

Soft Skills are Not Soft! Collective Empathy – a New Disposition for Software Engineering Teams?

Associate Professor Tony Clear

Department of Computer Science and Software Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz

Draws on Clear, T. (2024). Software Developers and Collective Empathy – Can they be Disposed to Care? [In Press]. *ACM Inroads, TBA*.

Mihi

- Hāere mai, Haere Mai, Haere Mai.
- Tēnā koutou katoa.
- Ko Tony Clear taku ingoa.
- Nō Pōneke ahau.
- Ko Maungakiekie taku maunga.
- Ko Waitematā taku moana.
- I te taha o taku matua, no Enniscorthy Ireland ahau.
- I te taha o aku whaea, no Cork Ireland ahau.
- Ko Tainui raua ko Ngapuhi nga iwi o nga mokopuna
- Tēnā koutou, Tēnā koutou, Tēnā tatou katoa.

Outline

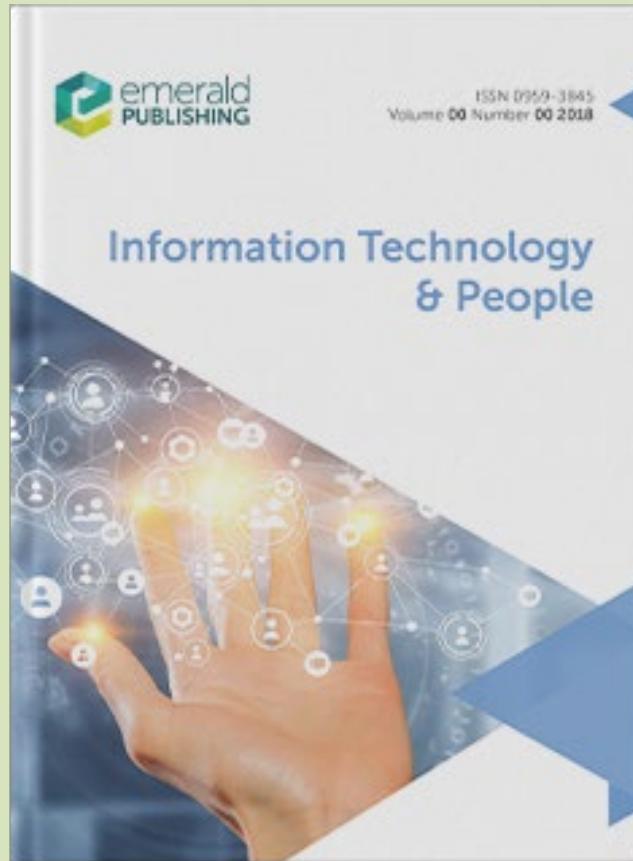
- Software, People and Teams
- “Soft Skills”
- Computing Competencies & CC2020
- Software Developers - Skills Requirements
- Empathy & Collective Empathy - Practitioners' views
- Teaching dispositions & Empathy as a disposition
- Job advertisements – what do employers want?
- Assessment, Collective Empathy, Competencies and Affect
- Dispositions, Assessment and Internalization
- Conclusion & key take aways
- References

Software



People

- Software
- People



People

Three Categories of Bad Boss

"A bad boss.
Me? You really
think so?"



Doesn't know
he's bad.

"I could do
better. I
just wish I
knew how!"



Knows he's bad.
Wants to improve.

"It's my way
or the
highway!"



Doesn't care
either way.

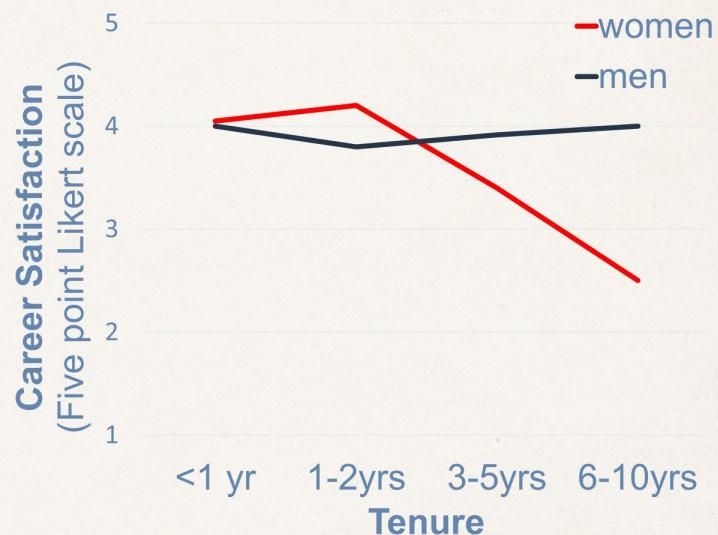
Tech's Diversity Problem

Diversity
problem

only 23%
w o m e n



29% leave in first 5 years



Career Satisfaction over time by Gender

James, T., Galster, M., Blincoe, K., Miller, G. (2017)

Empathy?



AUT Students and Healthy Together Technology

This year, the Healthy Together Technology team has been supporting a group of AUT Bachelor of Computer and Information Management students.

The students all majored in either Software Development or Service Science and have been completing their final year projects. They chose to work with CM Health on two projects – a chat-bot for the main CM Health website to help answer FAQs for the public, and a Wayfinding App to assist patients and whaanau to navigate around the Middlemore Hospital site, particularly to the Galbraith Infusion Centre and the Cardiac investigation unit.

They have been mentored by Megan Milmine, Deputy CIO, and Sally Dennis, IS Clinical Change Manager. Read more [here](#).



- There is **no single definition of soft skills** (“transferable”, “core” or “qualitative” skills) and no agreed set of soft skills [28].
 - Soft skills in our work are **“intra- and inter-personal (socio-emotional) skills”** essential for personal development, social participation and workplace success [29] and include “nontechnical, domain-independent skills that underpin our behavior in the workplace [28]”.
-
- Galster, M., Mitrovic, A., Malinen, S., Holland, J., & Peiris, P. (2023, 2023/08/01/). Soft skills required from software professionals in New Zealand. *Information and Software Technology*, 160, 107232. <https://doi.org/https://doi.org/10.1016/j.infsof.2023.107232>

SOFT SKILLS BROADLY DEFINED

- Some definitions **inclusive of**
 - “**personality traits, goals, motivations, and preferences valued in the labor market [30]**”
- ...define soft skills
 - “as the **combination of the abilities, attitudes, habits, and personality traits** that allow people to perform better in the workplace, complementing the technical skills [...] and influencing the way they behave and interact with others” [31].
- Galster, M., Mitrovic, A., Malinen, S., Holland, J., & Peiris, P. (2023, 2023/08/01/). Soft skills required from software professionals in New Zealand. *Information and Software Technology*, 160, 107232. <https://doi.org/https://doi.org/10.1016/j.infsof.2023.107232>

- ...**exclude personality traits** (e.g., neuroticism, extraversion, agreeableness [32]),
 - because personality traits **tend to be more stable over time**, depend on the social environment, and are **more difficult to train** (we may improve communication skills but still remain introverts) [33].
 - As Liang et al. argue, a skill is “**anything that an individual can actively work to improve on** [34]”.
-
- Galster, M., Mitrovic, A., Malinen, S., Holland, J., & Peiris, P. (2023, 2023/08/01/). Soft skills required from software professionals in New Zealand. *Information and Software Technology*, 160, 107232.
<https://doi.org/https://doi.org/10.1016/j.infsof.2023.107232>

SOFT SKILLS - FREQ

Galster, M., Mitrovic, A., Malinen, S., Holland, J., & Peiris, P. (2023, 2023/08/01/). Soft skills required from software professionals in New Zealand.

Information and Software Technology, 160, 107232.

<https://doi.org/https://doi.org/10.1016/j.infsof.2023.107232>

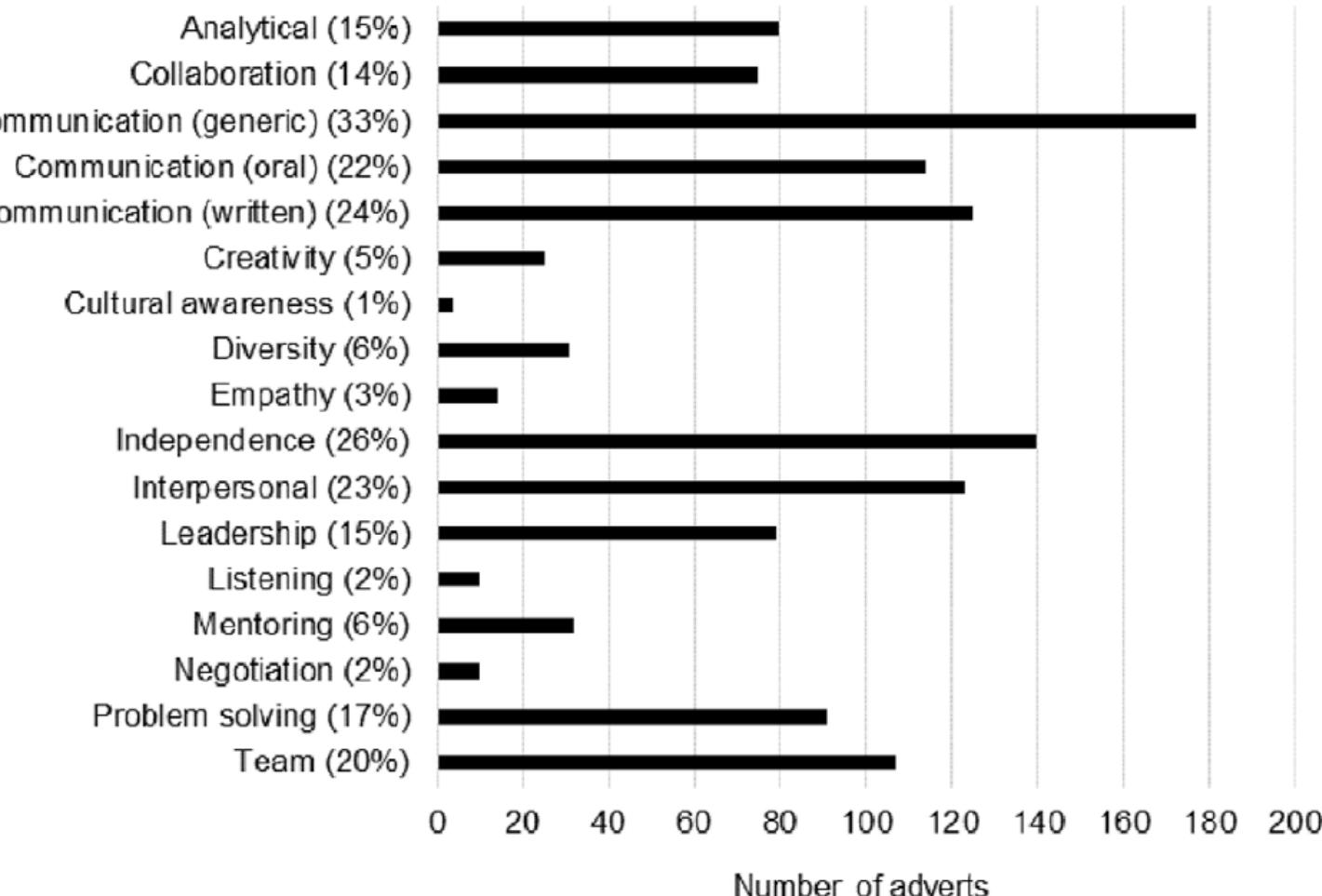


Fig. 2. Frequency of soft skills (alphabetical order).

DEFINITIONS AND INCONSISTENCY

- The Galster et al., (2023) approach – inconsistency between knowledge and skill models in comparisons,
 - SWEBOK; SFIA behavioral factors; Occupational information network (O*Net);
 - E.g. *judgement, decision making* – n/a task not skill ?
- Compared against 5 previous research studies
 - Approx 80% of “soft skills” not mapped
- **Defining ‘soft skills’ is hard!**
- The CC2020 Competency Model
- Breaks elements out
- Enables distinctions between
 - Professional Knowledge Areas
 - Dispositions
- Galster, M., Mitrovic, A., Malinen, S., Holland, J., & Peiris, P. (2023, 2023/08/01/). Soft skills required from software professionals in New Zealand. *Information and Software Technology*, 160, 107232.
<https://doi.org/https://doi.org/10.1016/j.infsof.2023.107232>

CC2020 PROJECT

CURRICULUM

DIRECTIONS



A Computing Curricula Series Report
2020 December 31

Computing Curricula 2020

CC2020

Paradigms for Global Computing Education

encompassing undergraduate programs in
Computer Engineering
Computer Science
Cybersecurity
Information Systems
Information Technology
Software Engineering
with data science

Clear, A., Parrish, A., & CC2020 Task Force. (2020). *Computing Curricula 2020 - CC2020 – Paradigms for Future Computing Curricula* Retrieved from New York:

<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>



Association for
Computing Machinery



IEEE
computer
society

CC2020 PROJECT MODEL

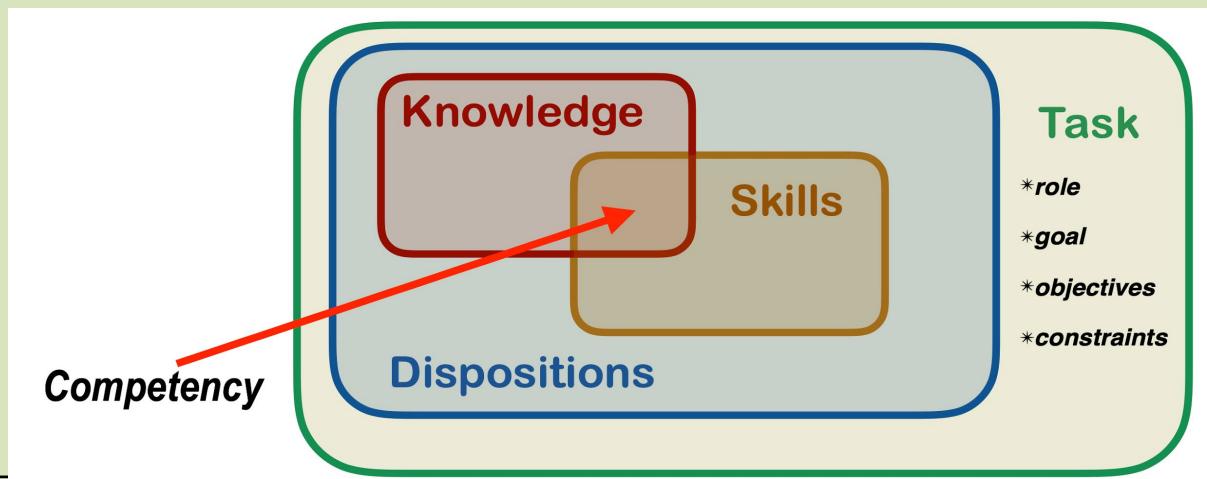
ASPECTS OF KNOWLEDGE BASED LEARNING [Established Curriculum Paradigm]

- KA's [knowledge areas]
- KU's [knowledge units]
- LO's [Learning outcomes]

COMPONENTS OF COMPETENCY BASED LEARNING [CC2020 MODEL]

– **Competency = [Knowledge+ Skills+ Dispositions] in Task**

- A competency structure (see Fig. 1) shows knowledge, skills, and dispositions that are observable in the accomplishment of a task, a task that prescribes purpose within a work context [24].



CC2020 – COMPETENCY BASED LEARNING

- Knowledge is the “*know-what*” dimension of competency that is factual.
- Skills express the “*know-how*” and usually develop over time and with practice.
- Dispositions frame the “*know-why*” dimension of competency, which prescribes a requisite character or quality in task performance.
- Task is the construct that frames the skilled application of knowledge and makes dispositions concrete.

VOCABULARIES

SUBJECT/CONTENT KNOWLEDGE

3.2.1 *Knowledge Vocabulary for Computer Science Competencies.*

The CS2013 document divides computer science knowledge into 18 KAs subdivided into 225 KUs. For example, the Software Engineering KA is divided into ten KUs:

- (1) SE/Software Processes
- (2) SE/Software Project Management
- (3) SE/Tools and Environments
- (4) SE/Requirements Engineering
- (5) SE/Software Design
- (6) SE/Software Construction
- (7) SE/Software Verification and Validation
- (8) SE/Software Evolution
- (9) SE/Software Reliability
- (10) SE/Formal Methods

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

PROFESSIONAL & FOUNDATIONAL KNOWLEDGE

Table 4: Professional & Foundational Knowledge Areas extending CS2013

Tag	Knowledge Area
PK-1	Oral Communication & Presentation
PK-2	Written Communication
PK-3	Problem Solving and Trouble Shooting
PK-4	Project/Task Organization & Planning
PK-5	Collaboration and Teamwork
PK-6	Research and Self-Learning
PK-7	Multi-Task Prioritization & Management
PK-8	Relationship Management
PK-9	Analytical and Critical Thinking
PK-10	Time Management
PK-11	Quality Assurance / Control
PK-12	Mathematics and Statistics
PK-13	Ethical Intercultural Perspectives

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

SKILLS

Table 4.3. Levels of Cognitive Skills Based on Bloom's Taxonomy

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Exhibit memory of previously learned materials by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, and giving descriptions.	Solve problems in new situations by applying acquired knowledge, facts, techniques, and rules in a different way.	Examine and break information into parts by identifying motives or causes; make inferences and find evidence to support solutions.	Present and defend opinions by making judgments about information, validity of ideas, or quality of material.	Compile information together in a different way by combining elements in a new pattern or by proposing alternative solutions.

Clear, A., Parrish, A., & CC2020 Task Force. (2020). *Computing Curricula 2020 - CC2020 – Paradigms for Future Computing Curricula* Retrieved from New York:
<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>

CC2020 DISPOSITIONS

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

ON TEACHING DISPOSITIONS?

- in discussing whether dispositions could be taught [Clear, 2021], I noted that “... *a disposition* “concerns not what abilities people have, **but how people are disposed to use those abilities**” [Schussler, 2006].

So here we are talking about a mindset and attitudinal dimensions, which raises the question can a disposition be taught or is it some innate part of a person’s character?”

[Clear, 2021]

Clear, T. (2021). THINKING ISSUES: Is Agility a Disposition and Can it be Taught? . *ACM Inroads*, 12(1), 13-14.
doi:10.1145/3447870

SPECIFYING COMPETENCIES?

KA-Ref#	Title	Competency Statement	Dispositions	Knowledge-Skill Pairs				Traced Competencies	Notes
				KU / Broader Knowledge Description	Topic/LO Description	Knowledge Element	Paired Skill Level		
Our reference number goes here. E.g., "KA-##"	Short Title	Competency statement goes here (Natural Language) that embeds the task/context and suggests the K-S-D breadth and depth	List top applicable dispositions: <ID+descriptor>. Note that dispositions are not aligned with KS-pairs.	CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level	List other competency reference #'s that this competency expects or depends upon. Often null.	Any notes or comments
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		Often null
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		
				CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	New IDs for extensions to CS2013	Bloom's Level	Note deviations from Bloom's level in CS2013	
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		

Figure 5: Competency specification in a tabular format

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

EXPRESSING COMPETENCIES?

“...most appropriate to express competencies at the level of entire KAs, **from the point of view of a graduate or professional.**

When we “designed” a competency statement, we focused on a particular knowledge area and **looked for collections of topics and/or learning outcomes (LOs) that could be observed in tasks and at particular skill levels.**

...there was significant variance in how this was approached, and how the resulting competency was formulated. **Dispositions were often implied at first, but refinement of the statements made these both more clear and more relevant.”**

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020*

ACM Conference on Innovation and Technology in Computer Science Education. New York: ACM.

A SAMPLE COMPETENCY STATEMENT

Table 7: IS-4: IS/Basic Machine Learning

Statement	Dispositions
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.	D-6 (Responsible) D-10 (Meticulous)

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

Table 7: IS-4: IS/Basic Machine Learning

Statement		Dispositions		
Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.			D-6 (Responsible)	D-10 (Meticulous)
IS/Basic Machine Learning	List the differences among the three main styles of learning: supervised, reinforcement, and unsupervised.	IS-BML-1	B-II	
	Identify examples of classification tasks, including the available input features and output to be predicted	IS-BML-2	B-II	
	Explain the difference between inductive and deductive learning	IS-BML-3	B-II	
	Describe over-fitting in the context of a problem	IS-BML-4	B-II	
	Apply the simple statistical learning algorithm such as Naive Bayesian Classifier to a classification task and measure the classifier's accuracy	IS-BML-5	B-III	
Prof. & Foundational Knowledge	Analytical and Critical Thinking	PK-9	B-III	
	Mathematics and Statistics	PK-12	B-III	
IS/Basic Machine Learning	Definition and examples of broad variety of machine learning tasks, including classification	IS-BML-a	B-II	
	Inductive Learning	IS-BML-b	B-II	
	Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees	IS-BML-c	B-III	
	The over-fitting problem	IS-BML-d	B-II	
	Measuring classifier accuracy	IS-BML-e	B-III	

'Skills' Requirements for LERO Practitioners - Junior to Senior

Johnny's study contrasted how “*Skills Requirements*” [or capabilities or competencies] for practitioners develop in the transition from Junior Software Developer to Senior Developer.

Reviewed job advertisements contrasting stated expectations.

In developing *competency statements* from this data source - **two key insights were derived:**

- for *professional knowledge areas* such as “teamwork” (which did not easily fit within a cognitive taxonomy) Johnny struggled to map them to the ”skills” aspect of the identified competency so rated them at level 0 of the ‘skills’ taxonomy;
- in subsequently allocating *dispositions*, distinguishing between *skills* element of the CC2020 competency model and junior and senior expectations of the degree of exhibition of a *disposition* has posed some challenges.

EMPATHY

- Empathy - phenomenon studied across many disciplines,
- psychology, nursing and medicine prominent
- “*a complex phenomenon with no one unified definition*” [10]
- In software, studies on empathy limited
- “*has been recognised as a human aspect that can help to understand software developer and stakeholder human interactions*”. [11]
- One definition of empathy “***the ability to experience the affective and cognitive states of another person, while maintaining a distinct self, in order to understand the other***” [12]

MEASURING EMPATHY

- approaches vary,
- two key methods identified:
 - *1) measuring empathy via self-assessments, and*
 - *2) via neurophysiological examination methods of studying different brain activities using brain images [10]*
- Observations of empathy behaviours through verbal and non verbal behaviours also conducted
- In a multi-method study a self-assessment instrument “*the questionnaire of cognitive and affective empathy (QCAE)*” deemed to best fit *understanding how empathy is practised between software developers and end users* [10]

COLLECTIVE EMPATHY

- beyond empathy at the individual level,
- for instance, investigating “*interpersonal empathy in work groups or teams...*,
- ***few studies have suggested empathy at the team level as a collective phenomenon, that is, collective empathy, in software development teams***”. [1]
- *Akgun and colleagues*
- ***“collective empathy exists when all team members (e.g., programmers, system developers, testers) perceive or imagine their teammates’ affects, partially feel what others are feeling, and then demonstrate prosocial behaviors within their team during the project”*** [1].

Dimensions of Collective Empathy

Within Software Development Project Teams

Concept	Definition	Realised Through
Cognitive empathy	<i>ability of team members to understand the feelings of others within the team.</i>	<i>taking the perspective of the other</i>
Affective empathy	<i>the affective reaction team members have to the affective states of their teammates</i>	<i>“we feel with you” or “we are feeling for you”</i>
Behavioral empathy	<i>ability of team members to respond to the feelings of others within the team</i>	<i>helping behavior, usually through communicative reaction to others’ feelings, enabling troubled team members to feel they are in an empathic team environment</i>

DEV community

analysis [8,15]

Research Question	Themes Identified	Selected Illustrative Quotes
RQ1: The meaning of empathy for software Practitioners	Understanding means comprehending another individual's thoughts and feelings	“Empathy means understanding how someone else is thinking and feeling.”
	Compassion refers to care for the well-being of others	try to care about what is important for the client as we care about what is important for us.
	Perspective taking means adopting someone else's subjective perspective	“empathy comes into picture where we should try to see it from other person point of view.”
	Embodiment entails imaginatively placing one's self in another's position	empathy is simply the ability to put yourself in the shoes of another human being.”
	Emotional sharing means the ability to share the emotional state of others.	[empathy is] “the ability to understand and share the feelings of another.”

DEV community
analysis [8,15]

Research Question	Themes Identified
RQ2: The value of empathy for software Practitioners	<i>We find that software practitioners consider empathy as important, undervalued, needed and wanted.</i>
Tony	

DEV community
analysis [8,15]

RQ3: Applying empathy in software engineering activities	Communication and Collaboration - practitioners consider empathy useful or important when communicating with colleagues, clients, and users.	software developers play different roles in their organizations...would involve talking to people and wherever you need to deal with people, empathy can play a key role
	Coding - practitioners discuss the need for empathy when they are coding or maintaining the code of other developers,	Something that I learned as the time passes was to have empathy with another developer's code."
	Management and Leadership - practitioners, view the need for empathy to successfully coordinate, communicate, motivate, and work with their teams and colleagues.	"To make an impact, our SRE leaders need to lead with empathy and help the rest of the organization engineer with empathy."
	Code review - practitioners consider empathy necessary in the code review process	Empathy for other engineers - ... Be mindful that...asking for their input is essentially asking them for their time

DEV community
analysis [8,15]

RQ4: Empathetic practices in Software Engineering	adopting good programming practices is related to multiple best coding practices. Practices mean caring for fellow developers and future maintainers of the code	Having a consistent code style is empathetic because it allows your teammates to understand the code faster."
	Understanding others means trying to understand what others think or feel.	"Each person has to act with empathy toward their frustrated users. Understand them. Care about them."
	Being compassionate relates to showing compassion towards others, being polite and kind to others, and communicating with kindness in a nonviolent way	Compassionate coding might, however, educate you to notice things you did not notice in the past. It might eventually show you how some of your actions cause suffering for others or for yourself."
	Being mindful means being aware of interactions with others,	by being mindful of situations where empathy is important, we can all develop this vital quality."
	Testing relates to automating tests to verify that the code works.	When we write tests ...we're making sure that the future maintainer of our code can feel safe about modifying it."

analysis [8,15]

RQ5: Effects of practicing empathy in software engineering	quality improvement is related to better software quality, better code, code that is easier to maintain, to read, and to understand	By focusing on empathy in the code we write, we will produce higher quality software as a byproduct."
	The effect better products means products with fewer bugs and that are better suited to meet the users' needs.	remembering the human beings using our software focuses our attention and helps prioritize our efforts, ultimately resulting in products better suited to our users."
	The effect build trust means creating a culture of trust in the workplace, where failures and mistakes are accepted	Cultivating a culture where failure is embraced and experimentation is cultivated is important and essential".
	The effect better work environment relates to creating a better, healthier work environment with fewer conflicts	If you are a fellow developer or a lead and if you practice being empathetic, you can create a better work environment for that person and also help him/her grow eventually
	Improve stakeholders' communication refers to more effective communication with clients, team members, and colleagues, in interviews, and between managers and the team.	Remembering the human beings using our software focuses our attention and helps prioritize our efforts, ultimately resulting in products better suited to our users...You Will Communicate More Effectively With Designers and Product Owners."

RETURNING TO DISPOSITIONS

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM

ON TEACHING DISPOSITIONS?

- “... *a disposition* “concerns not what abilities people have, but how people are disposed to use those abilities” [Schussler, 2006].

So here we are talking about a mindset and attitudinal dimensions, which raises the question can a disposition be taught or is it some innate part of a person’s character?” [Clear, 2021]

Clear, T. (2021). THINKING ISSUES: Is Agility a Disposition and Can it be Taught? . *ACM Inroads*, 12(1), 13-14.
doi:10.1145/3447870

EMPATHY AS A DISPOSITION?

- Dispositions such as ‘being proactive’ not cut and dried abilities, but “*imply a tendency or an inclination to act in certain ways as determined to be appropriate in the situation*” [9].
- relating that work to the idea of ‘empathy’, we can see that ‘acting with empathy’ is not simply an ability, but possesses the essential qualities of a ‘disposition’, namely being dependent on how people are disposed to act empathetically and also on them having the discretion to decide when and how to act with empathy.

EMPATHY VALUED BY EMPLOYERS?

- studying job advertisements to determine global software engineering capabilities desired by employers, confirmed acting with empathy seen as highly valued.
- “The attributes such as *Mentorship, Customer Engagement, Collaboration, Leadership, Interpersonal Skills, Knowledge Sharing, Relationship Building, Team Player, Respectful*, as...advertised by many companies having global clientele does allude towards a possible distributed or global DevOps role” [14].
- focus DevOps and global teams, but dispositions (termed ‘attributes’ in that study) listed above, have a strong fit with the concept of *collective empathy*.

COLLECTIVE EMPATHY AND DISPOSITIONS?

- Relating to CC2020 [5], and its competency-based approach
- vocabulary of dispositions defined [5],
- two closest matches to the concept of *collective empathy* were ***Collaborative***, and ***Responsive*** defined below:
- ***Collaborative/Team Player/Influencing***. *Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal,*
- ***Responsive/Respectful***. *Reacts quickly and positively. Respects the timing needs for communication and actions need to achieve the goals of the work.*

JUNIOR AND SENIOR DEVELOPER JOB ADS – FINDINGS



- dispositions such as being a *team player* or having a *propensity to share knowledge with others* which mapped to *collective empathy* were identified as being valued:
- “*most frequent dispositions are (collaborative), (responsible) and (purpose-driven). Junior job ads focus more on being collaborative, whereas senior job ads also put emphasis on being responsible and purpose-driven on top of being collaborative.*”[13].

JUNIOR AND SENIOR DEVELOPER JOB ADS – FINDINGS 2

- “mentorship appears frequently in senior level job ads, and is exclusive to senior developers. This suggests that mentorship is a desired and distinctive quality of senior software developers compared to junior developers, and is a skill that aspiring senior developers should aim to hone” [13].
- “When it comes to disposition, being collaborative often comes with being responsive, since these two form the basis of communication skills.” [13]

PRODUCT ROLE JOB ADS – FINDINGS

- Table 3. Themes categorised within the study of the *Skills of Product Managers, Product Owners, and Business Analysts in New Zealand, Australia, and India* [2]

Intrapersonal	Cognitive	Managerial	Technical	Knowledge Area	Interpersonal

the *intrapersonal* theme tended to strongly map to dispositions, with the term '*empathetic*' now occurring explicitly in the job advertisements, as opposed to its earlier implicit presence in advertisements.

PRODUCT ROLE JOB ADS – FINDINGS

examples of Job Ads with differing forms of *collective empathy*, for specific roles [2]

- *Product Manager Skills Required in New Zealand* According to 15 Ads on Seek.co.nz –
[Intrapersonal] *Empathetic (3)* 15 Job Ads [Managerial] *PM Practices and Disciplines (5)*; [Interpersonal] *Facilitation (5)*
- *Product Manager Skills Required in India* According to 15 Ads on Naukri.com –
[Intrapersonal] *Empathetic (1)* 15 Job Ad [Managerial] *Lead a Team of Product Owners, Scrum Masters and Business Analysts (2)*
- *Product Owner Skills Required in Australia* According to 15 Ads on Seek.com.au –
[Intrapersonal] *Self Motivated [D2]* – [Managerial] *Inclusive Leadership/Encourages diversity/ Environment of collaboration and creativity/Team empowerment (7)*;
[Interpersonal] *Skill Development/Knowledge Sharing/Mentorship and Coaching (4)*

Assessment - work vs. University

- Gonczi, A. (2013). Competency-based approaches: linking theory and practice in professional education with particular reference to health education. *Educational Philosophy and Theory*, 45(12), 1290-1306.
- The nature of workplace problems is that they are unpredictable and multidisciplinary and their solution often requires teamwork. Hence, it is not possible to rigidly specify the outcomes that are sought.
- This contrasts sharply with the typical university academic assessment where problems are clearly laid out and assessors have a predetermined answer against which to assess student performance.
- What is needed, by contrast, is a number of assessments that examines knowledge, skills, **dispositions and judgement** in an integrated way and that enables us to make:
 - judgements about actual performance
 - inferences about knowledge and dispositions that we believe are important.
 - inferences about reasoning capacity and judgement.

Assessment & Competence

- Gonczi, A. (2013). Competency-based approaches: linking theory and practice in professional education with particular reference to health education. *Educational Philosophy and Theory*, 45(12), 1290-1306.
- No one assessment event can do all of this.
- Rather, we need a range of assessments that include a performance assessment where one or more **practitioners make professional judgements about a person's competence** (helped by some kind of marking scale), plus some kind of reflective learning log or diary or **portfolio that assesses a person's learning over time: a process assessment that is ongoing rather than summative**.
- The performance assessment should be undertaken in a real-world setting if possible.
- However, **it is possible to use simulated settings** if the real-world setting is not conclusive and further testing needs to take place on specific issues

Assessment & Collective Empathy

- Some artifacts
 - Team Contract
 - Team Agreement
 - Code of conduct
 - Contributor Covenant
- <https://where.coraline.codes/>
- *best known as the creator of **Contributor Covenant**, the first and most popular **code of conduct for open source communities**. It's been adopted by over 40,000 projects including Linux, Golang, JRuby, Swift, F#, Rails, and the open source portfolios of organizations including Apple, Microsoft, Google, Salesforce, and Intel.*

Coraline Ada Ehmke

Big-time open source troublemaker
and sworn enemy of the status quo.



Collective Empathy

- Increasing Prevalence of Empathy as a term
- <https://us.pycon.org/2018/about/code-of-conduct/>
- Examples of behavior that contributes to creating a positive environment include:
 - Being kind to others
 - Behaving professionally
 - Using welcoming and inclusive language
 - Being respectful of differing viewpoints and experiences
 - Gracefully accepting constructive criticism
 - Focusing on what is best for the community
 - Showing empathy towards other community members

Collective Empathy & AI

- Relevance of Empathy as a term – not going away?
- Our survey shows that creators and heavy users prioritize workplace flexibility more than total compensation, and that they are seeking a sense of belonging, care, and reliability within their work community.
- They stay in their jobs when they are given flexibility, and they leave when they aren't.
- The other factors that make them stay are meaningful work, support for health and wellbeing, reliable and supportive coworkers, and a safe workplace environment.
- Aaron De Smet, Sandra Durth, Bryan Hancock, Marino Mugayar-Baldocchi, & Reich, A. (2024, 18/03/2024). **The human side of generative AI:** Creating a path to productivity. *McKinsey Quarterly*.
<https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/the-human-side-of-generative-ai-creating-a-path-to-productivity>

Collective Empathy

BCIS Team Contract

AUT Research & Development Final Year Project *Team Contract*

Date	Version	Author	Notes
April 2020	1.0	Storm Waugh	Create new Team Contract
May 2020	1.1	Storm Waugh	Updated meeting guidelines and communication channels
August 2020	1.2	Tania Johnson	Further amendments to work hours and meeting guidelines. Add team signatures

Code of Conduct: Collectively as a project team, we will:

- Treat each team member with dignity and kindness.
- Behave according to the project management ethics:
 - Participate by actively engaging in team meetings and client meetings by sharing opinions and asking questions relating to issues discussed in those meetings.
 - Being honest when facing difficulties, struggling to understand assigned tasks or with the development of the navigation app.
 - Encourage team members to be proactive with this project by exerting positive feedback and leading by example.
 - Be respectful of team members religion and cultural beliefs.
 - Be responsible by acknowledging your mistakes throughout the duration of the project and by managing your own time effectively.
- Produce quality work according to what is outlined in the QA Plan
- Not tolerate team members who discriminate, harass or bully other team members and individuals outside of the project.

Cognitive

Behavioral
Empathy

- “A second part of the taxonomy is the affective domain. It includes objectives which describe changes in interest, attitudes and values and the development of appreciations.
- ...classifying objectives under this domain...**has been a difficult task ...still far from complete.** Several problems make it so difficult. Objectives in this domain not stated very precisely; and in fact, teachers do not appear to be very clear about the learning experiences which are appropriate to these objectives.
- ...difficult to describe appropriate behaviors since **internal or covert feelings and emotions** as significant for this domain as the overt behavioral manifestations.
- Then too, our **testing procedures for the affective domain are still in the most primitive stages.** We hope to complete the task but are not able to predict a publication date.”

Krathwohl, D. R., Bloom, B. S., & Masia, B. B. (1956). *Taxonomy of educational objectives, the classification of educational goals. Handbook II: affective domain.* David McKay Co. Inc., New York, 1, 956.

- Krathwohl et al. 1964 found a great diversity of **imprecise affective terminology**.
- Terms such as interest, appreciation, attitudes, and values are commonly used to cover portions of **a large range of achievement**—from simple awareness to the point of absorption into internal structures guiding behavior.
- In response, **they settled on a single continuum, *internalization*, as an organizing principle**; and were able to construct **a hierarchy of five major levels along this continuum**. Fig. 1 illustrates this hierarchy, and the overlapping nature of some of the common terms. Progression from Level 1 to Level 5 **denotes an increasing level of internalization of interests, attitudes, and/or values**.

Lynch, D. R., Russell, J. S., Evans, J. C., & Sutterer, K. G. (2009). **Beyond the cognitive: The affective domain, values, and the achievement of the vision.** *Journal of professional issues in engineering education and practice*, 135(1), 47-56.

Bloom's Affective Domain

Ex. Lynch, D. R., Russell, J. S., Evans, J. C., & Sutterer, K. G. (2009). Beyond the cognitive: The affective domain, values, and the achievement of the vision.

Journal of professional issues in engineering education and practice, 135(1), 47-56.

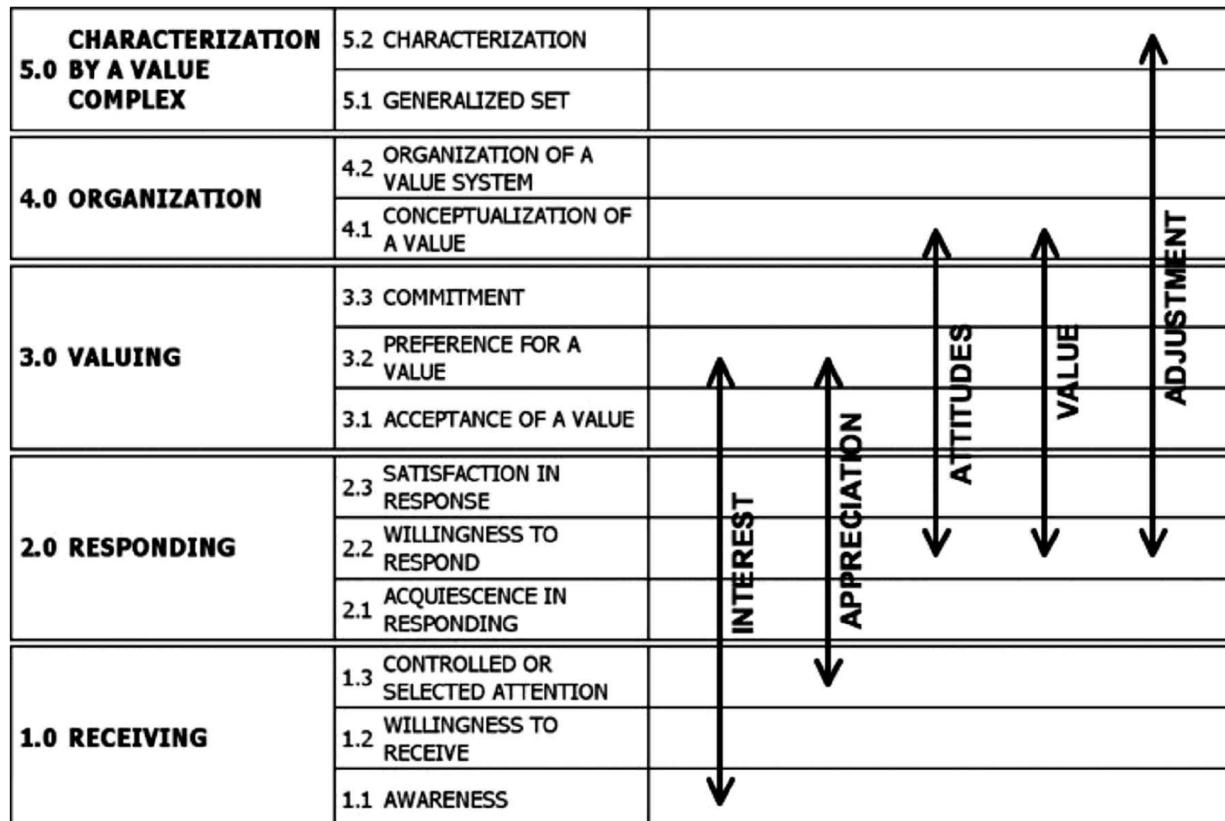


Fig. 1. Internalization continuum of affective domain (see also Appendix I). (Adapted from Krathwohl, David R., Bloom, Benjamin S. and Masia, Bertron B., *Taxonomy of Educational Objectives, Book 2: Affective Domain*. Published by Allyn and Bacon, Boston, Mass. Copyright © 1964 by Pearson Education. Adapted by permission of the publisher.)

Dispositions?

- ... encompasses socio-emotional skills, attitudes, and behaviors that characterize
- the **inclination to carry out tasks** and the **sensitivity to know when and how to engage in those tasks** .
- Values to be internalized

Raj, R., Sabin, M., Impagliazzo, J., Bowers, D., Daniels, M., Hermans, F., . . . McCauley, R. (2021). Professional Competencies in Computing Education: Pedagogies and Assessment. In *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education* (pp. 133-161).

Value Internalization continuum of Bloom's affective domain

Level	Internalization Level Definition	Sub-Level Definition	Disposition
			Collaborative [D8 - CC2020]
3.0	Valuing	3.3 Commitment	Always takes need to work with team members into account
		3.2 Preference for a value	Prefers to consider working with others in jointly prioritising tasks
		3.1 Acceptance of a value	Recognises client and team members need for collaboration
2.0	Responding	2.3 Satisfaction in response	pleased to participate in team tasks e.g. reviews on colleague's commits/releases
		2.2 Willingness to respond	willing to collaborate on team tasks e.g coding, reviews
		2.1 Acquiescence in responding	grudgingly tries to be collaborative
1.0	Receiving	1.3 Controlled or selected attention	sometimes collaborative, but inconsistently considers need to collaborate
		1.2 Willingness to receive	Not sure how to collaborate, but open to guidance
		1.1. Awareness	realises that collaborating is relevant but does not act on it
0	Unaware	0.0 Not considered or perceived as relevant	Acts alone and no sense that being collaborative has value

Assessing The Disposition “Being Collaborative”

Being Collaborative

More advanced levels

Value Internalization continuum of Bloom's affective domain		Disposition	
Level	Internalization Level Definition	Sub-Level Definition	Collaborative [D8 - CC2020]
5.0	Characterization by a value complex	5.1 Characterization	Promulgates collaborative software engineering
		5.2 Generalized set	Designs standards and frameworks to encourage and enable collaborative work
4.0	Organization	4.2 Organization of a value system	Implements frameworks and practices that emphasise collaboration
		4.1 Conceptualization of a value	selects a set of collaborative practices for a project/team

- Soft Skills not soft - hard to consistently define!
- Competencies incorporating dispositions useful framework
- Collective Empathy a concept of growing importance for IT Professionals and Students
- Collective Empathy can be understood as a disposition
- Collective Empathy can be viewed as a value that undergoes a process of internalization over time
- Collective Empathy can be assessed
- Collective Empathy can be developed



AUT

TE WĀNANGA ARONU
O TĀMAKI MAKAU RA

ftware for
etter world



CONCLUSION

- Beyond inconsistently defined and measured soft skills
- increasing interest and awareness of *collective empathy* in software teams,
 - (*as evidenced through job advertisements and the literature*).
- [3] conclude “*Our work emphasizes the value of empathy in software engineering. Nonetheless, it remains an under-researched subject*”
- Also “**For educators:** *SE educators might consider our results to approach the development of empathy throughout SE education...helpful for pedagogical interventions to build students' empathetic capacities*”[3].
- Links with work on dispositions and their assessment [6,7,9,17], and competency-based curriculum development [5, 4], and assessment strategies
- potential to more explicitly incorporate *collective empathy* into computing curriculum
- A direction for furthering research and educational practice??

References

1. Akgün, A.E., Keskin, H., Cebecioglu, A.Y. and Dogan, D. Antecedents and consequences of collective empathy in software development project teams. *Information & Management*, 52 (2). (2015), 247-259.
2. Balutia, P. *Agile project management- product manager, product owner and BA roles*, Master of Information Technology Project Management Dissertation, Auckland University of Technology, Auckland, 2023.
3. Cerqueira, L., Freire, S., Bastos, J., Spínola, R., Mendonça, M. and Santos, J. A Thematic Synthesis on Empathy in Software Engineering based on the Practitioners' Perspective *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. Association for Computing Machinery, (Campo Grande Brazil: ACM 2023), 332–341.
4. Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., Lunt, B., Maiorana, F., Pears, A. and Pitt, F. Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. (New York: ACM 2020).
5. Clear, A., Parrish, A. and CC2020 Task Force. Computing Curricula 2020 - CC2020 - Paradigms for Future Computing Curricula ACM and IEEE-CS eds. *A Computing Curricula Series Report* ACM, New York, 2020.
6. Clear, T. THINKING ISSUES: Computing Competencies, Dispositions and the Affective Taxonomy: More Work Still to Do? *ACM Inroads*, 14,3 (2023), 8–10.
7. Clear, T. THINKING ISSUES: Is Agility a Disposition and Can it be Taught? *ACM Inroads*, 12, 1 (2021), 13-14.
8. Dev. Dev Community. Forem. (2024), <http://dev.to/>. Accessed 20 March 2024.
9. Frezza, S., Clear, T. and Clear, A. Unpacking Dispositions in the CC2020 Computing Curriculum Overview Report. In *50th ASEE/IEEE Frontiers in Education Conference*. (Uppsala, Sweden: IEEE 2020).
10. Galster, M., Mitrovic, A., Malinen, S., Holland, J. and Peiris, P. Soft skills required from software professionals in New Zealand. *Information and Software Technology*, 160 (2023/08/01/ 2023), 107232.
11. Gunatilake, H., Grundy, J., Hoda, R. and Mueller, I. Enablers and Barriers of Empathy in Software Developer and User Interactions: A Mixed Methods Case Study. *ACM Trans. Softw. Eng. Methodol.* (2024), [just accepted]
12. Gunatilake, H., Grundy, J., Mueller, I. and Hoda, R. Empathy models and software engineering—A preliminary analysis and taxonomy. *Journal of Systems and Software*, 203. (2023), 111747.
13. Guthridge, M. and Giummarras, M.J. The taxonomy of empathy: a meta-definition and the nine dimensions of the empathic system. *Journal of Humanistic Psychology*. (2021), 00221678211018015.

