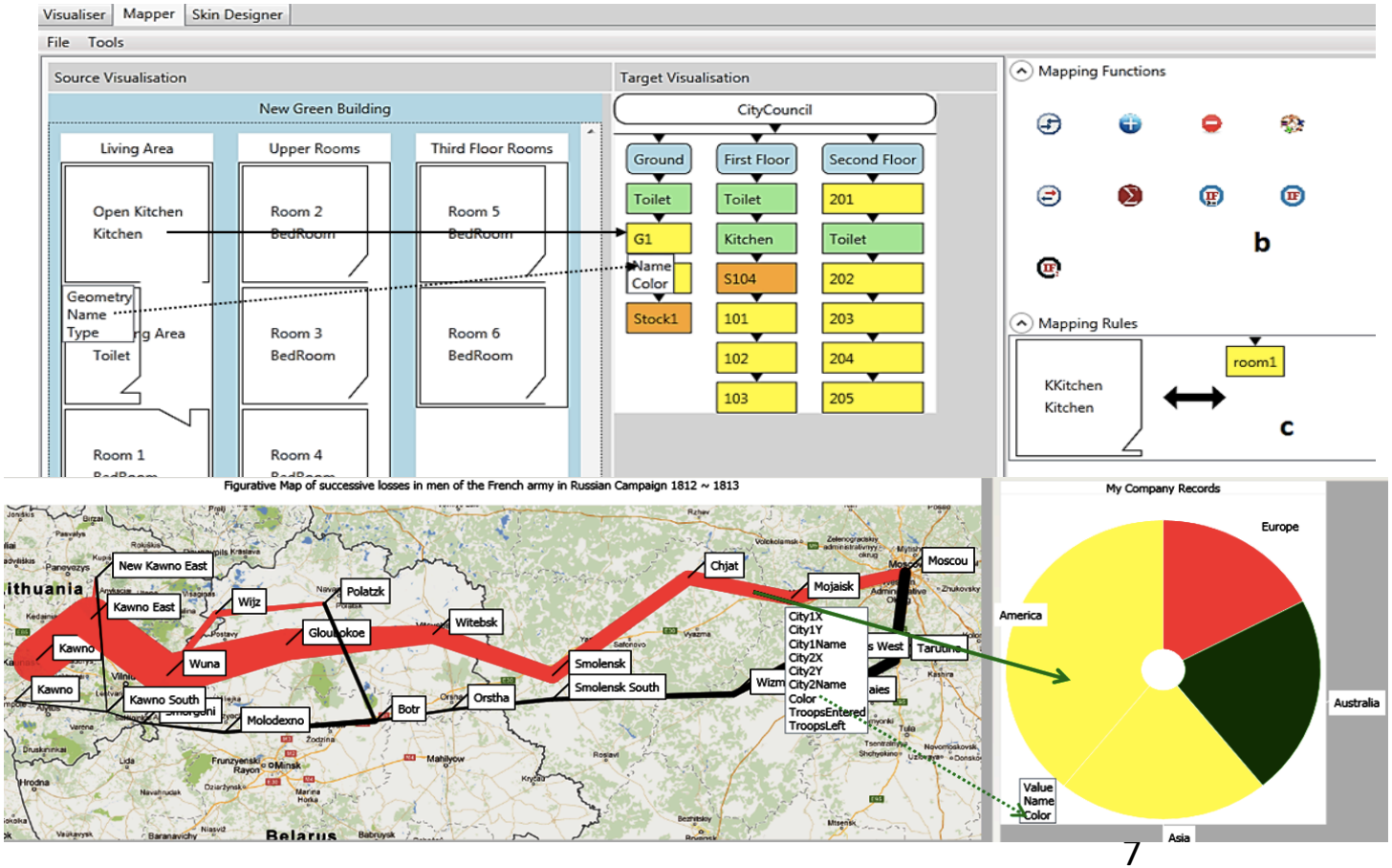
Form-based Mapper



JVLC2004

CONVErT – by-example based data mapping/integration/visualisation

JVLC2014

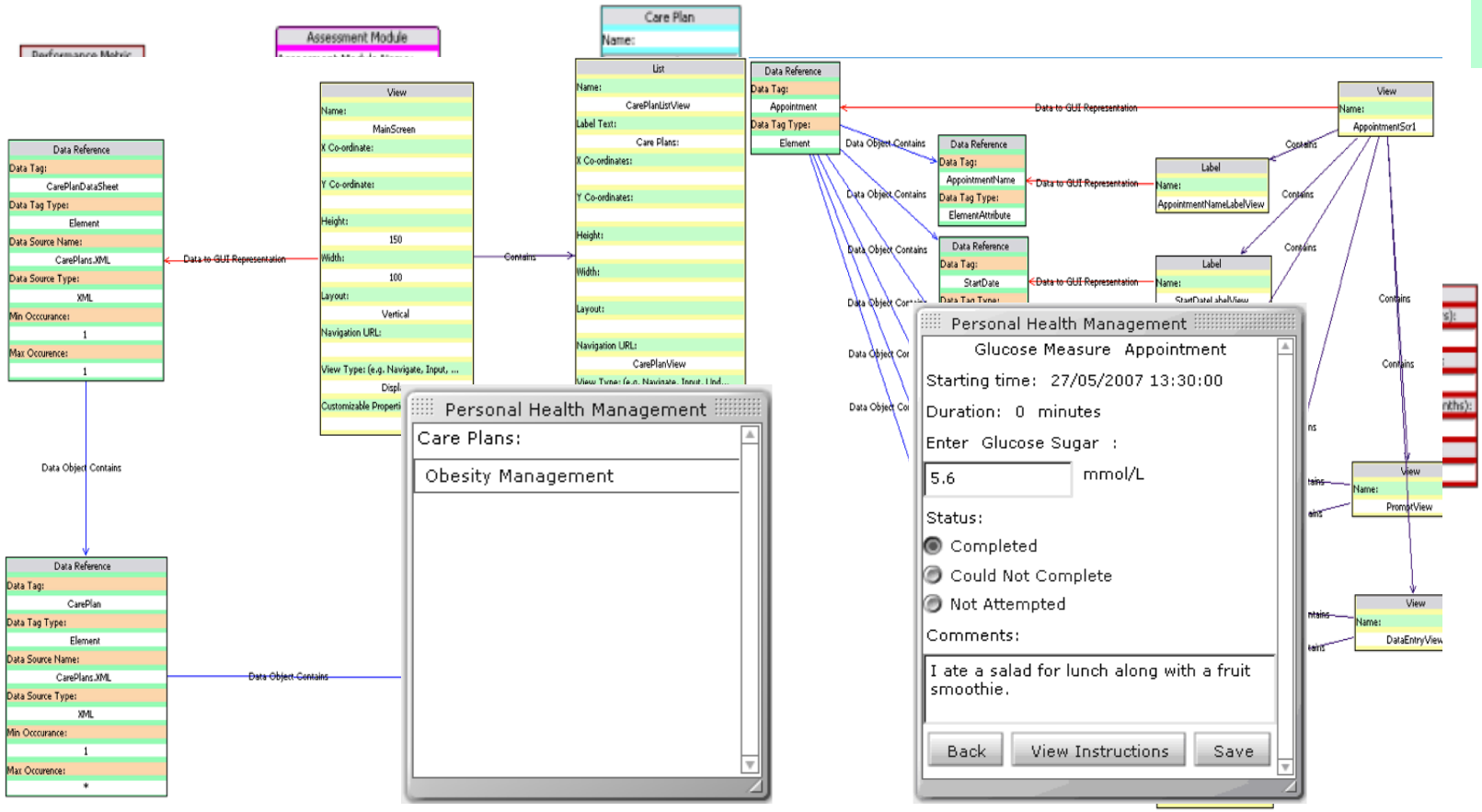


Example #2: Mobile Health app generation

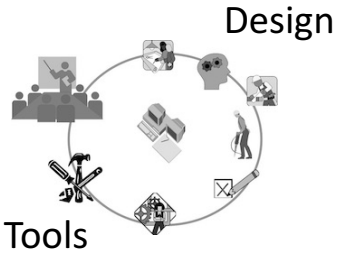
- Scenario: want to model, generate range of eHealth apps
- Mobile phone-based personal health care planning applications
- Two meta-models with associated DVSLs: Visual Health Care Planning Language, Visual Care Application Model
- Model generic care plan with a visual DSVL tool