References

14. He, J. *Skills Requirements from Junor Software Developer to Senior Developer*. Master of Computer and Information Sciences Dissertation. Auckland University of Technology, Auckland, 2022.
15. Hussain, W., Clear, T. and MacDonell, S. Emerging Trends for Global DevOps: A New Zealand Perspective. in Cruzes, D. and Sharma, A. eds. *Proceedings 2017 IEEE 12th International Conference on Global Software Engineering*. (Los Alamitos, California: IEEE 2017), 21-30.
16. Papoutsoglou, M., Wachs, J., & Kapitsaki, G. M. Mining DEV for social and technical insights about software development. In *18th International Conference on Mining Software Repositories (MSR)*. (IEEE/ACM 2021).
17. Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M., Systematic mapping studies in software engineering. in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*. (2008), 1-10.
18. Raj, R., Sabin, M., Impagliazzo, J., Bowers, D., Daniels, M., Hermans, F., Kiesler, N., Kumar, A.N., MacKellar, B. and McCauley, R. Professional Competencies in Computing Education: Pedagogies and Assessment. in *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education*. (ACM 2021), 133-161.
19. Schussler, D.L. Defining dispositions: Wading through murky waters. *The Teacher Educator*, 41, 4 (2006), 251-268.

Soft Skills are Not Soft! Collective Empathy – a New Disposition for Software Engineering Teams?

Questions?



Associate Professor Tony Clear
Department of Computer Science and Software Engineering
Auckland University of Technology

Tony.clear@aut.ac.nz



Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

Galster, M., Mitrovic, A., Malinen, S., Holland, J., & Peiris, P. (2023, 2023/08/01). Soft skills required from software professionals in New Zealand.

Information and Software Technology, 160, 107232.
<https://doi.org/https://doi.org/10.1016/j.infsof.2023.107232>

Table 1
Identified soft skills in alphabetical order (numbers in brackets indicate SEEK advert ID).

Soft skill	Example quote	Description
Analytical	“Have excellent analytical skills and be methodical” (55934622)	Ability to think critically, analyze different types of information and from different sources
Collaboration	“Strong collaborative focus.” (55997187)	Qualities and competencies to collectively progress toward a common goal with other stakeholders
Communication (generic)	“Excellent communication skills [...] to explain technical issues in a non technical manner.” (55996923)	Ability to communicate with others in “two-way process”, i.e., by exchanging (“sending” and “receiving”) information
Communication (oral)	“Strong written, verbal and presentation skills” (55965677)	Ability to exchange (“send” and “receive”) information in different form orally (e.g., in formal presentations or informal conversations)
Communication (written)	“Good communication skills with the ability to write clearly” (55964542)	Ability to exchange (“send” and “receive”) information in different forms of writing (e.g., technical documentation or customer documents)
Cultural awareness	“Cultural agility” (55963749)	Awareness and acceptance of other cultures and cultural identities related to the software itself (e.g., cultural responsiveness of apps) and its development and maintenance (e.g., culturally diverse teams)
Creativity	“... ability to think outside of the box...” (55885160)	Ability to imagine and form original ideas to create something new and potentially innovative
Diversity	“... ability to work effectively with a wide range of individuals in a diverse community” (55882970)	Ability to engage with people from a range of different social, educational, professional and ethnic backgrounds and of different genders
Empathy	“You have empathy for customers, teammates, and other stakeholders” (55988058)	Capacity to understand or feel what another person (including subordinates, peers, managers, clients) is experiencing from within their frame of reference (cognitively, emotionally and compassionately)
Independence	“Have demonstrated experience of self-managing and prioritising your work.” (55996346)	Ability to act free from the influence or control of another person, group or organization (e.g., without detailed instructions from team leads, clients or peers)
Interpersonal	“Relationship management skills.” (55996384)	Ability to be in, establishing, relating to, or involving relations between individuals or groups of persons (e.g., customers, clients)
Leadership	“[...] command respect and to create a sense of community amongst the members of the project teams.” (55978397)	Ability to influence and guide, motivate and inspire other members of a team, organization or community, including project, business, practice and technology leadership
Listening	“Exceptional communicator. You listen intently to customers and colleagues.” (55967867)	Ability to receive language from other stakeholders (internal and external) without judgment or immediate response (unlike communication, listening is a “one-way” process and could be considered a “subskill” of communication)
Mentoring	“Good leadership skills to train, guide and mentor the work of less experienced personnel.” (5996815)	Ability to form meaningful relationships with other individuals or groups (usually at the same or lower levels of hierarchy, experience or expertise) with the goal of professional, technology and personal development
Negotiation	“Strong problem solving, negotiation and decision-making skills.” (55996249)	Ability to efficiently and effectively discuss to reach agreement (maybe consensus) while handling (maybe resolving) conflicts with internal and external stakeholders
Problem solving	“Expert-level troubleshooting & problem-solving skills” (56002657)	Ability to determine why an “issue” is happening and how to resolve it by defining and measuring a problem, analyzing the problem and ultimately addressing the problem
Team	“... resilient and adaptable team player...” (55995239)	Abilities to work well with others during conversations, projects, meetings or other collaborations

Generative AI and its Implications for Competencies: Work in Progress

Associate Professor Tony Clear
Department of Computer Science
and Software Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz

Clear, T., Cajander, Å., Clear, A., McDermott, R., Bergqvist, A., Daniels, M., Divitini, M., Forshaw, M., Humble, N., Kasinidou, M., Kleanthous, S., Kultur, C., Parvini, G., Polash, M., & Zhu, T. (2024). **A Plan for a Joint Study into the Impacts of AI on Professional Competencies of IT Professionals and Implications for Computing Students.** In *Proceedings of the 2024 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 2). ACM. <https://doi.org/https://doi.org/10.1145/3649405.3659527>

Mihi

- Hāere mai, Haere Mai, Haere Mai.
- Tēnā koutou katoa.
- Ko Tony Clear taku ingoa.
- Nō Pōneke ahau.
- Ko Maungakiekie taku maunga.
- Ko Waitematā taku moana.
- I te taha o taku matua, no Enniscorthy Ireland ahau.
- I te taha o aku whaea, no Cork Ireland ahau.
- Ko Tainui raua ko Ngapuhi nga iwi o nga mokopuna
- Tēnā koutou, Tēnā koutou, Tēnā tatou katoa.

- An international ACM Innovation and Technology in Computer Science Education (ITiCSE) Conference working group
 - currently underway virtually and convening in July 2024:
- ***WG 2: A Multi-Institutional-Multi-National Study into the Impacts of AI on Work Practices of IT Professionals and Implications for Computing Students***
 - <https://iticse.acm.org/2024/working-groups/#wg2>
- *This coordinated, multinational working group is dedicated to examining the ramifications of AI integration within the IT sector.*
- *Employing qualitative research methods and conducting thematic analysis on interview data gathered from IT professionals [i.e. industry practitioners such as software developers] representing diverse contexts,*
- *the working group endeavours to uncover profound insights into how AI impacts work engagement, socio-technical dynamics, and the cultivation of professional competencies.*

OVERVIEW

- The concept of an ITiCSE working group
- Past working groups – history and context
- Progress of this working group
- Leaders & Rationale
- Members
- WG Steps taken so far
- Data Analysis Strategy
- Competencies
- Early insights?
- Implications for the teaching of software engineering ? - *tentative*

A Guide to ITiCSE Working Groups

Alison Clear & Tony Clear

*[from presentation to ITiCSE 2016 Working
Groups, Arequipa, Peru]*

What is an ITiCSE Working Group?

A concept unique to the ITiCSE conference

Investigating an interesting, unresolved topic in Computer Science Education

A diverse group of people, different countries, different cultures, different institutions

How is a Working Group Proposed

Concept proposed in January and submitted to ITiCSE

Successful proposals published on the ITiCSE website

Interested people apply to join

Groups of 5 to around 10 participants work electronically prior to commencement of the conference

Continue to work together for the duration of the conference

Work together face to face for the two days before the conference

Working Groups Reports Publication

Produce a mature draft report on the last day of the conference

Final report published in the ACM digital library

Rigorous cycle of blind reviewing

Outcome

- Substantial research paper
- greater than 20 pages
- Breadth and depth
- Published as a set of proceedings in the ACM Digital Library
- Unique experience

Historical Perspectives on the Computing Curriculum - Report of the ITiCSE'97 working group on Historical Perspectives in Computing Education

A Framework for Enhancing the Social Good in Computing Education: A Values Approach

Naturally Occurring Data as Research Instrument: Analyzing Examination Responses to Study the Novice Programmer

Research Perspectives on the Objects-Early Debate

Integrating cultural issues into the computer
and information technology curriculum

Computing educators oral history
project: seeking the trends

What's in a Name?: International
Interpretations of Computing
Education Terminology

Computing and sustainability: evaluating
resources for educators

Comments from Working Group Chairs

- “A chance to really leverage resources that you can’t do individually, gives you the ability to do a much wider search of the topic when you have 5-10 people together. A lot greater resources to tackle a topic”
- John Barr, July 2016

- “Builds a sense of trust”
- Tony Clear, July 2016

Comments from Working Group Chairs

- “Dynamic to be able to brainstorm to create and generate new ideas in person, just doesn’t work on your own”
- “Don’t have to be deep in a field, a great way to be initiated in the field and get up to speed quickly”
- “Get to meet people who have similar interests from all across the world”
- John Barr, July 2016

Comments from other WG participants

- Wonderful opportunity to meet new colleagues from around the world to meet on a problem of interest in CS Education
- Cary Laxer, July 2016
- One of the better community building ideas, f2f time with colleagues who have research interests in common
- Mats Daniels, Keynote speaker ITiCSE 2016

Planning the Work - Pre

- Think about preparatory activity
- Design and allocate tasks to members
- Design and prepare protocol, instruments questionnaires
- Arrange human subjects ethics protocols
- Collect examples
- Review selected literature
- Analyse data sets

Planning the Work - During

- Take stock of preparatory work completed
- Agree an agenda and plan
- Share contact details and conf. commitments
- Consider a draft structure for the report
- Work through ideas that require whole group input
- Divide the work and operate in parallel to produce the sections

Selected References

- [1] J. Noll, S. Beecham, and I. Richardson, "Global Software Development and Collaboration: Barriers and Solutions" *ACM Inroads*, vol. 1, no. 3, pp. 66-78, Sept 2010. [Online]. Available: <http://doi.acm.org.ezproxy.aut.ac.nz/10.1145/1709424.1709428>.
- [2] Clear, T., Beecham, S., Barr, J., Daniels, M., Mcdermott, R., Oudshoorn, M., Savickaite, A., and Noll, J., 2015. Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review. In *Proceedings of the Working Group Reports of the 2015 on Innovation & Technology in Computer Science Education Conference*, N. Ragonis and P. Kinnunen Eds. ACM, New York, 1-39. DOI= <http://dx.doi.org/http://dx.doi.org/10.1145/2858796.2858797>.
- [3] Beecham, S., Clear, T., Barr, J. and Noll, J. Protocol for Challenges and Recommendations for the Design and Conduct of Global Software Engineering Courses: A Systematic Review, Limerick, Ireland 2015.
- [4] T. Clear, S. Beecham, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, "Developments in Global Software Engineering Education," in *46th ASEE/IEEE Frontiers in Education Conference.*, D. Trytten, H. Matusovich, and M. Castro Eds. Erie, PA: IEEE, 2016.
- [5] S. Beecham, T. Clear, J. Barr, M. Daniels, M. Oudshoorn, and J. Noll, "Preparing Tomorrow's Software Engineers for Work in a Global Environment," *IEEE Software*, vol. 34, no. 1, pp. 9-12, Jan/Feb 2017, doi: 10.1109/MS.2017.16.
- [6] S. Beecham, T. Clear, D. Damian, J. Barr, J. Noll, and W. Scacchi, "How Best to Teach Global Software Engineering? Educators Are Divided," *IEEE Software*, vol. 34, no. 1, pp. 16-19, Jan/Feb 2017, doi: 10.1109/MS.2017.12.
- [7] T. Clear and S. Beecham, "Global Software Engineering Education Practice Continuum Special Issue of the ACM Transactions on Computing Education," *ACM Transactions on Computing Education (TOCE)*, vol. 19, no. 2, p. 7, 2019.
- [8] Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., Lunt, B., Maiorana, F., Pears, A. and Pitt, F. Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century in *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, ACM, New York, 2020.

Planning the Work – Pre 1

- Think about preparatory activity
 - Leaders initiated project and data collection [Sweden, UK, NZ]
 - MIMN design
- Design and allocate tasks to members
 - members selected – 15
 - countries and time zones
 - NZ [but on the move], Australia, Scotland, UK,
 - Norway, Sweden, US, Cyprus, Canada

Planning the Work – Pre 2

- Subgroups established – 4
 - Literature Review
 - Data Analysis
 - Practitioner Impacts
 - Competencies
 - Spreads the work and the time zone diffs.
- Design and prepare protocol, instruments questionnaires
- Arrange human subjects ethics protocols
 - leaders completed, separately by institution
 - Data sharing agreement developed

Planning the Work – Pre 3

- Collect examples
 - in country interviews under way
 - zoom and automatic transcription and translation service
 - (GDPR certified) - Sweden
 - MS -Teams and auto transcription – NZ
 - Zoom and manual transcription by exception
 - Interviews on the move, time zones and B&B wifi!
 - Getting buy-in
 - Information Sheets, consent forms
 - Semi-structured interviews common schedule

Planning the Work – Pre 4

- Review selected literature
 - subgroup allocated
 - Some subgroup specific literature searches
- Analyse data sets
 - subgroup allocated
 - secure central repository established [UU - ALLVIS]
 - common data sharing agreement established
 - draft data analysis protocol developed

Data Analysis Protocol - Options

- ## Reviewing Options

- literature on protocol templates has been consulted (Brereton et al., 2008; King et al., 2018; Stol and Fitzgerald, 2014).
- The excerpt below from Brereton et al., (2008), although positioned at the **overall case study level**, makes some highly relevant points:
 - 1) no existing data analysis template was found to be available,
 - 2) the two phases of “*creating instruments and protocols*” and “*analysing data*”
 - see over - were considered most applicable for this protocol.
- Brereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). *Using a protocol template for case study planning*. Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering,
- King, N., Brooks, J., & Tabari, S. (2018). *Template analysis in business and management research. Qualitative methodologies in Organization studies: volume II: methods and possibilities*, 179-206.
- Stol, K.-J., & Fitzgerald, B. (2014). *Research protocol for a case study of crowdsourcing software development* (Lero Technical Report -TR_2014_03). <https://cora.ucc.ie/handle/10468/7039>

Developing a Data Analysis Protocol

- Full Case Study Protocol
- *Eisenhardt (1989) concerned with using case studies to develop theories suggests the following activities:*
 - **Getting started** by defining the research question and *a priori* questions but not defining hypotheses or theory.
 - **Selecting cases** by considering a particular population and using theoretical concerns to focus on specific cases.
 - **Crafting instruments and protocols** using multiple data collection methods, using qualitative and quantitative data, and preferably multiple researchers.
 - **Entering the field** i.e. incorporating field notes with data analyses and using flexible and opportunistic data collection methods.
 - **Analysing the data** both within case and across cases.
 - **Shaping hypotheses** by iterative tabulation of evidence looking for identified constructs, replication logic across cases, and looking for evidence to explain why relationships exist.
 - **Enfolding the literature** i.e. comparing with existing similar and conflicting literature.
 - **Reaching closure** using the concept of “theoretical saturation” which says researchers stop looking for more cases/data when they believe more data will only give a marginal improvement to the existing results”. (Brereton et al., 2008)

Forms of Template Analysis

- King et al., (2018) presenting “*Template Analysis*” :
- “*Thematic analysis is widely acknowledged as an accessible and useful approach to the analysis of rich and meaningful qualitative data—indeed, Clarke and Braun (2013) describe thematic analysis as the ‘basic’ method of qualitative data analysis*”.
- *Template Analysis a particular style of thematic analysis*
- distinguish between *Generic Template Analysis* as a *method*, and as applied within a broader research *methodology* such as grounded theory or Interpretative Phenomenological Analysis [IPA],
- observe that issues about differing philosophical, theoretical or methodological positions can be better accommodated, as noted below:
- *Generic styles of thematic analysis can provide researchers more flexibility and adaptability to the particular requirements of their own work—rather than applying a methodology as a whole package.*

Steps in Template Analysis

- In template analysis the general steps in the process are defined by King et al., (2018) as below:
- *procedural steps characteristically followed in Template Analysis are:*
 - *Familiarization with the data*
 - *Preliminary coding*
 - *Clustering*
 - *Developing the initial template*
 - *Modifying the template*
 - *Defining the ‘final’ template*
 - *Using the template to interpret the data*
 - *Writing-up*
 -

WG Data Analysis Subgroup

- data analysis sub-group will conduct the three steps prior to developing the initial template “*Familiarization with the data, preliminary coding, clustering*”, using a subset of the transcript data, to derive an initial template using a defined spreadsheet for coding each transcript.
- The template will be refined as the process progresses as noted by (*King et al., 2018*):
 - *Revisions might include: re-defining themes to increase or narrow their scope (shown through moving them up or down hierarchical levels), moving themes between clusters, adding new themes—or even entire new clusters—and deleting themes that have become redundant as the template has developed.*
- Version one of the template allows for coding of each question.
- For some questions a predefined set of deductive codes as an initial set of codes for that question,
- for other questions a more inductive strategy may be more suitable.

WG Analysis - Work Allocation

- Decisions about coding parties need to be made.
- It is presumed at this stage that work will be distributed amongst the WG members to share the load, once the template is sufficiently stable.
- Members of the WG will generally code whole transcripts, possibly adopting a pair coding strategy, first individually coded then by comparison between pairs.
- But it may be preferable to assign questions to sub-groups, e.g. question 4.3 by the competencies sub-group?
-
- The transcripts will be stored in the Uppsala ALLVIS repository, and made available to members whose institutions have signed the data sharing agreement.

Planning the Work – During the WG Mtg

Partly In Progress

- Take stock of preparatory work completed
- Agree an agenda and plan
- Share contact details and conf. commitments
- Consider a draft structure for the report
- Work through ideas that require whole group input
- Divide the work and operate in parallel to produce the sections

Planning the Work – During -2 TBD

- Allocate a paper editor
- Review components and drafts as they come in
- Prepare for presentation day one of conference
 - Update audience on topic & progress
 - Seek feedback in selected areas
- Build consensus towards a final draft
- Produce consolidated mature draft and handover to WG Chairs on the last day of the conference before leaving
- Advise a list of three sound reviewers

Planning the Work – Post **TBD**

- Ensure responsibility allocated for final draft
- Share timetables [holidays and semester commitments]
- Work on final draft [version tracking mechanism in place] and share amongst whole or core team
- Submit and wait for reviews
- Make changes as advised by reviewers
- Submit camera ready copy and hope!

CC2020 PROJECT

CURRICULUM

DIRECTIONS

Clear, A., Parrish, A., &
CC2020 Task Force.
(2020). *Computing
Curricula 2020 - CC2020*

*Paradigms for Future
Computing Curricula*
Retrieved from New
York:

<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>

A Computing Curricula Series Report
2020 December 31

Computing Curricula 2020

CC2020

Paradigms for Global Computing Education

encompassing undergraduate programs in
Computer Engineering
Computer Science
Cybersecurity
Information Systems
Information Technology
Software Engineering
with data science

**A Framework for
Structuring
Competency
Related
Responses?**



Association for
Computing Machinery



IEEE
computer
society

CC2020 PROJECT MODEL

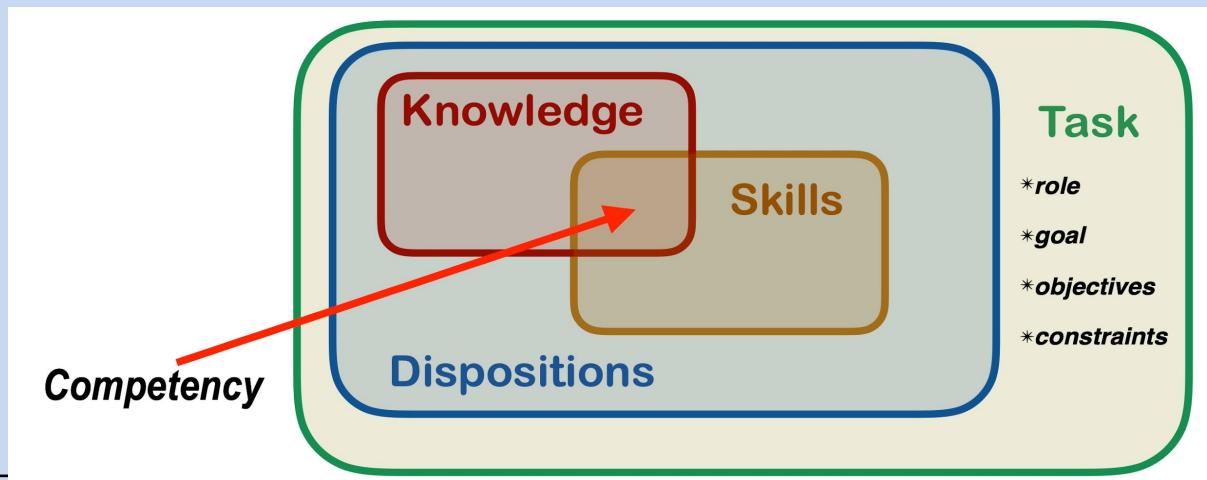
ASPECTS OF KNOWLEDGE BASED LEARNING [Established Curriculum Paradigm]

- KA's [knowledge areas]
- KU's [knowledge units]
- LO's [Learning outcomes]

COMPONENTS OF COMPETENCY BASED LEARNING [CC2020 MODEL]

– **Competency = [Knowledge+ Skills+ Dispositions] in Task**

- A competency structure (see Fig. 1) shows knowledge, skills, and dispositions that are observable in the accomplishment of a task, a task that prescribes purpose within a work context [24].



CC2020 – COMPETENCY BASED LEARNING

- Knowledge is the “*know-what*” dimension of competency that is factual.
- Skills express the “*know-how*” and usually develop over time and with practice.
- Dispositions frame the “*know-why*” dimension of competency, which prescribes a requisite character or quality in task performance.
- Task is the construct that frames the skilled application of knowledge and makes dispositions concrete.

SUBJECT/CONTENT KNOWLEDGE

3.2.1 *Knowledge Vocabulary for Computer Science Competencies.*

The CS2013 document divides computer science knowledge into 18 KAs subdivided into 225 KUs. For example, the Software Engineering KA is divided into ten KUs:

- (1) SE/Software Processes
- (2) SE/Software Project Management
- (3) SE/Tools and Environments
- (4) SE/Requirements Engineering
- (5) SE/Software Design
- (6) SE/Software Construction
- (7) SE/Software Verification and Validation
- (8) SE/Software Evolution
- (9) SE/Software Reliability
- (10) SE/Formal Methods

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

PROFESSIONAL & FOUNDATIONAL KNOWLEDGE

Table 4: Professional & Foundational Knowledge Areas extending CS2013

Tag	Knowledge Area
PK-1	Oral Communication & Presentation
PK-2	Written Communication
PK-3	Problem Solving and Trouble Shooting
PK-4	Project/Task Organization & Planning
PK-5	Collaboration and Teamwork
PK-6	Research and Self-Learning
PK-7	Multi-Task Prioritization & Management
PK-8	Relationship Management
PK-9	Analytical and Critical Thinking
PK-10	Time Management
PK-11	Quality Assurance / Control
PK-12	Mathematics and Statistics
PK-13	Ethical Intercultural Perspectives

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

SKILLS

Table 4.3. Levels of Cognitive Skills Based on Bloom's Taxonomy

Remembering	Understanding	Applying	Analyzing	Evaluating	Creating
Exhibit memory of previously learned materials by recalling facts, terms, basic concepts, and answers.	Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, and giving descriptions.	Solve problems in new situations by applying acquired knowledge, facts, techniques, and rules in a different way.	Examine and break information into parts by identifying motives or causes; make inferences and find evidence to support solutions.	Present and defend opinions by making judgments about information, validity of ideas, or quality of material.	Compile information together in a different way by combining elements in a new pattern or by proposing alternative solutions.

Clear, A., Parrish, A., & CC2020 Task Force. (2020). *Computing Curricula 2020 - CC2020 – Paradigms for Future Computing Curricula* Retrieved from New York:
<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>

CC2020 DISPOSITIONS

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., ... Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

SPECIFYING COMPETENCIES?

KA-Ref#	Title	Competency Statement	Dispositions	Knowledge-Skill Pairs				Traced Competencies	Notes
				KU / Broader Knowledge Description	Topic/LO Description	Knowledge Element	Paired Skill Level		
Our reference number goes here. E.g., "KA-##"	Short Title	Competency statement goes here (Natural Language) that embeds the task/context and suggests the K-S-D breadth and depth	List top applicable dispositions: <ID+descriptor>. Note that dispositions are not aligned with KS-pairs.	CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level	List other competency reference #'s that this competency expects or depends upon. Often null.	Any notes or comments
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		Often null
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		Note deviations from Bloom's level in CS2013
				CS2013 or other description of a KU	Text of LO or Topic or Professional Skill	New IDs for extensions to CS2013	Bloom's Level		
					Text of LO or Topic or Professional Skill	CS2013 LO or Topic ID	Bloom's Level		

Figure 5: Competency specification in a tabular format

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

EXPRESSING COMPETENCIES?

“...most appropriate to express competencies at the level of entire KAs, **from the point of view of a graduate or professional.**

When we “designed” a competency statement, we focused on a particular knowledge area and **looked for collections of topics and/or learning outcomes (LOs) that could be observed in tasks and at particular skill levels.**

...there was significant variance in how this was approached, and how the resulting competency was formulated. **Dispositions were often implied at first, but refinement of the statements made these both more clear and more relevant.”**

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

A SAMPLE COMPETENCY STATEMENT

Table 7: IS-4: IS/Basic Machine Learning

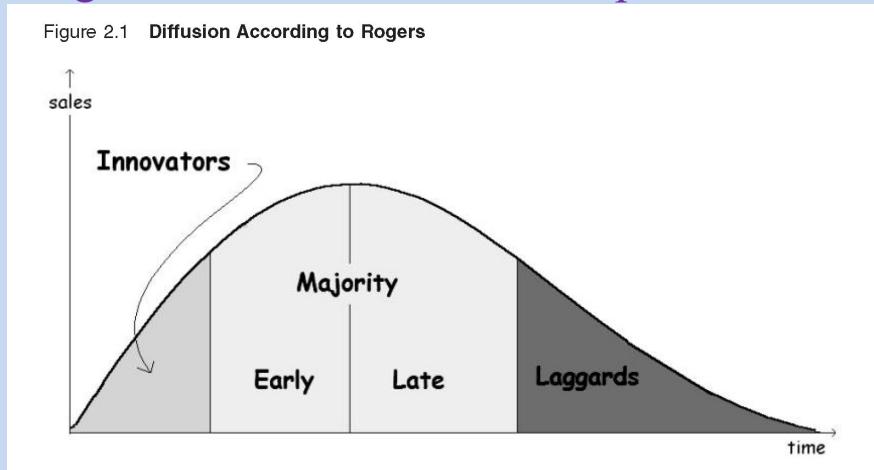
Statement	Dispositions
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.	D-6 (Responsible) D-10 (Meticulous)

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

Table 7: IS-4: IS/Basic Machine Learning

Statement		Dispositions		
Knowledge Unit	Topic/LO Description	Knowledge Element	Skill Level	Notes
Confidently apply relevant statistical techniques to differing modes of machine learning when undertaking an assigned classifying task, and show the ability to precisely measure the accuracy of the resulting classifier.			D-6 (Responsible)	D-10 (Meticulous)
IS/Basic Machine Learning	List the differences among the three main styles of learning: supervised, reinforcement, and unsupervised.	IS-BML-1	B-II	
	Identify examples of classification tasks, including the available input features and output to be predicted	IS-BML-2	B-II	
	Explain the difference between inductive and deductive learning	IS-BML-3	B-II	
	Describe over-fitting in the context of a problem	IS-BML-4	B-II	
	Apply the simple statistical learning algorithm such as Naive Bayesian Classifier to a classification task and measure the classifier's accuracy	IS-BML-5	B-III	
Prof. & Foundational Knowledge	Analytical and Critical Thinking	PK-9	B-III	
	Mathematics and Statistics	PK-12	B-III	
IS/Basic Machine Learning	Definition and examples of broad variety of machine learning tasks, including classification	IS-BML-a	B-II	
	Inductive Learning	IS-BML-b	B-II	
	Simple statistical-based learning, such as Naive Bayesian Classifier, decision trees	IS-BML-c	B-III	
	The over-fitting problem	IS-BML-d	B-II	
	Measuring classifier accuracy	IS-BML-e	B-III	

- Seems to be early days yet?
- Variable acceptance and adoption
- Outright ban to enthusiastic incorporation in ways of working



Libai, B., Mahajan, V., & Muller, E. (2017). Can you see the chasm?: Innovation diffusion according to rogers, bass, and moore. In *Review of marketing research* (pp. 38-57). Routledge.

- Active experimentation – personal and institutional pilot studies - FOMO
- Some vendor guided adoptions e.g. Microsoft and Copilot [Copilot not an Auto-pilot]
- Large Corporate risk assessment before a standard approach adopted

IMPLICATIONS FOR TEACHING SE 1?

- A shift to a dialogic model of interacting with systems
- Prompt Engineering and refinement of strategies
- API's to connect with backend chatbots easily implemented [e.g. Vercel
<https://vercel.com/changelog/next-js-ai-chatbot-2-0>]
- Good reviews:
 - Ebert, C., & Louridas, P. (2023). Generative AI for software practitioners. *IEEE Software*, 40(4), 30-38.
 - Ozkaya, I. (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software*, 40(3), 4-8.
<https://doi.org/10.1109/MS.2023.3248401>
- What to do about various forms of cheating?

IMPLICATIONS FOR TEACHING SE 2?

- What to do about various forms of cheating?
- How to shift to acceptable and valuable behaviour
- Just today's 4GL and end-user development fad?
- How to develop judgement and handling reviews and errors
- Hallucinations an inherent design of LLM technology?
- How to co-exist with Gen-AI services?
- How to redesign courses and assessments?

IMPLICATIONS FOR TEACHING SE 3?

- Commercial services and downsides
- Cost and accessibility issues?
- What to do about various forms of cheating?
- Observed violations of service agreements – who reports miscreants?
- Privacy and IP rights
- Whose work and evolving citation standards?
- AI systems and embedded biases
- Ethical awareness

ACADEMIC INTEGRITY - THEN?

You **can** use AI when writing or preparing a presentation if you use it to help you improve small aspects of your work, such as:

- grammar and punctuation, and
- formal terminology.

You **cannot** use AI to generate ideas when writing, preparing a presentation or creating an artwork/artefact (unless otherwise specified in your assessment instructions). The ideas have to come from you and your course materials.

<https://canvas.aut.ac.nz/courses/7624/pages/referencing>
9/01/2023 - and evolving



After drafting a paragraph for a group assignment, Jude uses the Grammarly AI writing assistant to check their writing. Grammarly identifies spelling mistakes and where the writing is too wordy. Jude reads the suggested changes and explanations about why some of their writing can be improved. They then make some improvements to their work.

Jude has acted appropriately because they have:

- only used AI to identify small errors in their own work,
- used the AI's explanations to learn more about academic writing, and
- made their own improvements.



Karl quickly writes an essay on the day it needs to be submitted. He provides ChatGPT with the assessment task instructions, a list of required readings, and his essay, and he then prompts ChatGPT to write a better version. ChatGPT completely reorganises the structure and content of Karl's work. Karl then submits ChatGPT's version because it looks better than his own.

Karl has not acted appropriately because he has:

- submitted work that he did not do himself (he submitted ChatGPT's work), and this is a breach of AUT's academic integrity guidelines. Even if Karl had acknowledged his use of ChatGPT in the essay, this would still be inappropriate because he did not write the essay himself.

Figure 4: Appropriate and inappropriate uses of AI when writing and presenting

Reference

AAIN Generative AI Working Group. (2023). *AAIN Generative Artificial Intelligence Guidelines*, Australian Academic Integrity Network. <https://doi.org/10.26187/sbwr-kq49> ↗
<https://doi.org/10.26187/sbwr-kq49>

ACADEMIC INTEGRITY - NOW?



Submit only your own work*

Acknowledge all sources of information you use by:

- using the appropriate referencing style, and
- paraphrasing or quoting any words/ideas that are not your own.



Do not submit work done by others, such as:

- other students*
- friends or relatives
- assignment writing services
- artificial intelligence software**, like ChatGPT.

Do not submit work that you have previously submitted for assessment.

<https://canvas.aut.ac.nz/courses/7624/pages/academic-integrity> 10/05/2024

- and evolving

Figure 1: How to maintain academic integrity

* Unless the work is for a designated (group) task

** If your assessment requires the approved use of artificial intelligence, you must acknowledge wherever you do so in your work with an appropriate in-text reference (see guidelines for APA [↗](https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312), Chicago [↗](https://aut.ac.nz.libguides.com/turabian/personalcomms#s-lg-box-22370438), and Harvard [↗](https://aut.ac.nz.libguides.com/c.php?g=919289&p=6648988#s-lg-box-22370440) referencing styles).

ACADEMIC INTEGRITY – NOW?

Artificial intelligence software

Referencing the information generated by an algorithm or artificial intelligence software tool, such as ChatGPT.

Credit the author of the algorithm/AI tool with a reference list entry and an in-text citation.

Reference list format

Who	When	What	Where
Author of AI tool.	(Year released).	<i>Title of tool</i> (Version) [Description].	URL to access tool

Reference list example

OpenAI. (2023). *ChatGPT* (Mar 14 version) [Large language model]. <https://chat.openai.com/chat>

In-text citation examples

OpenAI (2023) generated the following response when...

...that indicates a limitation of the software (OpenAI, 2023).

In your writing

- Describe how you used the tool.
- Provide the prompt you used.
- Provide any portion of the relevant text that was generated in response.
- Document the exact text created because tools like ChatGPT generate unique responses in each chat session, even if given the same prompt.

More information

<https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312>

<https://apastyle.apa.org/blog/how-to-cite-chatgpt>

ChatGPT – Terms of Use



What you can do. Subject to your compliance with these Terms, you may access and use our Services.

In using our Services, you must comply with all applicable laws as well as our Sharing & Publication Policy, Usage Policies, and any other documentation, guidelines, or policies we make available to you.

What you cannot do. You may not use our Services for any illegal, harmful, or abusive activity. For example, you may not:

- Use our Services in a way that infringes, misappropriates or violates anyone's rights.
- ...
- Represent that Output was human-generated when it was not.
-
- Use Output to develop models that compete with OpenAI.

OpenAI. (2024,
January 31 2024).
Terms of Use.
Retrieved
10/05/2024 from
<https://openai.com/policies/terms-of-use>

ChatGPT – Terms of Use

Your content. You may provide input to the Services (“Input”), and receive output from the Services based on the Input (“Output”). Input and Output are collectively “Content.”

“Content.” You are responsible for Content, including ensuring that it does not violate any applicable law or these Terms. You represent and warrant that you have all rights, licenses, and permissions needed to provide Input to our Services.

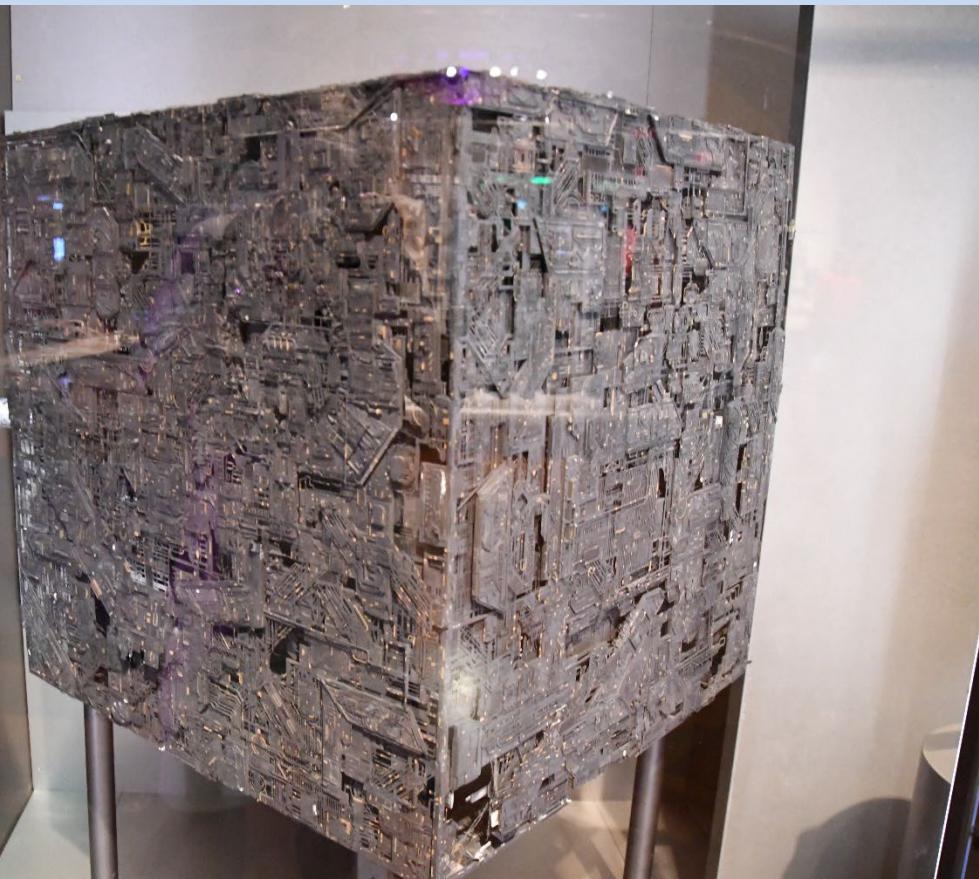
Our use of content. We may use Content to provide, maintain, develop, and improve our Services, comply with applicable law, enforce our terms and policies, and keep our Services safe.

Opt out. If you do not want us to use your Content to train our models, you can opt out by following the instructions in this Help Center article. Please note that in some cases this may limit the ability of our Services to better address your specific use case.

OpenAI. (2024, January 31 2024). *Terms of Use*. Retrieved 10/05/2024 from <https://openai.com/policies/terms-of-use>

BROADER IMPLICATIONS

- Giving data into the datacube of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
<https://commons.wikimedia.org/w/index.php?curid=58277452>



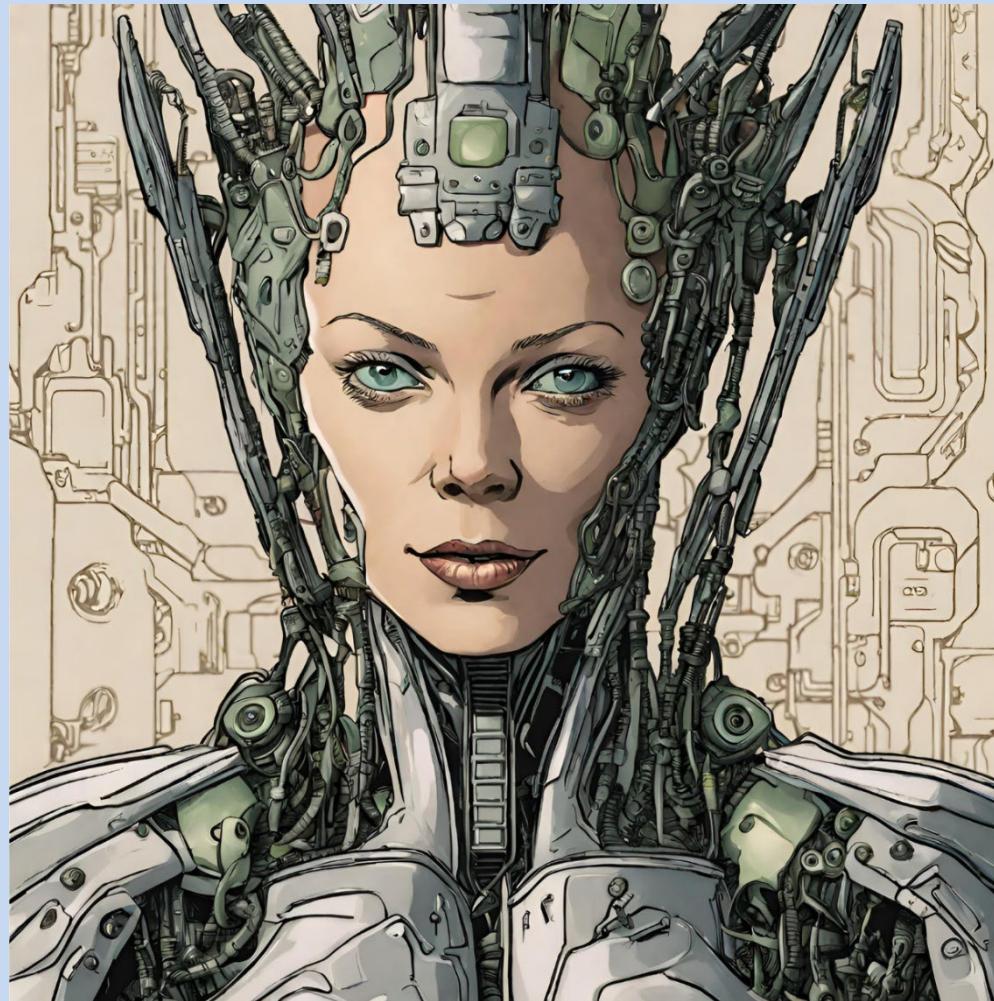
IMPLICATIONS?

- Giving data into the maw of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
https://upload.wikimedia.org/wikipedia/commons/5/52/Trekkie_-_Borg_-_Star_Trek_Convention_%2889504912575%29.jpg



IMPLICATIONS FOR TEACHING SE?

- Giving data into the clutches of the BORG and Getting it Back?



[https://www.mediawiki.org/wiki/File:
Borg_Queen_by_Canva_AI.png](https://www.mediawiki.org/wiki/File:Borg_Queen_by_Canva_AI.png)

Polski: Artystyczny wizerunek królowej Borg z uniwersum Star trek wygenerowany przez oprogramowanie Canva Magic Multimedia

English: Borg Queen from Star Trek universe artistic vision generated by Canva Magic Multimedia

4 April 2024

Own work

[Canva Magic Multimedia](#)

CONCLUSION

- Covered ITiCSE working groups and this one
 - *Implications of [Gen] AI for IT professionals and competencies*
- Covered WG creation and progress to date
- Outlined Data Analysis Strategy and Protocol
- Defined Competencies from a CC2020 perspective
- Discussed early insights about the state of the art?
- Explored tentative implications for the teaching of software engineering ?
- Still early work
- To be fleshed out in the final report [after WG meets 4 – 7 July and revisions by end of year - all going well ☺]

Generative AI and its Implications for Competencies: Work in Progress

Questions?



Associate Professor Tony Clear
Department of Computer Science and Software
Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz



Planning the Work – During -2

- Allocate a paper editor
- Review components and drafts as they come in
- Prepare for presentation day one of conference
 - Update audience on topic & progress
 - Seek feedback in selected areas
- Build consensus towards a final draft
- Produce consolidated mature draft and handover to WG Chairs on the last day of the conference before leaving
- Advise a list of three sound reviewers

Planning the Work - Post

- Ensure responsibility allocated for final draft
- Share timetables [holidays and semester commitments]
- Work on final draft [version tracking mechanism in place] and share amongst whole or core team
- Submit and wait for reviews
- Make changes as advised by reviewers
- Submit camera ready copy and hope!

DEFINING & DESCRIBING SKILLS

Table 5: Sample CC2020 Skill Level Vocabulary [3]

	I Remembering	II Understanding	III Applying	IV Analyzing	V Evaluating	VI Creating
Definitions	Exhibit memory of previously learned materials...	Demonstrate understanding of facts and ideas...	Solve problems in new situations by applying...	Examine and break information into parts...	Present and defend opinions by making judgments about information...	Compile information together in a new way...
Verbs	Choose, define, find, how, label, list, match, name, omit, recall, relate, select, show, spell, tell, what, when, where, which, who, why	Classify, compare, contrast, demonstrate, explain, extend, illustrate, infer, interpret, outline, relate, rephrase, show, summarize, translate	Apply, build, choose, construct, develop, experiment with, identify, interview, make use of, model, organize, plan, select, solve, utilize	Analyze, assume, categorize, classify, compare, contrast, discover, dissect, distinguish, divide, examine, function, infer, inspect, list, relate, simplify, survey, take part in, test for	Agree, appraise, assess, award, choose, conclude, criteria, criticize, decide, deduct, defend, determine, disprove, estimate, evaluate, explain, interpret, judge, justify, measure, prioritize, prove, rate, recommend, select	Adapt, build, change, combine, compose, construct, create, design, develop, elaborate, estimate, improve, invent, make up, maximize, minimize, modify, optimize, originate, plan, predict, propose solution, solve, test theory

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. New York: ACM.

ON TEACHING DISPOSITIONS?

- in discussing whether dispositions could be taught [Clear, 2021], I noted that “... *a disposition* “concerns not what abilities people have, **but how people are disposed to use those abilities**” [Schussler, 2006].

So here we are talking about a mindset and attitudinal dimensions, which raises the question can a disposition be taught or is it some innate part of a person’s character?”

[Clear, 2021]

Clear, T. (2021). THINKING ISSUES: Is Agility a Disposition and Can it be Taught? . *ACM Inroads*, 12(1), 13-14.
doi:10.1145/3447870

CC2020 DISPOSITIONS

Table 6: CC2020 Disposition Vocabulary

Label	Disposition	Elaboration [25]
D-1	Proactive	<i>With Initiative / Self-Starter.</i> Shows independence. Ability to assess and start activities independently without needing to be told what to do. Willing to take the lead, not waiting for others to start activities or wait for instructions.
D-2	Self-Directed	<i>Self-motivated / Self-Directed.</i> Demonstrates determination to sustain efforts to continue tasks. Direction from others is not required to continue a task toward its desired ends.
D-3	Passionate	<i>With Passion / Conviction.</i> Strongly committed to and enthusiastic about the realization of the task or goal. Makes the compelling case for the success and benefits of task, project, team or means of achieving goals.
D-4	Purpose-Driven	<i>Purposefully engaged / Purposefulness.</i> Goal-directed, intentionally acting and committed to achieve organizational and project goals. Reflects an attitude towards the organizational goals served by decisions, work or work products. E.g., business acumen.
D-5	Professional	<i>With Professionalism / Work ethic.</i> Reflects qualities connected with trained and skilled people: acting honestly, with integrity, commitment, determination and dedication to what is required to achieve a task.
D-6	Responsible	<i>With Judgement / Discretion / Responsible / Rectitude.</i> Reflects on conditions and concerns, then acts according to what is appropriate to the situation. Makes responsible assessments and takes actions using professional knowledge, experience, understanding and common sense. E.g., Responsibility, Professional astuteness.
D-7	Adaptable	<i>Adaptable / Flexible / Agile.</i> Ability or willingness to adjust approach in response to changing conditions or needs.
D-8	Collaborative	<i>Collaborative / Team Player / Influencing.</i> Willingness to work with others; engages appropriate involvement of other persons and organizations helpful to the task; strives to be respectful and productive in achieving a common goal.
D-9	Responsive	<i>Responsive / Respectful.</i> Reacts quickly and positively. Respects the timing needs for communication and actions needed to achieve the goals of the work.
D-10	Meticulous	<i>Attentive to Detail.</i> Achieves thoroughness and accuracy when accomplishing a task through concern for relevant details.
D-11	Inventive	<i>Exploratory / Inventive.</i> Looks beyond simple solutions; examines alternative ideas and solutions; seeks, produces and integrates appropriate alternative.

Clear, A., Clear, T., Vichare, A., Charles, T., Frezza, S., Gutica, M., . . . Pitt, F. (2020). Designing Computer Science

Competency Statements: A Process and Curriculum Model for the 21st Century In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science*

Education. New York: ACM.



Generative AI: a Critique

Associate Professor Tony Clear
Department of Computer Science
and Software Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz

Clear, T. (2024). THINKING ISSUES: Large Language Models, the 'Doctrine of Discovery' and 'Terra Nullius' Declared Again? *ACM Inroads*, 15(2), 6–9. <https://doi.org/10.1145/3638564>

- Large Language Models (LLMs) and
- recent hype around generative AI and ChatGPT,
- profound questions around data, rights, and ownership claims,
- how any such claims might be viewed critically by computing educators and their students

Clear, T. (2024). THINKING ISSUES: Large Language Models, the 'Doctrine of Discovery' and 'Terra Nullius' Declared Again? *ACM Inroads*, 15(2), 6–9. <https://doi.org/10.1145/3638564>

- Google™ at the vanguard of a “a third wave of colonization
- for countries such as New Zealand,
- **First by empire and the gun,**
- **Then by the dollar and economic might,**
- **Now by the shaping of discourse through distorted delivery of information.”**
- So, do large language models and the possibilities provided by generative AI merely represent another extension of this “third wave” or are they radically different?

Clear, T. (2006, Dec). Google™ - "Do No Evil" - Yeah Right! *SIGCSE Bulletin*, 38(4), 8-10.
<https://doi.org/https://doi.org/10.1145/1189136.1189142>

SURVEILLANCE CAPITALISM

- Shoshana Zuboff [11,12]
- strategies adopted by ‘big tech’ companies
- pre-emptively appropriating rights to newly conceived forms of data,
- expanding to the new world of ‘big data’
- and the economic sea change known as
- “surveillance capitalism.”

Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

- “‘Big data’ constituted by capturing small data
- from individuals’ computer-mediated actions and utterances in their
- pursuit of effective life.” [11]
- Big tech companies accumulating “not only surveillance assets and capital, but
- also rights … accomplished through a form of unilateral declaration that most
- closely resembles the social relations of a pre-modern absolutist authority.” [11]

Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

- Zuboff refers to a calculated secrecy, “concealing a new political equation in which
- Google’s concentrations of computational power
- **brush aside users’ decision rights** as easily as King Kong might shoo away
- an ant, all accomplished offstage where no one can see?” [12]

Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

- “euphemisms operate as those on the earliest maps of the North American continent,
- Whole regions labelled with terms such
- as “heathens,” “infidels,” “idolaters,”
- “primitives,” “vassals,” and “rebels.”
- On those euphemisms, native peoples—their places and claims—were
- deleted from the invaders’ moral and legal equations,
- legitimating the acts of taking and breaking that paved the way for church and monarchy”

Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

- New Zealand may frame itself as a “small, advanced economy,”
- let’s be quite clear—
- we, along with the citizens of many other countries, are the newly colonized!

- So how does this process of what we
- might term ‘neo-digi-colonization’ work?
- how might AI and the controllers of LLMs
- go about usurping our ‘places and claims’?
- One well known strategy is that of
- “Move fast and Break things” [10],
- where speed and greed rather than sense predominate

- “In high-stakes AI research,
- data work is often seen as low-level grunt
- work ... and incentive structures generally
- encourage a ‘move fast and break things’
- mentality over careful scientific work.

Liesenfeld, A., Lopez, A., & Dingemanse, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *arXiv preprint arXiv:2307.05532*.

THE NEED FOR OPENNESS

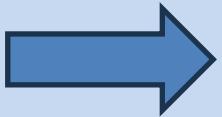
- Liesenfeld and colleagues
- argue for greater “openness in the fast-moving field of instruction-tuned large language models.
- We have found projects at varying stages of implementation,
- documentation, and useability.
- Most of them offer access to source code and some aspects of pre-training data,
- Sometimes in legally ambiguous ways.”

Liesenfeld, A., Lopez, A., & Dingemanse, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *arXiv preprint arXiv:2307.05532*.

- “There are many shades of openness...yet all of the projects surveyed here
- are significantly more open than ChatGPT.
- ChatGPT was announced in a company blog post
- rolled out to the public with an interface
- designed to capture as much free human labour as possible,
- but without any technical documentation.” [6]
- **‘Free labour’ has of course been a long-standing marker of colonization!**

Liesenfeld, A., Lopez, A., & Dingemanse, M. (2023). Opening up ChatGPT: Tracking openness, transparency, and accountability in instruction-tuned text generators. *arXiv preprint arXiv:2307.0553*

Data supports..



Prediction



Enables Control

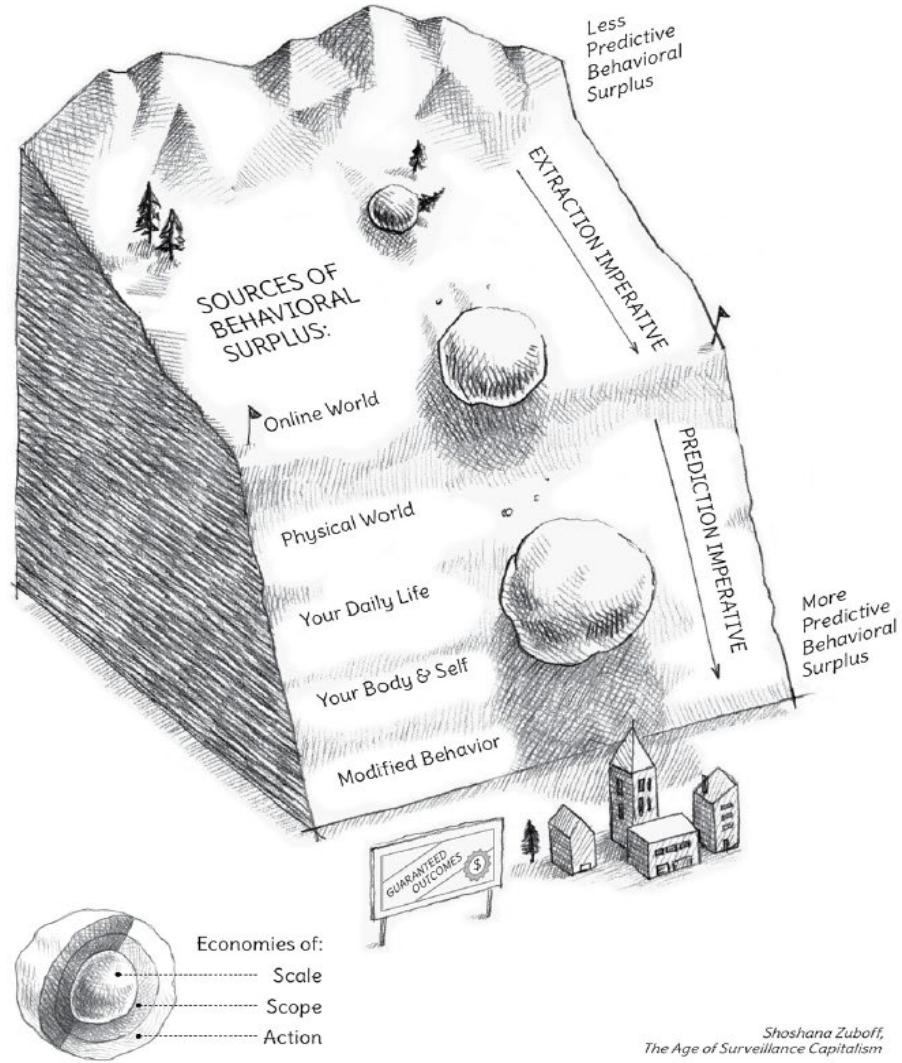


Figure 3: The Dynamic of Behavioral Surplus Accumulation

Zuboff, S.
(2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*. Profile Books.

- So if our data is to become our newly appropriated land
- how do we typically lay claims to property and land?
- Mechanisms for property claims based on Roman law (Simpson, 1993) and identify:
 - Occupation (only when ‘belonging to no-one’ or ‘terra nullius’ for land)
 - Conquest
 - Cession (by treaty)

Simpson, G. (1993). Mabo, international law, terra nullius and the stories of settlement: an unresolved jurisprudence. *Melb. UL Rev.*, 19, 195.

- European colonization had sanction from the Catholic Church, through
 - the ‘doctrine of discovery,’
 - provided legal justification for the occupation of ‘newly discovered’ territories by the colonizing authorities who laid claim to them.
 - “The doctrine was laid out in a series of papal “bulls,” or decrees; the first one was issued in 1452. They authorized colonial powers such as Spain and Portugal to seize lands and subjugate people in Africa and the “New World,” as long as people on the lands were not Christians.

DOCTRINE REPUDIATED

- Nearly 500 years after papal decrees were used to rationalize Europe's colonial conquests,
- the Vatican repudiated those decrees on Thursday,
- the "Doctrine of Discovery" that was used to justify snuffing out Indigenous people's culture and livelihoods
- is not part of the Catholic faith. [3]

Chappell, B. (2023, March). The Vatican repudiates 'Doctrine of Discovery,' which was used to justify colonialism.

NPR. <https://www.npr.org/2023/03/30/1167056438/vatican-doctrine-of-discovery-colonialism-indigenous>

- while the legitimacy of the doctrine has now belatedly been repudiated and
- “the mindset of cultural or racial superiority which allowed for that objectification or subjection of people has been renounced,” [3]
- much of the legal basis for land ownership in settler societies still relies upon it.
- As one example, Bess has reported that:
- The United States Supreme Court in 2005, relying on a series of Indian law cases going back to 1823, specifically cited the Doctrine in its decision denying the right of the Oneida Indian Nation of New York to regain its territory. Justice Ruth Bader Ginsburg wrote in the 2005 decision. “Under the Doctrine of Discovery ...fee title to the land occupied by Indians when the colonists arrived became vested in the sovereign – first the discovering European nation and later the original States and the United States.”

TERRA NULLIUS IN AUSTRALIA?

- Simpson's critique of the use of the doctrine and the tweaking of the definition of the term "terra nullius" in Australia,
- "precedent is a deity greater than universally accepted history in some cases... the judiciary ignored international law and history and called its decisions 'precedent';
- in Mabo, it rewrote international law and the common law, and
- called the decision 'justice'.
- What must the original inhabitants of this land make of such mysticism?"
- **So summarising the basis for claims below**

Simpson, G. (1993). Mabo, international law, terra nullius and the stories of settlement: an unresolved jurisprudence. *Melb. UL Rev.*, 19, 195.

"**Occupation** derives from the natural mode of acquisition in Roman law known as *occupatio*. *Occupatio* could only confer title over objects which were *res nullius* - i.e. belonging to no-one".

The doctrine, of course, became known as *terra nullius* when it was applied exclusively to land rather than objects generally. If land was *terra nullius* it could be acquired through occupation. The corollary to this was that title **could only be acquired through occupation if the land was terra nullius**. *Terra nullius* was land that was either deserted or uninhabited ...or inhabited by uncivilized or disorganized groups (this was the general international law view).

In cases where the land was occupied by peoples having a system of social organization, land could only be acquired or colonized through either **conquest** or **cession (treaty)**. This was typically the practice in Asia, Latin America and North America... In Asia, *terra nullius* was thought to have little relevance to the well-organized tribal societies in existence at the time of European colonization, and most territorial acquisitions occurred by **cession or treaty**... The Spanish, on the other hand, acquired sovereignty over Latin America by **conquest**..while in North America a whole variety of methods were used ranging from treaties to conquest, but generally not mere occupation."

In New Zealand the Maori people were thought to fall into category of **cession**, and therefore treaties were concluded between the indigenous inhabitants and the European settlers... Notoriously, of course, Australia was regarded as falling into the category of ***terra nullius***.

Figure 1: Methods of acquisition for Land and Property [Ex. 9]

AI AND PROPERTY RIGHTS?

- What about AI and property rights.
- How will these legal games play out in the face of today's new “terra nullius” being
- the data libraries, datasets, and scrapings from territories of the internet,
- Newly ‘discovered’ by the creators of LLMs such as ChatGPT
- and their design and use of generative AI systems based on these
- implicit territorial claims?



Simpson, G. (1993). Mabo, international law, terra nullius and the stories of settlement: an unresolved jurisprudence. *Melb. UL Rev.*, 19, 195.

- In the face of this new attempted form of colonization,
- major risks exist for users and owners of systems,
- and major battles lie ahead over intellectual property rights.
- In the New Zealand context, the users of chatbots such as ChatGPT should exercise caution over ownership issues with copyright experts questioning “who ‘owns’ parts of the essay, song lyrics, poems, speeches, blogs or other features that the chatbot spouts out? [arguing that] the chatbot technology arrived so quickly that users have not had time to think through the implications.”

- New Zealand copyright expert Moon has made the point that
 - “somewhere along the way, those words, sentence sequences,
 - images and sounds are likely to have been input by a human.
-
- And under New Zealand law, that content is automatically protected
 - by copyright for the life of the author or creator, and 50 years beyond that person’s
 - death.” [8]

Phare, J. (2023, 11/09/2023). Warning: Using AI chatbots like ChatGPT could get you sued for copyright breaches. *NZ Herald*. <https://www.nzherald.co.nz/nz/warning-using-ai-chatbots-like-chat-gpt-could-get-you-sued-for-copyright-breaches/BWZGMWDN6JHGVFAQI6U6DLYSSM/>

- “in the US, only works produced by a human can be registered.
- But in New Zealand, all computer-generated work, including ‘new’ content created by a chatbot like ChatGPT, is protected under copyright for 50 years.
- But apart from ChatGPT outputs, there is a copyright risk for AI users who “educate”
- or “train” their systems to generate material, Moon says. That includes AI systems
- like GitHub Copilot, used to help write computer programs, which are protected
- under literary works.” [8]

Phare, J. (2023, 11/09/2023). Warning: Using AI chatbots like ChatGPT could get you sued for copyright breaches. *NZ Herald*. <https://www.nzherald.co.nz/nz/warning-using-ai-chatbots-like-chat-gpt-could-get-you-sued-for-copyright-breaches/BWZGMWDN6JHGVFAQI6U6DLYSSM/>

- Further “Moon predicts there could well be a debate over whether New Zealand
 - should drop its copyright protection for computer-generated works altogether.
-
- He thinks it’s a debate worth having, given that few other countries offer that
 - protection.” [8]

Phare, J. (2023, 11/09/2023). Warning: Using AI chatbots like ChatGPT could get you sued for copyright breaches. *NZ Herald*. <https://www.nzherald.co.nz/nz/warning-using-ai-chatbots-like-chat-gpt-could-get-you-sued-for-copyright-breaches/BWZGMWDN6JHGVFAQI6U6DLYSSM/>

- In Australia concerns raised about the Books3 dataset,
- chief executive of Australia's Copyright Agency,
- described the Books3 development as 'a free kick to big tech' at the expense of Australia's creative and cultural life.
- 'We're going to need greater transparency – how these tools have been developed, trained, how they operate – before people can truly understand what their legal rights might be,' she said...
- Australian copyright law protects creators of original content from data scraping.

Burke, K. (2023, 28/09/2023). 'Biggest act of copyright theft in history': thousands of Australian books allegedly used to train AI model. *The Guardian*. <https://www.theguardian.com/australia-news/2023/sep/28/australian-books-training-ai-books3-stolen-pirated>

- Litigation in the US against ChatGPT creator OpenAI over use of allegedly pirated book datasets, Books1 and Books2 (which do not appear to be affiliated with Books3) has already commenced.
- *“The New York Times has sued OpenAI and Microsoft for the unpermitted use of Times articles to train GPT large language models. The case could have a significant impact on the relationship between generative AI and copyright law, particularly with respect to fair use, and could ultimately determine whether and how AI models are built.”* [Pope, 2024]

Pope, A. (2024, 26 July). NYT v. OpenAI: The Times's About-Face. *Blog Essay*.
<https://harvardlawreview.org/blog/2024/04/nyt-v-openai-the-timess-about-face/>

- risks and legal exposure for
- educational institutions and students using
- generative AI through LLMs and systems
- such as ChatGPT are unclear.

- **Problem**

- Proliferating data collection, advanced algorithms, and powerful computers have made it easy to piece together information about individuals' private lives from public information as controls over information privacy become increasingly ineffective

- **Policy Implications**

- Proliferating data collection, use, and publication present rapidly accumulating risks of private information disclosure that require regulation to mitigate.

- Traditional approaches to anonymization, deidentification, and disclosure control fail to protect information at its current scale and are entirely unable to deal with new ways of utilizing information, such as generative AI.

- Inherently imperfect legal and technical solutions must balance individuals' and stakeholders' needs for data privacy and accuracy.

- Altman, M., Cohen, A., & Nissim, K. (2024). *ACM TechBrief: Data Privacy Protection*. Association for Computing Machinery.

COLONIZATION - OLD AND NEW STRATEGIES

...well-rehearsed strategies for colonization shine through.

...mindsets of cultural and racial superiority that justified the legal subterfuge of “terra nullius,”

when ‘big tech’ seeks to arrive and take the digitized traces of our lives and use them without permission or recompense to generate copies and derivative analogues for their own commercial purposes.

A REGULATORY RACE?

In the face of this inexorable private ‘land grab’

An ongoing ‘regulatory race’ is one necessary response as a strategy for asserting human rights through the public sphere.

Capitalist societies have long shown the ability to regulate natural monopolies for the common good,

for instance the European Union’s definition of ‘big tech’ companies as ‘utility providers’ of “very large commercial online platforms.”

THE EU AND REGULATION

Ironically the European Union, the original source of colonisation, has shown insight through the ability to act with respect to AI

through its recent “Regulation on Artificial Intelligence (the EU AI Act)” [5].

The Act classifies AI into four levels of risk based on the intended use of a system:

- 1) unacceptable;
- 2) high;
- 3) limited, and
- 4) minimal risk,

where the Act is most concerned with ‘high-risk AI.’

THE EU AND REGULATION

I can envisage the dominant large language models in due course being similarly defined as ‘utility platforms’ with accompanying and evolving regulations,

as a likely development to rebalance the private and public spheres.

With our students we need to become aware that

resistance and vigilance on the part of citizens in these data wars

will be a necessary part of this ongoing struggle against neo-digi-colonization.

IMPLICATIONS FOR TEACHING SE 1?

- A shift to a dialogic model of interacting with systems
- Prompt Engineering and refinement of strategies
- API's to connect with backend chatbots easily implemented [e.g. Vercel
<https://vercel.com/changelog/next-js-ai-chatbot-2-0>]
- Good reviews of the technology and its implications:
 - Ebert, C., & Louridas, P. (2023). Generative AI for software practitioners. *IEEE Software*, 40(4), 30-38.
 - Ozkaya, I. (2023). Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software*, 40(3), 4-8.
<https://doi.org/10.1109/MS.2023.3248401>

IMPLICATIONS FOR TEACHING SE 2?

- What to do about various forms of cheating?
- How to shift to acceptable and valuable behaviour
- Just today's 4GL and end-user development fad?
- How to develop judgement and handling of reviews and errors
- Hallucinations an inherent design of LLM technology?
- How to co-exist with Gen-AI services?
- How to redesign courses and assessments?

JUDGEMENT AND ERRORS?

- How to develop judgement and handling of reviews and errors

...overall feedback very detailed, long, and not always well ordered.

48 % of the generated feedback is incomplete and/or not fully correct, containing incorrect classifications, redundancies, inconsistencies, or problematic explanations.

...can make it more difficult for students to understand the feedback, increasing the cognitive load [41].

Some comments in feedback mention, generics, concurrency, or improvements on the provided interface, ...likely to overwhelm novices who do not yet know these concepts.

To conclude, using GPT-4 Turbo for automatically generating feedback does not seem to be advisable. The same applies to students using it without guidance or prior instruction.

- Azaiz, I., Kiesler, N., & Strickroth, S. (2024). Feedback-Generation for Programming Exercises With GPT-4. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1 (pp. 31-37).

IMPLICATIONS FOR TEACHING SE 3?

- Commercial services and downsides
- Cost and accessibility issues?
- What to do about various forms of cheating?
- Observed violations of service agreements – who reports miscreants?
- Privacy and IP rights
- Whose work and evolving citation standards?
- AI systems and embedded biases
- Ethical awareness

ACADEMIC INTEGRITY - THEN?

You **can** use AI when writing or preparing a presentation if you use it to help you improve small aspects of your work, such as:

- grammar and punctuation, and
- formal terminology.

You **cannot** use AI to generate ideas when writing, preparing a presentation or creating an artwork/artefact (unless otherwise specified in your assessment instructions). The ideas have to come from you and your course materials.

<https://canvas.aut.ac.nz/courses/7624/pages/referencing>
9/01/2023 - and evolving



After drafting a paragraph for a group assignment, Jude uses the Grammarly AI writing assistant to check their writing. Grammarly identifies spelling mistakes and where the writing is too wordy. Jude reads the suggested changes and explanations about why some of their writing can be improved. They then make some improvements to their work.

Jude has acted appropriately because they have:

- only used AI to identify small errors in their own work,
- used the AI's explanations to learn more about academic writing, and
- made their own improvements.



Karl quickly writes an essay on the day it needs to be submitted. He provides ChatGPT with the assessment task instructions, a list of required readings, and his essay, and he then prompts ChatGPT to write a better version. ChatGPT completely reorganises the structure and content of Karl's work. Karl then submits ChatGPT's version because it looks better than his own.

Karl has not acted appropriately because he has:

- submitted work that he did not do himself (he submitted ChatGPT's work), and this is a breach of AUT's academic integrity guidelines. Even if Karl had acknowledged his use of ChatGPT in the essay, this would still be inappropriate because he did not write the essay himself.

Figure 4: Appropriate and inappropriate uses of AI when writing and presenting

Reference

AAIN Generative AI Working Group. (2023). *AAIN Generative Artificial Intelligence Guidelines*, Australian Academic Integrity Network. <https://doi.org/10.26187/sbwr-kq49> ↗
<https://doi.org/10.26187/sbwr-kq49>

ACADEMIC INTEGRITY - NOW?



Submit only your own work*

Acknowledge all sources of information you use by:

- using the appropriate referencing style, and
- paraphrasing or quoting any words/ideas that are not your own.



Do not submit work done by others, such as:

- other students*
- friends or relatives
- assignment writing services
- artificial intelligence software**, like ChatGPT.

Do not submit work that you have previously submitted for assessment.

<https://canvas.aut.ac.nz/courses/7624/pages/academic-integrity> 10/05/2024

- and evolving

Figure 1: How to maintain academic integrity

* Unless the work is for a designated (group) task

** If your assessment requires the approved use of artificial intelligence, you must acknowledge wherever you do so in your work with an appropriate in-text reference (see guidelines for APA [↗](https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312), Chicago [↗](https://aut.ac.nz.libguides.com/turabian/personalcomms#s-lg-box-22370438), and Harvard [↗](https://aut.ac.nz.libguides.com/c.php?g=919289&p=6648988#s-lg-box-22370440) referencing styles).

ACADEMIC INTEGRITY – NOW?

Artificial intelligence software

Referencing the information generated by an algorithm or artificial intelligence software tool, such as ChatGPT.

Credit the author of the algorithm/AI tool with a reference list entry and an in-text citation.

Reference list format

Who	When	What	Where
Author of AI tool.	(Year released).	<i>Title of tool</i> (Version) [Description].	URL to access tool

Reference list example

OpenAI. (2023). *ChatGPT* (Mar 14 version) [Large language model]. <https://chat.openai.com/chat>

In-text citation examples

OpenAI (2023) generated the following response when...

...that indicates a limitation of the software (OpenAI, 2023).

In your writing

- Describe how you used the tool.
- Provide the prompt you used.
- Provide any portion of the relevant text that was generated in response.
- Document the exact text created because tools like ChatGPT generate unique responses in each chat session, even if given the same prompt.

More information

<https://aut.ac.nz.libguides.com/APA7th/software#s-lg-box-22369312>

<https://apastyle.apa.org/blog/how-to-cite-chatgpt>

ChatGPT – Terms of Use

What you can do. Subject to your compliance with these Terms, you may access and use our Services.

In using our Services, you must comply with all applicable laws as well as our Sharing & Publication Policy, Usage Policies, and any other documentation, guidelines, or policies we make available to you.

What you cannot do. You may not use our Services for any illegal, harmful, or abusive activity. For example, you may not:

- Use our Services in a way that infringes, misappropriates or violates anyone's rights.
- ...
- Represent that Output was human-generated when it was not.
-
- Use Output to develop models that compete with OpenAI.

OpenAI. (2024,
January 31 2024).
Terms of Use.
Retrieved
10/05/2024 from
<https://openai.com/policies/terms-of-use>

ChatGPT – Terms of Use

Your content. You may provide input to the Services (“Input”), and receive output from the Services based on the Input (“Output”). Input and Output are collectively “Content.”

“Content.” You are responsible for Content, including ensuring that it does not violate any applicable law or these Terms. You represent and warrant that you have all rights, licenses, and permissions needed to provide Input to our Services.

Our use of content. We may use Content to provide, maintain, develop, and improve our Services, comply with applicable law, enforce our terms and policies, and keep our Services safe.

Opt out. If you do not want us to use your Content to train our models, you can opt out by following the instructions in this Help Center article. Please note that in some cases this may limit the ability of our Services to better address your specific use case.

OpenAI. (2024, January 31 2024). *Terms of Use*. Retrieved 10/05/2024 from <https://openai.com/policies/terms-of-use>

BROADER IMPLICATIONS

- Giving data into the datacube of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
<https://commons.wikimedia.org/w/index.php?curid=58277452>



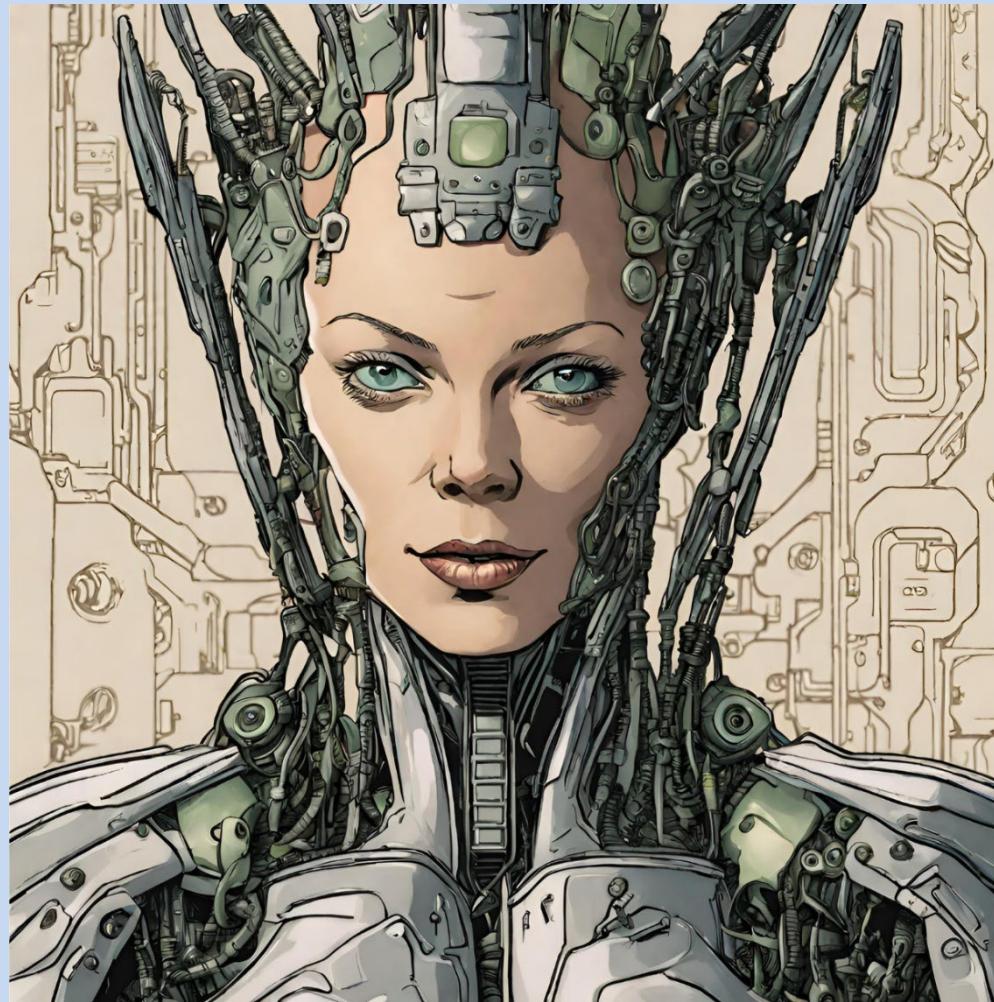
IMPLICATIONS?

- Giving data into the maw of the BORG!
- By Tomás Del Coro from Las Vegas, Nevada, USA - Trekkie - Borg - Star Trek Convention, CC BY-SA 2.0,
https://upload.wikimedia.org/wikipedia/commons/5/52/Trekkie_-_Borg_-_Star_Trek_Convention_%2889504912575%29.jpg



IMPLICATIONS FOR TEACHING SE?

- Giving data into the clutches of the BORG and Getting it Back?



[https://www.mediawiki.org/wiki/File:
Borg_Queen_by_Canva_AI.png](https://www.mediawiki.org/wiki/File:Borg_Queen_by_Canva_AI.png)

Polski: Artystyczny wizerunek królowej Borg z uniwersum Star trek wygenerowany przez oprogramowanie Canva Magic Multimedia

English: Borg Queen from Star Trek universe artistic vision generated by Canva Magic Multimedia

4 April 2024

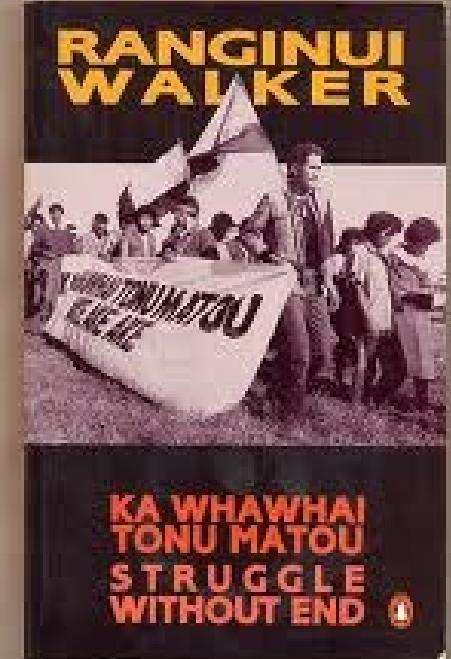
Own work

[Canva Magic Multimedia](#)

CONCLUSION

- Discussed Inroads column on LLM's
- Covered waves of colonization and technology
- Covered surveillance capitalism
- Our lives and data!
- Neo-digi-colonization
- The need for openness
- The doctrine of discovery
- Making property and land claims
- Links with AI and intellectual property rights
- Data Privacy
- Regulatory races
- Implications for Competencies and SE Education

Generative AI a Critique: Questions?



Associate Professor Tony Clear
Department of Computer Science and Software
Engineering,
Auckland University of Technology

Tony.clear@aut.ac.nz



AUT

ENSE 701 – Virtual Team Guidance (based on Comp501 presentation)



Course lecturer –
Assoc. Prof. Tony Clear



Outline

1. COVID19 and Working as a Virtual Team
2. Some Guidance from the literature
 1. Jarvenpaa & Leidner (1998)
 2. Gordon (2017)
3. Hybrid Workplaces
4. Stages in the life of a virtual team
5. Stages of your Team Project Assignment
6. Key actions at each stage



COVID19 and Working as a Virtual Team

1. Lockdown meant working online as a team
2. Need to balance home life and study
3. Need to work around any technology issues
4. Need to communicate actively
5. Let others know if you are not going to be able to:
 1. Be at a meeting
 2. Deliver your agreed contribution
 3. Perform your role
 4. Meet milestones
6. Agree a backup plan!
7. Be sensitive to one another and your needs in a difficult time
8. But committing to the work and one another will help
 1. Bring satisfaction through achievement
 2. Bring a sense of mutual support
9. Let engaging in the learning give you structure in an uncertain time

Virtual Teams – Some Guidance from the Literature

1. Virtual Teams have been an active field of study for 35 years
2. My own research spans 25 of those
3. But still challenging!
4. See the issues discussed in my recent *Inroads* column

Clear, T. (2021). Loosening Ties: Permanently Virtual Teams and the Melting Iceberg of Relationship. *ACM Inroads*, 12(3), 6-8. <https://doi.org/10.1145/3479419>

Beyond Virtual Teams –Hybrid Workplaces

1. The word “hybrid” has become one popular umbrella label attributed to various work-related terms.
2. These days, we often read about hybrid workplaces or hybrid offices, hybrid work or working, as well as hybrid teams.
3. Google Workspace experts define *hybrid work* as “*a spectrum of flexible work arrangements in which an employee’s work location and/or hours are not strictly standardized.*”
4. In other words, anything that lies in the middle of “in the office, nine till five” and “anywhere around the world at any time.”

Smite, D., Christensen, E. L., Tell, P., & Russo, D. (2023). The Future Workplace: Characterizing the Spectrum of Hybrid Work Arrangements for Software Teams. *IEEE Software*, 40(2), 34-41. <https://doi.org/10.1109/MS.2022.3230289>

Hybrid Workplaces and Team Typologies

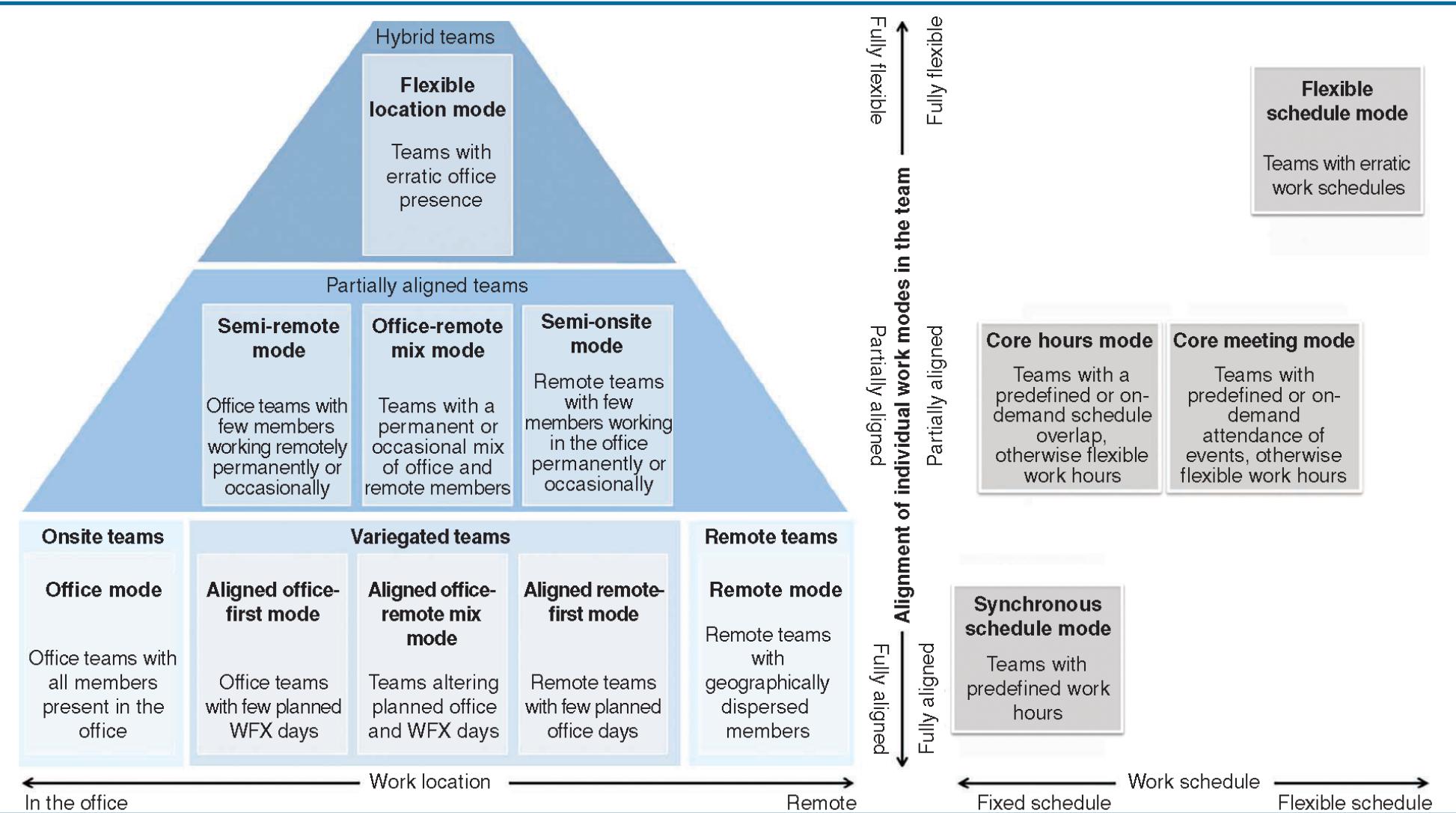


FIGURE 1. Team typology and the spectrum of work arrangements.

Navigating BCIS Team availability and location?

Manage My Lunch Team General Availability

Time	MON	TUE	WED	THU	FRI	SAT	SUN
8:00							
8:30							
9:00							
9:30							
10:00							
10:30							
11:00							
11:30							
12:00							
12:30							
13:00							
13:30							
14:00							
14:30							
15:00							
15:30							
16:00							
16:30							
17:00							

 Team is generally available
 A team member is not available

Virtual Teams – Jarvenpaa & Leidner (1999)

"in global virtual teams, trust might take on a form of swift trust with some variations.

Trust might be imported, but is more likely created via a communication behavior established in the first few keystrokes.

Communication that rallies around the project and tasks appears to be necessary to maintain trust.

Social communication that complements rather than substitutes for task communication may strengthen trust.

Finally, responding behaviors are as critical as initiating behaviors and members have to explicitly verbalize their commitment, excitement and optimism" (Jarvenpaa & Leidner, 1999).

Jarvenpaa, S., & Leidner, D. (1999). Communication and Trust in Global Virtual Teams. *Organization Science*, 10(6), 791-815.

Jarvenpaa, S., & Leidner, D. (1998). Communication and Trust in Global Virtual Teams. *Journal of Computer Mediated Communication*, 3(4).

Virtual Teams – Jarvenpaa & Leidner (1998)

The trust facilitating behaviours and actions are depicted in the table below.

Communication Behaviors that facilitated trust early in a group's life	Communication Behaviors that helped maintain trust later in a group's life
Social communication	Predictable communication
Communication of enthusiasm	Substantial and timely responses
Member actions that facilitated trust early in a group's life	Member actions that helped maintain trust later in a group's life
Coping with technical uncertainty	Successful transition from social to Procedural to task focus
Individual initiative	Positive leadership
	Phlegmatic response to crises

Table 6: Trust Facilitating Communication Behaviours And Member Actions (from Jarvenpaa & Leidner, 1998)

Virtual Teams – Gordon (2017)

In summary, the study shows that the most appropriate interactive leadership styles are significant and different from those of face-to-face teams. Furthermore, these styles are affected by, and affect, each team stage. Exercising the explaining and directive styles, along with the questioning style, in all stages of a team, but particularly in stages 1 and 2, may be the most effective way to elicit actions. As a result of this action research it was agreed to make the following six findings more explicit to the GlobCom team, including:

Gordon, A. (2017). *Leadership Interaction in Global Virtual Teams: Roles Models and Challenges* (PhD Thesis). Auckland University of Technology, Auckland. Retrieved from <http://hdl.handle.net/10292/10769>

Virtual Teams – Gordon (2017 cont'd)

1. Team stages have different levels of activity and demands and identifying them can help structure the workload.
2. Delegated roles are more likely to elicit actions and justify the creation of additional roles.
3. The directive and explanatory interaction styles are the most effective in eliciting actions with additional styles added depending on the situation.
4. Team quiescence occurs in the middle stages and needs to be reinvigorated to avoid a loss of team momentum.
5. The apologetic interaction style, despite it being socioemotional with its intention to be caring, is likely to diminish telepresence.
6. The team mentor role generates feedback and may need to be more active earlier.