- Configure generic care plan for individual
- Model mobile app UI for individual from tailored care plan with a visual DSVL tool
- Generate Flash, Windows Mobile, iPhone app code

VHCPL

ASE2008



Proactive design critics in software tools

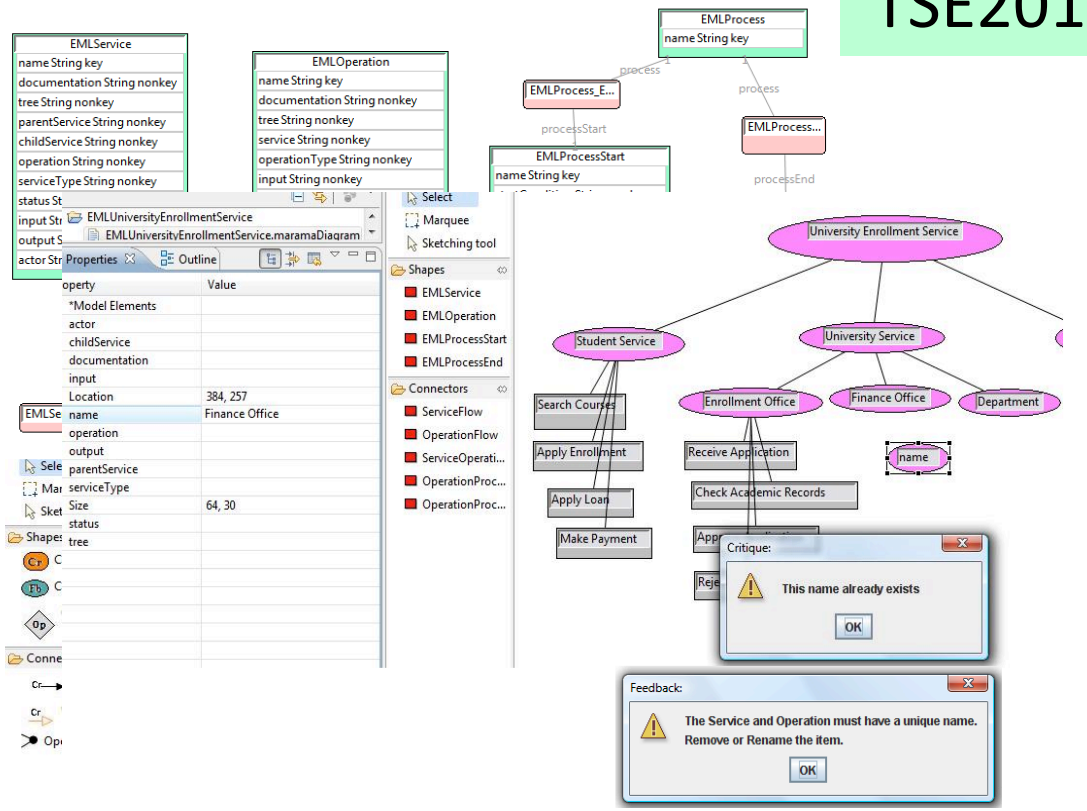


- Design in software engineering is a challenging task
- Issues of
  - Size of models
  - Complexity of models
  - Constraints
  - “Best practices”
- Augment (Marama) design tools with proactive ”design critics” to assist designer
- Basically a set of (rule-based) advisors/”agents”