Gordon, A. (2017). *Leadership Interaction in Global Virtual Teams: Roles Models and Challenges* (PhD Thesis). Auckland University of Technology, Auckland. Retrieved from <http://hdl.handle.net/10292/10769>

Stages in the Life of a Virtual Team

In a broad sense we can think of virtual teams having three stages:

- Beginning
- Middle
- End



Stages of your team project assignment

- Beginning
 - Team Member Allocation
 - Team Project Draft Proposal
 - **Team Proposal Presentation (assessed)**
- Middle
 - Portfolio Review
- End
 - **Team Project Presentation (assessed)**
 - **Team Portfolio Submission (assessed)**
 - **Team Portfolio [individual components] Submission (assessed)**

Key actions at each stage

- Beginning
 - (stage 1) Team Member Allocation
 - (stage 2) Team Project Draft Proposal
 - **(stage 3) Team Proposal Presentation (assessed)**
- Middle
 - (stage 4) Portfolio Progress Review
- End
 - **(Stage 5) Team Project Presentation (assessed)**
 - **(Stage 6) Team Portfolio Submission (assessed)**
 - **(Stage 6) Team Portfolio Submission [individual components] (assessed)**

Key actions at each stage – Virtual

- Beginning
 - (stage 1) Team Member Allocation – completed prior to lockdown but team engagement unclear
 - (stage 2) Team Project Draft Proposal – completed?? but team engagement unclear
 - **(stage 3) Team Proposal Presentation (assessed) – Due first week after break**

Additional Actions to Support Virtual Team Functioning

- Team icebreaker – using Team wiki (see next slide)
- Team Leader selection [complementing roles identified from stage 3]
- *Refer to Jarvenpaa & Leidner – actions early in team's life (social communication, communication of enthusiasm, coping with uncertainty, individual initiative)*
- *Refer to Gordon – defined roles elicit actions; interaction by giving directions and explanations helps elicit action*

Key actions at each stage – Icebreaker

- Team icebreaker – using Team wiki under groups on Blackboard
 - To check your team (under Teams tutorials – labs channel)
https://teams.microsoft.com/l/file/7B3E5B1E-D3EB-4A9D-9FCC-16CFB2C6B13A?tenantId=5e022ca1-5c04-4f87-8db7-d588726274e3&fileType=xlsx&objectUrl=https%3A%2F%2Fautuni.sharepoint.com%2Fsites%2FCOMP501_2021_02ComputingTechnologyinSocietySem22021%2FShared%20Documents%2FLectures%2FAssignment%203%20Team%20Assigned.xlsx&baseUrl=https%3A%2F%2Fautuni.sharepoint.com%2Fsites%2FCOMP501_2021_02ComputingTechnologyinSocietySem22021&serviceName=teams&threadId=19:6bb6686b76fc4085a61e645979b7c7c2@thread.tacv2&groupId=3fb30b0c-e8ab-43cf-998a-2af3a591f321
- Team Leader selection
- The Icebreaking phase involves virtual team (VT) members getting to know each other and appointing a group leader.
- **Task 1:** Participants in each VT become acquainted with one another and each student creates a “page” in their group’s wiki to introduce themselves to the team.
- Students in each VT interact in their VT wiki and each group member creates their own page there. It is suggested that in their page each student provide some information about things like their birthplace, favourite travel destination, interests and hobbies, their future career plans and their photo, and also some web links to make it more informative.
- **Task 2:** Based on their introductions, the students in each VT need to **select a leader** for their virtual team. Use the VT Discussion boards or Teams or other agreed way of communicating among your members – make sure you agree how you will communicate!
- Also agree how you will meet online – Doodle <https://doodle.com/create> is a helpful way to organise common meeting times

Key actions at each stage – Mid Project

- Middle
 - (stage 4) Portfolio Progress Review

Additional Actions to Support Virtual Team Functioning

- Team Leader initiative required
 - Plan time for joint review of separate components so they can be integrated
 - Consider establishing an editor role to ensure consistent style and approach
- *Refer to Jarvenpaa & Leidner – actions later in team's life (predictable communication; substantial and timely responses; successful transition from social, to procedural to task focus; positive leadership; phlegmatic response to crises)*
- *Refer to Gordon – team quiescence occurs in the middle stages and needs to be reinvigorated to avoid a loss of team momentum; delegated roles are more likely to elicit actions and justify the creation of additional roles*
 - Make sure to reinforce teams approach to how they will communicate!
 - Make sure team roles and tasks are planned and reinforced
 - Also reinforce approach and how team will meet online – Doodle <https://doodle.com/create> is a helpful way to organise common meeting times

Key actions at each stage – End Project

- End
- **(Stage 5) Team Project Presentation (assessed)**
- **(Stage 6) Team Portfolio Submission (assessed)**
- **(Stage 6) Team Portfolio Submission [individual components] (assessed)**

Additional Actions to Support Virtual Team Functioning

- Team Leader initiative required – also contact TA or Lecturer if any concerns
- Make sure team roles, tasks, deadlines are reinforced
- Confirm presentation roles (stage 5)
- Include time for joint review of separate components so they can be integrated
- Review the need and contribution of the editor role to ensure consistent style and approach
- *Refer to Jarvenpaa & Leidner – actions later in team's life (predictable communication; substantial and timely responses; successful transition from social, to procedural to task focus; positive leadership; phlegmatic response to crises)*
- *Refer to Gordon – team quiescence occurs in the middle stages and needs to be reinvigorated to avoid a loss of team momentum; ; delegated roles are more likely to elicit actions and justify the creation of additional roles*
- Make sure to reinforce teams approach to how they will communicate!
- Also reinforce approach and how team will meet online – Doodle <https://doodle.com/create> is a helpful way to organise common meeting times

Key actions at each stage – End Project

- End - Final
- **(Stage 5) Team Project Presentation (assessed)**
- **(Stage 6) Team Portfolio Submission (assessed)**
- **(Stage 6) Team Portfolio Submission [individual components] (assessed)**

Additional Actions to Support Virtual Team Functioning

- Confirm who is submitting the team portfolio
- Remember to submit your own components
- Remember to thank your team members for their contributions, their effort and collegial approach to supporting one another in doing good work ☺

Driving Process Improvement Via Comparative Agility Assessment

Laurie Williams¹, Kenny Rubin², Mike Cohn³

North Carolina State University, Raleigh, NC, USA, williams@csc.ncsu.edu

Innolution, USA, krubin@innolution.com

Mountain Goat Software, USA, mike@mountaingoatsoftware.com

Abstract—Rather than striving to be “perfectly agile,” some organizations desire to be more agile than their competition and/or the industry. The Comparative Agility™ (CA) assessment tool can be used to aid organizations in determining their relative agility compared with other teams who responded to CA. The results of CA can be used by a team to guide process improvement related to the use of agile software development practices. This paper provides an overview of industry trends in agility based upon 1,235 CA respondents in a range of domains and geographical locations. Additionally, the paper goes further in depth on the results of four industrial teams who responded to the CA, explaining why their results were relatively high or low based upon experiences with the teams. The paper also discusses the resultant process improvement reactions and plans of these teams subsequent to reviewing their CA results.

Keywords-component; agile software development, comparative agility survey

I. INTRODUCTION

Organizations primarily strive to beat out their competition rather than to be “perfect”. Some organizations may adopt agile practices to improve their ability to more rapidly and affordably create high quality products and services in their quest to obtain advantage over their competitors. These organizations do not pursue agility for the sake of being “perfectly” agile. Rather, these organizations strive to be more agile than their competition and other industrial organizations because of beliefs they have about the benefits of being agile.

How agile is agile enough? Hypothetically, an organization may be aware that their teams are struggling with the adoption of the test-driven development (TDD) [1] practice that is commonly associated with agile software development. This organization may be comforted if it knew that industry trends indicate low adoption of TDD. Conversely, the organization may heighten its intensity in adopting TDD if, instead, its adoption is lagging behind that of other industrial organizations. The Comparative Agility™ (CA) assessment tool developed by the second two authors, as will be explained in Section II, was created to aid organizations in understanding how their agile

adoption compares with others’ agile adoption. As such, teams can use CA as a vehicle for process improvement.

In this paper, we share industry trends in agile adoption based upon 1,235 CA respondents in a range of domains and geographical locations. We provide additional insight on these industry trends by sharing survey results and observations of 119 respondents from four industrial¹ teams working in one large US company. The teams were provided detailed comparison with industry trends. The teams were able to utilize the CA results to guide in process improvement.

Section II provides information about the CA assessment instrument. Section III briefly compares CA with two other assessment instruments. Section IV provides contextual information on survey respondents and on the industrial teams. Section V presents and explains survey responses. We summarize in Section VI.

II. COMPARATIVE AGILITY ASSESSMENT

CA is a survey-based assessment tool used by individuals and organizations wanting to compare their own agility to that of others. Any agile practitioner can visit the CA website² and, in exchange for investing his or her time to complete the survey, receive a free report that compares his or her survey results to the complete industry dataset. Alternatively, teams can request³ to have a customized collector. These team members then individually take the survey using a team-specific survey URL. These teams are provided a team-based comparison against some or all of the remaining dataset.

At the highest level, the CA approach assesses agility on seven *dimensions*: Teamwork; Requirements; Planning; Technical Practices; Quality; Culture; and Knowledge Creating. Each dimension is made up of three to six *characteristics* for a total of 32 characteristics. Each characteristic has approximately four *statements* that are assessed by the respondents for a total of 125 statements.

¹ The company prefers not to be identified.

² <http://www.comparativeagility.com/>

³ Those wishing to obtain a customized collector should contact the second author of this paper.

Results can be obtained/aggregated at the dimension, characteristic, or statement level.

Each statement is an agile practice for which the respondent indicates the truth of the statement relative to their team or organization. For example, one of the seven dimensions, Planning, has five characteristics: (1) planning levels; (2) critical variables; (3) progress tracking; (4) sources of dates and estimates; and (5) when do we plan. The “when do we plan” characteristic has four statements:

- Upfront planning is helpful without being excessive.
- Team members leave planning meetings knowing what needs to be done and have confidence they can meet their commitments
- Teams communicate the need to change release date or scope as soon as they are discovered.
- Effort spent on planning is spread approximately evenly throughout the project.

CA respondents choose the appropriate response given their situation, using a five point Likert scale: True; More true than false; Neither true nor false; More false than true; or False.

Through a combination of dimensions, characteristics, and statements, a team or organization can see how they compare to other organizations, or to themselves at an earlier time. For example, a team doing web development may compare itself to all other teams doing web development and find that they lag their competitors at adopting agile technical practices as shown by their score on the Technical Practices dimension. This information on its own could be enough for the team to structure a process improvement plan. But if the team wants more detail, they can look at the specific characteristics where they most lag their competitors. Example output is shown in Figure 1. The individual respondent or team receives information on the number of standard deviations of their response(s) versus the mean.

CA was designed to lead to actionable results. When an organization can see how it compares with other organizations, improvement efforts can be focused. CA does not provide a view to the purported business results of the respondents based upon their agile adoption.

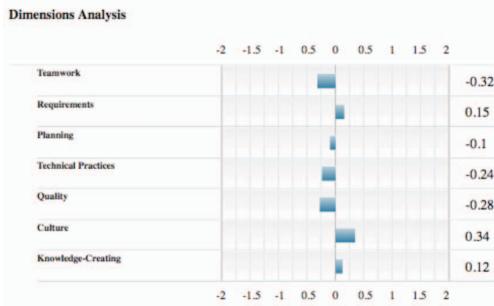


Figure 1: Sample CA Results (number of standard deviations of their response(s) versus the mean)

III. RELATED WORK

Two other assessment frameworks have been used to evaluate agile software development teams. One is the Extreme Programming Evaluation Framework (XP-EF)

[16]. The purpose of the XP-EF is to provide a structure for a case study such that the results of multiple, independent case studies can be combined and compared to create a family of related studies. For example, the results of case studies of industrial Extreme Programming (XP) [2] teams at IBM [16], Sabre Airline Solutions [10], and Tekelec [11] were structured via the XP-EF.

Another assessment framework is the Shodan survey [9]. Similar in intent to CA, the purpose of the Shodan survey is to assess “how XP” a team is. The Shodan survey is specifically focused on the development practices laid out in the XP methodology, and therefore consists of 15 questions rather than CA’s more broad 125 questions.

IV. DATA COLLECTION

The industry-wide data reported in this paper are based upon 1,235 respondents who had taken the survey between August 21, 2007 and February 24, 2010. Surveys were deleted from the database if they were anonymous surveys that were not part of customized collectors and the respondent had answered 20% or less of the questions. A slight majority (54%, N=669) of the responses came from teams who asked for their data to be analyzed via a customized collector, such as teams undergoing coaching and/or training by one of the authors. In other cases (46%, N=566), individuals found the Comparative Agility survey site, such as after seeing articles written about the survey [5], and decided to answer the questions. Based upon these circumstances, as a whole the respondents are considered to be part of agile teams or teams beginning an agile transition and not members of anti-agile and/or plan-driven [4] teams.

In this section, we provide demographic and contextual information about survey respondents from the four teams. The teams are all employees of the same successful US-based company with a positive company culture and job security. All teams work on data-intensive applications. Facilities in this company are structured such that each employee has their own office, which the employees have come to value. Office space constraints prevent additional collaborative team spaces to be created. Employees primarily have desktop computers and not laptops, preventing ease of impromptu collaborative work sessions. Top management at the company was supportive of a transition to agile software development, though had not issued a directive for teams to become agile.

A. Team 1

This team was the company’s pilot of agile software development beginning in early 2007. The team has focused on the use of the Scrum [12] project management practices rather than on the technical practices. They are a small ten-person team consisting of a development manager, five developers, two testers, a product manager, and a project manager. Seven of the team members are co-located in one building. Three of the developers work out of their homes in other geographic locations in the US, one in the same time zone as the rest of the team and the other with only a one-hour time zone difference.

The product manager works closely with the team on a daily basis to clarify requirements and provide acceptance test criteria. The responsibilities of the ScrumMaster are split between the project manager and the development manager, though the Scrum methodology would not advocate such a split. The team develops a product that

ships in conjunction with a corporate release of a larger product every 18-24 months.

This team's pilot was considered a success, which lead to the spread of agile software development to many other teams in the company. In the last pre-agile release, the team delivered 80% of the functionality committed in the release plan. In the first agile release, the team delivered 137% of the functionality committed in the release plan. In the first agile release, the product manager was able to remove requirements from the original release plan that were no longer desired and to add additional market-driven requirements, a practice that had not happened previously.

Nine of ten team members responded to the survey.

B. Team 2

This team began its transition to agile software development in August 2009. This team consists of approximately 63 members who primarily work in one location. However, the team has strong technical dependencies on components developed at another company location (same time zone) and in India. This team further subdivided into four sub-teams

The team began its agile transition with a new release of a product in which the old product was going to be re-architected from scratch. The team felt, at the time, that it had an unachievable release objective and felt uneasy about also using a radically different development process given this challenge. Some team members were positive about using agile as the only feasible means of delivering a product as scheduled while others were against the use of agile, and remain so after six months using agile. The role of ScrumMaster began with the project managers and then transitioned to development technical leads after approximately four months. Some agile methodologists do not advocate the role of ScrumMaster being filled by a technical team member [6].

The larger team was subdivided into four smaller teams. One of these teams was a "research team." The purpose of the research team was to investigate the technical implications of the stories in the backlog. The research team also determined which stories could be implemented in the next iteration based upon technical dependencies on other teams.

This survey was taken in two phases. During the first offering of the survey, only 12 team members responded. Management then encouraged participation, and all remaining team members took the survey. All responses are included in the paper.

C. Team 3

This team began its agile transition in early 2008. Similar to Team 2, this team was a large team that was subdivided into three sub-teams. The team primarily works in two locations in the same time zone but also have team members in England and India.

The product manager is active in story writing, story clarification, and acceptance test creation on a daily basis. This team also decided that only the acceptance testing would be done in the current iteration and most testing of features would occur in the iteration after the feature was implemented. Converse to the practice used by this team, most often agile teams test stories during the current iteration and leave each iteration with a potentially-shippable product [6].

The survey was completed by 20 of 31 team members.

D. Team 4

Team 4 has just begun its agile transition. Many on the team took agile training at the end of 2009. The product line originated as customized, service-based software developed for customers, and the developers were located around the US to be close to customers. Team 4 organized itself into three teams, each with a separate product in the same product family. Each team was able to create its own agile practice and shared experiences throughout the teams. The products also have to pass US Food and Drug Administration (FDA) audits of its process.

The survey was taken by 27 team members out of 35.

E. All Respondents

As discussed, the industry results reported in this paper are based upon input of 1,235 members of agile teams or teams desiring to be agile and are not representative of the industry at large. The respondents ranged from developer to tester to project manager to Chief Technology Officer. The respondents came from many regions in the world. The geographic distribution of responses is shown in Table 1. A respondent could indicate multiple geographic locations if answering for their larger organization so the total in Table 1 exceeds 100%.

Table 1: Geography of survey respondents.
(Multiple responses allowed)

Continent	% of respondents
Africa	0.4%
Antarctica	0.2%
Asia	32.0%
Australia	1.9%
Europe	35.0%
North America	62.0%
South America	1.5%

As shown in Table 2, the respondents had a range of experience with agile software development practices. The majority of the respondents were relatively new to agile software development. This phenomenon is both likely due to the encouragement of the authors to have the teams they are coaching and training to take the survey and also because agile software development is becoming increasingly popular with companies.

Table 2: Experience of respondents.

Experience	% of respondents
0-6 months	54%
7-12 months	14%
1 year	14%
Longer	18%

V. SURVEY RESULTS

In this section, we discuss survey results structured around the seven dimensions of CA. A comparative view of industry adoption of the dimensions in Figure 2. The number presented for each dimension is the mean score of the responses on the set of statements related to that dimension whereby for each question a response of True=5; More true than false=4; Neither true nor false=3; More false than true=2; and False=1.

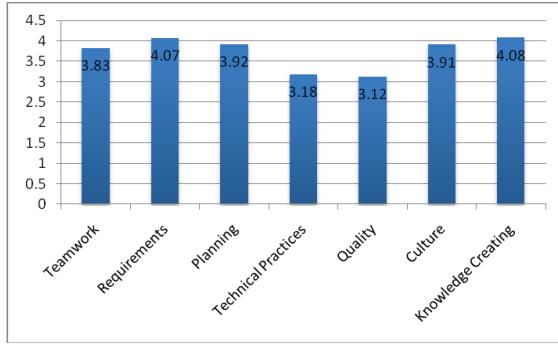


Figure 2: Industry average responses

In the following sub-sections, we will examine each of these dimensions in further depth. The results will reference the survey summary presented in the Appendix.

A. Teamwork

The Teamwork dimension includes the following five characteristics:

- *Team composition*: the extent to which the team operates under a small, focused “whole team” arrangement whereby the multidisciplinary team members all work together for a common goal.
- *Team management*: the extent to which a team is self-empowered and freed from non-value-adding tasks.
- *Focus*: assesses whether the team’s priorities change within an iteration and whether the team feels they are all headed toward the same goal.
- *Communication*: the extent to which the team communicates face-to-face daily and has effective daily Scrum meetings.
- *Team member location*: provides a view to whether team members work on the same floor, same building, same campus, same city, or within the same time zone.

1) *Industry Trends*: The mean response for the Teamwork dimension was 3.83. Some of the particularly strong elements of teamwork among respondents were testers and programmers work on the same team, people were on two or fewer teams, and teams are kept together as long as possible. These teams feel they are all headed toward the same goal. Additionally, these teams have daily Scrum meetings that are most often less than fifteen minutes and are effective at synchronizing work. Finally, more often than not, teams work with substantially overlapping hours, although they less frequently work in close proximity to each other.

Some weaker trends among participants are that teams do not determine who is on or off the team. Additionally, teams deal with non-value-adding tasks and changing management priorities. Lastly, few respondents indicated that they worked in a shared physical environment, such as would be done in a “war room” or team room.

2) *Four Teams*: As a whole, the four teams are particularly below the industry mean, particularly in the team management characteristic. The teams feel significantly less empowered and freed from non-value-

added tasks relative to others who took the survey. Another trend for all five teams was a continuing emphasis on written documentation and signoffs. This company utilized a gate model for project management that perpetuated the need for documentation and signoffs despite their transition to agile software development. At each gate, the continuation of the development process is decided by the management team based upon a review of pre-determined artifacts. Additionally, the process for releasing their product had not been “agilized” so all releases still ended with at least four months of release tasks. Finally, many of the teams were geographically distributed.

Not all teams shared the same difficulties in transitioning to agile practices. In particular, Teams 2 and 4 had team composition challenges whereby the team members were not working on small, focused teams. Team 2 also had priority changes mid-iteration and did not feel that the whole team was headed toward the same goal. Conversely, Team 3 was ahead of the industry averages relative to the whole team heading toward the same goal.

B. Requirements

The Requirements dimension includes the following four characteristics:

- *Communication focus*: the extent to which the requirements are discussed and clarified in “just-in-time” discussions during an iteration.
- *Level of detail*: the requirements are able to start at a high level with little detail; details are negotiated during the iteration.
- *Emergence*: requirements can change based upon changing needs and development recommendations.
- *Technical design*: the technical design for a product is created collaboratively and occurs iteratively throughout a project.

1) *Industry Trends*: The mean response for the Requirements dimension was 4.07, among the highest for all dimensions in the survey indicating an overall “more true than false” response to the 14 requirements-related statements. The respondents strongly agreed that written requirements had to be augmented with discussion. Additionally, product owners willingly acknowledge that features can turn out to be bigger than anticipated.

Conversely, the survey results indicate that detailed requirements documents and technical design can sometimes still be written, and product owners can be criticized for requirements changes.

2) *Four Teams*: To a significant extent, the teams’ results mirrored the industry results. Team 2 who has started its agile transition only six months prior to this data collection seems to still value detailed requirements and technical design. Conversely, Teams 1 and 3 has above average results in all requirements characteristics.

C. Planning

The Planning dimension includes the following five characteristics:

- *Planning levels*: assesses whether there is a release plan that is updated throughout the project and a task plan for each iteration.

- *Critical variables:* the extent to which scope is traded off for schedule and that the product owner does the prioritization.
- *Progress tracking:* the extent to which release burndown and iteration burndown charts are used and tracked.
- *Sources of dates and estimates:* the estimates are developed by the people who will do the work.
- *When do we plan:* planning is done prior and throughout the product cycle.

1) *Industry Trends:* The mean response for the Planning dimension was 3.92. The respondents strongly agreed that the people who are actually doing the work should make estimates and plans. These plans are then broken down into task plans for each iteration. Product owners negotiate tradeoffs between scope and schedule. Upfront planning is helpful without being excessive.

Teams were least likely to create and track to a release burndown chart. A release burndown chart shows the amount of committed features for the release on the y-axis and the number of iterations remaining on the x-axis. Teams were more likely to track to an iteration burndown chart that shows the number of hours of work remaining in the release on the y-axis and the number of days left in the iteration on the x-axis.

2) *Four Teams:* Many of the industrial trends were similar with the teams. However, the teams were less likely to create an initial plan prior to the first iteration of the release that shows an incremental release of features throughout the release cycle. Team 2 had additional difficulties trading off scope for schedule with the team getting prioritization of their work based upon input from the product managers. Additionally, this team had trouble spreading work evenly and having confidence they could meet their commitments. Team 3 continued the trend of having above average results in all planning characteristics. In particular, Team 3's team members felt especially positive about their participation in the estimation process.

D. Technical Practices

The Technical Practices dimension includes the following six characteristics:

- *Test-driven development:* assesses the extent to which the team writes unit tests before implementation code [1].
- *Pair programming:* the extent to which the team utilizes the pair programming practice whereby two team members work side-by-side at one computer, collaborating on the same work artifact [15].
- *Refactoring:* the extent to which explicit effort is spent to improve the design of code without increasing the functionality [7].
- *Continuous integration:* the extent to which new code is integrated into the team code base at least once per day.
- *Coding standards:* the extent to which the team has a set of rules for writing source code.
- *Collective code ownership:* the extent to which anyone on the team can change any code in the code base.

1) *Industry Trends:* The mean response for the Technical Practices dimension was 3.18, among the lowest of all dimensions. Other surveys [3, 13, 14] indicate that many agile teams follow the Scrum methodology [12]. Scrum prescribes project management practices but not technical practices. Our results substantiate these prior surveys because the mean for the technical practices dimension was lower than most other dimensions.

The technical practices most adopted by teams, in decreasing order of frequency, are continuous integration, coding standard, collective code ownership, refactoring, TDD, and pair programming. The first four of these are generally considered best practices among both agile and plan-driven teams. The last two, TDD and pair programming, are often associated with agile software development though these practices could be used by plan-driven teams.

2) *Four Teams:* Similar to many of the respondents, the teams had not generally embraced the technical practices. In particular, none of the teams practice collective code ownership or pair programming. However Teams 2, 3, and 4 did utilize the TDD practice.

E. Quality

The Quality dimension includes the following three characteristics:

- *Automated testing:* assesses the extent to which the team automates their unit tests and runs the tests frequently.
- *Customer acceptance testing:* the extent to which the team has acceptance tests written in conjunction with the product manager and these acceptance test are automated and run each day.
- *Timing:* assesses how early in the development process testing occurs.

1) *Industrial Trends:* The mean response for the Quality dimension was 3.12, the lowest of all the dimensions. Overall, the quality dimension assesses the team's attention to quality throughout all iterations from the start of the iteration and involving the whole team.

Automated unit testing was not practiced by many teams. Acceptance tests were automated even less frequently than unit tests. Acceptance testing is a formal process that is conducted to determine whether or not a system satisfies a set of criteria (i.e. "acceptance criteria") that are pre-determined by the customer to enable the customer to determine whether or not to accept the system [8]. On agile teams, the acceptance tests are to be written in conjunction with the customer or product owner. Survey results indicated that a significant amount of testing is still done manually. Finally, survey results indicate that performance, integration, and scalability testing occurs at the end of the development process. Many respondents indicated that a feature was not considered done until its acceptance tests pass.

2) *Four Teams:* Two main trends occurred with all the teams. First, the teams have an above-average practice of obtaining acceptance test criteria from the product owner. Team 2 had a high adoption rate of automated unit and acceptance testing. Additionally, the teams had a greater propensity toward testing later in the process. In

particular, teams are more likely to test completed features in the following or later iterations.

F. Culture

The Culture dimension includes the following six characteristics:

- *Management style*: the extent to which the team feels pressure to meet deadlines without having their autonomy taken away or unreasonable pressured.
- *Responses to stress*: the extent to which the team responds to pressure by re-prioritizing or re-scoping versus by working overtime or adding people.
- *Customer involvement*: assesses the access the team has to its product owner.
- *Title and salary alignment*: assesses the extent to which the team has incentive and a culture to work together as a team rather than as individuals.
- *Infrastructure*: the extent to which the team has technology and a work environment to support agility and synchronous communication between team member.
- *People*: assesses the skill level, accessibility, and agile attitude of the team members.

1) *Industry Trends*: The mean response for the Culture dimension was 3.91. Most questions had responses around the mean score indicating an overall response of “more true than false.” The items that were higher than the average concerned the product owner and the project manager. Teams were satisfied with their access to these team members and with the job they were doing, even though often the product owner was not co-located with the team.

One item that rated lower was bonuses, annual reviews, and compensation promoting team behavior.

2) *Four Teams*: The teams’ responses followed the overall industry trends just discussed though overall more positive. However, Team 2 had concerns related to response to stress and management style. These results indicate the team is experiencing considerable stress and pressure to meet their deadlines.

G. Knowledge Creating

The Knowledge Creating dimension includes the following three characteristics:

- *Reflection*: the extent to which the team conducts iteration reviews and retrospectives.
- *Timeboxes*: the extent to which the team works in short iterations culminating in working software.
- *Team learning*: assesses the culture within the team to question and learn from each other.

1) *Industry Trends*: The mean response for the Knowledge Creating dimension was 4.08, the highest of all the dimensions. The results indicate that many teams conduct end-of-iteration reviews and retrospectives. Almost all indicate these iterations are no more than 30 days long. The respondents indicated a high propensity toward questioning and learning from each other; valuing new approaches, technologies, skills and practices; sharing a set of common principles; and discussing problems.

2) *Four Teams*: The teams indicated a higher than average active participation in post-iteration reviews. All teams except Team 3 scored lower than average on having

potentially-deliverable software ready at the end of each iteration and iterations not being mini-waterfalls. Indications are that Team 2 has knowledge-creating challenges relative to learning and gaining from retrospectives and each other.

VI. SUMMARY

Using the results of Comparative Agility assessment tool, this paper provided a view into the “state of agility” in the industry based upon 1,235 responses. Industry trends indicate the highest adoption of agile practices occur in the areas of embracing emergent requirements and creating knowledge throughout the iteration and release. The lowest industry adoption occurs relative to utilizing technical practices and focusing on quality throughout all iterations. The short-term focus of iterations coupled with a lack of prescribed engineering practices in the popular Scrum methodology may lead to trouble. “Flaccid Scrum”⁴ refer to teams that utilize only Scrum’s project management practices. Progress eventually slows for Flaccid Scrum teams, according to Fowler, because the team has not paid enough attention to the quality of the code produced during each iteration. In many cases, only the easiest scenario of a feature is demonstrated at the end of the iteration. The feature can then be considered to be “done”, and focus turns to a new set of features for the next iteration.

The four teams utilized their CA results to structure process improvement. Teams 1 and 3 heightened its awareness on the need to automate tests. Team 2 made several changes. First, they began preparing earlier for the upcoming iteration. To further aid in this preparation, they also decided to disband their dedicated research team and to put these members into each of the sub-teams to prepare for future iterations. All Team 2 sub-teams began to have one joint Sprint Review meeting so that all could become aware of the overall progress toward meeting the release goals. Team 3 reviewed its stature relative to industry trends and its peer teams and was proud that they were generally ahead in their agile adoption. Team 4 determined that it needed more training and coaching in several areas. In particular, they determined that team members sometimes had differing views on some agile practices and more training could help form a more united view.

ACKNOWLEDGMENT

The authors would like to thank members of the North Carolina State University Research group for their helpful comments on this paper. Many thanks to the members of the industrial organization. Partial funding for this research was provided by the ScrumAlliance.

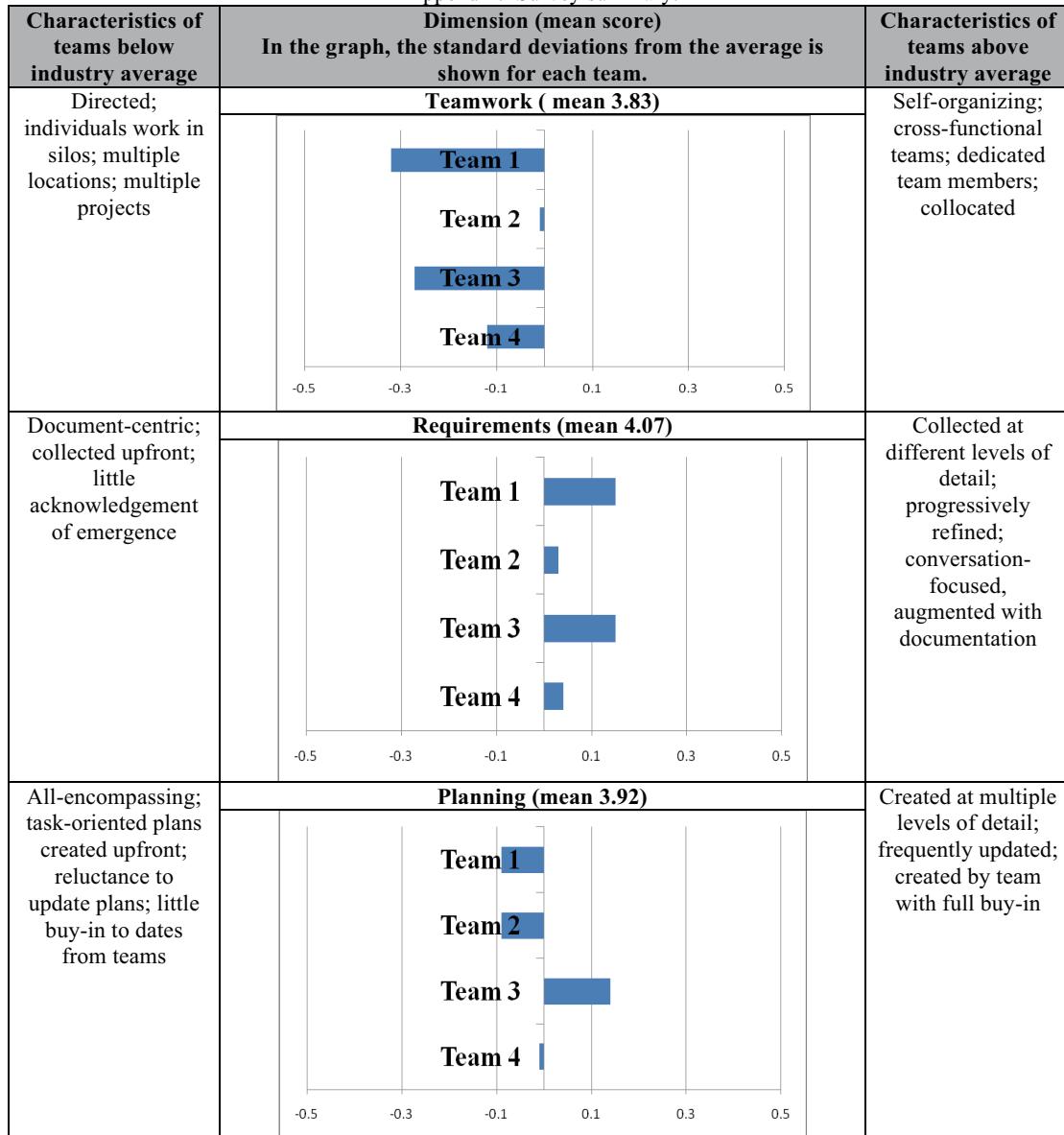
REFERENCES

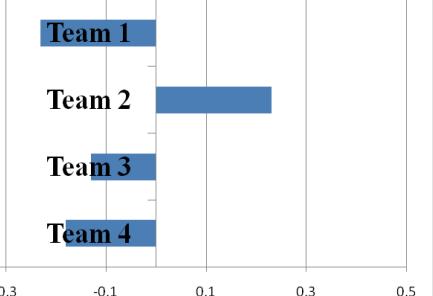
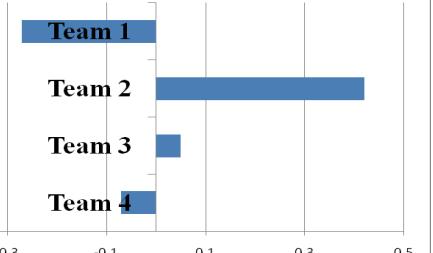
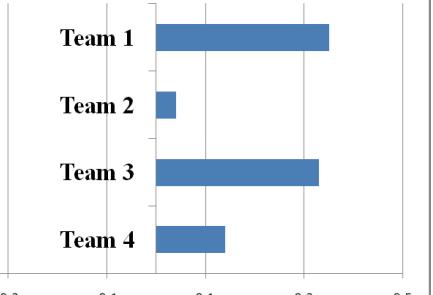
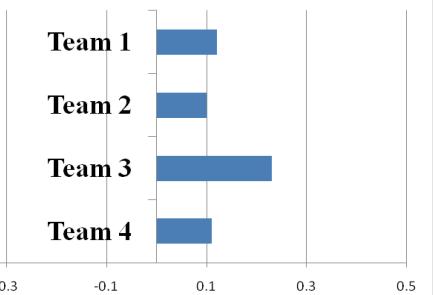
- [1] K. Beck, *Test Driven Development -- by Example*. Boston: Addison Wesley, 2003.
- [2] K. Beck, *Extreme Programming Explained: Embrace Change*, Second ed. Reading, MA: Addison-Wesley, 2005.
- [3] A. Begel and N. Nagappan, "Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study," in *Empirical Software Engineering and Measurement Conference (ESEM)*, Madrid, Spain, 2007, pp. 255-264.

⁴ <http://www.martinfowler.com/bliki/FlaccidScrum.html>

- [4] B. Boehm and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods," *IEEE Computer*, vol. 36, pp. 57-66, June 2003.
- [5] M. Cohn, "Determining How Agile You Are Comparatively," January 12, 2010, <http://www.agilejournal.com/articles/columns/column-articles/2588>.
- [6] M. Cohn, "Succeeding with Agile: Software Development Using Scrum," Massachusetts: Addison Wesley, 2010.
- [7] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*. Reading, Massachusetts: Addison Wesley, 1999.
- [8] IEEE, "IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology," 1990.
- [9] W. Krebs, "Turning the Knobs: A Coaching Pattern for XP Through Agile Metrics," in *Extreme Programming/Agile Universe*, Chicago, IL, 2002.
- [10] L. Layman, L. Williams, and L. Cunningham, "Exploring Extreme Programming in Context: An Industrial Case Study," in *Agile Development Conference*, Salt Lake City, UT, 2004, pp. 32-41.
- [11] L. Layman, L. Williams, D. Damian, and H. Buresc, "Essential Communication Practices for Extreme Programming in a Global Software Development Team" *Information and Software Technology (IST)*, vol. 48, pp. 781-794, 2005.
- [12] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*. Upper Saddle River, NJ: Prentice-Hall, 2002.
- [13] Version One, "Second Annual Survey 2007 The State of Agile Development," 2007, http://www.versionone.com/pdf/StateOfAgileDevelopment2_FullDataReport.pdf.
- [14] Version One, "Third Annual Survey 2008 The State of Agile Development," 2008, http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf.
- [15] L. Williams and R. Kessler, *Pair Programming Illuminated*. Reading, Massachusetts: Addison Wesley, 2003.
- [16] L. Williams, L. Layman, and W. Krebs, "Extreme Programming Evaluation Framework for Object-Oriented Languages -- Version 1.4," North Carolina State University, Raleigh, NC Computer Science TR-2004-18 <http://www.csc.ncsu.edu/research/tech/reports.php>, 2004.

Appendix: Survey summary.



Characteristics of teams below industry average	Dimension (mean score) In the graph, the standard deviations from the average is shown for each team.	Characteristics of teams above industry average										
Code written by programmers working alone; little emphasis on testing; code becomes harder to maintain over time; infrequent integration and system builds	<p>Technical Practices (mean 3.18)</p>  <table border="1"> <caption>Data for Technical Practices (mean 3.18)</caption> <thead> <tr> <th>Team</th> <th>Mean Score</th> </tr> </thead> <tbody> <tr> <td>Team 1</td> <td>-0.1</td> </tr> <tr> <td>Team 2</td> <td>0.2</td> </tr> <tr> <td>Team 3</td> <td>-0.1</td> </tr> <tr> <td>Team 4</td> <td>-0.1</td> </tr> </tbody> </table>	Team	Mean Score	Team 1	-0.1	Team 2	0.2	Team 3	-0.1	Team 4	-0.1	Code written in pairs using test-driven development; code not allowed to
Team	Mean Score											
Team 1	-0.1											
Team 2	0.2											
Team 3	-0.1											
Team 4	-0.1											
Quality is tested in after development; little emphasis on or effective use of automation	<p>Quality (mean 3.12)</p>  <table border="1"> <caption>Data for Quality (mean 3.12)</caption> <thead> <tr> <th>Team</th> <th>Mean Score</th> </tr> </thead> <tbody> <tr> <td>Team 1</td> <td>-0.1</td> </tr> <tr> <td>Team 2</td> <td>0.4</td> </tr> <tr> <td>Team 3</td> <td>-0.05</td> </tr> <tr> <td>Team 4</td> <td>-0.1</td> </tr> </tbody> </table>	Team	Mean Score	Team 1	-0.1	Team 2	0.4	Team 3	-0.05	Team 4	-0.1	Quality is built into the product during each iteration; automated unit and acceptance tests
Team	Mean Score											
Team 1	-0.1											
Team 2	0.4											
Team 3	-0.05											
Team 4	-0.1											
Satisfied with status quo; meets deadlines through heroic effort; command-and-control	<p>Culture (mean 3.91)</p>  <table border="1"> <caption>Data for Culture (mean 3.91)</caption> <thead> <tr> <th>Team</th> <th>Mean Score</th> </tr> </thead> <tbody> <tr> <td>Team 1</td> <td>0.35</td> </tr> <tr> <td>Team 2</td> <td>-0.05</td> </tr> <tr> <td>Team 3</td> <td>0.35</td> </tr> <tr> <td>Team 4</td> <td>0.1</td> </tr> </tbody> </table>	Team	Mean Score	Team 1	0.35	Team 2	-0.05	Team 3	0.35	Team 4	0.1	Trusting; collaborative, and adaptive
Team	Mean Score											
Team 1	0.35											
Team 2	-0.05											
Team 3	0.35											
Team 4	0.1											
Infrequent or ineffective reflection and team interaction; inconsistent use of iterations	<p>Knowledge Creating (mean 4.08)</p>  <table border="1"> <caption>Data for Knowledge Creating (mean 4.08)</caption> <thead> <tr> <th>Team</th> <th>Mean Score</th> </tr> </thead> <tbody> <tr> <td>Team 1</td> <td>0.1</td> </tr> <tr> <td>Team 2</td> <td>0.1</td> </tr> <tr> <td>Team 3</td> <td>0.25</td> </tr> <tr> <td>Team 4</td> <td>0.1</td> </tr> </tbody> </table>	Team	Mean Score	Team 1	0.1	Team 2	0.1	Team 3	0.25	Team 4	0.1	All work performed in strictly adhered-to iterations; frequent reflections; focus on team learning
Team	Mean Score											
Team 1	0.1											
Team 2	0.1											
Team 3	0.25											
Team 4	0.1											

Influence in Software Ecosystem and Software Communities

Jingchang Chen

What is a software ecosystem (SECO)?

An evolution from software product line. (Bosch, 2009)

What is a software ecosystem?

“A set of businesses functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently under-pinned by a common technological platform or market and operate through the exchange of information, resources and artifacts” (Jansen et al., 2009)

“A software ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions” (Bosch, 2009)

A set of connected services based on a common technology foundation in a shared market.

Real-world examples

Microsoft Windows

Apple iOS, MacOS

Linux

Google Chrome

...

VS Code

React

USB Standard

Why SECO is popular?

Benefits (Barbosa & Alves, 2011):

- Reduce the (individual) development cost
- Welcome new players
- Accelerate the evolution
- Create new markets...

Especially good for small to medium enterprises,
which is the majority in the industry (Fischer et al.,
2000)

Challenge:

- Requirement engineering process - Influence matters! (Schaarschmidt et al., 2015; Linåker et al., 2019; Damian et al., 2021)

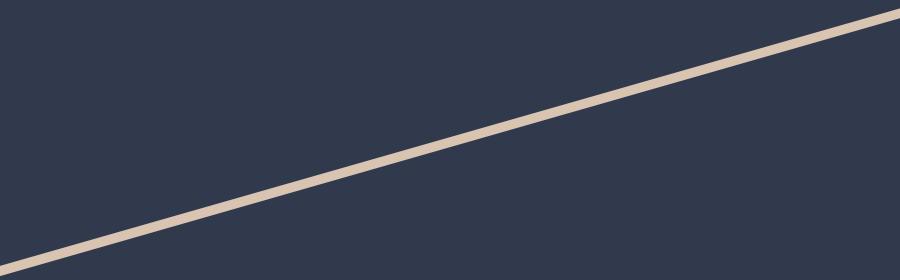
What brings influence?

There is a tie between contribution and influence,
especially in open-source software communities.

- Lines of code
- Number of pull requests

Overall, changes to the source code

Methodology, Data Source, Tools



Methodology

De

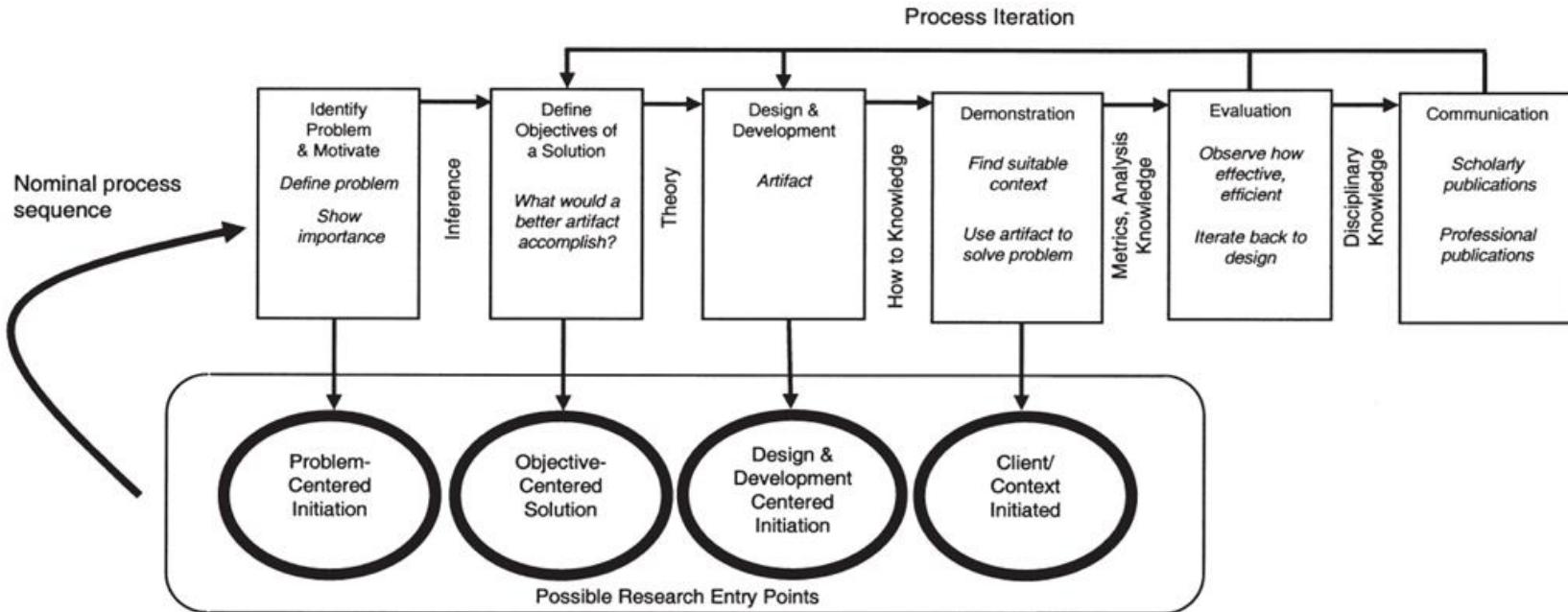


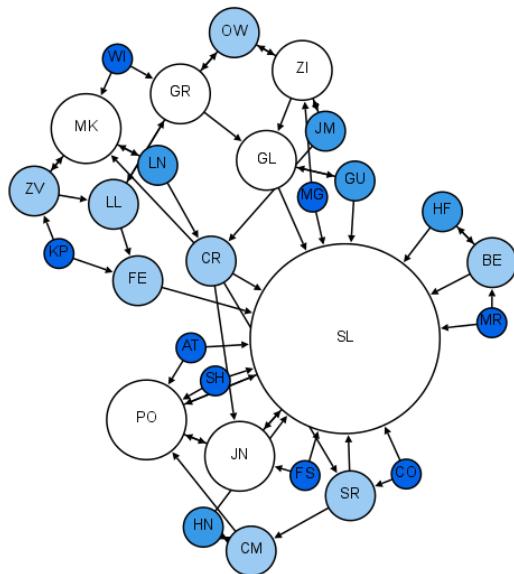
Figure 1. DSRM Process Model

Data Source

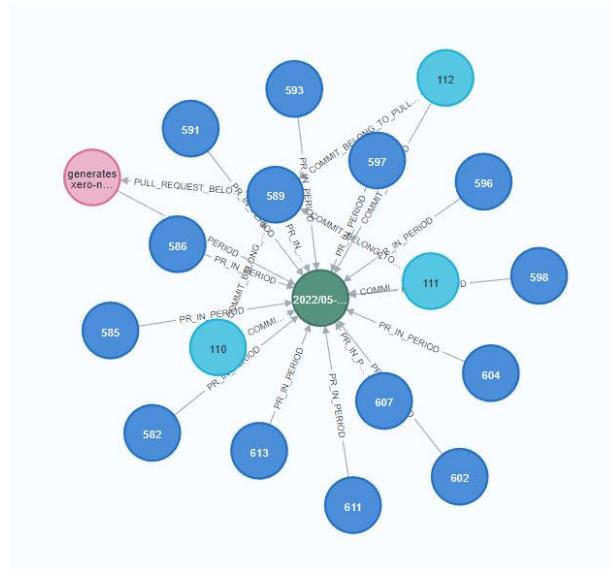
- Private
- Open-source
 - GitHub
 - GitTorrent (Gousios & Spinellis, 2012)
 - GArchive

Tools

- Social Network Analysis
 - A tool to study relationships
- Neo4j
 - A graphic relational database



** A social network. Cited from Wikipedia

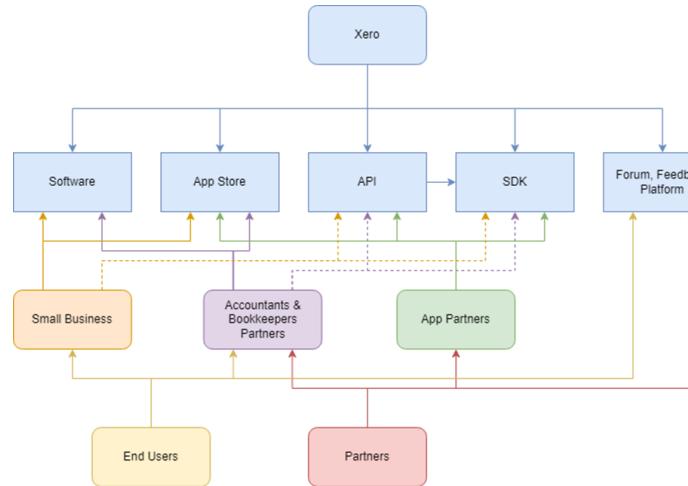


My study



Xero

1. Breaking down its structure



2. Identifying available public data sources

- Official website.
- Developer forum (deprecated).
- User forum.
- Xero API customer feedback platform.
- Application store.
- Official documents.
- GitHub repositories.