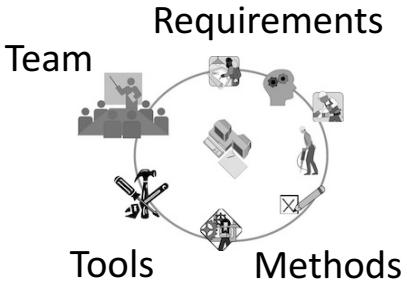
MaramaCritic

TSE2013

Critic Domain						
Critiquing Approach	Mode of Critic	Critic Rule Authoring	Critic Realisation Approach	Critic Strategy	Type of Critic Feedback	Type of critic
Comparative critiquing	Textual	Insert new critic rule	Rule-based	Active	Explanation	Completeness critics
Analytical critiquing	Graphical & 3Dimension Visualisation	Modify critic rule	Predicates	Passive	Argumentation	Consistency critics
	Multi-modal	Delete critic rule	Knowledge-based	Reactive	Suggestions	Optimization critics
		Authoring rule facility	Pattern-matching	Proactive	Examples (or precedents)	Alternative critics
		Enable/ disable critic rules	Programming code	Local	Simulation	Evolvability critics
			Object constraint language (OCL)	Global	Demonstration	Presentation critics
					Interpretations	Tool critics
					Positive	Experiential critics
					Negative	Organizational critics
					Constructive	Pattern critics
						Structure critics
						Naming critics
						Metric critics



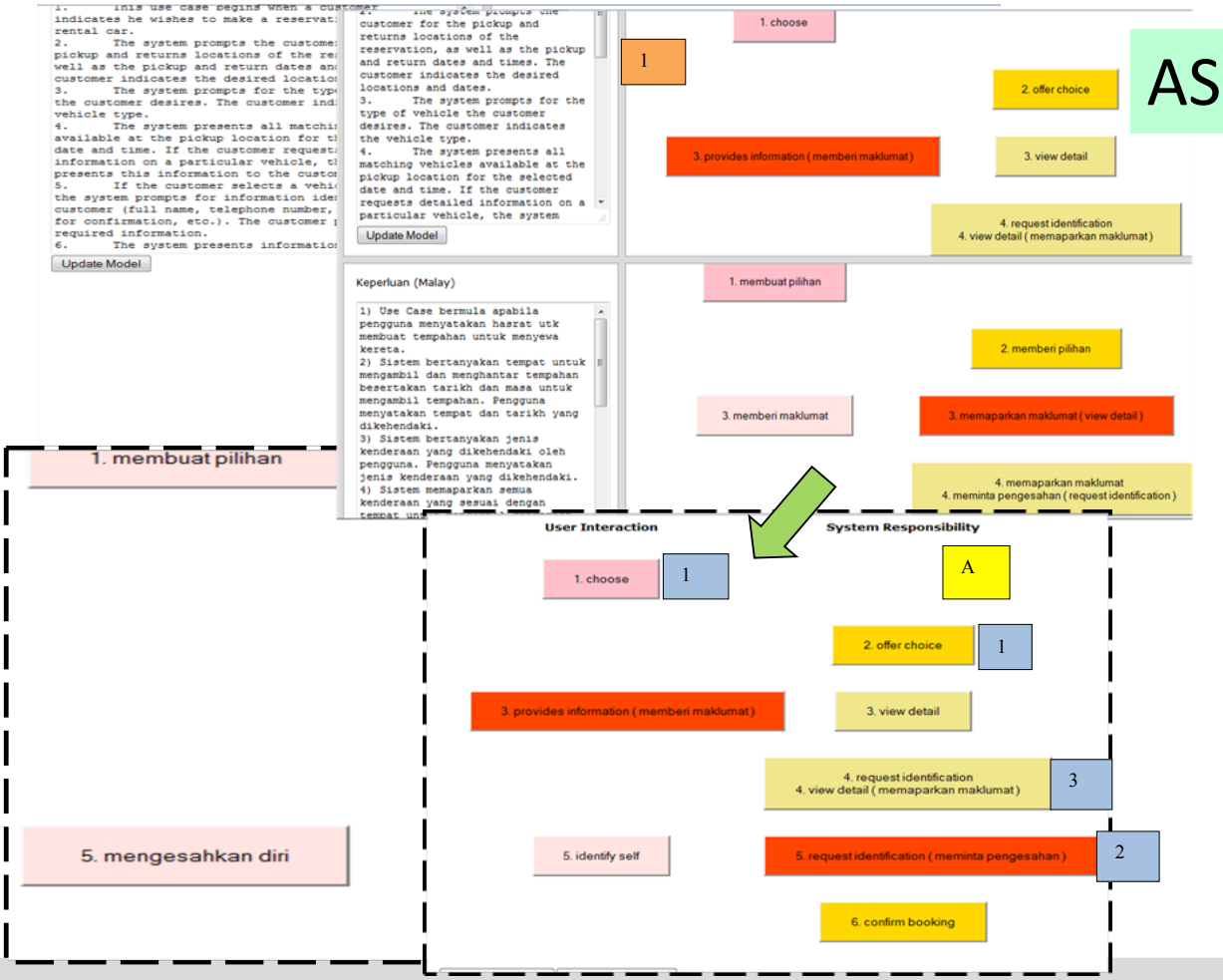
Multi-lingual Requirements Engineering



- Software developed by teams
- Teams may be diverse in many ways
  - Location
  - Language
  - Gender
  - Culture
  - Organization
- Explored one aspect in Malaysian context with multi-lingual teams (also have multi-cultural aspect)
- Added multi-lingual support to Essential use case-based requirements tool