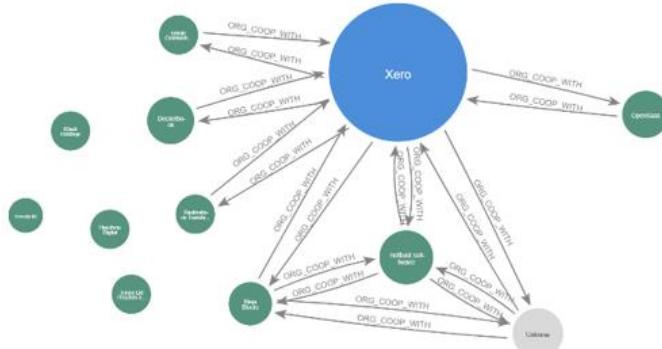
Xero

3. Mining GitHub

- GitHub RESTful API

4. Building networks

- Node: Individual/Organisation
- Edge: Both commit in one patch (pull request)
- Aggregated by time period (versions/tags)



Xero

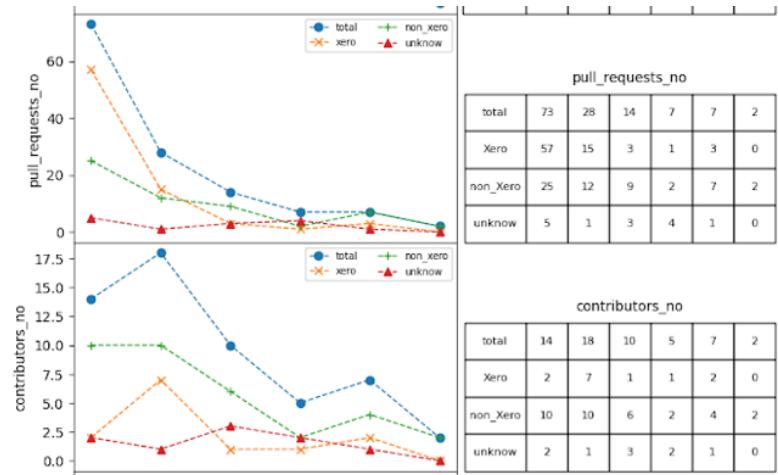
Results:

1. A large number of outsourcers are hired

Period	Orchestrator	Niche Player	End User	External Developer	Outsource	Unclear
2017/05-2017/10	1	1	2	0	6	1
2017/11-2018/04	1	1	0	2	4	3
2018/05-2018/10	1	0	1	0	1	3
2018/11-2019/04	1	0	0	0	2	0
2019/05-2019/10	1	0	1	1	1	1
2019/11-2020/04	0	0	1	0	1	0
Total	1	2	4	3	12	8

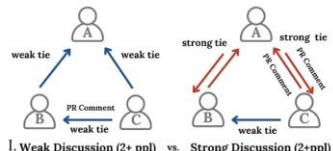
1. Most contributors are short-term contributors, focusing on the issue they encountered

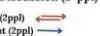
3. In an SDK project, the amount of contribution and contributors decreases when the product becomes more mature



React

1. Mining React GitHub repo
 - o Use GArchive and GitTorrent to recover the missing pieces
 2. Building social network
 - o Node: Individual
 - o Edge: Comment in the same pull request
 - o Aggregated by time period (versions/tags)
 - o Strong tie VS weak tie
3. Finding cliques
 - o Bron-Kerbosch algorithm
 4. Labeling comments
 - o Manually label (code) a part of them
 - o Train machine learning model to label the rest



Strong Tie = Reciprocated PR Comment (2ppl) 
Weak Tie = Non-Reciprocated PR Comment (2ppl) 



Raw Quote Example	Theme	Relevant Codes
<i>"You guys told me to do the sketchy hack instead of always reflecting the correct value. I don't like the solution and sebastian advocated for, but I needed to choose my battles in the interest of moving this diff forward."</i>	Conflict	Disagreement, Dissapointment, Doubt, Highlighting Issues, Correction, Contrasting Needs or Perspective
<i>"I've opened #2273 to track the status of 'setInherHTML' vs 'createNodesFromMarkup'"</i>	Project Workflow	Process, Making a Plan, Labelling, Taking Responsibility, Assignment, Ending Discussions, Redirection
<i>"@jsfb you made this change. What did you mean?"</i>	Interdependence	Relying on Social Connections, Accountability, Accepting Work, Reference, Demonstration
<i>"I know you are just following what's already here, but I think we should consolidate these constants."</i>	Improvement	Giving Advice, Giving Clarification, Agreement, Refinement, Making Suggestions, Gratitude
<i>"Would you mind explaining what this is good for?"</i> <i>"raised_hands: woohoo!"</i>	Inquiry	Seeking Advice, Seeking Clarification, Asking for Help
	Miscellaneous	-

React

Results:

- Conflicts exist in discussions in higher diverse teams

(Diversity Blau Index Score)	Ethnic	Nationality	Gender
Conflict Present-(23/39)	0.096	0.515	0.152
Conflict Absent-(16/39)	0.041	0.486	0.145

- These conflicts are usually beneficial (productive conflicts) (Schulz-Hardt et al., 2002), leading to more merged pull requests

Collaborative Groups	Conflict	No Conflict
Average Merged Pull Requests / Group	172.00	53.00
Average Merged Pull Requests / Group / Version	20.31	10.75

- The teams with conflicts also have more interdependence and inquiry comments in discussions, which shows a higher level of psychological safety

	Interdependence	Inquiry	p value
Groups (Conflict Present)	16	971	0.000
Groups (Conflict Absent)	0	194	0.000

- These teams also last longer, compared with groups with lower diversity and low/no conflicts

Thanks for watching!

References

- J. Bosch, "From software product lines to software ecosystems," in *Proceedings of the 13th International Software Product Line Conference*, 2009, vol. 9, pp. 111-119.
- S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in *2009 31st International Conference on Software Engineering - Companion Volume*, 16-24 May 2009 2009, pp. 187-190, doi: 10.1109/ICSE-COMPANION.2009.5070978.
- O. Barbosa and C. Alves, "A systematic mapping study on software ecosystems," in *Proceedings of the Third International Workshop on Software Ecosystems*, 2011, pp. 15-26.
- E. Fischer, E. Fisher, and R. Reuber, Industrial clusters and SME promotion in developing countries (no. 3). *Commonwealth Secretariat*, 2000.
- M. Schaarschmidt, G. Walsh, and H. F. O. von Kortzfleisch, "How do firms influence open source software communities? A framework and empirical analysis of different governance modes," *Information and Organization*, vol. 25, no. 2, pp. 99-114, 2015/04/01/ 2015, doi: <https://doi.org/10.1016/j.infoandorg.2015.03.001>.
- J. Linåker, B. Regnell, and D. Damian, "A Community Strategy Framework—How to obtain influence on requirements in meritocratic open source software communities?," *Information and Software Technology*, vol. 112, pp. 102-114, 2019.
- D. Damian, J. Linåker, D. Johnson, T. Clear, and K. Blincoe, "Challenges and Strategies for Managing Requirements Selection in Software Ecosystems," *IEEE Software*, vol. 38, no. 6, pp. 76-87, 2021.
- K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45-77, 2007.
- A. Marin and B. Wellman, "Social network analysis: An introduction," *The SAGE handbook of social network analysis*, pp. 11-25, 2011.
- S. Schulz-Hardt, M. Jochims, and D. Frey, "Productive conflict in group decision making: Genuine and contrived dissent as strategies to counteract biased information seeking," *Organizational Behavior and Human Decision Processes*, vol. 88, no. 2, pp. 563–586, 2002.

AUT

Week 6 ex. COMP 501 – Smart Cities



Course lecturer –
Assoc. Prof. Tony Clear



Lecture Outline

- Smart Cities as a Technology/Topic
- Opportunities
- Risks
- Choices
- Ethical Issues



Smart Cities as a Technology/Topic

Smart Cities Promise

- Poorly Defined? [Baecker, 2019]
 - Goal of reducing energy consumption in a city
 - Marketing slogan to advance a city's reputation
- *Among the many definitions that exist of smart cities, Caragliu et al. [5]. Consider that “a city is smart when investments in human and social capital and traditional and modern (ICT) communication infrastructure fuel sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance”.* [Perles-Ribes & Ivars-Baidal, 2021].

Smart Cities Infrastructure

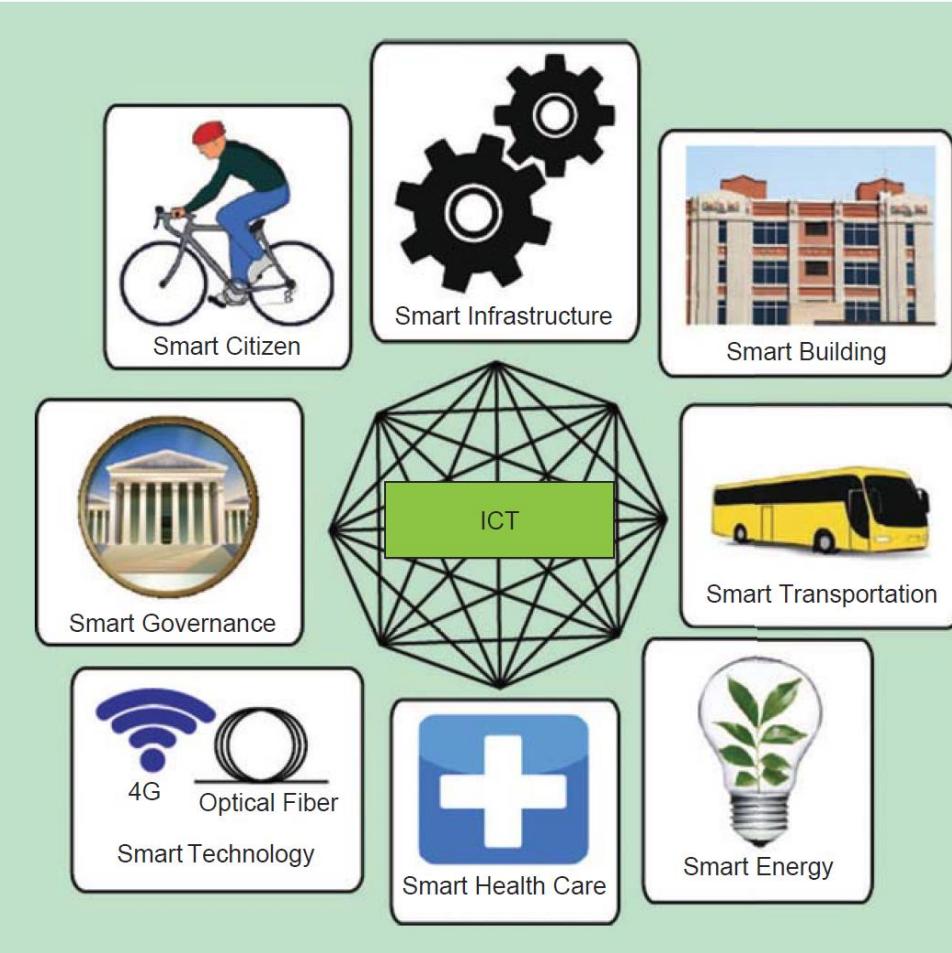


FIGURE 1. A broad overview of smart city components.

Smart cities are greener, safer, faster, and friendlier.

The different components of a smart city include *smart infrastructure, smart transportation, smart energy, smart health care, and smart technology*.

These components are what make the cities smart and efficient.

Information and communication technology (ICT) are enabling keys for transforming traditional cities into smart cities. Two closely related emerging technology frameworks, the **Internet of Things (IoT)** and **big data (BD)**, make smart cities efficient and responsive. [Mohanty et al., 2016]



Surveillance Technology

- A “smart sensor” is a generic term for a device that can take inputs from the physical environment, collect data, and share it with other similarly connected devices. For example, a smart pollution sensor might sample water or air quality from the physical environment and convert it to a readable score, and then share that information with a collecting sensor. Smart sensors tend to be small, low cost, wireless, energy efficient, and can be combined with other devices and networked together.
- Sensors are now embedded in the infrastructure of American cities. Smart roads, smart streetlights, smart homes, and smart electrical grids offer entirely new means of monitoring citizens living in “smart cities.” This municipal data collection involves state actors surveilling citizens, literally tagging them, touching them, and tracking them—all the while aggregating personal data for government purposes over long periods of time. [Andrew Ferguson, 2020]

Smart City Concepts and Technologies

- ABSTRACT
- “Smart cities” is a hot topic in debates about urban policy and practice across the globe. There is, however, limited knowledge and understanding about trending smart city concepts and technologies; relationships between popular smart city concepts and technologies; policies that influence the perception and utilization of smart city concepts and technologies.
- The aim of this study is to evaluate how smart city concepts and technologies are perceived and utilized in cities.
- The methodology involves a social media analysis approach—i.e., systematic geo-Twitter analysis— that contains descriptive, content, policy, and spatial analyses.
- For the empirical investigation, the Australian context is selected as the testbed.

Smart City Concepts and Technologies (cont.)

- The results reveal that:
- (a) innovation, sustainability, and governance are the most popular smart city concepts;
- (b) internet-of-things, artificial intelligence, and autonomous vehicle technology are the most popular technologies;
- (c) a balanced view exists on the importance of both smart city concepts and technologies;
- (d) Sydney, Melbourne, and Brisbane are the leading Australian smart cities;
- (e) systematic geo-Twitter analysis is a useful methodological approach for investigating perceptions and utilization of smart city concepts and technologies.
- The findings provide a clear snapshot of community perceptions on smart city concepts and technologies, and can inform smart city policymaking”.

Trending Smart City Concepts and Technologies

- Of the 3,073 usable tweets, 1,179 (38 percent) were original, and 1,894 (62 percent) were retweeted, reflecting the highly interactive nature of users.
- All Twitter discussions developed in total 28 hashtags.
- The hashtag analysis identified (excluding #smartcities and #smartcity) 16 key hashtags among them as the most strongly associated ones with the smart city domain.
- These were:
- #autonomousvehicle; #transport;
- #5G; #sustainability; #mobility;
- #internet-of-things; #energy;
- #innovation;
- #governance;
- #artificialintelligence; #blockchain; #bigdata;
- #robotics; #opendata;
- #waste; #startups.

Tan Yigitcanlar, Nayomi Kankamamge & Karen Vella (2021) How Are Smart City Concepts and Technologies Perceived and Utilized? A Systematic Geo-Twitter Analysis of Smart Cities in Australia, Journal of Urban Technology, 28:1-2, 135-154, DOI: 10.1080/10630732.2020.1753483

What Are the Official Smart City Policies That Influence Perception and Utilization of Smart City Concepts and Technologies?

- Transport-related policies are the most prominent.
- Energy-related policies of Australia are concerned about balancing energy supply and energy demand reduction through smart energy use
- Economy-related policies received considerably less attention across Australia, even though the economy has weakened in recent years.
- Governance-related policies are gaining momentum.

Tan Yigitcanlar, Nayomi Kankamge & Karen Vella
(2021) How Are Smart City Concepts and
Technologies Perceived and Utilized? A Systematic
Geo-Twitter Analysis of Smart Cities in Australia,
Journal of Urban Technology, 28:1-2, 135-154, DOI:
[10.1080/10630732.2020.1753483](https://doi.org/10.1080/10630732.2020.1753483)



Opportunities with Smart Cities

Taking Advantage of Citizen Data

- *Imagine what the smart sensor-enabled city of the future can do. It can monitor where citizens walk, drive, live, play, eat, what medical services they need, what they buy, what they like, who they visit and associate with, and who they love. The city becomes the platform for data collection.*
- *The data is potentially available at a granular level to track individuals, at an associational level to map networks of contacts, and at a pattern level to monitor the number of people involved in any activity. This “sensorveillance” data is tied to geography, time, and date, and can be visualized across days, weeks, or years. [Andrew Ferguson, 2020]*

Taking Advantage of Commercial Data

- *It was Google that recognized the gold dust in the detritus of its interactions with its users and took the trouble to collect it up....Google exploits information that is a by-product of user interactions, or data exhaust, which is automatically recycled to improve the service or create an entirely new product.*
- *What had been regarded as waste material—“**data exhaust**” spewed into Google’s servers during the combustive action of Search—was quickly reimagined as a critical element in the transformation of Google’s search engine into a reflexive process of continuous learning and improvement.*
- *The summary of these developments is that the behavioral surplus upon which Google’s fortune rests can be considered as surveillance assets. These assets are **critical raw materials** in the pursuit of surveillance revenues and their translation into surveillance capital. The entire logic of this capital accumulation is most accurately understood as surveillance capitalism, which is the foundational framework for a surveillance-based economic order: a surveillance economy.[Zuboff, 2019]*



Converting Our Lives to Data

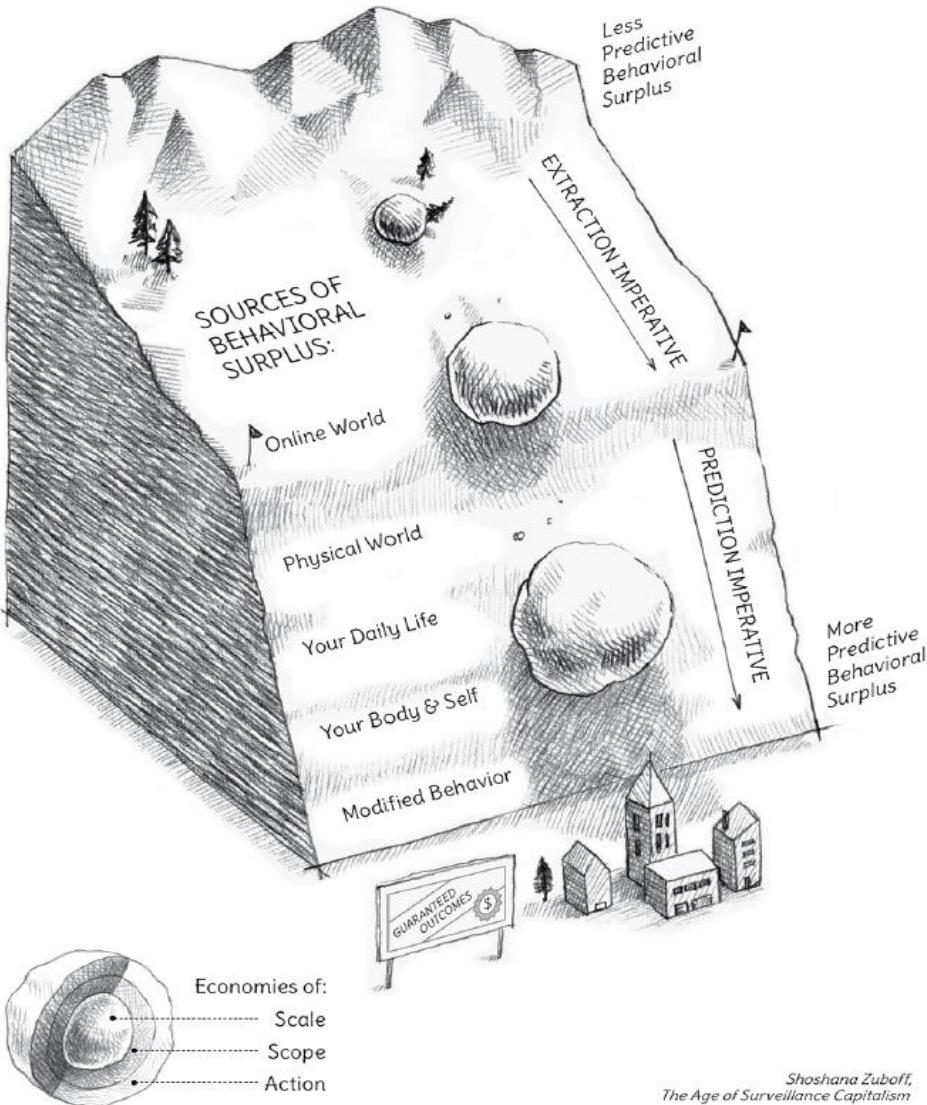
- *Industrial capitalism transformed nature's raw materials into commodities, and surveillance capitalism lays its claims to the stuff of human nature for a new commodity invention. Now it is human nature that is scraped, torn, and taken for another century's market project.*
- *It is obscene to suppose that this harm can be reduced to the obvious fact that users receive no fee for the raw material they supply. That critique is a feat of misdirection that would use a pricing mechanism to institutionalize and therefore legitimate the extraction of human behavior for manufacturing and sale. It ignores the key point that the essence of the exploitation here is the rendering of our lives as behavioral data for the sake of others' improved control of us.* [Zuboff, 2019]

Opportunities for Platform Vendors

- Surveillance Capitalism as a business model?
- But must it be so?
- As Tom Berners-Lee has reflected:
 - “But then you should step back. It’s making a good case about **how one particular wave of social networks can work, if you train the AI to maximise the engagement of the teenager.**
 - **But if you train the AI to maximise the happiness of the teenager, or the efficiency of the teenager – well, then the whole thing would produce a very different outcome.** You use the same software – you just turn the dial.
 - **You could imagine two social networks where most of the code is mostly the same – it’s just that one is optimised for one thing, and the other is optimised for another. And the unintended consequences in each case are completely different.”**
 - <https://www.theguardian.com/lifeandstyle/2021/mar/15/tim-berners-lee-we-need-social-networks-where-bad-things-happen-less>

The Dynamic of Behavioral Surplus Accumulation

Surveillance capitalism's master motion is the accumulation of new sources of behavioral surplus with more predictive power. The goal is predictions comparable to guaranteed outcomes in real-life behavior. Extraction begins online, but the prediction imperative increases the momentum, driving extraction toward new sources in the real world.



Technological Sovereignty and Citizen Control

- The City of Barcelona stands out internationally as a smart city that aims to harness technology to empower citizens. This has been led by Francesca Bria, the Chief Technology and Digital Innovation Officer for the City of Barcelona.
- Bria advocates for technological sovereignty and the use of technology in order to increase citizens' capacity (as both individuals and collectives) to design the city's technological and nontechnological infrastructure to inform the ends that it serves (see Morozov & Bria, 2018, p. 22).
- The Barcelona Digital City Plan (Ajuntament de Barcelona, 2019) aims to transcend mere technological objectives, to rethink a smart city that serves its citizens, by grounding technology “at the service of people and not people at the service of technology” (p. 6).
- The main public policy actions outlined in the plan encompass establishing Barcelona as a global city of commons and collaboration production, by ending privatization and promoting remunicipalization of critical urban infrastructure. This will build data-driven models of the economy, including a city data commons, and promote collective collaboration above centralized state and market solutions. [Mann et al., 2021]



Technology in Service of Democracy

- The goal is to “use digitisation to benefit all citizens and transition towards a more sustainable, democratic, equitable and circular city” that is “committed to innovation and sees the city as an urban platform for establishing connection... to contribute to solving the city's pressing social and environmental problems” (Ajuntament de Barcelona, 2019, p. 33).
- Technological sovereignty and data sovereignty are at the heart of the Barcelona Digital City Plan (2015–2019) and the Digital Barcelona Plan (2017–2020; Bria, 2016).
- The initiatives are described as providing the “capacity to decide” (Galdon, 2017), via governance frameworks and technological solutions that promote citizen capacity to know what is happening, and control the flow of their personal information. This involves discerning and directing their personal data, as per Leszczynski's anxieties of control.
- According to the Deputy Mayor of Barcelona Gerard Pisarell, “In a democratic city, technology should serve to digitally empower citizens, to protect their privacy from abuses by the public and private powers. [...] That has a name: conquering technological, digital sovereignty, for the common good” (as quoted in Galdon, 2017, para. 4, March & Ribera-Fumaz, 2018, p. 235). [Mann et al., 2021]



Risks with Smart Cities

Smart Cities – Law and Surveillance as “Search”

- *The Fourth Amendment provides that [t]he right of the people to be secure in their persons, houses, papers, and effects, against unreasonable searches and seizures, shall not be violated, and no Warrants shall issue, but upon probable cause, supported by Oath or affirmation, and particularly describing the place to be searched, and the persons or things to be seized.*
- *...under current Supreme Court doctrine, automated, continuous, aggregated, long-term acquisition of personal data by smart sensors triggers Fourth Amendment scrutiny and thus could violate the Constitution.*
- *Building a smart city without a legal layer is a design flaw that not only will raise Fourth Amendment concerns (if the arguments in this paper are convincing), but will raise a whole host of avoidable problems. A legal layer will be the only way to proactively respond to criticisms that a smart city is really just a city of surveillance or data capitalism or both.* [Andrew Ferguson, 2020]

Smart Cities – Law and Surveillance as “Search” [2]

- In his book, *The Fourth Amendment in an Age of Surveillance*, Professor Gray undertakes to build out some of this Fourth Amendment scaffolding. He suggests designing legal constraints around the eight operational stages of big data collection:
- (1) deployment; (2) data gathering; (3) data aggregation; (4) data storage; (5) data access; (6) data analytics; (7) accessing the results of data analytics; and (8) uses of data analytics.
- In his book, Gray applies each stage to a series of technologies, including many which will make an appearance in a smart city.
- The point is that this type of structured legal and technological analysis at each moment of data collection can be accomplished.
- In the same way computer science engineers must make decisions about possible outcomes based on possible inputs, challenges, and changes, so must the lawyers.
- Step by step, collection point by collection point, the moments of data collection can be regulated at the outset. [Andrew Ferguson, 2020]



Vendor and Social Risks - Do Users Want Privacy if it is an option?

- App Tracking Transparency —
- 96% of US users opt out of app tracking in iOS 14.5, analytics find
- <https://arstechnica.com/gadgets/2021/05/96-of-us-users-opt-out-of-app-tracking-in-ios-14-5-analytics-find/>



Regulatory Risks?

- Eric Schmidt, former CEO of Google, embodies this self-regulatory ethos regarding the role of governments regulating technology in the smart city future:
- “The problem is [that] if you write a rule, inevitably, you fix the solution on a specific [technology], but the technology moves so quickly ... **It's generally better to let the tech companies do these things**” (quoted in Zahn and Serwer 2019).
- This statement underscores the view that technology vendors behind the smart city initiatives should be allowed to self-regulate as they move to provide the next phase of democracy through coded engagement. [Tenney, 2021]



Choices with Smart Cities



Smart City of Toronto?

- Our primary argument is that the City of Toronto's efforts to transform Toronto into a world-renowned smart city is one of **conflicting visions of smart city governance**.
- The **first of these visions** positions the use of big data and smart technology as a form of participatory bread to feed corporate-run smartautomata, where troubles of civic engagement are solved through technological solutions.
- In this vision, the smart city becomes an incubator for billion-dollar technology development, while sacrificing the rights and privacy of its citizenry.
- The **alternative vision** of smart city governance is one that is strategically developed through participatory mechanisms endorsed by the city's citizens and ideally led by their interests and needs.
- In this model of smart city governance, the focus is on when, why, and how the same smart city technologies come to be implemented. [Tenney, 2021]



Smart City of Toronto - Governance Models?

- Fundamentally, governance at the municipal level should include a variety of mechanisms for engaging citizens in the decision-making process and provide a means of making municipal operations transparent and accountable (Scassa 2018).
- Currently, most smart city governance (SCG) models are primarily focused on setting limits on smart city technology and routinely lack a mechanism to provide an oversight role to the citizen, through formal engagement processes, which would engage them in deciding which technologies to procure and deploy and for what purposes (Lauriault et al. 2018b).
- Instead, many SCG models predominantly revolve around providing private-industry public resources under the gratis model (e.g., free-to-use resources like data, infrastructure, or experimental space), that allows them to pilot, develop, and further solicit their products with lower financial risk.[Tenney, 2021]

Smart City of Toronto vs Open Smart City?

- Conversely, an open smart city is one in which there is collaboration by a range of stakeholders (residents, private sector, civil society, and academics) in the ethical, accountable, and transparent mobilization of technologies and data for an equitable and balanced governance of the city (Lauriault et al. 2018a).
- The SCG model that incorporates such citizen-centric processes in the design and regulation of the smart city prior to the implementation, instead of the promise to be included in an automata-type version of public engagement after the fact, seems befitting of the notion of an open smart city and the libre model of SCG (e.g., free to access resources). [Tenney, 2021]

Smart City of Toronto - IoTES Pilot?

- In 2018, several City divisions started a pilot project with the aim of approaching the traditional IT procurement process from a different angle by using open source IoT environmental sensors (IoTES).
- The goal of the IoTES pilot was to observe the devices during a four-month proof-of-concept study to gauge the effectiveness of the sensors across different environments (Coop 2018).
- Compared to other IoT sensor arrays at the City, IoTES devices stood in stark contrast in both how they are designed and implemented, and how they were created to encourage public engagement (City Staff).
- A key design aspect of the IoTES was to allow citizens, organizations, and other municipalities to utilize the architecture. They were built so others could repurpose both their architecture and the data collected.
- **Designing them in this way not only leveraged the one-time investment for a range of others to benefit, but it also reduced the overall cost by being a repeatable design, with no additional licensing fees for additional units (City Staff).**
- The City staff behind the design and operation of these devices publicly post their design schematics and source code, and have created a public repository to enable others to build their own devices, adding an opportunity to crowdsource the effort of data collection.
- Furthermore, since the devices will be placed in public spaces (e.g., street trees, public furniture), the IoTES devices did not aim to blend seamlessly into the background and go unnoticed by the passing citizen.
- Each device was to be adorned by a scannable QR code and a message explaining the project and its purpose, providing direction to educational materials for those who choose to inquire.
- While it is unclear what type of interest the initiative will garner among the public, **there was a conscious and deliberate effort to provide the option for engagement with the public** (City Staff). [Tenney, 2021]

Smart City of Toronto – A Different Vision?

- “The initial plans were definitely scary for a lot of people ... it felt like [Quayside] was going to become a technology-based gated community, where everything was monitored behaviour, personal data is collected by thousands of cameras,” said Jasmine Mohamed, a Toronto resident completing her graduate degree in urban planning. “I’m glad that’s no longer a priority for the new vision.”
- **Instead, the new plan centres on affordability, low-carbon design and an emphasis on local and minority-owned businesses.**
- While it retains a number of elements from the initial glossy renderings – such as wooden skyscrapers – **the new plan focuses more on integrating elements of Lake Ontario into parks and recreational amenities.**
- For a city already grappling with a housing shortage and affordability crisis, the plans need to be ambitious if Toronto hopes to achieve any degree of equity, warns Mohamed.
- “The pandemic has brought disparities in the city to the forefront. It’s hit low-income areas so hard. And so this it’s an opportunity for the city to be bold”.

Technology Design Responses

- *Gray's recommendations align with those earlier espoused by Ann Cavoukian...*
- *Privacy by Design* advances the view that the future of privacy cannot be assured solely by compliance with regulatory frameworks; rather, privacy assurance must ideally become an organization's default mode of operation.
- *Privacy by Design* extends to a “Trilogy” of encompassing applications: 1) IT systems; 2) accountable business practices; and 3) physical design and networked infrastructure.
- Principles of *Privacy by Design* may be applied to all types of personal information, but should be applied with special vigour to sensitive data such as medical information and financial data. The strength of privacy measures tends to be commensurate with the sensitivity of the data.
- The objectives of *Privacy by Design* — ensuring privacy and gaining personal control over one's information and, for organizations, gaining a sustainable competitive advantage — may be accomplished by practicing the following 7 Foundational Principles. [Ann Cavoukian, 2011]



Privacy By Design Principles

- 1. ***Proactive*** not Reactive; ***Preventative*** not Remedial
- 2. Privacy as the ***Default Setting***
- 3. Privacy ***Embedded*** into Design
- 4. Full Functionality — ***Positive-Sum***, not Zero-Sum
- 5. End-to-End Security — ***Full Lifecycle Protection***
- 6. ***Visibility*** and ***Transparency*** — Keep it ***Open***
- 7. ***Respect*** for User Privacy — Keep it ***User-Centric***

[Cavoukian, 2010]



Ethical Issues with Smart Cities

The Need for Professional Sensitivity

- *Organizations struggle to comply with legal requirements as well as customers' calls for better data protection.*
- *On the implementation level, incorporation of privacy protections in products and services depends on the commitment of the engineers who design them.*
- *We interviewed six senior engineers, who work for globally leading IT corporations and research institutions, to investigate their motivation and ability to comply with privacy regulations.*
- *Our findings point to a lack of perceived responsibility, control, autonomy, and frustrations with interactions with the legal world.*
- *While we increasingly call on engineers to go beyond functional requirements and be responsive to human values in our increasingly technological society, we may be facing the dilemma of asking engineers to live up to a challenge they are currently not ready to embrace.*
- [Bednar et al., 2019]



The Need for Professional Sensitivity (cont.)

- *One of the two engineers who mentioned having design autonomy contradicted himself, later saying that he did not have the autonomy, or only had some. Another engineer mentioned on several occasions that he had neither the choice nor the final control.*
- *Such lack of autonomy and control is especially startling as all interviewees hold senior positions and hence should be in the position to strongly influence (if not determine) how privacy is dealt with in their teams and projects.*
- *Interestingly, while engineers pointed at lawyers in our interviews, the same fingerpointing can be observed in the legal world, which is frustrated with engineers' reluctance to embrace privacy. [Bednar et al., 2019]*

The Challenges for Computing Professionals – Embedded Agents??

- Moshe Vardi - former Editor-in-Chief of *Communications of the ACM* has just commented in ACM's flagship magazine,
- “*Under the mantra of ‘Information wants to be free’ several tech companies have turned themselves into advertising companies*”
- *and as a consequence*
- “*AI technology is used by large and powerful companies to support a business model that is, arguably, unethical*” (Vardi, 2022).
- *Also cf. related issues with the Business Models underpinning the “attention economy”*
- Clear, T. (2022). Algorithms of Anger: Lost Down the Rabbit-Hole! *ACM Inroads*, 13(2), 6-8.
<https://doi.org/10.1145/353001>



Professional Competence

- **Principle 4: Client and Employer**
- Software engineers shall, consistent with the public health, safety, and welfare, always act in professional matters as faithful agents and trustees of their client or employer. In particular, software engineers shall:
- **4.01 Provide service only in areas of their competence.**

Professional Competence?

“The idea of a smart city sounds really nice. It’s new, it’s innovative and it’s world-leading. It promises to make hard problems simpler and easier and faster to solve,” said Shoshanna Saxe, a civil engineering professor at the University of Toronto. “But when you dig into the details of these pitches, they really tend not to work very well.”

As an example, she pointed to the **“smart” rainwater monitoring system proposed by Sidewalk Labs as a way to avoid flooding.** But the idea, she says, was **premised both on the notion that weather systems that caused the worst damage could be predicted – and that the city wouldn’t lose power during a severe storm.**

Saxe highlighted the need for decision makers to take a far longer view of city infrastructure.

“We’re often easily distracted by the idea of something new and flashy, but then we learn it’s quite hard to deliver on those promises,” she said.

“And while we’re chasing something flashy, the problems become more entrenched and it becomes harder to deal with them.” [Mann et al., (2020)]

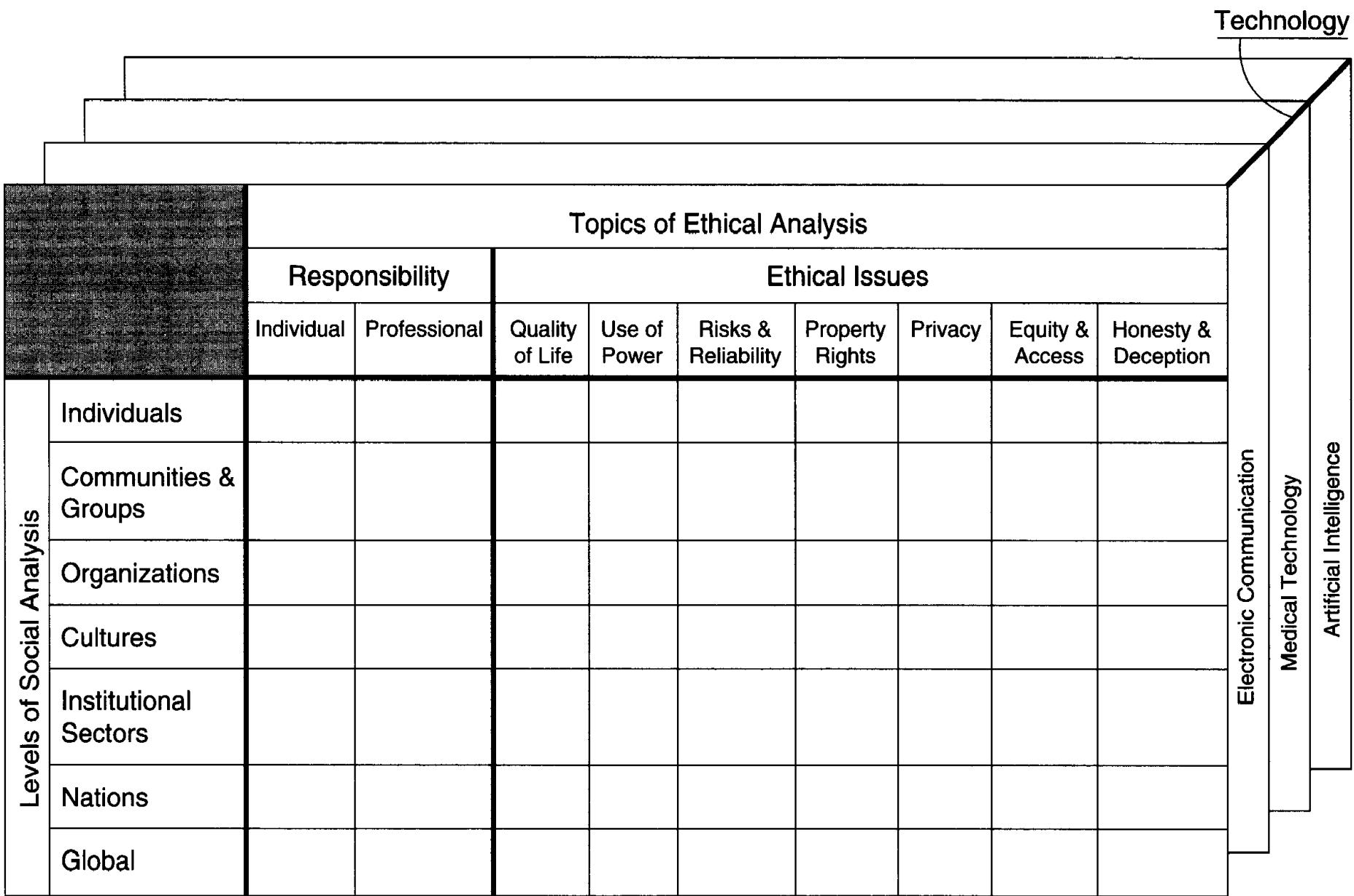


Figure 1. The three-dimensional knowledge space

Professional Codes of Ethics - ACM

- Increased interest from the public and popular media drive change. For example, ACM revised its “Code of Ethics and Professional Conduct” in 2018, for the first time since 1992. Its origins date back to 1973, when the initial code of ethics for computing professionals was developed. It is quite timely that the software engineering and computing professions understand ethical engineering principles regarding how we practice as well as how we design software systems, with or without AI. Can these two codes of conduct, the ACM/IEEE’s “Software Engineering Code of Ethics” (1997) and ACM’s Code of Ethics and Professional Conduct (2018), provide solid principles for those of us who are actively designing and developing software systems? The ethics principles outlined in these two documents are in fact clear, even for our new challenges: increased AI and data-related security, privacy, fairness, and bias issues. For example, the 2018 ACM code includes the following principles:
 - Principle 1.4, emphasizing fairness and taking action to not discriminate
 - Principle 1.6, regarding privacy expectations
 - Principle 2.9, directing the design and implementation of systems that are robustly and usably secure
 - Principle 3.7, **recognizing and taking special care of systems that become integrated into society’s infrastructure.** [Ozkaya, 2019]

Professional Codes of Ethics - SE

- One viewpoint is that developments in technology, big data, and algorithms, along with the pervasive nature of software, necessitated the changes in the 2018 ACM code. As a pleasant surprise, the 1997 software engineering code also outlines clear guidelines, particularly under its product principles. Here are a few.
- 3.12: Work to develop software and related documents **that respect the privacy** of those who will be affected by that software.
- 3.13: Be careful to **use only accurate data that was derived by ethical and lawful means, and use it only in properly authorized ways.**
- 3.14: **Maintain the integrity of data**, being sensitive to outdated or flawed occurrences. [Ozkaya, 2019]

Ethics as a Design Concern

- Similar to other engineering disciplines, in software engineering, **we should treat ethics and its related concerns as fundamental design constraints.**
- Should the practice of software engineering also consider the legal implications of ethical conduct?
- Perhaps the release of European Union's General Data Protection Regulation (GDPR) hints that software engineering may be moving in a direction similar to other engineering disciplines as well.
- GDPR suggests consequence on lack of compliance on the listed data protection rules, including legal and financial consequences of unethical conduct.
- **Embracing ethics as an explicit, nonnegotiable software design concern will be a start toward conscious progress.**
- **Treating ethics as a design concern starts with identifying key quality attributes that all systems must implement.**
- Deploying successful systems that address business and user goals within cost, resource, and expected quality constraints require tradeoffs.
- It will be necessary to change our position when it comes to concerns related to ethics; design tradeoffs we make should not compromise these concerns. **At a minimum, security, privacy, data management, transferability, and explainability concerns should be addressed for designing ethics in.** [Ozkaya, 2019]



A theatre of machines: Automata circuses and digital bread in the smart city of Toronto

- The power of technology poses opportunities and risks when it is focused on addressing urban issues and masked as civic participation.
- The model of implementation for technological solutions can determine if a city is actually smart and adheres to a democratic model of governance.
- Until the citizen is included and co-designs the smart city, outcomes will do little more than propagate uncertainty and bias in a rapidly changing industry of smart city technologies.
[Tenney et al., 2021]

SMART CITIES

Chris Hankin, Lead Author



PROBLEM

How to deploy information and communication technology (ICT) to create smart cities without compromising security, privacy, fairness, and sustainability.

<https://www.acm.org/articles/bulletins/2022/may/techbrief-on-smart-cities>

POLICY IMPLICATIONS

- *Cybersecurity risks* must be considered at every stage of every smart city technology's life cycle.
- *Effective privacy protection mechanisms* must be an essential component of any smart city technology deployed.
- Such *mechanisms should be transparently fair* to all city users, not just residents.
- The *climate impact of smart city infrastructures* must be fully understood as they are being designed and regularly assessed after they are deployed.