MEReq

English Essential Interaction Patterns Library		Malay Essential Interaction Patterns Library	
Essential Interaction	**Abstract Interaction**	**Essential Interaction**	**Abstract Interaction**
1. Save record 2. Save information 3. Save data	Save information	1. Menyimpan data *(save data)* 2. Menyimpanmaklumatperibadi*(save personal information)* 3. Menyimpanrekodjualan *(save sales record)*	SimpanMaklumat*(save information)*



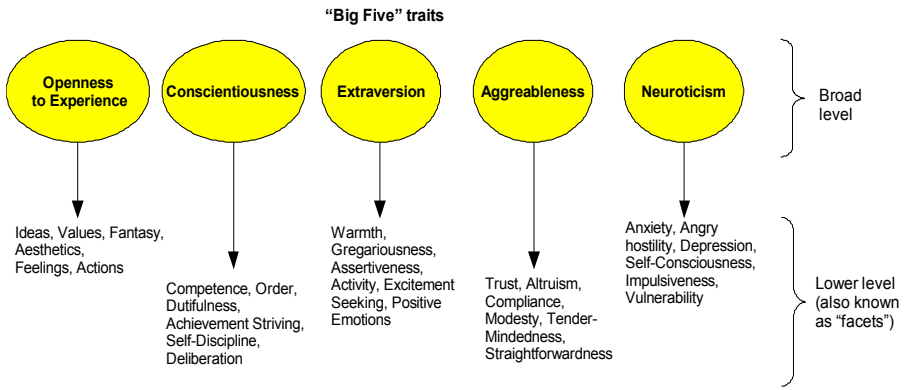
ASEJ2017

Personality impact on aspects of software development

- What impact does personality have on software engineering?
- Pair-programming – influence on the pairs?
  - Set of experiments with student teams @ uni Auckland
- Testing – influence on the individual, the team, the organisation?
  - Surveys, observation, job ad analysis, performance assessment, work logs

Personality & pair-programming (teaching)

ESE2014



Experiment	*Exp 1*	*Exp 2*	*Exp 3*	*Exp 4*
Semester:	Summer 2009	Semester 1, 2009	Semester 2, 2009	Semester 1, 2010
Sample size:	48	214	118	137
Course:	CS101	CS101	CS101	CS101
Subjects:	First year undergraduate	First year undergraduate	First year undergraduate	First year undergraduate
Tutorial settings:	• Compulsory • 2 hours • Closed-lab	• Compulsory • 2 hours • Closed-lab	• Compulsory • 2 hours • Closed-lab	• Compulsory • 2 hours • Closed-lab
Personality factor (IV):	Conscientiousness	Conscientiousness	Neuroticism	Openness to Experience

Personality Factor	Exp.	Correlation (*r*)		
		Assign.	MidTerm	Final
Conscientiousness	1*	0.29**	0.07	-0.05
	2*	-0.03	-0.11	-0.08
	3	0.19**	0.19**	0.15
	4	0.17**	0.19**	0.18**
Neuroticism	1	-0.17	-0.04	-0.03
	2	0.02	-0.04	-0.04
	3*	0.05	-0.01	0.01
	4	0.04	-0.02	-0.00
Openness to	1	0.15	0.35**	0.29**
	2	0.21**	0.13	0.22**
	3	0.01	0.23**	0.15
	4*	0.15	0.18**	0.17**

N(*Exp 1*) = 48; N(*Exp 2*) = 214; N(*Exp 3*) = 118, N(*Exp 4*)=137
(*) Personality factor is controlled
(**) Significant at $\alpha < 0.05$

Tester personality & appraisals

IST2014

DIMENSION 2- BUG REPORT (EASE OF REPLICATION):

DESCRIPTION: The extent to which reported bugs can be easily replicated from the bug reports. Qualities of the bug report to be considered include: the presence of all necessary information; clear description of steps to reproduce the bugs; and the presence of necessary details regarding input and environment. Please consider all the bug reports produced by the software tester during the period of evaluation and select a rating based on the average quality.

INSTRUCTION: Please consider bug reports (both- written and verbal) produced by the software tester being appraised using this dimension and indicate the rating you will give to those reports.

Label	Score	Definition	Rating
Perfect	5	Bugs can be replicated based on the bug reports. Sufficient information about the required sequence of actions, input and environment is provided. No unnecessary information is provided.	○
High	4	Bugs can be replicated based on the bug reports. Necessary information is clear in the report however may not always be to-the-point.	○
Satisfactory	3	Bugs can be replicated based on the bug reports, however not all information is present and clear in the report. Sometimes unnecessary information is provided, or steps to replicate are confused or "mixed up".	○
Difficult	2	It is difficult to replicate the bugs based on the bug reports only. Necessary information is missing or poorly presented, and/or too much unnecessary information is provided.	○
Impossible	1	Completely impossible to replicate the bugs from the information in the bug reports. Necessary information is unavailable.	○

DIMENSION 5- TEST PLANNING:

DESCRIPTION: Frequency of preparing good quality test plans. Quality attributes of a test plan include: efficiency in finding bugs; ability of assessing high risk area; and selection of efficient test strategy.

INSTRUCTIONS: Please consider the test plans made by the software tester being appraised using this dimension and indicate the rating you will give to those plans.

Label	Score	Definition	Rating
Always	5	Consistently prepares satisfactory test plan.	○
Mostly	4	Almost always prepares satisfactory test plan.	○
Usually	3	Sometimes prepares satisfactory test plan.	○
Rarely	2	Test plan is most often not satisfactory.	○
Never	1	Test plan is never of a satisfactory standard.	○

DIMENSION 3- BUG COUNT (COMPARED TO EASE OF FINDING):

DESCRIPTION: Number of bugs found in comparison to difficulty of finding them.

INSTRUCTIONS: In the following table five labels of frequency of finding bugs are defined in the vertical direction (columns) and three levels of bug "difficulty" (how difficult the bug in question is to find - please note that this is a subjective judgement) are defined in the horizontal direction (rows). Please consider the bugs reported by the software tester being appraised and, for each level of difficulty, select the box indicating the relative number of bugs found by this tester. Since the average number of bugs, and difficulty levels, are dependent on the project, this should be decided by the appraiser.

Difficult to find bugs are weighted more heavily in calculating the tester's rating in this dimension.

Label	Bug Count		Very high	High	Average	Low	Very low
Difficulty of finding	Definitions		Found well above average number of bugs	Found above average number of bugs	Found average number of bugs	Found below average number of bugs	Found well below average number of bugs
		score weight	5	4	3	2	1
Difficult	Very difficult to find bugs (Not easily found by all)	0.5	○	○	○	○	○
Normal	Bugs with average difficulty of finding (Needs careful attention)	0.3	○	○	○	○	○
Easy	Obvious bugs	0.2	○	○	○	○	○

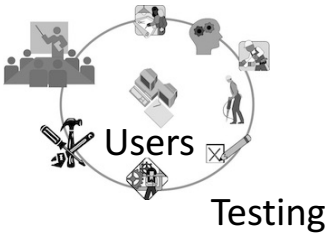
PAF: SECTION FOUR - OVER ALL PERFORMANCE ASSESSMENT

DESCRIPTION: The over all performance assessment score is obtained by dividing the summation of individual scores in seven dimensions by 7.

INTERPRETATION OF OVERALL SCORE: The over all performance assessment score is interpretated according to the following table.

Over all score	Interpretation
0-0.99	Poor
1-1.99	Marginal
2-2.99	Satisfactory
3-3.99	Good
4-5	Outstanding

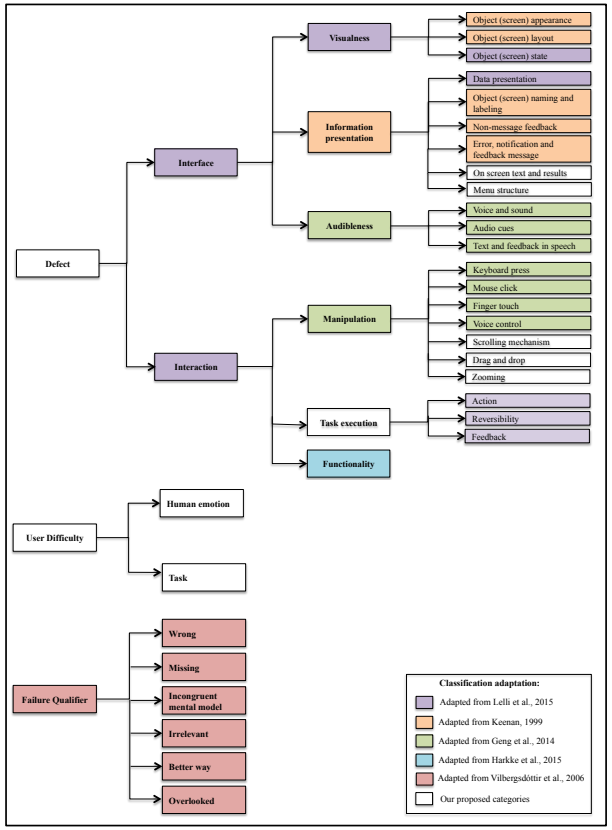
Reporting usability defects

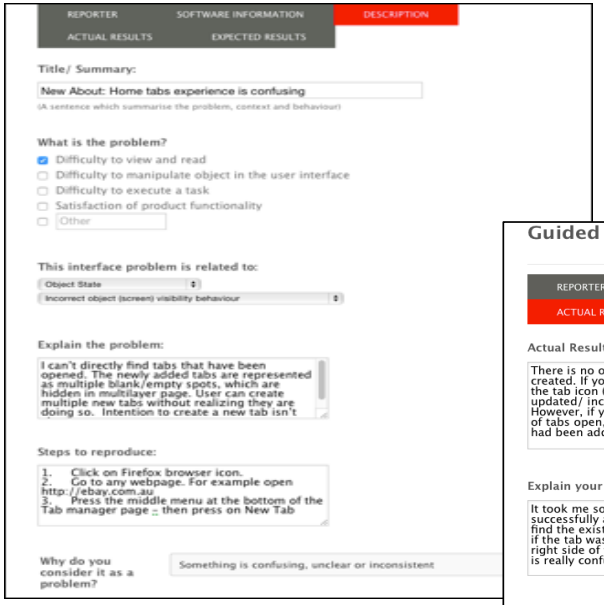


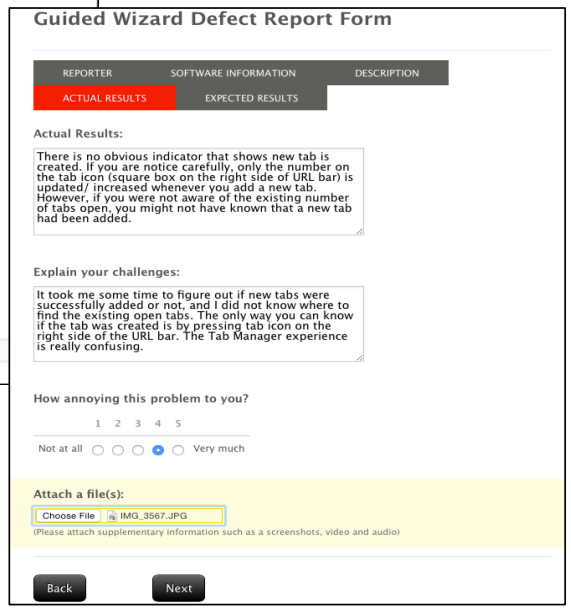
- Software typically has a bunch of “defects”
- Functional and non-functional
- One under-researched non-functional area are usability defects
  - Problems with how users interact with the software
- How do we currently find, report, fix these?
- How can we improve the reporting?
- Better understand current reporting needs: survey, repository mining, observation
- New usability defect taxonomy to better characterise usability defects
- New usability defect reporting tool