References

- Baecker, R. M. (2019). *Computers and society: Modern perspectives*: Oxford University Press, USA.
- Bednar, K., Spiekermann, S., & Langheinrich, M. (2019). Engineering Privacy by Design: Are engineers ready to live up to the challenge? *The Information Society*, 35(3), 122-142.
- Cavoukian, A. (2010). Privacy by design: the definitive workshop. A foreword by Ann Cavoukian, Ph. D. *Identity in the Information Society*, 3(2), 247-251.
- Ferguson, A. G. (2020). Structural Sensor Surveillance. *Iowa Law Review*, 106(1), 47-112.
- Gotterbarn, D., Bruckman, A., Flick, C., Miller, K., & Wolf, M. J. (2017). ACM code of ethics: a guide for positive action. *Communications of the ACM*, 61(1), 121-128.
- Martin, C. D., Huff, C., Gotterbarn, D., & Miller, K. (1996). A framework for implementing and teaching the social and ethical impact of computing. *Education and Information Technologies*, 7(2), 101-122.
- Mohanty, S. P., Choppali, U., & Kougianos, E. (2016). Everything you wanted to know about smart cities: The internet of things is the backbone. *IEEE Consumer Electronics Magazine*, 5(3), 60-70.
- Ozkaya, I. (2019). Ethics Is a Software Design Concern. *IEEE Software*, 36(3), 4-8.
- Perles-Ribes, J. F., & Ivars-Baidal, J. A. (2021). The Pathway from Smartness to Sustainability: Exploring the Transmission Mechanisms *Springer International Publishing*. Cham. Abstract retrieved from 10.1007/978-3-030-65785-7_39
- Zuboff, S. (2019). *The age of surveillance capitalism: The fight for a human future at the new frontier of power*: Profile Books.
- Zuboff, S. (2015). Big other: surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1), 75-89.

References (2)

- Mann, M., Mitchell, P., Foth, M., & Anastasiu, I. (2020). # BlockSidewalk to Barcelona: Technological sovereignty and the social license to operate smart cities. *Journal of the Association for Information Science and Technology*, 71(9), 1103-1115.
- Tan Yigitcanlar, Nayomi Kankamamge & Karen Vella (2021) How Are Smart City Concepts and Technologies Perceived and Utilized? A Systematic Geo-Twitter Analysis of Smart Cities in Australia, *Journal of Urban Technology*, 28:1-2, 135-154, DOI: 10.1080/10630732.2020.1753483
- Tenney, M., Garnett, R., & Wylie, B. (2020). A theatre of machines: Automata circuses and digital bread in the smart city of Toronto. *The Canadian Geographer/Le Géographe canadien*, 64(3), 388-401.
- Vardi, M. Y. (2022). ACM, ethics, and corporate behavior. *Commun. ACM*, 65(3), 5.
<https://doi.org/10.1145/3516423>
- <https://www.theguardian.com/world/2021/mar/12/toronto-canada-quayside-urban-centre>
- <https://www.theguardian.com/technology/2020/may/07/google-sidewalk-labs-toronto-smart-city-abandoned>
- <https://www.theguardian.com/cities/2019/jun/06/toronto-smart-city-google-project-privacy-concerns>
- <https://www.theguardian.com/lifeandstyle/2021/mar/15/tim-berners-lee-we-need-social-networks-where-bad-things-happen-less>

Privacy By Design Principles

- 1. **Proactive** not Reactive; **Preventative** not Remedial
- The *Privacy by Design* (PbD) approach is characterized by proactive rather than reactive measures. It anticipates and prevents privacy invasive events *before* they happen. PbD does not wait for privacy risks to materialize, nor does it offer remedies for resolving privacy infractions once they have occurred — it aims to *prevent* them from occurring. In short, *Privacy by Design* comes before-the-fact, not after.
- 2. Privacy as the **Default Setting**
- We can all be certain of one thing — the default rules! *Privacy by Design* seeks to deliver the maximum degree of privacy by ensuring that personal data are automatically protected in any given IT system or business practice. If an individual does nothing, their privacy still remains intact. No action is required on the part of the individual to protect their privacy — it is built into the system, *by default*.
- 3. Privacy **Embedded** into Design
- *Privacy by Design* is embedded into the design and architecture of IT systems and business practices. It is not bolted on as an add-on, after the fact. The result is that privacy becomes an essential component of the core functionality being delivered. Privacy is integral to the system, without diminishing functionality.
- 4. Full Functionality — **Positive-Sum**, not Zero-Sum
- *Privacy by Design* seeks to accommodate all legitimate interests and objectives in a positive-sum “win-win” manner, not through a dated, zero-sum approach, where unnecessary trade-offs are made. *Privacy by Design* avoids the pretense of false dichotomies, such as privacy **vs.** security, demonstrating that it *is* possible to have both.

Privacy By Design Principles (Cont'd)

- 5. End-to-End Security — ***Full Lifecycle Protection***
- *Privacy by Design*, having been embedded into the system prior to the first element of information being collected, extends securely throughout the entire lifecycle of the data involved — strong security measures are essential to privacy, from start to finish. This ensures that all data are securely retained, and then securely destroyed at the end of the process, in a timely fashion. Thus, *Privacy by Design* ensures cradle to grave, secure lifecycle management of information, end-to-end.
- 6. ***Visibility*** and ***Transparency*** — Keep it ***Open***
- *Privacy by Design* seeks to assure all stakeholders that whatever the business practice or technology involved, it is in fact, operating according to the stated promises and objectives, subject to independent verification. Its component parts and operations remain visible and transparent, to users and providers alike. Remember, trust but verify.
- 7. ***Respect*** for User Privacy — Keep it ***User-Centric***
- Above all, *Privacy by Design* requires architects and operators to keep the interests of the individual uppermost by offering such measures as strong privacy defaults, appropriate notice, and empowering user-friendly options. Keep it user-centric. [Ann Cavoukian, 2011]



June 27, 2023

PRINCIPLES FOR THE DEVELOPMENT, DEPLOYMENT, AND USE OF GENERATIVE AI TECHNOLOGIES*

Introduction

Generative Artificial Intelligence (AI) is a broad term used to describe computing techniques and tools that can be used to create new content, including: text, speech and audio, images and video, computer code, and other digital artifacts.¹ While such systems offer tremendous opportunities for benefits to society, they also pose very significant risks.² The increasing power of generative AI systems, the speed of their evolution, broad application, and potential to cause significant or even catastrophic harm means that great care must be taken in researching, designing, developing, deploying, and using them. Existing mechanisms and modes for avoiding such harm likely will not suffice.

* Lead authors of this document for USTPC were Ravi Jain, Jeanna Matthews, and Alejandro Saucedo. Important contributions were made by Harish Arunachalam, Brian Dean, Advait Deshpande, Simson Garfinkel, Andrew Gross, Jim Hendler, Lorraine Kisselburgh, Srivatsa Kundurthy, Marc Rotenberg, Stuart Shapiro, and Ben Shneiderman. Assistance also was provided by: Ricardo Baeza-Yates, Michel Beaudouin-Lafon, Vint Cerf, Charalampos Chelmis, Paul DeMarinis, Nicholas Diakopoulos, Janet Haven, Ravi Iyer, Carlos E. Jimenez-Gomez, Mark Pastin, Neeti Pokhriyal, Jason Schmitt, and Darryl Scriven.

¹ The first set of generative AI advances rest on very large AI models that are trained on an extremely large corpus of data. Examples that are text-oriented include BLOOM, Chinchilla, GPT-4, LaMDA, and OPT, as well as conversation oriented models like Bard, ChatGPT, and others. By definition, this is a rapidly evolving area. This list of examples, therefore, is by no means intended to be exhaustive. Similarly, the principles advanced in this document also are certain to evolve in response to changing circumstances, technological capabilities, and societal norms.

² Generative AI models and tools offer significant new opportunities for enhancing numerous online experiences and services, automating tasks normally done by humans, and assisting and enhancing human creativity. From another perspective, such models and tools also have raised significant concerns about multiple aspects of information and its use, including accuracy, disinformation, deception, data collection, ownership, attribution, accountability, transparency, bias, user control, confidentiality, privacy, and security. Generative AI also raises important questions outside the scope of this document, including many about the replacement of human labor and jobs by AI-based machines and automation.

This statement puts forward principles and recommendations for best practices in these and related areas based on a technical understanding of generative AI systems.³ The first four principles, which are specific to generative AI, address issues regarding limits of use, ownership, personal data control, and correctability. The following four principles were derived and adapted from the joint *ACM Statement on Principles for Responsible Algorithmic Systems*⁴ released in October 2022. These pertain to transparency, auditability and contestability, limiting environmental impacts, and security and privacy.⁵

This statement also reaffirms and includes five principles from the joint statement as originally formulated and has been informed by the January 2023 [ACM TechBrief: Safer Algorithmic Systems](#).

The following instrumental principles, consistent with the ACM Code of Ethics,⁶ are intended to foster fair, accurate, and beneficial decision-making concerning generative and all other AI technologies:

Generative AI-Specific Principles

- 1. Limits and guidance on deployment and use:** In consultation with all stakeholders, current law and regulation should be reviewed and applied as written or revised to limit the deployment and use of generative AI technologies when required to minimize harm. No high-risk AI system should be allowed to operate without clear and adequate safeguards, including a “human in the loop” and clear consensus among relevant stakeholders that the system’s benefits will substantially outweigh its potential negative impacts.

³ Technical considerations do not, however, exist in a vacuum. In many cases, they thus have led us to also recommend that legal, regulatory, and policy issues raised by generative AI be discussed transparently among multiple stakeholders. The goal of such efforts must be appropriately robust frameworks for oversight of these technologies grounded firmly in technical fundamentals and practice. The safe and responsible use of generative AI will be possible only with the transparent and consistent collaboration over time of all impacted stakeholders.

⁴ [Statement on Principles for Responsible Algorithmic Systems](#), ACM Technology Policy Council and its Europe and U.S. Technology Policy Committees (October 26, 2022) <https://www.acm.org/binaries/content/assets/public-policy/final-joint-ai-statement-update.pdf> (Joint Statement).

⁵ Multiple additional principles articulated in the joint statement also remain germane and are restated in the last section of this document. They concern legitimacy and competency, minimizing harms, interpretability and explainability, maintainability, and accountability and responsibility.

⁶ The ACM Code of Ethics and Professional Conduct was designed to inspire and guide the ethical conduct of all computing professionals, including current and aspiring practitioners, instructors, students, influencers, and anyone who uses computing technology in an impactful way. The Code includes principles formulated as statements of responsibility, based on the understanding that the public good is always the primary consideration. Each principle is supplemented by guidelines, which provide explanations to assist computing professionals in understanding and applying the principle. See <https://www.acm.org/code-of-ethics>.

Providers⁷ should undertake extensive impact assessments prior to the deployment of such technologies to thoughtfully ensure that the benefits to society of any such deployment outweigh its risks. One approach is to define a hierarchy of risk levels, with unacceptable risk at the highest level and minimal risk at the lowest level.⁸ Such categorizations must include the risk that users who attribute human characteristics or behavior to generative AI systems inappropriately, may be more likely to rely upon such systems' outputs and experience harm.

Providers of generative AI systems released to the general public should provide recommendations for the correct and responsible use of those systems, and also provide sufficient information about such systems to permit expert evaluation of their risks and impacts.⁹ Finally, providers should enable mechanisms to allow generative AI systems to be deactivated unilaterally by external means in emergency situations.

2. Ownership: Inherent aspects of how generative AI systems are structured and function are not yet adequately accounted for in intellectual property (IP) law and regulation.¹⁰ Such

⁷ "Providers" is used in this document to mean all entities that deliver generative AI technologies, components, systems, or applications to users or other entities. This may include developers; model, dataset, subsystem, platform, system, or application providers; and parties such as sellers, resellers, integrators, or marketers.

⁸ Various bodies such as the National Institute of Standards and Technology (NIST), the Institute of Electrical and Electronics Engineers (IEEE), and the European Union (EU) have made recommendations that are relevant in this regard. (NIST has formulated a risk management framework while the IEEE and EU articulate a risk hierarchy.) See respectively: National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, January 2023 [<https://doi.org/10.6028/NIST.AI.100-1>]; *IEEE Standard for System, Software, and Hardware Verification and Validation*, 1012-2016 [<https://ieeexplore.ieee.org/document/8055462>]; and *Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts*, 2021/0106 (COD), April 21, 2023 [https://eurlex.europa.eu/resource.html?uri=cellar:e0649735-a372-11eb-9585-01aa75ed71a1.0001.02/DOC_1&format=PDF].

Risk assessments of generative AI systems should be done by teams of cross-disciplinary experts and public, private, and non-governmental bodies, and with broad public input. We also note that generative AI systems are complex, not yet fully understood, and may demonstrate emergent behaviors and emergent risks that are not predictable simply by extrapolating from their existing capabilities. This is an area that thus needs substantial further research. Another such area is that of bias which, while a risk in AI systems in general, has become a particularly significant concern with the large language models used in generative AI.

⁹ Generative AI providers should also provide meta-information about models to enable experts and trained members of the community to understand them and evaluate their impacts. Such information might productively include datasheets, model cards, model whitepapers, factsheets, and detailed impact assessments. Well-designed dashboards also could give users a clearer understanding of the impact of their decisions of how best to use generative AI systems, and greater control over their output.

¹⁰ It is not currently possible, for example, for users or creators of generative AI systems to definitively say which portions of a training dataset adhere to which copyrights or licenses, which portions of that dataset may have directly or indirectly contributed to a particular generated artifact, and consequently what the copyright and licensing implications of that artifact may be. This not only creates an issue for creators whose works have been used to generate artifacts, but also for users of those artifacts who may be exposed to the risk of substantial penalties for copyright violations.

regimes thus should be reviewed and, where necessary, revised to strengthen protections for human creators without placing undue restrictions on lawful permissive access to copyrighted material (e.g., pursuant to fair use or fair dealing provisions in the US and Europe)¹¹ or diminishing the overall creative commons.¹²

3. **Personal data control:** Generative AI systems should allow a person to opt out of their data being used to train the system or facilitate its generation of information. In many cases, the default choice should be for a person to explicitly opt into their data being used. At minimum, such systems should provide mechanisms to allow any person to opt out of their personal data, including their biometric data, being used for such purposes.¹³ If a person opts out of providing data once a model has been trained, there should be a mechanism in place to update the model to remove that individual's data.
4. **Correctability:** Providers of generative AI systems should create and maintain public repositories where errors made by the system can be noted and, optionally, corrections made. If an error is discovered and noted, providers should develop transparent mechanisms that allow stakeholders to track providers' progress toward eliminating errors, including the retraining of models and other mitigations as needed.

¹¹ In the United States, a person's original and creative works are automatically copyrighted when first "fixed in a medium of tangible expression." Generally, absent prior approval by the copyright holder, works cannot be used unless deemed a "fair use" under a four-factor statutory test, or they are subject to a limited number of express other statutory exceptions. Other countries may or may not provide similar protection for works created within their own jurisdictions.

¹² Areas of creative work that have traditionally fallen outside of IP controls, such as artistic style, become contentious when a generative AI tool is able to reduce demand for the efforts of human creators through automated mimicry, especially without citation of the works of human creators in a training set. This is especially critical since, unlike a human, the tool can do so quickly and at large scale. At the same time, while traditional notions of fair and acceptable use of copyrighted works allow for certain digital processes to be carried out on them (e.g., to display the works on a screen), it is not clear that this "authorization" will include their use as training data for AI to generate further artifacts in all jurisdictions. Other unforeseen scenarios or outcomes about the uses of generative AI for creative works that either test the boundaries of existing laws and regulations or lack any legal precedent may emerge in the future. We note with concern, for example, attempts at for-profit monetization of human-generated work available through a creative commons and/or publicly available dataset with explicit or implicit human IP attached that contravenes the original intent of or arrangements under which the IP was made available. Such use cases, and doubtless many others, must be addressed by new statutes or judicially resolved on a case-by-case basis.

¹³ Biometric data has been afforded particular protection in some jurisdictions. In the United States, for example, regulation of its use is a matter of state law, both common and statutory. See, e.g., the *Illinois Biometric Information Privacy Act*, 740 ILCS 14 (2008), which places limits on the use of personal images and likenesses [<https://www.ilga.gov/legislation/ilcs/ilcs3.asp?ActID=3004&ChapterID=57>]. The European Union's General Data Protection Regulation provides broad similar protection. *Regulation (EU) 2016/679 of the European Parliament and of the Council on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC*, April 2016 [<https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679>].

Adapted Prior Principles

5. **Transparency:** Any application or system that utilizes generative AI should conspicuously disclose that it does so to the appropriate stakeholders. In particular, where generative AI is being used to simulate human agents, at all times individuals must be promptly and clearly informed that they are interacting with a system as opposed to a human.¹⁴ Further, generative AI systems should warn users that information the system generates may contain errors, and that their authoritative tone or other attributes may be misleading. In addition, to prevent unintended or malicious misrepresentation (e.g., “deepfakes”), generative AI systems should provide a mechanism that permits information they generate to be unambiguously identified by third parties as having been AI produced. Such techniques may include cryptographic or steganographic markers.
6. **Auditability and contestability:** Providers of Generative AI systems should ensure that system models, algorithms, data, and outputs can be recorded where possible (with due consideration to privacy), so that they may be audited and/or contested in appropriate cases. It is also important that providers of Generative AI systems have appropriate auditing strategies in place so citizens, consumer groups, and industry bodies can review and comment on them over time to facilitate their correction and potential retraining.
7. **Limiting environmental impacts:** Given the large environmental impacts of Generative AI models,¹⁵ we recommend that consensus on methodologies be developed to measure, attribute, and actively reduce such impacts. In particular, the total environmental costs to society, including those that are externalized by providers of the technology, must be determinable and attributed to the relevant entities in the ecosystem. Finally, sustainability issues also should be considered and accounted for during a system's entire life cycle.¹⁶

¹⁴ The necessity for transparency becomes even more critical when generative AI intentionally simulates human agents as some users may anthropomorphize such systems inappropriately. A related issue is that some generative AI systems can present their outputs in authoritative language and a manner that conveys their confidence to users. However, the systems are ultimately limited by their training datasets, and the quantity and quality of training sets as well the techniques used can lead to subtle errors. This may cause users to miss errors that have been generated by the AI (sometimes called “hallucinations”) or be lulled into not checking for them adequately, if at all.

¹⁵ The cumulative estimated carbon emissions of recently released Generative AI models have been estimated to greatly exceed those of more traditional AI models. As the use of Generative AI grows such emissions could increase significantly. See C.J. Wu et al., Sustainable AI: Environmental Implications, Challenges and Opportunities, Conference on Machine Learning and Systems (MLSys), 2022.

[https://www.researchgate.net/publication/355843251_Sustainable_AI_Environmental_Implications_Challenges_and_Opportunities]

¹⁶ Such analysis must extend beyond simply focusing on operational efficiency during system training or inference to include, e.g., the tradeoff between AI performance and environmental impact, or techniques to reduce or reuse model training runs or artifacts.

8. **Heightened security and privacy:** Generative AI systems are susceptible to a broad range of new security¹⁷ and privacy¹⁸ risks, including new attack vectors and malicious data leaks, among others. Their use, therefore, requires heightened risk-mitigation controls to ensure that relevant security and privacy best practices are verifiably and consistently employed throughout the model life cycle, and that these can be effectively audited, both internally and as appropriate by third parties.

Reaffirmed Principles

Five additional principles articulated in our October 2022 *joint statement* also continue to apply as originally written to generative and other AI systems. They are reaffirmed and included here for completeness and ease of reference:

9. **Legitimacy and competency:** Designers of algorithmic systems should have the management competence and explicit authorization to build and deploy such systems. They also need to have expertise in the application domain, a scientific basis for the systems' intended use, and be widely regarded as socially legitimate by stakeholders impacted by the system.¹⁹ Legal and ethical assessments must be conducted to confirm that any risks introduced by the systems will be proportional to the problems being addressed, and that any benefit-harm trade-offs are understood by all relevant stakeholders.
10. **Minimizing harm:** Managers, designers, developers, users, and other stakeholders of algorithmic systems should be aware of the possible errors and biases involved in their design, implementation, and use, and the potential harm that a system can cause to individuals and society. Organizations should routinely perform impact assessments on systems they employ to determine whether the system could generate harm, especially discriminatory harm, and to apply appropriate mitigations. When possible, they should learn from measures of actual performance, not solely patterns of past decisions that may themselves have been discriminatory.

¹⁷ For example, the use of generative AI models to generate computer code presents substantial security risks. Such models are typically trained on code repositories. If any credentials are stored with the code, malicious actors could exploit the model to output valid keys. Indeed, they could go even further and introduce malware in response to queries, whether by poisoning training data or corrupting system outputs. Analogous security risks also exist for many other types of generative AI models.

¹⁸ The inherently necessary use of large training dataset and model sizes for generative AI systems can lead to privacy issues becoming more likely or severe than for smaller models or datasets. Models may directly or indirectly infer personally identifiable information (such as employment, home address, and family data) of particular individuals, which are then susceptible to data leaks. Similarly, there are risks of reverse engineering training data from trained models. (Although models amalgamate training data, it has been proven that training examples may nonetheless be recovered in this process.) See for example, Carlini et al., *Quantifying Memorization Across Neural Language Models*, Conference on Learning Representation, 2023. [<https://iclr.cc/virtual/2023/oral/12637>]

¹⁹ Projects with no clear scientific basis (e.g., inferring personality traits from facial images) should not be deployed.

- 11. Interpretability and explainability:** Managers of algorithmic systems are encouraged to produce information regarding both the procedures that the employed algorithms follow (interpretability) and the specific decisions that they make (explainability). Explainability may be just as important as accuracy, especially in public policy contexts or any environment in which there are concerns about how algorithms could be skewed to benefit one group over another without acknowledgement. It is important to distinguish between explanations and after-the-fact rationalizations that do not reflect the evidence, or the decision-making process used to reach the conclusion being explained.
- 12. Maintainability:** Evidence of all algorithmic systems' soundness should be collected throughout their life cycles, including documentation of system requirements, the design or implementation of changes, test cases and results, and a log of errors found and fixed.²⁰ Proper maintenance may require retraining systems with new training data and/or replacing the models employed.
- 13. Accountability and responsibility:** Public and private bodies should be held accountable for decisions made by algorithms they use, even if it is not feasible to explain in detail how those algorithms produced their results. Such bodies should be responsible for entire systems as deployed in their specific contexts, not just for the individual parts that make up a given system. When problems in automated systems are detected, organizations responsible for deploying those systems should document the specific actions that they will take to remediate the problem and under what circumstances the use of such technologies should be suspended or terminated.

²⁰ Otherwise, the system may become less appropriate as inputs drift from those originally anticipated, or if the underlying real-world conditions change (e.g., facial recognition systems are used on a wider or different demographic than was present in the training data).



IT Professionals

NEW ZEALAND



IT Professionals NZ

Code of Ethics

The mandatory code outlining ethical and professional requirements for IT Professionals in New Zealand.

May 2017

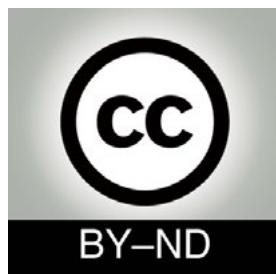
About ITP

IT Professionals New Zealand (ITP) is the professional body of the IT sector in New Zealand.

ITP has a proud history spanning over 50 years and has been a part of the computing and IT sector in New Zealand since formation in 1960.

ITP Vision

ITP is the authoritative voice of the IT profession that leads professional development and good practice in IT.



This document is made available under the following license:

Creative Commons Attribution-No Derivative Works 3.0 New Zealand

You are free to **share** (copy, distribute and transmit the work) under the following conditions:

- ▶ **Attribution:** You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
- ▶ **No Derivative Works:** You may not alter, transform, or build upon this work.

Waiver: Note that any of these conditioned may be waived if you get permission from IITP.

Full details at <http://creativecommons.org/licenses/by-nd/3.0/nz/>

The ITP Code of Ethics

This ITP Code of Ethics document is split into three sections:

Section 1 - Code of Ethics _____ **Page 4**

This is the ITP Code of Ethics in its entirety. The Code is made up of 8 Tenets, each related to the others but also covering an important area related to ethics and conduct.

Section 2 - Guidelines and Interpretation _____ **Page 6**

This material does not form a part of the Code, but rather is used to assist in the interpretation and implementation of the Code of Ethics.

Section 3 - Breaches and Disciplinary Process _____ **Page 14**

This section is taken from Schedule Four of the ITP Bylaws, and outlines the procedures by which alleged breaches of the Code will be dealt with, including the provision for a hearing of the *Discipline and Professional Conduct Board*.

This ITP Code of Ethics was formally adopted by resolution of a Special General Meeting held for that purpose in Wellington, New Zealand, on Friday 28th May 2010. From 2010 to 2015 this was known as the “Code of Professional Conduct”, however was renamed back to Code of Ethics via a Special General Meeting on 30 June 2015.

The Code replaced the previous ITP Code of Ethics.

Section 1:

The ITP New Zealand Code of Ethics

ITP Code of Ethics

IT Professionals will practice with:

- 1. Good faith** - Members shall treat people with dignity, good faith and equality; without discrimination; and have consideration for the values and cultural sensitivities of all groups within the community affected by their work;
- 2. Integrity** - Members shall act in the execution of their profession with integrity, dignity and honour to merit the trust of the community and the profession, and apply honesty, skill, judgement and initiative to contribute positively to the well-being of society;
- 3. Community-focus** - Members' responsibility for the welfare and rights of the community shall come before their responsibility to their profession, sectional or private interests or to other members;
- 4. Skills** - Members shall apply their skills and knowledge in the interests of their clients or employers for whom they will act without compromising any other of these Tenets;
- 5. Continuous Development** - Members shall develop their knowledge, skills and expertise continuously through their careers, contribute to the collective wisdom of the profession, and actively encourage their associates to do likewise;
- 6. Informed Consent** - Members shall take reasonable steps to inform themselves, their clients or employers of the economic, social, environmental or legal consequences which may arise from their actions;
- 7. Managed Conflicts of Interest** - Members shall inform their clients or employers of any interest which may be, or may be perceived as being, in conflict with the interests of their clients or employers, or which may affect the quality of service or impartial judgement;
- 8. Competence** - Members shall follow recognised professional practice, and provide services and advice carefully and diligently only within their areas of competence.

These Tenets comprise the essence of the Code of Ethics. Breaches shall be dealt with as provided in the ITP Bylaws. They should be read and applied in conjunction with Supplements to the Code of Ethics.

The Supplements will be maintained and updated by the Institute so as to be timely and relevant. Members of the Institute have a duty to be cognisant of the Supplements' content especially those which are applicable to the areas of professional expertise and activity in which they are involved.

Section 2:

Supplement to the Code: **Guidance and Interpretation**

Supplement to the ITP Code of Ethics:

Guidance and Interpretation

This supplement does not form a part of the Code, however should be read in conjunction with the code to provide context and interpretation.

This Supplement includes the following sections:

1: Context of the Tenets of the Code	8
2: General Matters	8
3: The Community	9
4: Qualifications and Competence	10
5: Clients and Employers	11
6: Ethical Dilemmas	12
7: Maintenance of the Code	13

1: Context of the Tenets of the Code

- 1.1** The Code of Ethics is issued under the provisions of clause 9 of the Constitution of the Institute of IT Professionals NZ Incorporated (trading as IT Professionals New Zealand) and is binding on all members in all grades of the Institute, including the Institute itself.
- 1.2** This Code of Ethics is to be read and interpreted in conjunction with the Institute's Constitution, Bylaws and this Guidance and Interpretation.
- 1.3** The respect which society accords the technology professions is earned and maintained by its members demonstrating a strong and consistent commitment to ethical values. These commitments are additional to the obligations, which every member of society is required to observe, such as obeying the law of the legislative authority within which they are working as well as those of their own country as applicable, and reflect the additional responsibility expected of all professionals.
- 1.4** Therefore the Institute must maintain an appropriate Code of Ethics, to make it available for the information of the public and to enforce it impartially. This Code must be responsive to the changing expectations of both the public and the profession and the global standards to which the Institute of IT Professionals subscribes.
- 1.5** This code is based upon the principles of:
- Interests of the community
 - Respect for the individual
 - Interests of the client
 - Professional integrity

and supported by the values of:

- Competence
- Truth
- Social justice and
- Ethical behaviour

- 1.6** Eight ethical Tenets form the basis of the Code to guide members in achieving the high ideals of professional life. To assist in the interpretation of the Code, guidelines are set out below to support these Tenets.

2: General Matters

- 2.1** The purpose of this supplement is to assist members in applying the Tenets of the Code of Ethics which have been written in broad terms. Whilst not specifically part of the Code of Ethics, the supplement should be read in conjunction with the Code and members will be expected to have considered the content of the Code, this and any other supplements in any matter of professional conduct.
- 2.2** The supplements are not prescriptive but form the basis, in conjunction with the Constitution and Bylaws, of the Institute's concept of Professional Conduct and Practice.

3: The Community

3.1 “Community”, in the context of the Code, refers to all groups in society including members’ own workplaces. The first three Tenets of the Code refer to the Community and may be considered to include:

- Acting and working in a way such that the health, safety and well-being of employees and colleagues are not endangered;
- Ensuring that work undertaken meets community expectations by adopting the norms of recognised professional practice; or communicating any attendant risks or limitations, and their effect, in any work undertaken which does not accord with convention;
- Being vigilant in ‘duty of care’ toward members of the community;
- Communicating the results of work undertaken in a clear and unambiguous way;
- Raising real or perceived conflicts of interest, or issues which may not be in the community interest, at an early stage of involvement;
- Commitment to the principles of sustainable development of the planet’s resources and seeking to minimise adverse environmental impacts of their work or applications of technology for both present and future generations;
- Not being involved in any activity which is known to be fraudulent, dishonest or not in the interests of the community (as described); and
- Not accepting reward or compensation from any more than one party, without the clear understanding and acceptance of all parties.

3.2 In summary, these Tenets of the Code require members to be mindful of more than their technical and professional responsibilities and their immediate employer or client.

3.3 Many of the requirements demand no more than sound management practices such as Occupational Health and Safety Plans and care of colleagues and staff. The Code goes further in its obligation for members to be aware of the consequences of all of their actions in the practice of their professions. The essence of “professionalism” is in remaining aware of these obligations and in making sound and informed decisions when faced with any conflict of responsibilities which may, and likely will, arise.

4: Qualifications and Competence

- 4.1** Qualifications denote the foundation of knowledge that a member has achieved through formal education, experience, post graduate learning or a combination from all of these sources.
- 4.2** Competence is demonstrated by application of knowledge and skills to provide service, advice or opinion to clients or employers.
- 4.3** The Tenets of the Code which relate to these themes may be considered to include an expectation of members to:
- not misrepresent the qualifications and competences of themselves or those in their employ or under their supervision;
 - not undertake any assignment which is outside their competence; and if requested to do so, to bring to their client's or employer's attention to the need to access further expertise;
 - seek expert assistance on encountering any professional issue or problem outside the range of experience or competence;
 - not expect their employees to undertake work for which the employees have little or no demonstrated competence, other than in a supervised capacity; and
 - keep themselves competent and informed by continuous professional development.
- 4.4** By carefully limiting the professional work undertaken within the limits of their qualifications and competence, members protect the interests of the community, clients, employers and themselves.
- 4.5** A mistake or error of judgement that a member might make within the limits of competence and qualification, even though it may be judged as negligence, will not be considered as unethical behaviour.

5: Clients and Employers

- 5.1** Members have a duty to provide loyal and competent service to their clients and employers. “Loyalty” implies looking out for their interests, giving fearless advice, providing strict confidentiality.
- 5.2** A client is the recipient of professional service, advice or opinion. Members are encouraged to be always mindful of the question “who is the client?” In some circumstances it may also be the employer. An employee of a private company has dual duties to employer and client. An academic employed by a University has students as clients.
- 5.3** When a member is serving both an employer and a client, there is potential for competing and conflicting interests.
- 5.4** The Tenets of the Code which relate to these themes include an expectation of members to:
- not disclose or use any confidential information gained in the course of their employment without permission, unless disclosure is a legal requirement or withholding the information would be to the detriment of the community;
 - be truthful, factual and free from exaggeration in advertising or promoting their services to potential clients, either by advertisement or direct approach;
 - keep clients and employers informed of any known or potential conflict of interest;
 - not accept payment for particular service or information from more than one source without disclosure to all parties, unless it is apparent that the service or information is intended for multiple use;
 - not undertake assignments under conditions which may compromise their ability to satisfy client or employer needs in a professional manner;
 - advise clients and employers of the level of risk, or any possible adverse consequences, of any instruction given which is outside the norms of conventional practice;
 - not attempt to supplant others who are already engaged by a client; nor to damage in any way the reputation of competitors by way of comparison or denigration;
 - when acting as an expert witness to a court or tribunal, be mindful of their primary obligation as an expert witness to the court or the hearing, and not to the engaging party.

6: Ethical Dilemmas

- 6.1** An ethical dilemma occurs when one Tenet of the Code cannot be honoured without apparent breach of another. It is recommended that any member finding themselves in such a situation should consult with the affected parties and attempt an ethical resolution or, failing that, refer the matter to the Discipline and Professional Conduct Board for advice.
- 6.2** In any professional career, ethical challenges or dilemmas will arise. Teachers may be caught between the demands of their employers and the needs of their students; employees may be caught between the performance targets of their employers and the expectations of their clients; consultants may be caught between the expectations of their clients and the interests of the community; public service professionals may be caught between the directives of superiors and the well being of staff.
- 6.3** Resolution of ethical dilemmas is ultimately a matter of personal responsibility, taking into account the principles which lie behind the Tenets of the Code. Members of the Institute are encouraged to share their ethical dilemmas with a trusted colleague, or refer such matters to the Institute's National Board or CEO if they wish. Either of those may further refer the matter to the PCB for consideration
- 6.4** Members seeking guidance in higher ethical issues are recommended to consider documents such as the United Nations Declaration of Human Rights, the Treaty of Waitangi and the New Zealand Bill of Rights. It is recognised that members may also feel subject to religious obligations which may impose ethical and moral dilemmas. In such cases, members are advised to consult with their religious guides and may refer such matters to the Institute's National Board or CEO if they wish. Either of those may further refer the matter to the Discipline and Professional Conduct Board for consideration.
- 6.5** Another aspect of ethical dilemma is the occurrence of a conflict of interest. A conflict of interest may be real, potential or perceived. It arises when relationships exist among such entities as clients, employers, employees, vendors, colleagues or any combination that can, may or actually influence another entity to a possibly detrimental extent. These extents include diminished commitment, pecuniary or other gain, undue influence
- 6.6** The issue can also be difficult because a perceived conflict of interest may be as consequential as a real occurrence. This happens when those who perceive a conflict act in ways that are influenced by the perception. As such, perceived conflicts of interest may have to be managed or otherwise dealt with as if real. The Code of Practice details some strategies relevant to conflict of interest resolution or management.

7: Maintenance of the Code

7.1 The role of IT Professionals NZ is:

- to provide rigorous and fair processes for dealing with complaints and charges made against members; and
- to provide counsel and support to members confronting ethical dilemmas or any other difficulties in relation to their own or others' ethical behaviour.

7.2 ITP has a Discipline and Professional Conduct Board which "is responsible for the implementation and oversight of professional conduct standards as they apply to all membership classes and to maintain alignment of such standards with kindred bodies and international guidelines."¹

7.3 Where appropriate, the Discipline and Professional Conduct Board will, as circumstances warrant, propose additions, changes or deletions within both this supplement and Supplement 2, the Code of Practice. Members are encouraged to submit material for Board consideration where they feel that guidance and interpretation and/or the Code of Practice might be improved.

7.4 The *Code of Practice* endeavours to provide guidance and examples within specific areas of Information Technology practice. As such, it is likely to change far more frequently than the other documents mentioned. Members are encouraged to refer to the Code regularly (either on the Institute's web site or in downloadable form or as a printed copy by request from the Institute).

7.5 The two Codes, particularly, form the basis upon which your conduct as a member of the Institute might be measured by the public, employers, clients and your colleagues. It is also the basis against which any disciplinary proceedings of the Institute will seek comparison.

7.6 Schedule 4 of the ITP Bylaws, dealing with breaches of the Code of Ethics, provides additional information.

¹ Professional Conduct Board – Mandate, Matthews, P., et al., 2009. *Information Technology Certified Professional, IITP ITCP Certification Model*. New Zealand Computer Society Inc.

Section 3:

ITP Bylaws Schedule 4: **ITP Disciplinary Process**

ITP Bylaws Schedule Four:

Breach of the Code of Ethics

1 Preamble

"The processes undertaken in the consideration of complaints or the resolution of disputes will be in strict accord with Rules determined by the Institute of IT Professionals NZ's National Board. The rules of natural justice will be paramount in all processes. The rules of natural justice include the right for a respondent to know the details of a complaint and the supporting evidence; the right to provide evidence in defence; and the right for an unbiased determination made by those who hear all the evidence."

- Institute of IT Professionals NZ, Code of Ethics

2 Procedure

The procedure for dealing with an alleged breach of the Code of Ethics by a member shall be as follows:

Lodging the Complaint

- 2.1 A complaint, being an allegation of a breach of the Code of Ethics, may be made by any individual or entity concerning any person who is a Institute member or was a member at the time relevant to the complaint.
- 2.2 Placing such a complaint shall signify acceptance that the matter shall be determined in the manner prescribed in this Schedule of the Bylaws, including the clauses dealing with publication of the findings of this consideration of the complaint. The complainant also specifically waives any right to take civil action against the Institute should he or she disagree with the process or findings of the Institute.
- 2.3 Nothing in this Schedule shall preclude the complainant or others taking civil or criminal proceedings against the member in question, and this process should not be seen as an alternative to doing so if appropriate.
- 2.4 The complaint must be made on the prescribed form setting out particulars of the alleged breach and attaching any documentation or other relevant details. The complaint shall be addressed to the President, or if it is in relation to the conduct of the President, to the Deputy President. This person shall be the "Receiver" of the complaint.

Receiving the Complaint

- 2.5 The President (or Deputy President in the case of a complaint against the President) may delegate to the Chief Executive or any member of the National Board to act on his or her behalf as Receiver to discharge the remainder of his or her responsibilities in relation to the matter if he or she sees fit. For example, if there is a Conflict of Interest, or if he or she is unable to fulfill the Receiver's requirements within the time required.

- 2.6 The Receiver must acknowledge receipt of the complaint in writing, and notify the member of the complaint in writing (attaching a copy of the complaint and all documentation provided with the complaint), including that the complaint has been referred to the Discipline and Professional Conduct Board who will hold a hearing into the complaint.

Referral to Discipline and Professional Conduct Board

- 2.7 The Receiver must refer the complaint to the Chair of the Discipline and Professional Conduct Board within 7 days of receipt.
- 2.8 If the Chair of the Discipline and Professional Conduct Board is not available to complete the requirements in this Schedule in a timely manner the IITP President may authorise another member of the Board or any member of the IITP National Board to Chair proceedings and discharge the duties of the Chair as outlined in this Schedule.

Frivolous or Vexatious Complaints

- 2.9 At any stage of proceedings if the Chair of the Discipline and Professional Conduct Board believes that the complaint is frivolous or vexatious they may immediately call for a vote of the Discipline and Professional Conduct Board as to whether it is indeed frivolous or vexatious. This vote may be conducted in any way prescribed by the Chair, including electronic via email.
- 2.10 If the Discipline and Professional Conduct Board votes by a three quarters majority that the complaint is frivolous or vexatious it shall be immediately dismissed.

Hearing of the Discipline and Professional Conduct Board

- 2.11 The Chair of the Discipline and Professional Conduct Board shall forward a copy of the complaint and all supporting information to the members of the Discipline and Professional Conduct Board and arrange a hearing of the Board, which shall be between 3 and 6 weeks from the date the complaint is referred. The Chair of this Board may choose a hearing time outside this timeframe if it is necessary to ensure a fair hearing, however the hearing must be conducted in a timely fashion.
- 2.12 The quorum for the hearing shall be two thirds of the membership of the Discipline and Professional Conduct Board. The hearing may be held by Teleconference or in person.
- 2.13 Hearings are confidential and the evidence provided during a hearing is not published, other than that to support a published verdict (see below).
- 2.14 The complainant will have the option of addressing the hearing of the Discipline and Professional Conduct Board in person (where the hearing is held in person) or by Teleconference for the purpose of providing further verbal evidence. The complainant may equally opt not to do so. If they are to address the hearing they may only be present for the portion of the hearing set aside for that purpose.

- 2.15 The Discipline and Professional Conduct Board may call on any witnesses or other persons to provide verbal evidence to the hearing, but has no authority to compel. Any witness may opt to provide evidence by way of a signed statement. Any witness providing verbal evidence may only be present for the portion of the hearing set aside for that purpose.
- 2.16 In keeping with the principles of natural justice, the member alleged to have committed the breach will have the option of being present during all verbal evidence but may not address the complainant, witnesses, or the Board at that time.
- 2.17 The member alleged to have committed the breach will have the option of addressing the hearing of the Discipline and Professional Conduct Board in person (where the hearing is held in person) or by Teleconference for the purpose of providing verbal evidence. The member may also choose to provide evidence in writing in advance by signed statement. The complainant may equally opt not to do either and this shall not be construed as evidence of acceptance of the allegation of breach.
- 2.18 The member alleged to have committed the breach may choose to be represented.

Outcome of Hearing

- 2.19 The Discipline and Professional Conduct Board shall deliberate in private until such time that:
 - 2.19.1 The Board rules by a three quarters majority that a significant breach has occurred. In this case a breach shall be found proved and the finding, along with a detailed justification for the finding, shall be referred to the National Board;
 - 2.19.2 The Board rules by a three quarters majority that a technical breach occurred, but finds that the breach is trivial or trifling and the matter is dismissed;
 - 2.19.3 The Board rules by a three quarters majority that no breach has occurred and the matter is dismissed;
 - 2.19.4 The Board cannot agree by a three quarters majority that a breach has occurred, and the Chair determines that further deliberation would be fruitless. The matter is therefore found to be unproved and is dismissed.
- 2.20 The complainant and the member shall be informed of the outcome without delay.

Penalty Imposed

- 2.21 In the case of a breach, the Discipline and Professional Conduct Board shall agree a penalty consistent with both the gravity of the breach and previous penalties for similar breaches from the options provided in the Institute's Constitution. A report outlining the findings and penalty shall be forwarded to the National Board without delay.
- 2.22 In the case of a breach, the Discipline and Professional Conduct Board shall solely decide the penalty, if any, from the remedies provided in the Constitution. The penalty is not subject to appeal.

- 2.23 The Discipline and Professional Conduct Board shall determine whether the existence, details, finding, complainant identity, and/or penalty shall be released publicly, and no other member or participant shall release in part or full any determination or detail related to the hearing or complaint other than that which this Board chooses to release.
- 2.24 If the hearing uncovers significant criminal or civil wrongdoing the National Board may opt to refer the matter to the NZ Police or other law enforcement agency by way of a formal complaint, or commence other legal proceedings as it sees fit.

Appealing the Determination of the Hearing

- 2.25 An Appeal against any finding may be made by either the Complainant or Member concerned for any of the following reasons:
 - 2.25.1 If further evidence that was not previously available becomes available that is materially different to any considered during the hearing and which, on balance, could change the outcome;
 - 2.25.2 If the procedure outlined in this schedule was not adhered to, and the breach is more than trivial and may have materially changed the outcome of the hearing;
 - 2.25.3 A member of the Discipline and Professional Conduct Board had a significant undeclared Conflict of Interest which may have impacted upon his or her impartiality.
- 2.26 Simply not agreeing with the determination or penalty shall not be grounds for appeal.
- 2.27 Any appeal should be made to the original Receiver within 21 days of the determination of the Discipline and Professional Conduct Board. The Receiver will initially determine whether, on the balance of probability, the appeal meets the criteria outlined in 2.25 and if so, shall formally forward the appeal to the National Board.
- 2.28 The National Board shall consider the matters raised in the appeal and determine whether the appeal shall stand. If the appeal stands the Board may modify the determination or penalty as it sees fit.

ACM Ethics

The Official Site of the Association for
Computing Machinery's Committee on
Professional Ethics

The joint ACM/IEEE-CS Software Engineering Code was published as: Don Gotterbarn, Keith Miller, and Simon Rogerson. 1997. Software engineering code of ethics. *Commun. ACM* 40, 11 (November 1997), 110-118.

DOI: [10.1145/265684.265699](https://doi.org/10.1145/265684.265699)

Note that this code is for anyone that is a member of the software engineering profession, regardless of ACM membership status. You may also wish to consult [The Code for all ACM members](#) (regardless of profession).

Thanks to [SEERI](#) for these translations of the SE Code: 

Want to contribute a translation? Get it touch via the [Contact Us](#) page.

The Software Engineering Code of Ethics and Professional Practice

Software Engineering Code of Ethics and Professional Practice (Version 5.2) as recommended by the ACM/IEEE-CS Joint Task Force on Software Engineering Ethics and Professional Practices and jointly approved by the ACM and the IEEE-CS as the standard for teaching and practicing software engineering.

Software Engineering Code of Ethics and Professional Practice (Short Version)

PREAMBLE

The short version of the code summarizes aspirations at a high level of the abstraction; the clauses that are included in the full version give examples and details of how these aspirations change the way we act as software engineering professionals. Without the aspirations, the details can become legalistic and tedious; without the details, the aspirations can become high sounding but empty; together, the aspirations and the details form a cohesive code.

Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. PUBLIC – Software engineers shall act consistently with the public interest.
2. CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.

3. PRODUCT – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT – Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES – Software engineers shall be fair to and supportive of their colleagues.
8. SELF – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Software Engineering Code of Ethics and Professional Practice (Full Version)

PREAMBLE

Computers have a central and growing role in commerce, industry, government, medicine, education, entertainment and society at large. Software engineers are those who contribute by direct participation or by teaching, to the analysis, specification, design, development, certification, maintenance and testing of software systems. Because of their roles in developing software systems, software engineers have significant opportunities to do good or cause harm, to enable others to do good or cause harm, or to influence others to do good or cause harm. To ensure, as much as possible, that their efforts will be used for good, software engineers must commit themselves to making software engineering a beneficial and respected profession. In accordance with that commitment, software engineers shall adhere to the following Code of Ethics and Professional Practice.