Usability Defect Taxonomy & Reporting

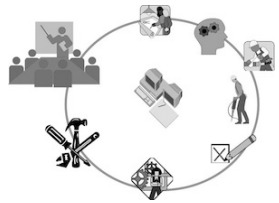
TSE2017





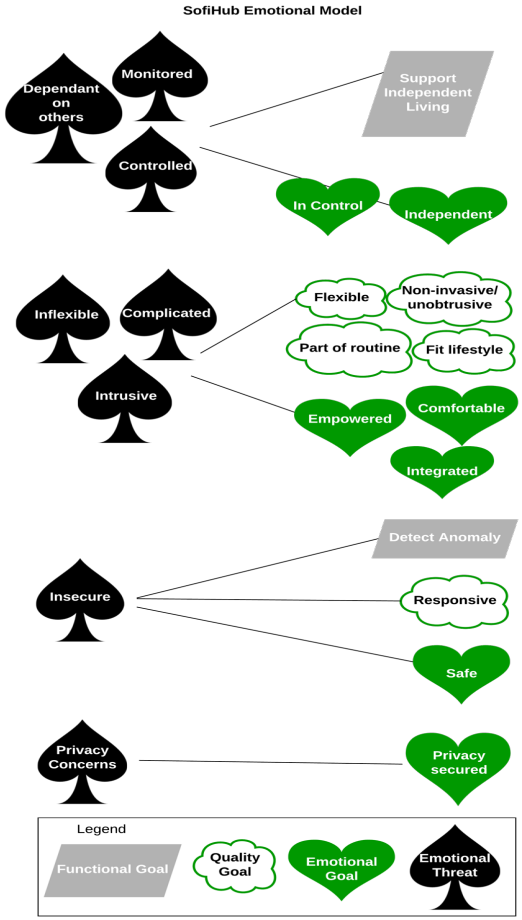


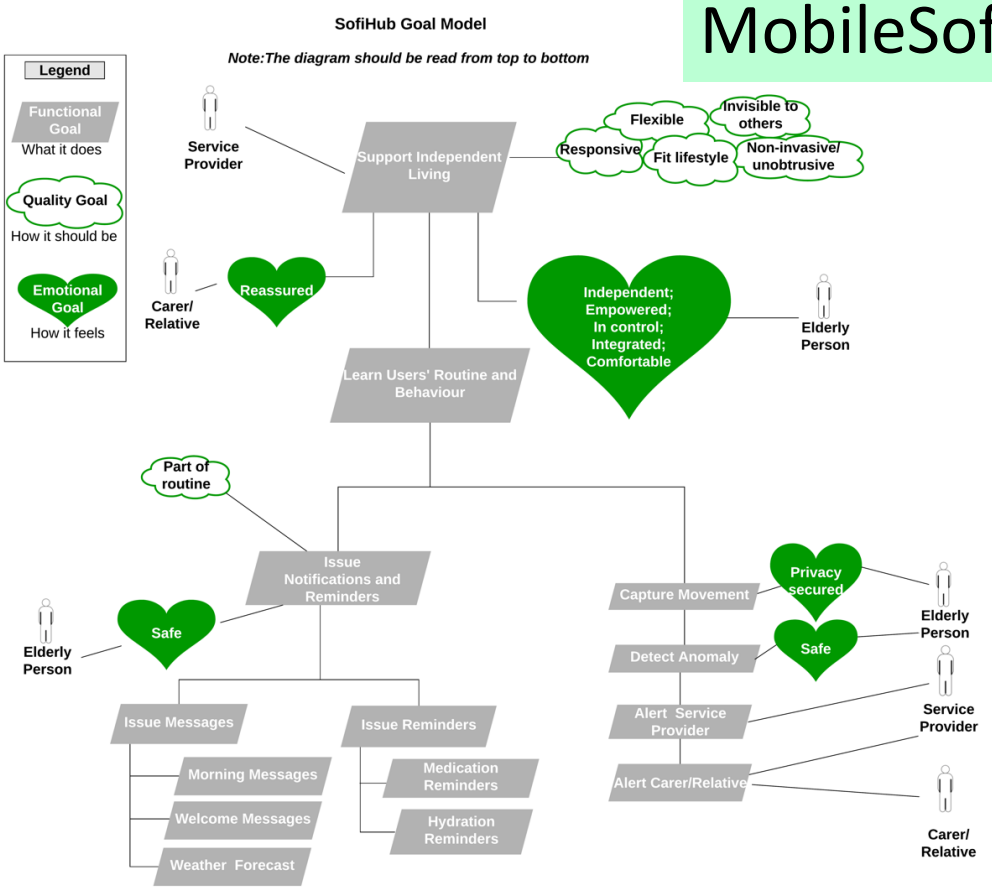
Incorporating end user emotions into software requirements engineering



- People use software
- Software is designed to help people perform tasks, solve problems
- But – people react to software / tasks / situations in various ways
- One (under-researched) way is emotional reactions to software usage
- Incorporating emotions / emotional reactions into software requirements, design, evaluation
- Applying to eHealth systems