The Code contains eight Principles related to the behavior of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession. The Principles identify the ethically responsible relationships in which individuals, groups, and organizations participate and the primary obligations within these relationships. The Clauses of each Principle are illustrations of some of the obligations included in these relationships. These obligations are founded in the software engineer's humanity, in special care owed to people affected by the work of software engineers, and the unique elements of the practice of software engineering. The Code prescribes these as obligations of anyone claiming to be or aspiring to be a software engineer.

It is not intended that the individual parts of the Code be used in isolation to justify errors of omission or commission. The list of Principles and Clauses is not exhaustive. The Clauses should not be read as separating the acceptable from the unacceptable in professional conduct in all practical situations. The Code is not a simple ethical algorithm that generates ethical decisions. In some situations standards may be in tension with each other or with standards from other sources. These situations require the software engineer to use ethical judgment to act in a manner which is most consistent with the spirit of the Code of Ethics and Professional Practice, given the

circumstances.

Ethical tensions can best be addressed by thoughtful consideration of fundamental principles, rather than blind reliance on detailed regulations. These Principles should influence software engineers to consider broadly who is affected by their work; to examine if they and their colleagues are treating other human beings with due respect; to consider how the public, if reasonably well informed, would view their decisions; to analyze how the least empowered will be affected by their decisions; and to consider whether their acts would be judged worthy of the ideal professional working as a software engineer. In all these judgments concern for the health, safety and welfare of the public is primary; that is, the “Public Interest” is central to this Code.

The dynamic and demanding context of software engineering requires a code that is adaptable and relevant to new situations as they occur. However, even in this generality, the Code provides support for software engineers and managers of software engineers who need to take positive action in a specific case by documenting the ethical stance of the profession. The Code provides an ethical foundation to which individuals within teams and the team as a whole can appeal. The Code helps to define those actions that are ethically improper to request of a software engineer or teams of software engineers.

The Code is not simply for adjudicating the nature of questionable acts; it also has an important educational function. As this Code expresses the consensus of the profession on ethical issues, it is a means to educate both the public and aspiring professionals about the ethical obligations of all software engineers.

PRINCIPLES

Principle 1: PUBLIC

Software engineers shall act consistently with the public interest. In particular, software engineers shall, as appropriate:

1.01. Accept full responsibility for their own work.

1.02. Moderate the interests of the software engineer, the employer, the client and the users with the public good.

1.03. Approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy or harm the environment. The ultimate effect of the work should be to the public good.

1.04. Disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents.

1.05. Cooperate in efforts to address matters of grave public concern caused by software, its installation, maintenance, support or documentation.

1.06. Be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools.

1.07. Consider issues of physical disabilities, allocation of resources, economic disadvantage and other factors that

can diminish access to the benefits of software.

1.08. Be encouraged to volunteer professional skills to good causes and contribute to public education concerning the discipline.

Principle 2: CLIENT AND EMPLOYER

Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest. In particular, software engineers shall, as appropriate:

2.01. Provide service in their areas of competence, being honest and forthright about any limitations of their experience and education.

2.02. Not knowingly use software that is obtained or retained either illegally or unethically.

2.03. Use the property of a client or employer only in ways properly authorized, and with the client's or employer's knowledge and consent.

2.04. Ensure that any document upon which they rely has been approved, when required, by someone authorized to approve it.

2.05. Keep private any confidential information gained in their professional work, where such confidentiality is consistent with the public interest and consistent with the law.

2.06. Identify, document, collect evidence and report to the client or the employer promptly if, in their opinion, a project is likely to fail, to prove too expensive, to violate intellectual property law, or otherwise to be problematic.

2.07. Identify, document, and report significant issues of social concern, of which they are aware, in software or related documents, to the employer or the client.

2.08. Accept no outside work detrimental to the work they perform for their primary employer.

2.09. Promote no interest adverse to their employer or client, unless a higher ethical concern is being compromised; in that case, inform the employer or another appropriate authority of the ethical concern.

Principle 3: PRODUCT

Software engineers shall ensure that their products and related modifications meet the highest professional standards possible. In particular, software engineers shall, as appropriate:

3.01. Strive for high quality, acceptable cost and a reasonable schedule, ensuring significant tradeoffs are clear to and accepted by the employer and the client, and are available for consideration by the user and the public.

3.02. Ensure proper and achievable goals and objectives for any project on which they work or propose.

3.03. Identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects.

3.04. Ensure that they are qualified for any project on which they work or propose to work by an appropriate combination of education and training, and experience.

3.05. Ensure an appropriate method is used for any project on which they work or propose to work.

3.06. Work to follow professional standards, when available, that are most appropriate for the task at hand, departing from these only when ethically or technically justified.

3.07. Strive to fully understand the specifications for software on which they work.

3.08. Ensure that specifications for software on which they work have been well documented, satisfy the users' requirements and have the appropriate approvals.

3.09. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work and provide an uncertainty assessment of these estimates.

3.10. Ensure adequate testing, debugging, and review of software and related documents on which they work.

3.11. Ensure adequate documentation, including significant problems discovered and solutions adopted, for any project on which they work.

3.12. Work to develop software and related documents that respect the privacy of those who will be affected by that software.

3.13. Be careful to use only accurate data derived by ethical and lawful means, and use it only in ways properly authorized.

3.14. Maintain the integrity of data, being sensitive to outdated or flawed occurrences.

3.15 Treat all forms of software maintenance with the same professionalism as new development.

Principle 4: JUDGMENT

Software engineers shall maintain integrity and independence in their professional judgment. In particular, software engineers shall, as appropriate:

4.01. Temper all technical judgments by the need to support and maintain human values.

4.02 Only endorse documents either prepared under their supervision or within their areas of competence and with which they are in agreement.

4.03. Maintain professional objectivity with respect to any software or related documents they are asked to evaluate.

4.04. Not engage in deceptive financial practices such as bribery, double billing, or other improper financial practices.

4.05. Disclose to all concerned parties those conflicts of interest that cannot reasonably be avoided or escaped.

4.06. Refuse to participate, as members or advisors, in a private, governmental or professional body concerned with software related issues, in which they, their employers or their clients have undisclosed potential conflicts of interest.

Principle 5: MANAGEMENT

Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance . In particular, those managing or leading software engineers shall, as appropriate:

5.01 Ensure good management for any project on which they work, including effective procedures for promotion of quality and reduction of risk.

5.02. Ensure that software engineers are informed of standards before being held to them.

5.03. Ensure that software engineers know the employer's policies and procedures for protecting passwords, files and information that is confidential to the employer or confidential to others.

5.04. Assign work only after taking into account appropriate contributions of education and experience tempered with a desire to further that education and experience.

5.05. Ensure realistic quantitative estimates of cost, scheduling, personnel, quality and outcomes on any project on which they work or propose to work, and provide an uncertainty assessment of these estimates.

5.06. Attract potential software engineers only by full and accurate description of the conditions of employment.

5.07. Offer fair and just remuneration.

5.08. Not unjustly prevent someone from taking a position for which that person is suitably qualified.

5.09. Ensure that there is a fair agreement concerning ownership of any software, processes, research, writing, or other intellectual property to which a software engineer has contributed.

5.10. Provide for due process in hearing charges of violation of an employer's policy or of this Code.

5.11. Not ask a software engineer to do anything inconsistent with this Code.

5.12. Not punish anyone for expressing ethical concerns about a project.

Principle 6: PROFESSION

Software engineers shall advance the integrity and reputation of the profession consistent with the public interest. In particular, software engineers shall, as appropriate:

6.01. Help develop an organizational environment favorable to acting ethically.

- 6.02. Promote public knowledge of software engineering.
- 6.03. Extend software engineering knowledge by appropriate participation in professional organizations, meetings and publications.
- 6.04. Support, as members of a profession, other software engineers striving to follow this Code.
- 6.05. Not promote their own interest at the expense of the profession, client or employer.
- 6.06. Obey all laws governing their work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest.
- 6.07. Be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful.
- 6.08. Take responsibility for detecting, correcting, and reporting errors in software and associated documents on which they work.
- 6.09. Ensure that clients, employers, and supervisors know of the software engineer's commitment to this Code of ethics, and the subsequent ramifications of such commitment.
- 6.10. Avoid associations with businesses and organizations which are in conflict with this code.
- 6.11. Recognize that violations of this Code are inconsistent with being a professional software engineer.
- 6.12. Express concerns to the people involved when significant violations of this Code are detected unless this is impossible, counter-productive, or dangerous.
- 6.13. Report significant violations of this Code to appropriate authorities when it is clear that consultation with people involved in these significant violations is impossible, counter-productive or dangerous.

Principle 7: COLLEAGUES

Software engineers shall be fair to and supportive of their colleagues. In particular, software engineers shall, as appropriate:

- 7.01. Encourage colleagues to adhere to this Code.
- 7.02. Assist colleagues in professional development.
- 7.03. Credit fully the work of others and refrain from taking undue credit.
- 7.04. Review the work of others in an objective, candid, and properly-documented way.
- 7.05. Give a fair hearing to the opinions, concerns, or complaints of a colleague.
- 7.06. Assist colleagues in being fully aware of current standard work practices including policies and procedures

for protecting passwords, files and other confidential information, and security measures in general.

7.07. Not unfairly intervene in the career of any colleague; however, concern for the employer, the client or public interest may compel software engineers, in good faith, to question the competence of a colleague.

7.08. In situations outside of their own areas of competence, call upon the opinions of other professionals who have competence in that area.

Principle 8: SELF

Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession. In particular, software engineers shall continually endeavor to:

8.01. Further their knowledge of developments in the analysis, specification, design, development, maintenance and testing of software and related documents, together with the management of the development process.

8.02. Improve their ability to create safe, reliable, and useful quality software at reasonable cost and within a reasonable time.

8.03. Improve their ability to produce accurate, informative, and well-written documentation.

8.04. Improve their understanding of the software and related documents on which they work and of the environment in which they will be used.

8.05. Improve their knowledge of relevant standards and the law governing the software and related documents on which they work.

8.06 Improve their knowledge of this Code, its interpretation, and its application to their work.

8.07 Not give unfair treatment to anyone because of any irrelevant prejudices.

8.08. Not influence others to undertake any action that involves a breach of this Code.

8.09. Recognize that personal violations of this Code are inconsistent with being a professional software engineer.

This Code was developed by the ACM/IEEE-CS joint task force on Software Engineering Ethics and Professional Practices (SEPP):

Executive Committee: Donald Gotterbarn (Chair), Keith Miller and Simon Rogerson;

Members: Steve Barber, Peter Barnes, Ilene Burnstein, Michael Davis, Amr El-Kadi, N. Ben Fairweather, Milton Fulghum, N. Jayaram, Tom Jewett, Mark Kanko, Ernie Kallman, Duncan Langford, Joyce Currie Little, Ed Mechler, Manuel J. Norman, Douglas Phillips, Peter Ron Prinzivalli, Patrick Sullivan, John Weckert, Vivian Weil, S. Weisband and Laurie Honour Werth.

This Code may be published without permission as long as it is not changed in any way and it carries the copyright notice. Copyright (c) 1999 by the Association for Computing Machinery, Inc. and the Institute for Electrical and Electronics Engineers, Inc.

ACM Ethics

Proudly powered by WordPress.

Substantial net improvements in programming quality and productivity have been obtained through the use of formal inspections of design and of code. Improvements are made possible by a systematic and efficient design and code verification process, with well-defined roles for inspection participants. The manner in which inspection data is categorized and made suitable for process analysis is an important factor in attaining the improvements. It is shown that by using inspection results, a mechanism for initial error reduction followed by ever-improving error rates can be achieved.

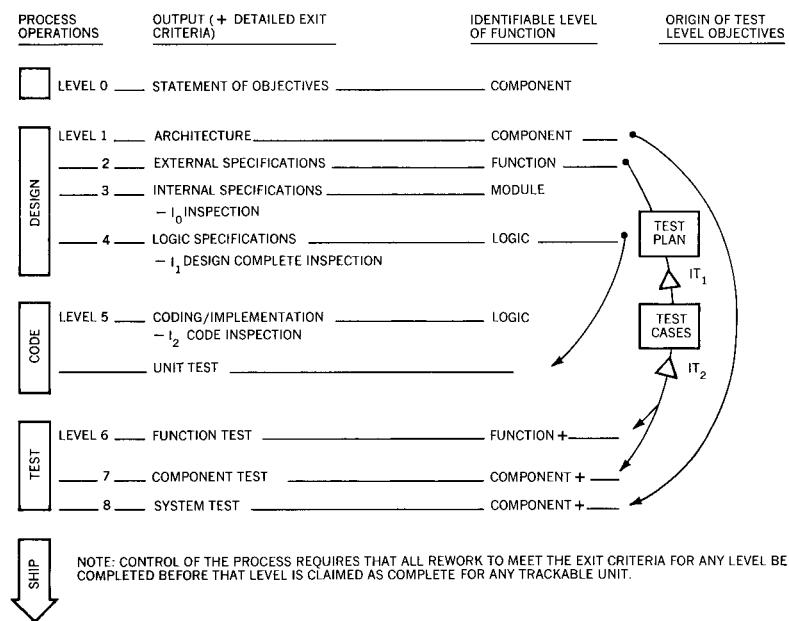
Design and code inspections to reduce errors in program development

by M. E. Fagan

Successful management of any process requires planning, measurement, and control. In programming development, these requirements translate into defining the programming process in terms of a series of operations, each operation having its own exit criteria. Next there must be some means of measuring completeness of the product at any point of its development by inspections or testing. And finally, the measured data must be used for controlling the process. This approach is not only conceptually interesting, but has been applied successfully in several programming projects embracing systems and applications programming, both large and small. It has not been found to "get in the way" of programming, but has instead enabled higher predictability than other means, and the use of inspections has improved productivity and product quality. The purpose of this paper is to explain the planning, measurement, and control functions as they are affected by inspections in programming terms.

An ingredient that gives maximum play to the planning, measurement, and control elements is consistent and vigorous *discipline*. Variable rules and conventions are the usual indicators of a lack of discipline. An iron-clad discipline on all rules, which can stifle programming work, is not required but instead there should be a clear understanding of the flexibility (or nonflexibility) of each of the rules applied to various aspects of the project. An example of flexibility may be waiving the rule that all main paths will be tested for the case where repeated testing of a given path will logically do no more than add expense. An example of necessary inflexibility would be that *all* code must be

Figure 1 Programming process



inspected. A clear statement of the project rules and changes to these rules along with faithful adherence to the rules go a long way toward practicing the required project discipline.

A prerequisite of process management is a clearly defined series of operations in the process (Figure 1). The miniprocess within each operation must also be clearly described for closer management. A clear statement of the criteria that must be satisfied to exit each operation is mandatory. This statement and accurate data collection, with the data clearly tied to trackable units of known size and collected from specific points in the process, are some essential constituents of the information required for process management.

In order to move the form of process management from qualitative to more quantitative, process terms must be more specific, data collected must be appropriate, and the limits of accuracy of the data must be known. The effect is to provide more precise information in the correct process context for decision making by the process manager.

In this paper, we first describe the programming process and places at which inspections are important. Then we discuss factors that affect productivity and the operations involved with inspections. Finally, we compare inspections and walk-throughs on process control.

The process

**a
manageable
process**

A process may be described as a set of operations occurring in a definite sequence that operates on a given input and converts it to some desired output. A general statement of this kind is sufficient to convey the notion of the process. In a practical application, however, it is necessary to describe the input, output, internal processing, and processing times of a process in very specific terms if the process is to be executed and practical output is to be obtained.

In the programming development process, explicit requirement statements are necessary as input. The series of processing operations that act on this input must be placed in the correct sequence with one another, the output of each operation satisfying the input needs of the next operation. The output of the final operation is, of course, the explicitly required output in the form of a verified program. Thus, the objective of each processing operation is to receive a defined input and to produce a definite output that satisfies a specific set of exit criteria. (It goes without saying that each operation can be considered as a miniprocess itself.) A well-formed process can be thought of as a continuum of processing during which sequential sets of exit criteria are satisfied, the last set in the entire series requiring a well-defined end product. Such a process is not amorphous. It can be measured and controlled.

**exit
criteria**

Unambiguous, explicit, and universally accepted exit criteria would be perfect as process control checkpoints. It is frequently argued that universally agreed upon checkpoints are impossible in programming because all projects are different, etc. However, *all* projects do reach the point at which there is a project checkpoint. As it stands, any trackable unit of code achieving a clean compilation can be said to have satisfied a universal exit criterion or checkpoint in the process. Other checkpoints can also be selected, albeit on more arguable premises, but once the premises are agreed upon, the checkpoints become visible in most, if not all, projects. For example, there is a point at which the design of a program is considered complete. This point may be described as the level of detail to which a unit of design is reduced so that one design statement will materialize in an estimated three to 10 source code instructions (or, if desired, five to 20, for that matter). Whichever particular ratio is selected across a project, it provides a checkpoint for the process control of that project. In this way, suitable checkpoints may be selected throughout the development process and used in process management. (For more specific exit criteria see Reference 1.)

The cost of reworking errors in programs becomes higher the later they are reworked in the process, so every attempt should be made to find and fix errors as early in the process as possible. This cost has led to the use of the inspections described later and to the description of exit criteria which include assuring that all errors known at the end of the inspection of the new "clean-compilation" code, for example, have been correctly fixed. So, rework of all known errors up to a particular point must be complete before the associated checkpoint can be claimed to be met for any piece of code.

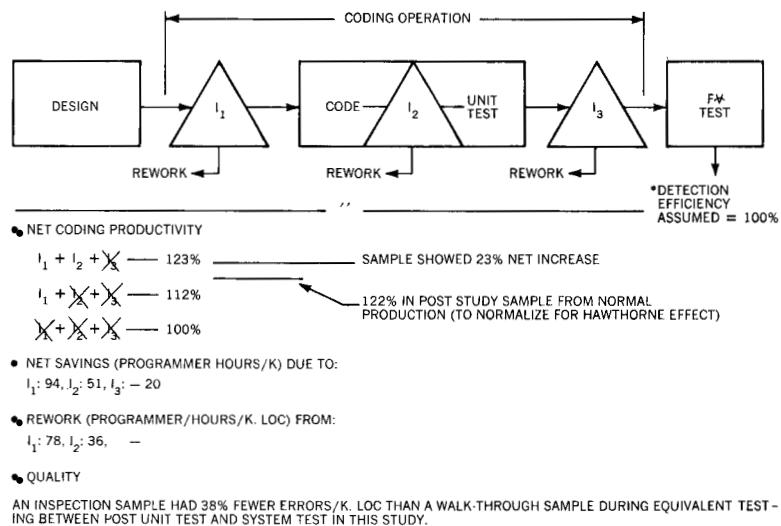
Where inspections are not used and errors are found during development or testing, the cost of rework as a fraction of overall development cost can be surprisingly high. For this reason, errors should be found and fixed as close to their place of origin as possible.

Production studies have validated the expected quality and productivity improvements and have provided estimates of standard productivity rates, percentage improvements due to inspections, and percentage improvements in error rates which are applicable in the context of large-scale operating system program production. (The data related to operating system development contained herein reflect results achieved by IBM in applying the subject processes and methods to representative samples. Since the results depend on many factors, they cannot be considered representative of every situation. They are furnished merely for the purpose of illustrating what has been achieved in sample testing.)

The purpose of the test plan inspection IT_1 , shown in Figure 1, is to find voids in the functional variation coverage and other discrepancies in the test plan. IT_2 , test case inspection of the test cases, which are based on the test plan, finds errors in the test cases. The total effects of IT_1 and IT_2 are to increase the integrity of testing and, hence, the quality of the completed product. And, because there are less errors in the test cases to be debugged during the testing phase, the overall project schedule is also improved.

A process of the kind depicted in Figure 1 installs all the intrinsic programming properties in the product as required in the statement of objectives (Level 0) by the time the coding operation (Level 5) has been completed—except for packaging and publications requirements. With these exceptions, all later work is of a verification nature. This verification of the product provides no contribution to the product during the essential development (Levels 1 to 5); it only adds error detection and elimination (frequently at one half of the development cost). I_0 , I_1 , and I_2 inspections were developed to measure and influence intrinsic

Figure 2 A study of coding productivity



quality (error content) in the early levels, where error rework can be most economically accomplished. Naturally, the beneficial effect on quality is also felt in later operations of the development process and at the end user's site.

An improvement in productivity is the most immediate effect of purging errors from the product by the I_0 , I_1 , and I_2 inspections. This purging allows rework of these errors very near their origin, early in the process. Rework done at these levels is 10 to 100 times less expensive than if it is done in the last half of the process. Since rework detracts from productive effort, it reduces productivity in proportion to the time taken to accomplish the rework. It follows, then, that finding errors by inspection and reworking them earlier in the process reduces the overall rework time and increases productivity even within the early operations and even more over the total process. Since less errors ship with the product, the time taken for the user to install programs is less, and his productivity is also increased.

The quality of documentation that describes the program is of as much importance as the program itself for poor quality can mislead the user, causing him to make errors quite as important as errors in the program. For this reason, the quality of program documentation is verified by publications inspections (PI_0 , PI_1 , and PI_2). Through a reduction of user-encountered errors, these inspections also have the effect of improving user productivity by reducing his rework time.

A study of coding productivity

A piece of the design of a large operating system component (all done in structured programming) was selected as a study sample (Figure 2). The sample was judged to be of moderate complexity. When the piece of design had been reduced to a level of detail sufficient to meet the Design Level 4 exit criteria² (a level of detail of design at which one design statement would ultimately appear as three to 10 code instructions), it was submitted to a design-complete inspection (100 percent), I₁. On conclusion of I₁, all error rework resulting from the inspection was completed, and the design was submitted for coding in PL/S. The coding was then done, and when the code was brought to the level of the first clean compilation,² it was subjected to a code inspection (100 percent), I₂. The resultant rework was completed and the code was subjected to unit test. After unit test, a unit test inspection, I₃, was done to see that the unit test plan had been fully executed. Some rework was required and the necessary changes were made. This step completed the coding operation. The study sample was then passed on to later process operations consisting of building and testing.

The inspection sample was considered of sufficient size and nature to be representative for study purposes. Three programmers designed it, and it was coded by 13 programmers. The inspection sample was in modular form, was structured, and was judged to be of moderate complexity on average.

Because errors were identified and corrected in groups at I₁ and I₂, rather than found one-by-one during subsequent work and handled at the higher cost incumbent in later rework, the overall amount of error rework was minimized, even within the coding operation. Expressed differently, considering the inclusion of *all* I₁ time, I₂ time, and resulting error rework time (with the usual coding and unit test time in the total time to complete the operation), a *net* saving resulted when this figure was compared to the no-inspection case. This net saving translated into a 23 percent increase in the productivity of the coding operation alone. Productivity in later levels was also increased because there was less error rework in these levels due to the effect of inspections, but the increase was not measured directly.

An important aspect to consider in any production experiment involving human beings is the Hawthorne Effect.³ If this effect is not adequately handled, it is never clear whether the effect observed is due to the human bias of the Hawthorne Effect or due to the newly implemented change in process. In this case a *control sample* was selected at random from many pieces of work *after the I₁ and I₂ inspections were accepted as commonplace*. (Previous experience without I₁ and I₂ approximated the net cod-

inspection
sample

coding
operation
productivity

ing productivity rate of 100 percent datum in Figure 2.) The difference in coding productivity between the experimental sample (with I_1 and I_2 for the first time) and the control sample was 0.9 percent. This difference is not considered significant. Therefore, the measured increase in coding productivity of 23 percent is considered to validly accrue from the only change in the process: addition of I_1 and I_2 inspections.

control sample The control sample was also considered to be of representative size and was from the same operating system component as the study sample. It was designed by four programmers and was coded by seven programmers. And it was considered to be of moderate complexity on average.

net savings Within the coding operation only, the net savings (including inspection and rework time) in programmer hours per 1000 Non-Commentary Source Statements (K.NCSS)⁴ were I_1 : 94, I_2 : 51, and I_3 : -20. As a consequence, I_3 is no longer in effect.

If personal fatigue and downtime of 15 percent are allowed in addition to the 145 programmer hours per K.NCSS, the saving approaches one programmer month per K.NCSS (assuming that our sample was truly representative of the rest of the work in the operating system component considered).

error rework The error rework in programmer hours per K.NCSS found in this study due to I_1 was 78, and 36 for I_2 (24 hours for design errors and 12 for code errors). Time for error rework must be specifically scheduled. (For scheduling purposes it is best to develop rework hours per K.NCSS from history depending upon the particular project types and environments, but figures of 20 hours for I_1 , and 16 hours for I_2 (*after the learning curve*) may be suitable to start with.)

quality The only comparative measure of quality obtained was a comparison of the inspection study sample with a fully comparable piece of the operating system component that was produced similarly, except that walk-throughs were used in place of the I_1 and I_2 inspections. (Walk-throughs⁵ were the practice before implementation of I_1 and I_2 inspections.) The process span in which the quality comparison was made was seven months of testing beyond unit test after which it was judged that both samples had been equally exercised. The results showed the inspection sample to contain 38 percent less errors than the walk-through sample.

Note that up to inspection I_2 , no machine time has been used for debugging, and so machine time savings were not mentioned. Although substantial machine time is saved overall since there are less errors to test for in inspected code in later stages of the process, no actual measures were obtained.

Table 1 Error detection efficiency

<i>Process Operations</i>	<i>Errors Found per K.NCSS</i>	<i>Percent of Total Errors Found</i>
Design		
I ₁ inspection	38*	82
Coding		
I ₂ inspection		
Unit test		
Preparation for acceptance test	8	18
Acceptance test	0	
Actual usage (6 mo.)	0	
Total	46	100

*51% were logic errors, most of which were missing rather than due to incorrect design.

In the development of applications, inspections also make a significant impact. For example, an application program of eight modules was written in COBOL by Aetna Corporate Data Processing department, Aetna Life and Casualty, Hartford, Connecticut, in June 1975.⁶ Two programmers developed the program. The number of inspection participants ranged between three and five. The only change introduced in the development process was the I₁ and I₂ inspections. The program size was 4,439 Non-Commentary Source Statements.

An automated estimating program, which is used to produce the normal program development time estimates for all the Corporate Data Processing department's projects, predicted that designing, coding, and unit testing this project would require 62 programmer days. In fact, the time actually taken was 46.5 programmer days including inspection meeting time. The resulting saving in programmer resources was 25 percent.

The inspections were obviously very thorough when judged by the inspection error detection efficiency of 82 percent and the later results during testing and usage as shown in Table 1.

The results achieved in Non-Commentary Source Statements per Elapsed Hour are shown in Table 2. These inspection rates are four to six times faster than for systems programming. If these rates are generally applicable, they would have the effect of making the inspection of applications programs much less expensive.

inspections in applications development

Inspections

Inspections are a *formal, efficient, and economical* method of finding errors in design and code. All instructions are addressed

Table 2 Inspection rates in NCSS per hour

<i>Operations</i>	<i>I₁</i>	<i>I₂</i>
Preparation	898	709
Inspection	652	539

Table 3. Inspection process and rate of progress

Process operations	Rate of progress*(loc/hr)		Objectives of the operation
	Design I_1	Code I_2	
1. Overview	500	not necessary	Communication education
2. Preparation	100	125	Education
3. Inspection	130	150	Find errors
4. Rework	20	16	Rework and resolve errors found by inspection
	hrs/K.NCSS	hrs/K.NCSS	
5. Follow-up	—	—	See that all errors, problems, and concerns have been resolved

*These notes apply to systems programming and are conservative. Comparable rates for applications programming are much higher. Initial schedules may be started with these numbers and as project history that is keyed to unique environments evolves, the historical data may be used for future scheduling algorithms.

at least once in the conduct of inspections. Key aspects of inspections are exposed in the following text through describing the I_1 and I_2 inspection conduct and process. I_0 , IT_1 , IT_2 , PI_0 , PI_1 , and PI_2 inspections retain the same essential properties as the I_1 and I_2 inspections but differ in materials inspected, number of participants, and some other minor points.

the people involved

The inspection team is best served when its members play their particular roles, assuming the particular vantage point of those roles. These roles are described below:

1. *Moderator*—The key person in a successful inspection. He must be a competent programmer but need not be a technical expert on the program being inspected. To preserve objectivity and to increase the integrity of the inspection, it is usually advantageous to use a moderator from an unrelated project. The moderator must manage the inspection team and offer leadership. Hence, he must use personal sensitivity, tact, and drive in balanced measure. His use of the strengths of team members should produce a synergistic effect larger than their number; in other words, *he is the coach*. The duties of moderator also include scheduling suitable meeting places, reporting inspection results within one day, and follow-up on rework. *For best results the moderator should be specially trained.* (This training is brief but very advantageous.)
2. *Designer*—The programmer responsible for producing the program design.
3. *Coder/Implementor*—The programmer responsible for translating the design into code.
4. *Tester*—The programmer responsible for writing and/or executing test cases or otherwise testing the product of the designer and coder.

If the coder of a piece of code also designed it, he will function in the designer role for the inspection process; a coder from some related or similar program will perform the role of the coder. If the same person designs, codes, and tests the product code, the coder role should be filled as described above, and another coder—preferably with testing experience—should fill the role of tester.

Four people constitute a good-sized inspection team, although circumstances may dictate otherwise. The team size should not be artificially increased over four, but if the subject code is involved in a number of interfaces, the programmers of code related to these interfaces may profitably be involved in inspection. Table 3 indicates the inspection process and rate of progress.

The total time to complete the inspection process from overview through follow-up for I_1 or I_2 inspections with four people involved takes about 90 to 100 people-hours for systems programming. Again, these figures may be considered conservative but they will serve as a starting point. Comparable figures for applications programming tend to be much lower, implying lower cost per K.NCSS.

Because the error detection efficiency of most inspection teams tends to dwindle after two hours of inspection but then picks up after a period of different activity, it is advisable to schedule inspection sessions of no more than two hours at a time. Two two-hour sessions per day are acceptable.

The time to do inspections and resulting rework must be scheduled and managed with the same attention as other important project activities. (After all, as is noted later, for one case at least, it is possible to find approximately two thirds of the errors reported during an inspection.) If this is not done, the immediate work pressure has a tendency to push the inspections and/or rework into the background, postponing them or avoiding them altogether. The result of this short-term respite will obviously have a much more dramatic long-term negative effect since the finding and fixing of errors is delayed until later in the process (and after turnover to the user). Usually, the result of postponing early error detection is a lengthening of the overall schedule and increased product cost.

Scheduling inspection time for modified code may be based on the algorithms in Table 3 and on judgment.

Keeping the objective of each operation in the forefront of team activity is of paramount importance. Here is presented an outline of the I_1 inspection process operations.

scheduling inspections and rework

I_1 inspection process

Figure 3 Summary of design inspections by error type

VP Individual Name	Inspection file				
	Missing	Wrong	Extra	Errors	Error %
CD CB Definition	16	2		18	3.5
CU CB Usage	18	17	1	36	6.9
FS FPFS	1			1	.2
IC Interconnect Calls	18	9		27	5.2
IR Interconnect Reqts	4	5	2	11	2.1
LO Logic	126	57	24	207	39.8
L3 Higher Lvl Docu	1		1	2	.4
MA Mod Attributes	1			1	.2
MD More Detail	24	6	2	32	6.2
MN Maintainability	8	5	3	16	3.1
OT Other	15	10	10	35	6.7
PD Pass Data Areas		1		1	.2
PE Performance	1	2	3	6	1.2
PR Prologue/Prose	44	38	7	89	17.1
RM Return Code/Msg	5	7	2	14	2.7
RU Register Usage	1	2		3	.6
ST Standards					
TB Test & Branch	12	7	2	21	4.0
	295	168	57	520	100.0
	57%	32%	11%		

Figure 4 Summary of code inspections by error type

VP Individual Name	Inspection file				
	Missing	Wrong	Extra	Errors	Error %
CC Code Comments	5	17	1	23	6.6
CU CB Usage	3	21	1	25	7.2
DE Design Error	31	32	14	77	22.1
F1		8		8	2.3
IR Interconnect Calls	7	9	3	19	5.5
LO Logic	33	49	10	92	26.4
MN Maintainability	5	7	2	14	4.0
OT Other					
PE Performance	3	2	5	10	2.9
PR Prologue/Prose	25	24	3	52	14.9
PU PL/S or BAL Use	4	9	1	14	4.0
RU Register Usage	4	2		6	1.7
SU Storage Usage	1			1	.3
TB Test & Branch	2	5		7	2.0
	123	185	40	348	100.0

1. Overview (whole team) — The designer first describes the overall area being addressed and then the specific area he has designed in detail—logic, paths, dependencies, etc. Documentation of design is distributed to all inspection participants on conclusion of the overview. (For an I₂ inspection, no overview is necessary, but the participants should remain the same. Preparation, inspection, and follow-up proceed as for I₁, but, of course, using code listings and design specifications

as inspection materials. Also, at I_2 the moderator should flag for special scrutiny those areas that were reworked since I_1 errors were found and other design changes made.)

2. *Preparation* (individual) — Participants, using the design documentation, literally do their homework to try to understand the design, its intent and logic. (Sometimes flagrant errors are found during this operation, but in general, the number of errors found is not nearly as high as in the inspection operation.) To increase their error detection in the inspection, the inspection team should first study the ranked distributions of error types found by recent inspections. This study will prompt them to concentrate on the most fruitful areas. (See examples in Figures 3 and 4.) Checklists of clues on finding these errors should also be studied. (See partial examples of these lists in Figures 5 and 6 and complete examples for I_0 in Reference 1 and for I_1 and I_2 in Reference 7.)
3. *Inspection* (whole team) — A “reader” chosen by the moderator (usually the coder) describes how he will implement the design. He is expected to paraphrase the design as expressed by the designer. Every piece of logic is covered at least once, and every branch is taken at least once. All higher-level documentation, high-level design specifications, logic specifications, etc., and macro and control block listings at I_2 must be available and present during the inspection.

Now that the design is understood, *the objective is to find errors*. (Note that an error is defined as any condition that causes malfunction or that precludes the attainment of expected or previously specified results. Thus, deviations from specifications are clearly termed errors.) The finding of errors is actually done during the implementor/coder’s discourse. Questions raised are pursued only to the point at which an error is recognized. It is noted by the moderator; its type is classified; severity (major or minor) is identified, and the inspection is continued. Often the solution of a problem is obvious. If so, it is noted, but no specific solution hunting is to take place during inspection. (The inspection is *not* intended to redesign, evaluate alternate design solutions, or to find solutions to errors; it is intended just to find errors!) A team is most effective if it operates with only one objective at a time.

Within one day of conclusion of the inspection, the moderator should produce a written report of the inspection and its findings to ensure that all issues raised in the inspection will be addressed in the rework and follow-up operations. Examples of these reports are given as Figures 7A, 7B, and 7C.

Figure 5 Examples of what to examine when looking for errors at I₁

I₁ Logic

Missing

1. Are All Constants Defined?
2. Are All Unique Values Explicitly Tested on Input Parameters?
3. Are Values Stored after They Are Calculated?
4. Are All Defaults Checked Explicitly Tested on Input Parameters?
5. If Character Strings Are Created Are They Complete, Are All Delimiters Shown?
6. If a Keyword Has Many Unique Values, Are They All Checked?
7. If a Queue Is Being Manipulated, Can the Execution Be Interrupted; If So, Is Queue Protected by a Locking Structure; Can Queue Be Destroyed Over an Interrupt?
8. Are Registers Being Restored on Exits?
9. In Queuing/Dequeueing Should Any Value Be Decremented/Incremented?
10. Are All Keywords Tested in Macro?
11. Are All Keyword Related Parameters Tested in Service Routine?
12. Are Queues Being Held in Isolation So That Subsequent Interrupting Requestors Are Receiving Spurious Returns Regarding the Held Queue?
13. Should any Registers Be Saved on Entry?
14. Are All Increment Counts Properly Initialized (0 or 1)?

Wrong

1. Are Absolutes Shown Where There Should Be Symbolics?
2. On Comparison of Two Bytes, Should All Bits Be Compared?
3. On Built Data Strings, Should They Be Character or Hex?
4. Are Internal Variables Unique or Confusing If Concatenated?

Extra

1. Are All Blocks Shown in Design Necessary or Are They Extraneous?

4. *Rework*—All errors or problems noted in the inspection report are resolved by the designer or coder/implementor.

5. *Follow-Up*—It is imperative that every issue, concern, and error be entirely resolved at this level, or errors that result can be 10 to 100 times more expensive to fix if found later in the process (programmer time only, machine time not included). It is the responsibility of the moderator to see that all issues, problems, and concerns discovered in the inspection operation have been resolved by the designer in the case of I₁, or the coder/implementor for I₂ inspections. If more than five percent of the material has been reworked, the team should reconvene and carry out a 100 percent reinspection. Where less than five percent of the material has been reworked, the moderator at his discretion may verify the quality of the rework himself or reconvene the team to reinspect either the complete work or just the rework.

**commencing
inspections**

In Operation 3 above, it is one thing to direct people to find errors in design or code. It is quite another problem for them to find errors. Numerous experiences have shown that people have to be taught or prompted to find errors effectively. Therefore, it

Figure 6 Examples of what to examine when looking for errors at I₂

INSPECTION SPECIFICATION

I₂ Test Branch

- Is Correct Condition Tested (If X = ON vs. IF X = OFF)?
- Is (Are) Correct Variable(s) Used for Test
- (If X = ON vs. If Y = ON)?
- Are Null THENS/ELSEs Included as Appropriate?
- Is Each Branch Target Correct?
- Is the Most Frequently Exercised Test Leg the THEN Clause?

I₂ Interconnection (or Linkage) Calls

- For Each Interconnection Call to Either a Macro, SVC or Another Module:
 - Are All Required Parameters Passed Set Correctly?
 - If Register Parameters Are Used, Is the Correct Register Number Specified?
 - If Interconnection Is a Macro,
 - Does the Inline Expansion Contain All Required Code?
 - No Register or Storage Conflicts between Macro and Calling Module?
 - If the Interconnection Returns, Do All Returned Parameters Get Processed Correctly?

is prudent to condition them to seek the high-occurrence, high-cost error types (see example in Figures 3 and 4), and then describe the clues that usually betray the presence of each error type (see examples in Figures 5 and 6).

One approach to getting started may be to make a preliminary inspection of a design or code that is felt to be representative of the program to be inspected. Obtain a suitable quantity of errors, and analyze them by type and origin, cause, and salient indicative clues. With this information, an inspection specification may be constructed. This specification can be amended and improved in light of new experience and serve as an on-going directive to focus the attention and conduct of inspection teams. The objective of an inspection specification is to help maximize and make more consistent the error detection efficiency of inspections where

Error detection efficiency

$$= \frac{\text{Errors found by an inspection}}{\text{Total errors in the product before inspection}} \times 100$$

The reporting forms and form completion instructions shown in the Appendix may be used for I₁ and I₂ inspections. Although these forms were constructed for use in systems programming development, they may be used for applications programming development with minor modification to suit particular environments.

The moderator will make hand-written notes recording errors found during inspection meetings. He will categorize the errors

**reporting
inspection
results**

Figure 7A Error list

1. PR/M/MIN Line 3: the statement of the prologue in the REMARKS section needs expansion.
2. DA/W/MAJ Line 123: ERR-RECORD-TYPE is out of sequence.
3. PU/W/MAJ Line 147: the wrong bytes of an 8-byte field (current-data) are moved into the 2-byte field (this year).
4. LO/W/MAJ Line 169: while counting the number of leading spaces in NAME, the wrong variable (I) is used to calculate "J".
5. LO/W/MAJ Line 172: NAME-CHECK is PERFORMED one time too few.
6. PU/E/MIN Line 175: In NAME-CHECK, the check for SPACE is redundant.
7. DE/W/MIN Line 175: the design should allow for the occurrence of a period in a last name.

Figure 7B Example of module detail report

MOD/MAC:	CHECKER	CODE INSPECTION REPORT							
		MODULE DETAIL							
		SUBCOMPONENT/APPLICATION							
SEE NOTE BELOW									
PROBLEM TYPE:	MAJOR*						MINOR		
	M	W	E	M	W	E			
		9					1		
	LD: LOGIC								
	TB: TEST AND BRANCH								
	EL: EXTERNAL LINKAGES								
	RU: REGISTER USAGE								
	SU: STORAGE USAGE								
	DA: DATA AREA USAGE			2					
	PU: PROGRAM LANGUAGE		2				1		
	PE: PERFORMANCE								
	MN: MAINTAINABILITY						1		
	DE: DESIGN ERROR						1		
	PR: PROLOGUE					1			
	CC: CODE COMMENTS								
OT: OTHER									
TOTAL:			13			5			

REINSPECTION REQUIRED? Y

*A PROBLEM WHICH WOULD CAUSE THE PROGRAM TO MALFUNCTION; A BUG. M = MISSING, W = WRONG, E = EXTRA.
NOTE: FOR MODIFIED MODULES, PROBLEMS IN THE CHANGED PORTION VERSUS PROBLEMS IN THE BASE SHOULD BE SHOWN IN THIS MANNER: 3(2), WHERE 3 IS THE NUMBER OF PROBLEMS IN THE CHANGED PORTION AND 2 IS THE NUMBER OF PROBLEMS IN THE BASE

and then transcribe counts of the errors, by type, to the module detail form. By maintaining cumulative totals of the counts by error type, and dividing by the number of projected executable source lines of code inspected to date, he will be able to establish installation averages within a short time.

Figures 7A, 7B, and 7C are an example of a set of code inspection reports. Figure 7A is a partial list of errors found in code inspection. Notice that errors are described in detail and are classified by error type, whether due to something being missing,

Figure 7C Example of code inspection summary report

CODE INSPECTION REPORT SUMMARY												Date 11/20/-	
To: Design Manager	KRAUSS	Development Manager	GIOTTI										
Subject: Inspection Report for	CHECKER	Inspection date	11/19/-										
System/Application		Release	Build										
Component		Subcomponents(s)											
Mod/Mac Name Mod Name N	New or Part Insp. Full or Part Insp. McGINLEY	Programmer HALE	Tester 348	ELOC Added, Modified, Deleted						Inspection People-hours (X.X)			
				Pre-insp	Est Post	Rework	Prep	Insp	Meetg	Re-work	Follow-up	Sub-component	
				A	M	D	A	M	D	A	M	D	
Totals													
Reinspection required? YES Length of inspection (clock hours and tenths) 2.2 Reinspection by (date) 11/25/- Additional modules/macros? NO DCR #'s written C-2 Problem summary: Major 13 Minor 5 Total 18 Errors in changed code: Major Minor Errors in base code: Major Minor LARSON McGINLEY HALE Initial Desr Detailed Dr Programmer Team Leader Other				Moderator's Signature									

wrong, or extra as the cause, and according to major or minor severity. Figure 7B is a module level summary of the errors contained in the entire error list represented by Figure 7A. The code inspection summary report in Figure 7C is a summary of inspection results obtained on all modules inspected in a particular inspection session or in a subcomponent or application.

Inspections have been successfully applied to designs that are specified in English prose, flowcharts, HIPO, (Hierarchy plus Input-Process-Output) and PIDGEON (an English prose-like meta language).

The first code inspections were conducted on PL/S and Assembler. Now, prompting checklists for inspections of Assembler, COBOL, FORTRAN, and PL/I code are available.⁷

One of the most significant benefits of inspections is the detailed feedback of results on a relatively real-time basis. The programmer finds out what error types he is most prone to make and their quantity and how to find them. This feedback takes place within a few days of writing the program. Because he gets early indications from the first few units of his work inspected, he is able to show improvement, and usually does, on later work even during the same project. In this way, feedback of results from inspections must be counted for the programmer's use and benefit: *they should not under any circumstances be used for programmer performance appraisal.*

Skeptics may argue that once inspection results are obtained, they will or even must count in performance appraisals, or at

**inspections
and
languages**

**personnel
considerations**

Figure 8 Example of most error-prone modules based on I_1 and I_2

Module name	Number of errors	Lines of code	Error density, Errors/K. Loc
Echo	4	128	31
Zulu	10	323	31
Foxtrot	3	71	28
Alpha	7	264	27 ← Average
Lima	2	106	19 Error
Delta	3	195	15 Rate
:	:	:	:
	67		

least cause strong bias in the appraisal process. The author can offer in response that inspections have been conducted over the past three years involving diverse projects and locations, hundreds of experienced programmers and tens of managers, and so far he has found no case in which inspection results have been used negatively against programmers. Evidently no manager has tried to "kill the goose that lays the golden eggs."

A preinspection opinion of some programmers is that they do not see the value of inspections because they have managed very well up to now, or because their projects are too small or somehow different. This opinion usually changes after a few inspections to a position of acceptance. The quality of acceptance is related to the success of the inspections they have experienced, the *conduct of the trained moderator*, and the *attitude demonstrated by management*. The acceptance of inspections by programmers and managers as a beneficial step in making programs is well-established amongst those who have tried them.

Process control using inspection and testing results

Obviously, the range of analysis possible using inspection results is enormous. Therefore, only a few aspects will be treated here, and they are elementary expositions.

most error-prone modules

A listing of either I_1 , I_2 , or combined $I_1 + I_2$ data as in Figure 8 immediately highlights which modules contained the highest error density on inspection. If the error detection efficiency of each of the inspections was fairly constant, the ranking of error-prone modules holds. Thus if the error detection efficiency of inspection is 50 percent, and the inspection found 10 errors in a

Figure 9 Example of distribution of error types

	<i>Number of errors</i>	<i>%</i>	<i>Normal/usual distribution, %</i>
Logic	23	35	44
Interconnection/Linkage (Internal)	21	31 ?	18
Control Blocks	6	9	13
—	.	8	10
—	.	7	7
—	.	6	6
—	.	4	2
		100%	100%

module, then it can be estimated that there are 10 errors remaining in the module. This information can prompt many actions to control the process. For instance, in Figure 8, it may be decided to reinspect module "Echo" or to redesign and recode it entirely. Or, less drastically, it may be decided to test it "harder" than other modules and look especially for errors of the type found in the inspections.