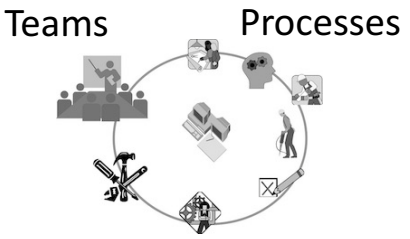
Example: requirements for the Smart Home





MobileSoft2018

Understanding interpersonal issues in agile practices



- Team climate
  - How does a “team climate” impact Agile software development?
  - Can we characterise good (and bad) team climates
  - Extending the TCI (Team Climate Inventory) to Agile Software TCI…
- Recruitment, engagement, translation in ESE research
  - This one is about human reactions to Empirical Software Engineering RESEARCH(ERS)!
  - How do we recruit participants to our ESE studies?
  - How can we keep them engaged, participating, answering surveys etc?
  - How do we translate lessons learned to SE practice?

Team Climate +Agile SE

IST2016

Table 11: Significant Personality Factors Affecting Software Team Performance

Study ID	Personality Factors	
	Positive Impact	Negative Impact
[PTC3][PTC4][PTC12] [PTC21]	Extraversion	N/A
[PTC15][PTC16]	Interpersonal Communication	N/A
[PTC9]	Self efficacy	N/A
[PTC15]	Comfortable and Compromise	N/A
[PTC4]	Mental Ability and Expertise	N/A
[PTC2]	N/A	Self-esteem, locus of control, introversion/extroversion, authoritarianism, dogmatism and dependability

Table 12: Team Climate Compositions

Term	Factors of Composition	Study ID#
Team climate	Vision, participative safety, task orientation and support for innovation	[PTC7][PTC10]
Organizational climate	High standards of work tasks, effective supervision, intrinsic fulfilment and role clarity	[PTC11]
Group cohesion	Members' affiliation with the group, mutual liking, cooperation, and task responsibility	[PTC13][PTC14][PTC17]
Collaboration	Aggression, cooperation, and individuals' affiliation with other individuals	[PTC2][PTC3]
Triggering factors	Play advantages and abilities, job importance, clear work requirements, teamwork and support, commit to doing high quality work, and recognize or praise	[PTC5]
Agile team environment	Whole team involvement, agile values, culture of action and change, and collective thinking	[PTC9]
Through-put	Team members' interaction, exchanging of information, decision-making participation pattern and social support	[PTC19]
Team Processes	Communication, conflict and cohesion	[PTC14]
IT Team climate	N/A	[PTC33]

Table 13: Significant Team climate factors affecting Team Performance

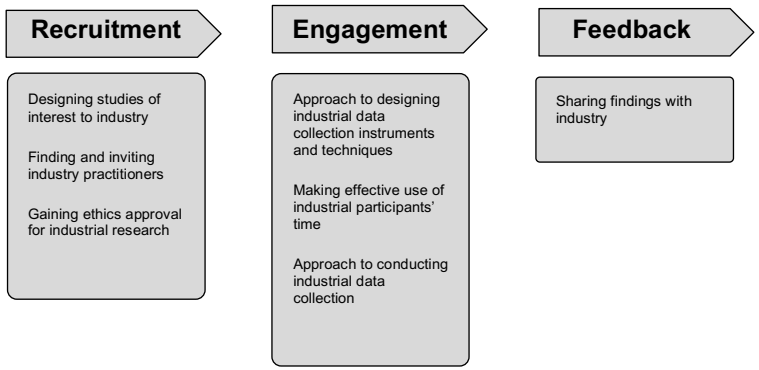
Studies	Significant	Not Significant
[PTC3][PTC9][PTC13]	Collaboration, cooperation, coordination	N/A
[PTC5]	Play advantages and abilities, job importance, clear work requirements, teamwork and support, commit to doing high quality work, and recognize or praise	N/A
[PTC7]	Commitment, trust, and coordination	N/A
[PTC9]	Agile values(trust, openness and respect) and collective thinking	N/A
[PTC2][PTC11]	Role allocation	N/A
[PTC7] [PTC10]	Participatory safety	N/A
[PTC18]	Composition and project task	N/A
[PTC17]	User representativeness and team members' involvement in system design	Cohesion
[PTC32][PTC34]	Communication	N/A

Table 14: Measures used for Team Performance

Study ID#	Measures
PTC4	Correctness, Duration, Methodology, Extensibility, Cost Effectiveness, Redesign, Regression grade
PTC7	Hoegl and Gemuenden (2001)
PTC13	Group Characteristics(Group Cohesiveness, Group Experience and Group Capability)
PTC14	Cost, Time and Scope
PTC19	Quantity, Quality, Speed, Customer Satisfaction Degree
PTC17, PTC18, PTC20	Henderson & Lee (1992)
PTC28	Jiang et al. (1997)
PTC35	Effective leadership, Intra-team communication, Group cohesion and Personality heterogeneity

Empirical Software Engineering Research + practitioners

IST2018



Industrial Research Phase	Challenges/Questions	Recommendations
RECRUITMENT	Designing studies of interest to industry *"Why aren't they interested in my really important software engineering research?"*	• Network with local practitioner community to identify their interest and refine research focus accordingly • Pilot study early to acquire practitioners' interest • Use practitioner feedback to guide future studies
	Finding and Inviting Industry Practitioners *"I don't know them and they don't know me"*	• Get genuinely involved with and contribute to the local practitioner community to build a strong reputation as a genuine researcher and contributor. • Approach managers and team coaches as they are critical source of access to recruit more individuals in their teams, and sometimes the full teams. • Approach online groups through moderators to improve authenticity. • Craft the call for participations (CfPs) carefully to avoid a 'spam effect' • Prepare a small invitation email with catchy slogan to attract participation • Hire enumerators where necessary to help recruit participants • Perform snowball sampling (or word of mouth
ENGAGEMENT	Approach to designing industrial data collection instruments and techniques *"How do I design data collection instruments and techniques that promote industrial engagement?"*	• Perform pilot data collection and approaching industrial participants to improve industrial relevance • Design demographic surveys to capture basic details prior to the main data collection session to customize and make the most of face-to-face time (primarily applicable for interview-based and observational studies) • Questions should be designed to achieve high clarity to help elicit useful responses (simple language, clear instructions and avoiding jargon) • Surveys should aim for an appealing presentation, and adequate layout to improve completion rates • Specialised data collection tools must be secure, reliable and accessible as well as professional-looking to attract and sustain industry interest
	Making effective use of industrial participants' time *"How do I make the most out of my industrial participant's time?"*	• Be flexible with meeting schedules to accommodate busy professionals • Schedule one or two additional backup slots in case of schedule changes • Schedule observations between/around interviews on site to utilize participant's time effectively • Ask for the minimum data as needed to answer research questions to prevent participants feeling overwhelmed
FEEDBACK TO INDUSTRY	Sharing findings with Industry *"What mechanisms can I use to share research findings with my participants and other industry practitioners?"*	• Various industry-friendly formats and mediums should be employed to share findings with the industry, including short videos, posters, brief reports of main takeaways, and talks or presentations to industry at industry-focused events to meet industry standards and expectations • Results should be shared after all data collection at a given company is complete so as not to bias other participants from the same company.

Challenges ; Outstanding issues

- Often software engineers don't understand / appreciate human aspects of SE
- Neither it seems do MBIE (NZ) or ARC Assessors…. ☹
- Designing and conducting experiments is hard, time-consuming
- Often need access to practitioners ; convincing them/their bosses can also be a challenge
- Many issues not yet well explored, but increasing interest in SE community
- I find them more challenging – but also in many ways more interesting – projects than the purely technical ones I do
- Recruiting (very good) students / post-docs to work on can be hard, but I've been pretty lucky to date…
- IMO – good research in these areas can make a major difference to pracitce

Future work

- Adding Emotions, accessibility, personalilty etc -> UML etc models
- Capturing, using further human-centric issues: values, emotions, usability, accessibility, culture, language, gender, age, … & evaluating software for these
- Incorporating multi-lingual, multi-cultural aspects into requirements, design
- Deep learning + design critics + PM
- Agile SE Team Climate Inventory & applying in practice
- Personality of requirements engineers, software architects, project managers
- DSVLs for Big Data applications, end user config incl security
- Better principles, tools for human-centric DSVL design & evaluation

Summary

- Human aspects of Software Engineering are fascinating!!
- There is lots of scope for work here
- Can apply other discipline approaches, knowledge – Information Systems, Social Sciences, etc
- Ultimately humans PRODUCE software and humans USE software
- Incorporating human perspectives critical to improve software and its production

Questions…

References

- Grundy, J.C, Hosking, J.G., Amor, R., Mugridge, W.B., Li, M. Domain-specific visual languages for specifying and generating data mapping system, Journal of Visual Languages and Computing, vol. 15, no. 3-4, June-August 2004, Elsevier, pp 243-263
- Avazpour, I., Grundy, J.C., Grunske, L. Specifying Model Transformations by Direct Manipulation using Concrete Visual Notations and Interactive Recommendations, Journal of Visual Languages and Computing, Volume 28, June 2015, Elsevier,pp 195–211.
- Abizer Khambati, John Grundy, John Hosking, and Jim Warren, Model-driven Development of Mobile Personal Health Care Applications, In Proceedings of the 2008 IEEE/ACM International Conference on Automated Software Engineering, L'Aquilla, Italy, 15-19 September 2008, IEEE CS Press
- Kamalrudin, M., Grundy, J.C., Hosking, J.G., MaramaAIC: Tool Support for Consistency Management and Validation of Requirements, Automated Software Engineering, Springer, 2017, vol 24, no 1, pp. 1-45
- Sallah, N., Mendes, E., Grundy, J.C. Investigating the effects of personality traits on pair programming in a higher education setting through a family of experiments, Empirical Software Engineering, vol. 19, no. 3, Springer, 2014, pp. 714-752.
- Kanij, T., Merkel, R., Grundy, J.C. Performance Appraisal of Software Testers, Information and Software Technology, Elsevier, vol. 56, no. 5, May 2014, Pages 495–505
- Yusop, N.S.M., Grundy, J.C., Vasa, R. Reporting Usability Defects: A Systematic Literature Review, IEEE Transactions on Software Engineering, vol. 43, no. 9, 2017, pp. 848-867.
- Ali, N.M., Hosking, J.G., Grundy, J.C., A Taxonomy and Mapping of Computer-based Critiquing Tools, IEEE Transactions on Software Engineering, vol. 39, no. 11, November 2013.
- Grundy, J.C. Abdelrazek, M., Kissoon, M., Vision: Improved development of mobile eHealth applications, IEEE/ACM International Conference on Mobile Software Engineering and Systems (MobileSoft 2018), 27-28 May 2018, Gothenberg, Sweden, ACM Press.
- Salleh, N., Hoda, R., Su, M.T., Kanij, T. and Grundy, J.C. Recruitment, Engagement and Feedback in Industrial Empirical Software Engineering Studies, to appear in Information and Software Technology, Elsevier. –
- Soomro, A.B., Salleh, N., Mendes, E., Grundy, J.C., Burch, G., Nordin, A., The Effect of Software Engineers' Personality traits on Team Climate and Performance: a Systematic Literature Review, Information and Software Technology, vol 73, Elsevier, pp 52-65.