If a ranked distribution of error types is obtained for a group of "error-prone modules" (Figure 9), which were produced from the same Process A, for example, it is a short step to comparing this distribution with a "Normal/Usual Percentage Distribution." Large disparities between the sample and "standard" will lead to questions on why Process A, say, yields nearly twice as many internal interconnection errors as the "standard" process. If this analysis is done promptly on the first five percent of production, it may be possible to remedy the problem (if it is a problem) on the remaining 95 percent of modules for a particular shipment. Provision can be made to test the first five percent of the modules to remove the unusually high incidence of internal interconnection problems.

Analysis of the testing results, commencing as soon as testing errors are evident, is a vital step in controlling the process since future testing can be guided by early results.

Where testing reveals excessively error-prone code, it may be more economical and saving of schedule to select the most error-prone code and inspect it before continuing testing. (The business case will likely differ from project to project and case to case, but in many instances inspection will be indicated). The selection of the most error-prone code may be made with two considerations uppermost:

distribution of error types

inspecting error-prone code

Table 4. Inspection and walk-through processes and objectives

Inspection		Walk-through	
Process Operations	Objectives	Process Operations	Objectives
1. Overview	Education (Group)	—	—
2. Preparation	Education (Individual)	1. Preparation	Education (Individual)
3. Inspection	Find errors! (Group)	2. Walk-through	Education (Group) Discuss design alternatives
4. Rework	Fix problems	—	Find errors
5. Follow-up	Ensure all fixes correctly installed	—	

Note the separation of objectives in the inspection process.

Table 5 Comparison of key properties of inspections and walk-throughs

Properties	Inspection	Walk-Through
1. Formal moderator training	Yes	No
2. Definite participant roles	Yes	No
3. Who "drives" the inspection or walk-through	Moderator	Owner of material (Designer or coder)
4. Use "How To Find Errors" checklists	Yes	No
5. Use distribution of error types to look for	Yes	No
6. Follow-up to reduce bad fixes	Yes	No
7. Less future errors because of detailed error feedback to individual programmer	Yes	Incidental
8. Improve inspection efficiency from analysis of results	Yes	No
9. Analysis of data → process problems → improvements	Yes	No

1. Which modules head a ranked list when the modules are rated by test errors per K.NCSS?
2. In the parts of the program in which test coverage is low, which modules or parts of modules are most suspect based on $(I_1 + I_2)$ errors per K.NCSS and programmer judgment?

From a condensed table of ranked "most error-prone" modules, a selection of modules to be inspected (or reinspected) may be made. Knowledge of the error types already found in these modules will better prepare an inspection team.

The reinspection itself should conform with the I₂ process, except that an overview may be necessary if the original overview was held too long ago or if new project members are involved.

Inspections and walk-throughs

Walk-throughs (or walk-thrus) are practiced in many different ways in different places, with varying regularity and thoroughness. This inconsistency causes the results of walk-throughs to vary widely and to be nonrepeatable. Inspections, however, having an established process and a formal procedure, tend to vary less and produce more repeatable results. Because of the variation in walk-throughs, a comparison between them and inspections is not simple. However, from Reference 8 and the walk-through procedures witnessed by the author and described to him by walk-through participants, as well as the inspection process described previously and in References 1 and 9, the comparison in Tables 4 and 5 is drawn.

Figure 10A describes the process in which a walk-through is applied. Clearly, the purging of errors from the product as it passes through the walk-through between Operations 1 and 2 is very beneficial to the product. In Figure 10B, the inspection process (and its feedback, feed-forward, and self-improvement) replaces the walk-through. The notes on the figure are self-explanatory.

Inspections are also an excellent means of measuring completeness of work against the exit criteria which must be satisfied to complete project checkpoints. (Each checkpoint should have a clearly defined set of exit criteria. Without exit criteria, a checkpoint is too negotiable to be useful for process control).

Inspections and process management

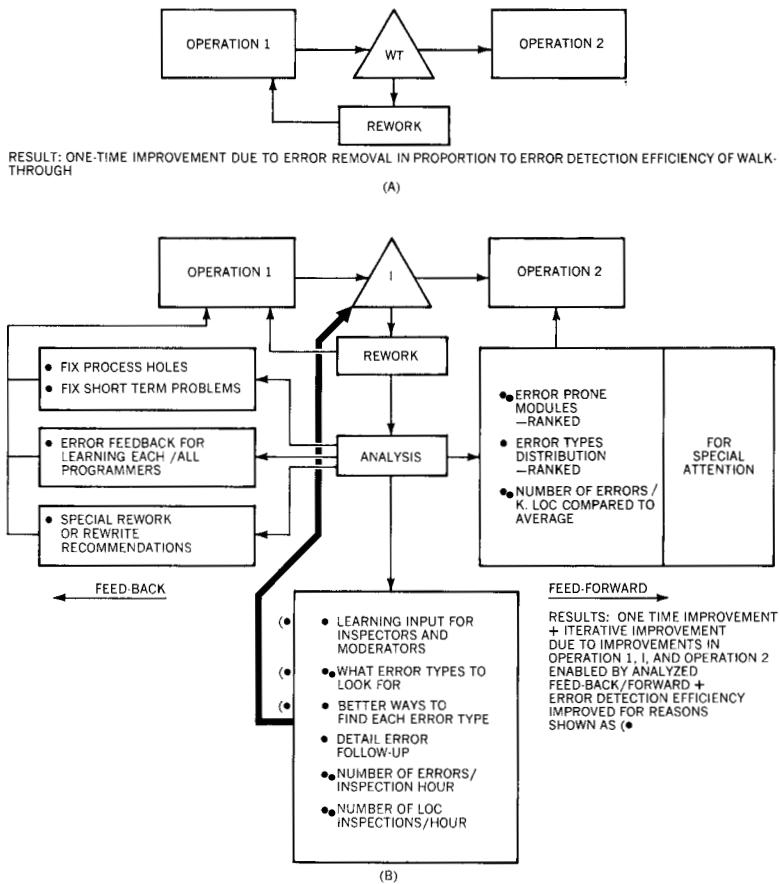
The most marked effects of inspections on the development process is to change the old adage that, "design is not complete until testing is completed," to a position where a very great deal must be known about the design before even the coding is begun. Although great discretion is still required in code implementation, more predictability and improvements in schedule, cost, and quality accrue. The old adage still holds true if one regards inspection as much a means of verification as testing.

Observations in one case in systems programming show that approximately two thirds of all errors reported during development are found by I₁ and I₂ inspections prior to machine testing.

effects on
development
process

percent of
errors found

Figure 10 (A) Walk-through process, (B) Inspection process



The error detection efficiencies of the I₁ and I₂ inspections separately are, of course, less than 66 percent. A similar observation of an application program development indicated an 82 percent find (Table 1). As more is learned and the error detection efficiency of inspection is increased, the burden of debugging on testing operations will be reduced, and testing will be more able to fulfill its prime objective of verifying quality.

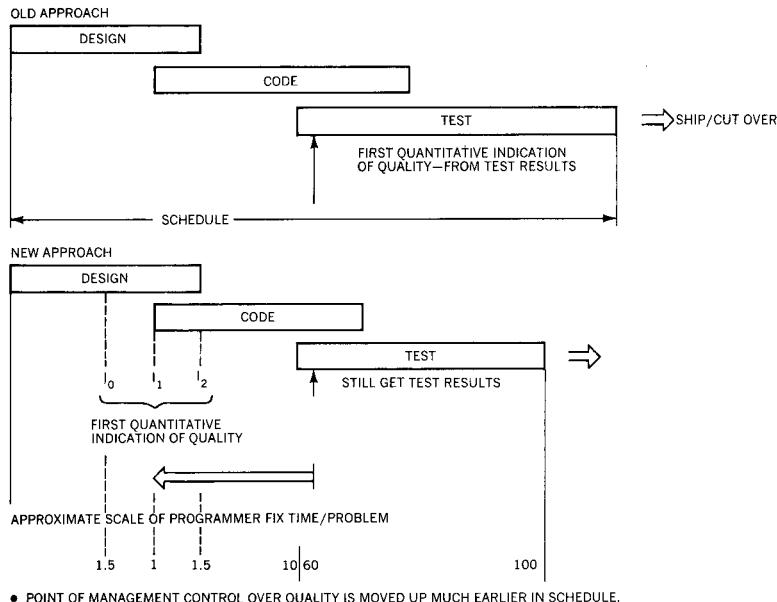
effect on cost and schedule

Comparing the "old" and "new" (with inspections) approaches to process management in Figure 11, we can see clearly that with the use of inspection results, error rework (which is a very significant variable in product cost) tends to be managed more during the first half of the schedule. This results in much lower cost than in the "old" approach, where the cost of error rework was 10 to 100 times higher and was accomplished in large part during the last half of the schedule.

process tracking

Inserting the I₁ and I₂ checkpoints in the development process enables assessment of project completeness and quality to be

Figure 11 Effect of inspection on process management



made early in the process (during the first half of the project instead of the latter half of the schedule, when recovery may be impossible without adjustments in schedule and cost). Since individually trackable modules of reasonably well-known size can be counted as they pass through each of these checkpoints, the percentage completion of the project against schedule can be continuously and easily tracked.

The overview, preparation, and inspection sequence of the operations of the inspection process give the inspection participants a high degree of product knowledge in a very short time. This important side benefit results in the participants being able to handle later development and testing with more certainty and less false starts. Naturally, this also contributes to productivity improvement.

An interesting sidelight is that because designers are asked at pre-I₁ inspection time for estimates of the number of lines of code (NCSS) that their designs will create, and they are present to count for themselves the actual lines of code at the I₂ inspection, the accuracy of design estimates has shown substantial improvement.

For this reason, an inspection is frequently a required event where responsibility for design or code is being transferred from

effect on product knowledge

one programmer to another. The complete inspection team is convened for such an inspection. (One-on-one reviews such as desk debugging are certainly worthwhile but do not approach the effectiveness of formal inspection.) Usually the side benefit of finding errors more than justifies the transfer inspection.

**inspecting
modified
code**

Code that is changed in, or inserted in, an existing module either in replacement of deleted code or simply inserted in the module is considered modified code. By this definition, a very large part of programming effort is devoted to modifying code. (The addition of entirely new modules to a system count as new, not modified, code.)

Some observations of errors per K.NCSS of modified code show its error rate to be considerably higher than is found in new code; (i.e., if 10.NCSS are replaced in a 100.NCSS module and errors against the 10.NCSS are counted, the error rate is described as number of errors per 10.NCSS, not number of errors per 100.NCSS). Obviously, if the number of errors in modified code are used to derive an error rate per K.NCSS for the whole module that was modified, this rate would be largely dependent upon the percentage of the module that is modified: this would provide a meaningless ratio. A useful measure is the number of errors per K.NCSS (modified) in which the higher error rates have been observed.

Since most modifications are small (e.g., 1 to 25 instructions), they are often erroneously regarded as trivially simple and are handled accordingly; the error rate goes up, and control is lost. In the author's experience, *all* modifications are well worth inspecting from an economic and a quality standpoint. A convenient method of handling changes is to group them to a module or set of modules and convene the inspection team to inspect as many changes as possible. But all changes must be inspected!

Inspections of modifications can range from inspecting the modified instructions and the surrounding instructions connecting it with its host module, to an inspection of the entire module. The choice of extent of inspection coverage is dependent upon the percentage of modification, pervasiveness of the modification, etc.

**bad
fixes**

A very serious problem is the inclusion in the product of bad fixes. Human tendency is to consider the "fix," or correction, to a problem to be error-free itself. Unfortunately, this is all too frequently untrue in the case of fixes to errors found by inspections and by testing. The inspection process clearly has an operation called Follow-Up to try and minimize the bad-fix problem, but the fix process of testing errors very rarely requires scrutiny of fix quality before the fix is inserted. Then, if the fix is bad, the whole elaborate process of going from source fix to link edit, to

test the fix, to regression test must be repeated at needlessly high cost. The number of bad fixes can be economically reduced by some simple inspection after clean compilation of the fix.

Summary

We can summarize the discussion of design and code inspections and process control in developing programs as follows:

1. Describe the program development process in terms of operations, and define exit criteria which must be satisfied for completion of each operation.
2. Separate the objectives of the inspection process operations to keep the inspection team focused on one objective at a time:

<i>Operation</i>	<i>Objective</i>
Overview	Communications/education
Preparation	Education
Inspection	Find errors
Rework	Fix errors
Follow-up	Ensure all fixes are applied correctly
3. Classify errors by type, and rank frequency of occurrence of types. Identify *which types* to spend most time looking for in the inspection.
4. Describe *how* to look for presence of error types.
5. Analyze inspection results and use for constant process improvement (until process averages are reached and then use for process control).

Some applications of inspections include function level inspections I_0 , design-complete inspections I_1 , code inspections I_2 , test plan inspections IT_1 , test case inspections IT_2 , interconnections inspections IF , inspection of fixes/changes, inspection of publications, etc., and post testing inspection. Inspections can be applied to the development of system control programs, applications programs, and microcode in hardware.

We can conclude from experience that inspections increase productivity and improve final program quality. Furthermore, improvements in process control and project management are enabled by inspections.

ACKNOWLEDGMENTS

The author acknowledges, with thanks, the work of Mr. O. R. Kohli and Mr. R. A. Radice, who made considerable contributions in the development of inspection techniques applied to program design and code, and Mr. R. R. Larson, who adapted inspections to program testing.

Figure 12 Design inspection module detail form

DETAILED DESIGN INSPECTION REPORT					
MODULE DETAIL					
MOD/MAC:	SUBCOMPONENT/APPLICATION				
SEE NOTE BELOW					
PROBLEM TYPE:	MAJOR*			MINOR	
	M	W	E	M	W
LO: LOGIC					
TB: TEST AND BRANCH					
DA: DATA AREA USAGE					
RM: RETURN CODES/MESSAGES					
RU: REGISTER USAGE					
MA: MODULE ATTRIBUTES					
EL: EXTERNAL LINKAGES					
MD: MORE DETAIL					
ST: STANDARDS					
PR: PROLOGUE OR PROSE					
HL: HIGHER LEVEL DESIGN DOC.					
US: USER SPEC.					
MN: MAINTAINABILITY					
PE: PERFORMANCE					
OT: OTHER					
TOTAL:					

REINSPECTION REQUIRED?

*A PROBLEM WHICH INDIVIDUALLY CAUSES THE PROGRAM TO MALFUNCTION A BUG. M = MISSING, W = WRONG, E = EXTRA.
NOTE: FOR MODIFIED MODULES, PROBLEMS IN THE CHANGED PORTION VERSUS PROBLEMS IN THE BASE SHOULD BE SHOWN IN THIS MANNER: 3(2), WHERE 3 IS THE NUMBER OF PROBLEMS IN THE CHANGED PORTION AND 2 IS THE NUMBER OF PROBLEMS IN THE BASE.

CITED REFERENCES AND FOOTNOTES

1. O. R. Kohli, *High-Level Design Inspection Specification*, Technical Report TR 21.601, IBM Corporation, Kingston, New York (July 21, 1975).
2. It should be noted that the exit criteria for I_1 (design complete where one design statement is estimated to represent 3 to 10 code instructions) and I_2 (first clean code compilations) are checkpoints in the development process through which every programming project must pass.
3. The Hawthorne Effect is a psychological phenomenon usually experienced in human-involved productivity studies. The effect is manifested by participants producing above normal because they know they are being studied.
4. NCSS (Non-Commentary Source Statements), also referred to as "Lines of Code," are the sum of executable code instructions and declaratives. Instructions that invoke macros are counted once only. Expanded macroinstructions are also counted only once. Comments are not included.
5. Basically in a walk-through, program design or code is reviewed by a group of people gathered together at a structured meeting in which errors/issues pertaining to the material and proposed by the participants may be discussed in an effort to find errors. The group may consist of various participants but always includes the originator of the material being reviewed who usually plans the meeting and is responsible for correcting the errors. How it differs from an inspection is pointed out in Tables 2 and 3.
6. *Marketing Newsletter*, Cross Application Systems Marketing, "Program inspections at Aetna," MS-76-006, S2, IBM Corporation, Data Processing Division, White Plains, New York (March 29, 1976).

7. J. Ascoly, M. J. Cafferty, S. J. Gruen, and O. R. Kohli, *Code Inspection Specification*, Technical Report TR 21.630, IBM Corporation, Kingston, New York (1976).
8. N. S. Waldstein, *The Walk-Thru—A Method of Specification, Design and Review*, Technical Report TR 00.2536, IBM Corporation, Poughkeepsie, New York (June 4, 1974).
9. Independent study programs: *IBM Structured Programming Textbook*, SR20-7149-1, *IBM Structured Programming Workbook*, SR20-7150-0, IBM Corporation, Data Processing Division, White Plains, New York.

GENERAL REFERENCES

1. J. D. Aron, *The Program Development Process: Part I: The Individual Programmer*, Structured Programs, 137–141, Addison-Wesley Publishing Co., Reading, Massachusetts (1974).
2. M. E. Fagan, *Design and Code Inspections and Process Control in the Development of Programs*, Technical Report TR 00.2763, IBM Corporation, Poughkeepsie, New York (June 10, 1976). This report is a revision of the author's *Design and Code Inspections and Process Control in the Development of Programs*, Technical Report TR 21.572, IBM Corporation, Kingston, New York (December 17, 1974).
3. O. R. Kohli and R. A. Radice, *Low-Level Design Inspection Specification*, Technical Report TR 21.629, IBM Corporation, Kingston, New York (1976).
4. R. R. Larson, *Test Plan and Test Case Inspection Specifications*, Technical Report TR 21.586, IBM Corporation, Kingston, New York (April 4, 1975).

Appendix: Reporting forms and form completion instructions

Instructions for Completing Design Inspection Module Detail Form

This form (Figure 12) should be completed for each module/macros that has valid problems against it. The problem-type information gathered in this report is important because a history of problem-type experience points out high-occurrence types. This knowledge can then be conveyed to inspectors so that they can concentrate on seeking the higher-occurrence types of problems.

1. MOD/MAC: The module or macro name.
2. SUBCOMPONENT: The associated subcomponent.
3. PROBLEM TYPE: Summarize the number of problems by type (logic, etc.), severity (major/minor), and by category (missing, wrong, or extra). For modified modules, detail the number of problems in the changed design versus the number in the base design. (Problem types were developed in a systems programming environment. Appropriate changes, if desired, could be made for application development.)

Figure 13 Design inspection summary form

DESIGN INSPECTION REPORT SUMMARY												Date _____		
To:	Design Manager _____			Development Manager _____										
Subject:	Inspection Report for _____			Inspection date _____										
System/Application _____				Release _____			Build _____							
Component _____				Subcomponents(s) _____										
Mod/Mac Name	New or Mod	Full or Part Insp.	Detailed Designer	Programmer	ELOC Added, Modified, Deleted						Inspection People-hours (X.X.)			
					Est. Pre	Est. Post	Rework	Over-view & Prep.	Insp Meetg	Re-work	Follow-up	Sub-component		
A	M	D	A	M	D	A	M	D						
				Totals										
										Reinspection required? _____ Length of inspection (clock hours and tenths) _____				
										Reinspection by (date) _____ Additional modules/macros? _____				
										DCR #'s written _____				
										Problem summary: Major _____ Minor _____ Total _____				
										Errors in changed code: Major _____ Minor _____ Errors in base code: Major _____ Minor _____				
Initial Desr	Detailed Dr	Programmer	Team Leader			Other			Moderator's Signature					

4. REINSPECTION REQUIRED?: Indicate whether the module/module macro requires a reinspection.

All valid problems found in the inspection should be listed and attached to the report. A brief description of each problem, its error type, and the rework time to fix it should be given (see Figure 7A, which describes errors in similar detail to that required but is at a coding level).

Instructions for Completing Design Inspection Summary Form

Following are detailed instructions for completing the form in Figure 13.

1. TO: The report is addressed to the respective design and development managers.
2. SUBJECT: The unit being inspected is identified.
3. MOD/MAC NAME: The name of each module and macro as it resides on the source library.
4. NEW OR MOD: "N" if the module is new; "M" if the module is modified.
5. FULL OR PART INSP: If the module/module macro is "modified," indicate "F" if the module/module macro was fully inspected or "P" if partially inspected.
6. DETAILED DESIGNER: and PROGRAMMER: Identification of originators.
7. PRE-INSP EST ELOC: The estimated executable source lines of code (added, modified, deleted). Estimate made prior to the inspection by the designer.

Figure 14 Code inspection module detail form

	DATE _____																																																														
CODE INSPECTION REPORT																																																															
MODULE DETAIL																																																															
MOD/MAC: _____	SUBCOMPONENT/APPLICATION: _____																																																														
SEE NOTE BELOW																																																															
PROBLEM TYPE:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="3" style="text-align: center; background-color: #cccccc;">MAJOR*</th> <th colspan="3" style="text-align: center; background-color: #cccccc;">MINOR</th> </tr> <tr> <th style="text-align: center;">M</th> <th style="text-align: center;">W</th> <th style="text-align: center;">E</th> <th style="text-align: center;">M</th> <th style="text-align: center;">W</th> <th style="text-align: center;">E</th> </tr> </thead> <tbody> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td></tr> </tbody> </table>			MAJOR*			MINOR			M	W	E	M	W	E																																																
	MAJOR*			MINOR																																																											
	M	W	E	M	W	E																																																									
LO: LOGIC _____																																																															
TB: TEST AND BRANCH _____																																																															
EL: EXTERNAL LINKAGES _____																																																															
RU: REGISTER USAGE _____																																																															
SU: STORAGE USAGE _____																																																															
DA: DATA AREA USAGE _____																																																															
PU: PROGRAM LANGUAGE _____																																																															
PE: PERFORMANCE _____																																																															
MN: MAINTAINABILITY _____																																																															
DE: DESIGN ERROR _____																																																															
PR: PROLOGUE _____																																																															
CC: CODE COMMENTS _____																																																															
OT: OTHER _____																																																															
TOTAL: _____																																																															

REINSPECTION REQUIRED? _____

*A PROBLEM WHICH WOULD CAUSE THE PROGRAM TO MALFUNCTION. A BUG. M = MISSING, W = WRONG, E = EXTRA.
NOTE: FOR MODIFIED MODULES, PROBLEMS IN THE CHANGED PORTION VERSUS PROBLEMS IN THE BASE SHOULD BE SHOWN IN THIS MANNER: 3(2), WHERE 3 IS THE NUMBER OF PROBLEMS IN THE CHANGED PORTION AND 2 IS THE NUMBER OF PROBLEMS IN THE BASE.

8. POST-INSP EST ELOC: The estimated executable source lines of code. Estimate made after the inspection.
9. REWORK ELOC: The estimated executable source lines of code in rework as a result of the inspection.
10. OVERVIEW AND PREP: The number of people-hours (in tenths of hours) spent in preparing for the overview, in the overview meeting itself, and in preparing for the inspection meeting.
11. INSPECTION MEETING: The number of people-hours spent on the inspection meeting.
12. REWORK: The estimated number of people-hours spent to fix the problems found during the inspection.
13. FOLLOW-UP: The estimated number of people-hours spent by the moderator (and others if necessary) in verifying the correctness of changes made by the author as a result of the inspection.
14. SUBCOMPONENT: The subcomponent of which the module/macros is a part.
15. REINSPECTION REQUIRED?: Yes or no.
16. LENGTH OF INSPECTION: Clock hours spent in the inspection meeting.
17. REINSPECTION BY (DATE): Latest acceptable date for reinspection.

IBM SYSTEMS

1. PROGRAMMER AND TESTER: Identifications of original participants involved with code.
2. PRE-INSP. ELOC: The noncommentary source lines of code (added, modified, deleted). Count made prior to the inspection by the programmer.
3. POST-INSP. ELOC: The estimated noncommentary source lines of code. Estimate made after the inspection.

InSTRUCTIONS FOR COMPLETING CODE INSPECTION SUMMARY FORM

This form (Figure 14) should be completed according to the information for completing the design inspection module detail form.

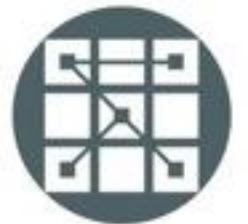
Instructions for Completing Code Inspection Module Detail Form

18. ADDITIONAL MODULES/MACROS?: For these subcompo-nents, are additional modules/macros yet to be inspected?
19. DCR #'S WRITTEN: The identification of Design Change Requests, DCR(s), written to cover problems in rework.
20. PROBLEM SUMMARY: Totals taken from Module Detail forms(s).
21. INITIAL DESIGNER, DETAILED DESIGNER, etc.: Identifier of members of the inspection team.

Figure 15 Code inspection summary form

4. REWORK ELOC: The estimated noncommentary source lines of code in rework as a result of the inspection.
5. PREP: The number of people hours (in tenths of hours) spent in preparing for the inspection meeting.

Reprint Order No. G321-5033.



AUT SOFTWARE ENGINEERING
RESEARCH LABORATORY

AUT

Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study



Assoc. Prof. Tony Clear
Tony.clear@aut.ac.nz



Mihi

- Hāere mai, Haere Mai, Haere Mai.
- Tēnā koutou katoa.
- Ko Tony Clear taku ingoa.
- Nō Pōneke ahau.
- Ko Maungakiekie taku maunga.
- Ko Waitematā taku moana.
- I te taha o taku matua, no Enniscorthy Ireland ahau.
- I te taha o aku whaea, no Cork Ireland ahau.
- Ko Tainui raua ko Ngapuhi nga iwi o nga mokopuna
- Tēnā koutou, Tēnā koutou, Tēnā tatou katoa.



Profile and Current Activities

1. Co-Director of Software Engineering Research lab at AUT <https://serl.aut.ac.nz/>
2. Global Software Engineering
 1. Scaled Agile
 2. Software Ecosystems – RSNZ Catalyst Leaders with Prof Daniela Damian
 3. GSE Education
 4. Global Collaboration
3. Computing Education
 1. Curriculum & Competencies [CC2020]
 2. Onboarding Software Professionals - SIGCSE Special Project [SERL]
 3. Editorial Roles [*ACM TOCE*, *ACM Inroads*, *Computer Science Education*], PC various conferences - CS Ed and SE
 4. Teaching BCIS Papers *Contemporary Issues in Software Engineering*, *Computing Technology In Society*, *R&D Project*, *B Eng(Hons)Industrial Project*, plus *Masters & PHD Supervision*

Do Scaling Agile Frameworks Address Global Software Development Risks? An Empirical Study

Sarah Beecham, Tony Clear, Ramesh Lal, John Noll
Journal of Systems and Software, Jan/Feb 2021
Journal First

Int'l Conference on Global Software Engineering (ICGSE) 2021

Presentation 19th May 2021
Tony Clear, Sarah Beecham, Ramesh Lal, John Noll



Overview

- Focus of the paper
- Phase one - Developing a GSD Risk Catalog
- Phase Two – Theoretical Mapping
- Phase Three – Empirical Assessment
- Conclusions

Paper Focus: Scaling Agile Frameworks and GSD Risks

- A **three-phase** process (lit review/practice-risk mapping/empirical validation),
- Illustrated how **two scaling agile frameworks**
 - –**DAD** and **SAFe**–
- Largely address **63 software development risks**
- Identified in a ***GSD Risk Catalog***
- But stronger in **some areas** than others

Phase One – Developing a GSD Risk Catalog

First phase:

- Identifying **Global Software Development** risks faced by software development organizations,
- By examining the **literature on risks**
In both **conventional** and **GSD contexts**

Result:

- A **GSD Risk Catalog** of **63 risks**,
- Divided into **four quadrants** following [**Wallace and Keil \(2004\)**](#):
 1. Customer Mandate,
 2. Scope and Requirements,
 3. Execution, and
 4. Environment

GSD Risk Catalog

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4), 68-73.

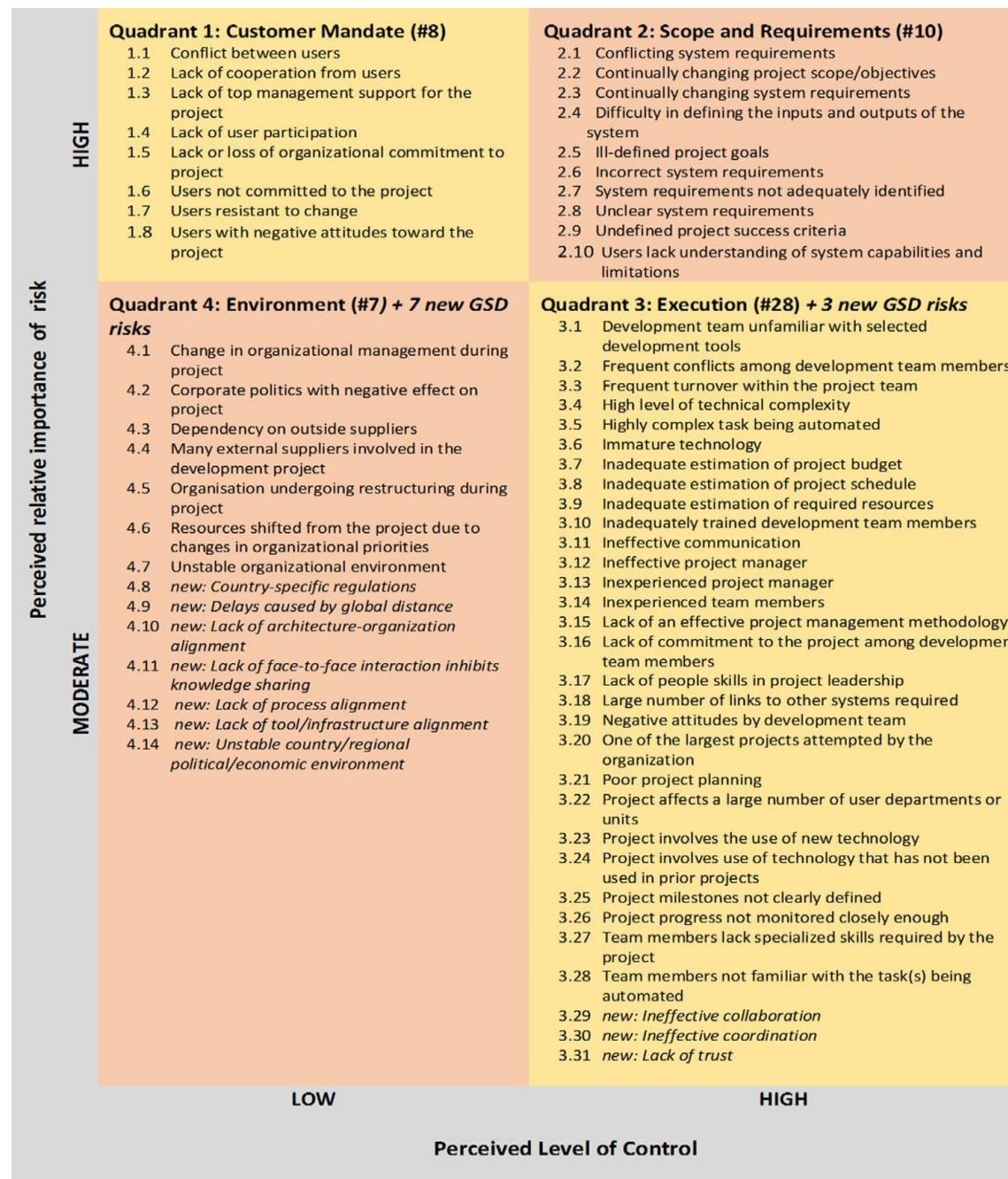


Fig. 2. GSD Risk Catalog derived from Wallace and Keil (2004) and Verner et al. (2014).

Verner, J. M., Brereton, O. P., Kitchenham, B. A., Turner, M., & Niazi, M. (2014). Risks and risk mitigation in global software development: A tertiary study. *Information and Software Technology*, 56(1), 54-78.

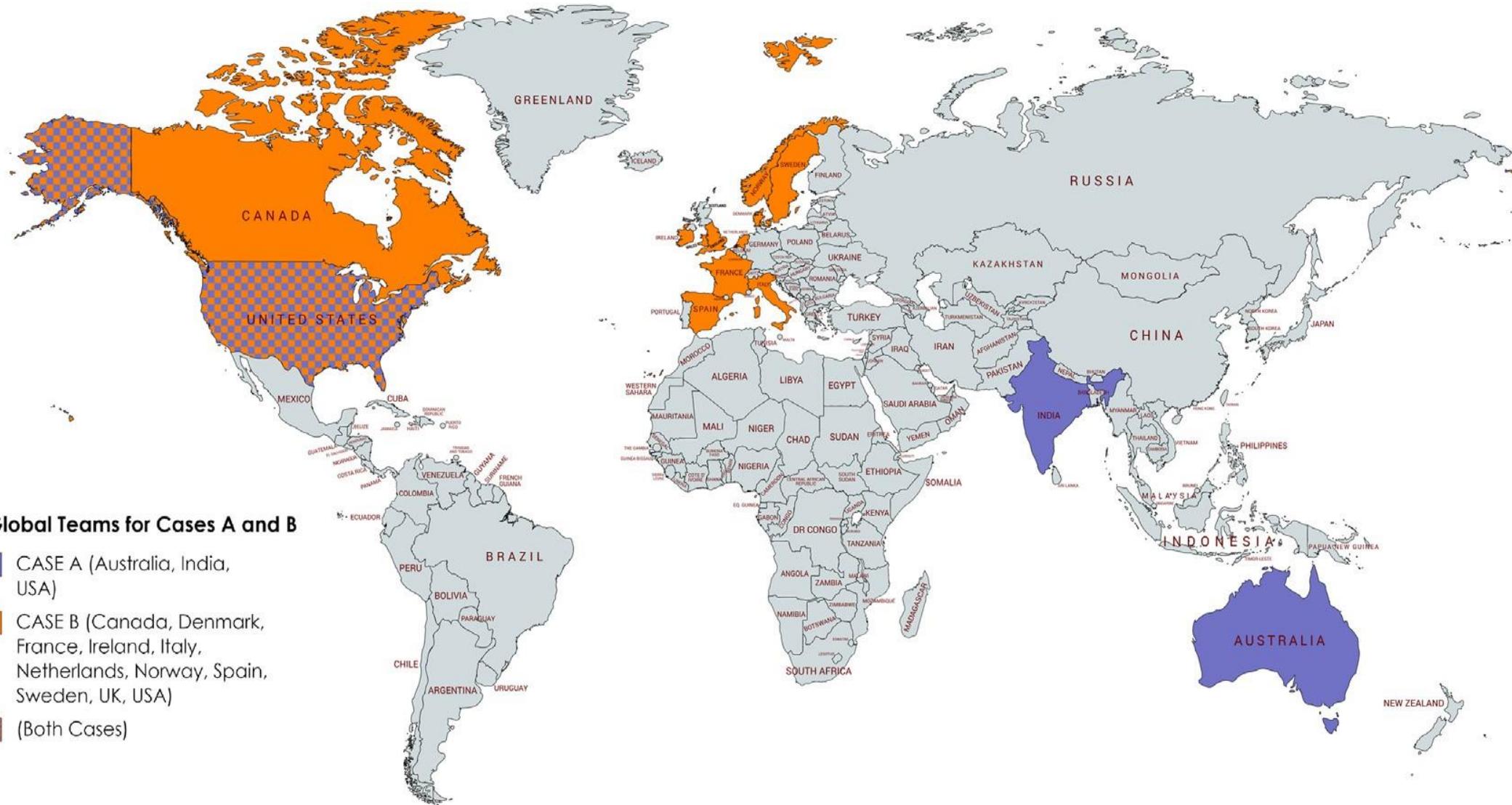
Phase Two - Theoretical Mapping

- Identified potential **risk mitigation** and **elimination practices** in the **two scaling agile frameworks (DAD and SAFe)**.
- We **compared** these extracted **practices** (from **DAD** and **SAFe**) to risks in the ***GSD Risk Catalog*** (developed in Phase one)
- **mapped practices to risks** indicates how scaling agile **practices *might* eliminate or mitigate** those risks

Phase Three – Empirical Assessment

- Assessed the strength of the scaling agile frameworks to mitigate or eliminate risk,
Avoiding criticism that we “*speculated that the strategy would have helped observed problems*” ([Verner et al., 2014](#)),
- Performed an **empirical assessment** of the **theoretical mappings** from Phase 2.
- To determine:
 - **frequency** with which **practices** in each framework **performed** in two companies,
 - and the **risks encountered** by those companies.

2 Global Cases and Locations



Created with [mapchart.net](#) ®

Fig. 5. Case locations.

Phase Three – Empirical Assessment (Cont'd)

- Examined **observation** and **interview notes** and **transcripts**, **self-assessment survey results**,
- In multiple case study of two **global software companies**
- To understand
 - Extent to which DAD and SAFe practices (from theoretical mapping) were implemented in the company
 - Evidence of any of the risks (in the mapping or GSD Risk Catalogue) occurring in either company

If **practice is implemented** in a company and



risk NOT seen = theory supported

risk seen = theory unsupported

Phase Three – Broad Conclusion

Adds to **limited empirical evidence of efficacy of scaling agile frameworks.**

suggesting claims

Scaling Agile Frameworks address risk and are driven by value

have some validity!

Conclusions From Paper

Of the **four quadrants** in the **GSD Risk Catalog**

1. **Customer Mandate risks** quadrant appears to be **better addressed through the SAFe framework than DAD**
2. **Scope and Requirements risks** are **addressed well by both methods**
3. **Execution risks** are **better mitigated by DAD than SAFe**
4. **Environment risks** are **less well addressed by either approach**

Suggests **Environment set of risks** are **less amenable to being addressed by a process framework.**

GSD Risks – not fully addressed by Scaling Agile Practices?

17 (out of 63) risks observed in both cases, eight are GSD risks specific:

1. Ineffective collaboration,
2. Ineffective coordination,
3. Lack of trust,
4. Country-specific regulations,
5. Delays caused by global distance,
6. Lack of architecture-organization alignment,
7. Lack of face-to-face interaction inhibits knowledge sharing,
8. Lack of process alignment.

Implications for all remote and home working?

Further Conclusions From Paper

Creating the **GSD Risk Catalog**

- Found many **Global Software Development risks, not identified** in the Wallace and Keil inventory
- **Ten new risks** GSD risks added to inventory
- Eight of these ten **GSD risks observed** (as shown on previous slide) in both companies, except:
 - Lack of tool/infrastructure alignment and
 - Unstable country/regional political/economic environment

These **new risks** appear to be **endemic** and suggest
a risk tariff in GSD

Final Takeaway

The **result** of a three phased methodology created a **scaling agile risk theoretical mapping** applied in a multiple case study showing how **two** scaling agile frameworks **Disciplined Agile Delivery** and the **Scaled Agile Framework** can potentially **eliminate** or **mitigate** the majority of 'software project' risks

- many **global software development** risks still pervade.

Scaling Agile Frameworks do not support every GSD risk.



Expansion

- GSD Risk Catalog
 - Process of dev't elaborated in Supplementary Technical Report pp. 5 - 7
 - Beecham, S., Clear, T., Lal, R., & Noll, J. (2020). *Companion to manuscript: Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study* https://www.lero.ie/sites/default/files/Beecham_2020_TR003.pdf
- Practices by Framework
 - *The result of these three phases is a scaling agile risk theoretical mapping that shows how two scaling agile frameworks— Disciplined Agile Delivery and the Scaled Agile Framework— can potentially eliminate or mitigate software project risks in global software development. [p. 21 also process elaborated on p.23 of TR]*
 - Beecham, S., Clear, T., Lal, R., & Noll, J. (2021). Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study. *Journal of Systems and Software*, 171(110823). <https://doi.org/https://doi.org/10.1016/j.jss.2020.110823>
 - Table 11 of paper – DAD p.22
 - Table 12 of paper – SAFE p. 25



Utility and Impact?

- BCIS Current R&D Project
- Security Policy for a Globally Distributed SME [Various Asia/Pacific sites]
 - Paananen, H., Lapke, M., & Siponen, M. (2020). State of the art in information security policy development. *Computers & Security*, 88, 101608.
 - Beecham, S., Clear, T., Lal, R., & Noll, J. (2021). Do Scaling Agile Frameworks Address Risk in Global Software Development? An Empirical Study. *Journal of Systems and Software*, 171(110823).
<https://doi.org/https://doi.org/10.1016/j.jss.2020.110823>
- Startup Software Dev't Org
- Markets a Software Solution which helps MSPs (Managed Service Providers) to automate common processes associated with IT support and administration



Case context?

- **Culture of company**
- Freewheeling style
- Historical flow of contract staff
- Inconsistent and Individualistic practices
- Personal use of open-source tools
- Security risks to be analysed and addressed
- Move towards more team-based development
- Desire to grow and scale
- **How to develop and implement a security policy?**
- **Standardization, prioritizing, what to mandate vs. encourage?**



Framing the situation ?

- How to develop and implement a security policy?
- Standardization, prioritizing, what to mandate vs. encourage?

- Possible contribution from our scaled agile risk model?
- Focus on risks to determine priorities?
- Focus on practices to identify areas of concentration and sequence of activities?
- Potential contribution to framing both the problem and solution
- Issues highlighted from *GSD Risk Catalog* below

GSD Risk Catalog

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4), 68-73.

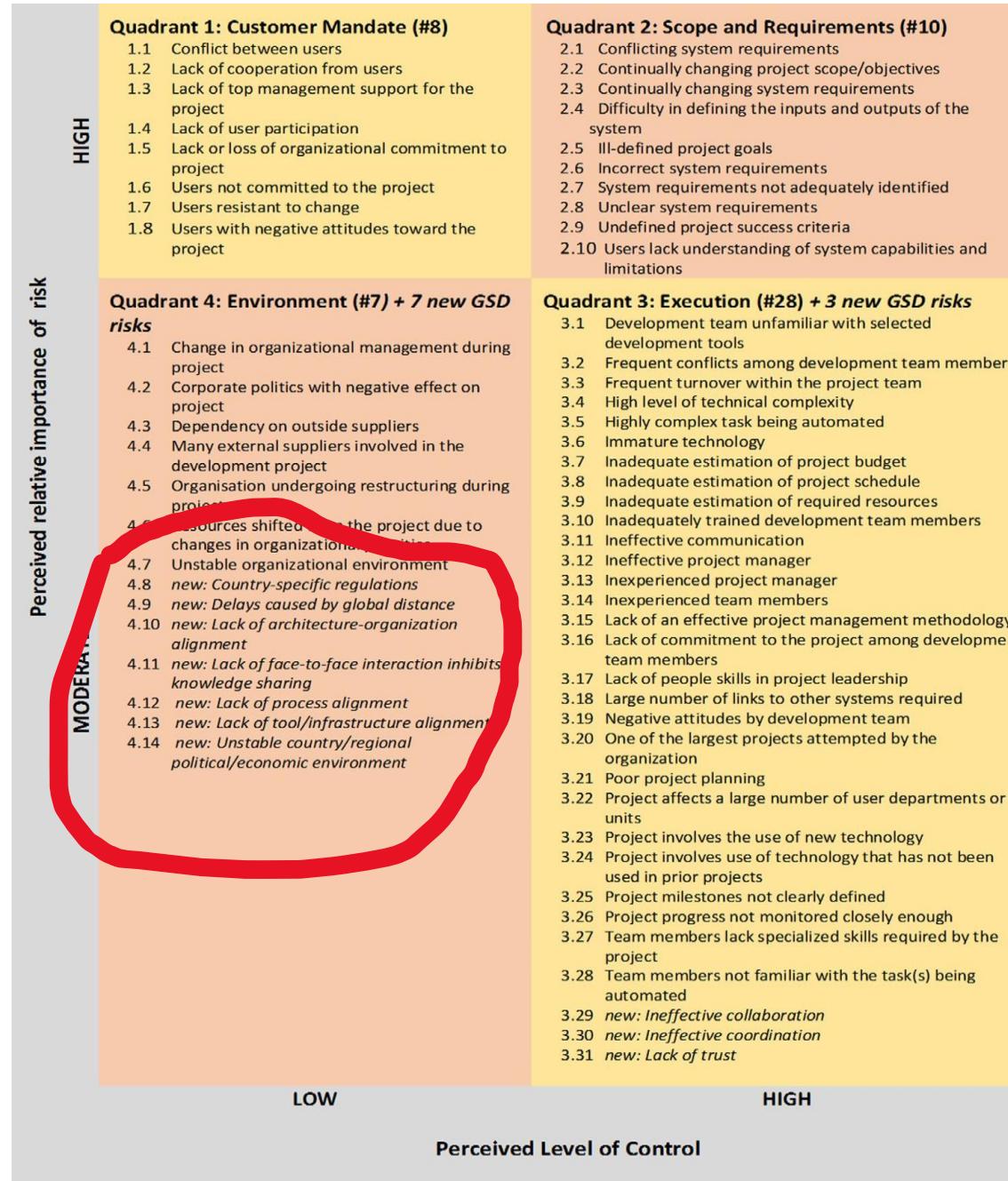


Fig. 2. GSD Risk Catalog derived from Wallace and Keil (2004) and Verner et al. (2014).

Verner, J. M.,
Brereton, O. P.,
Kitchenham, B.
A., Turner, M.,
& Niazi, M.
(2014). Risks
and risk
mitigation in
global
software
development:
A tertiary
study.
*Information
and Software
Technology*,
56(1), 54-78.

GSD Risk Catalog

Wallace, L., & Keil, M. (2004). Software project risks and their effect on outcomes. *Communications of the ACM*, 47(4), 68-73.

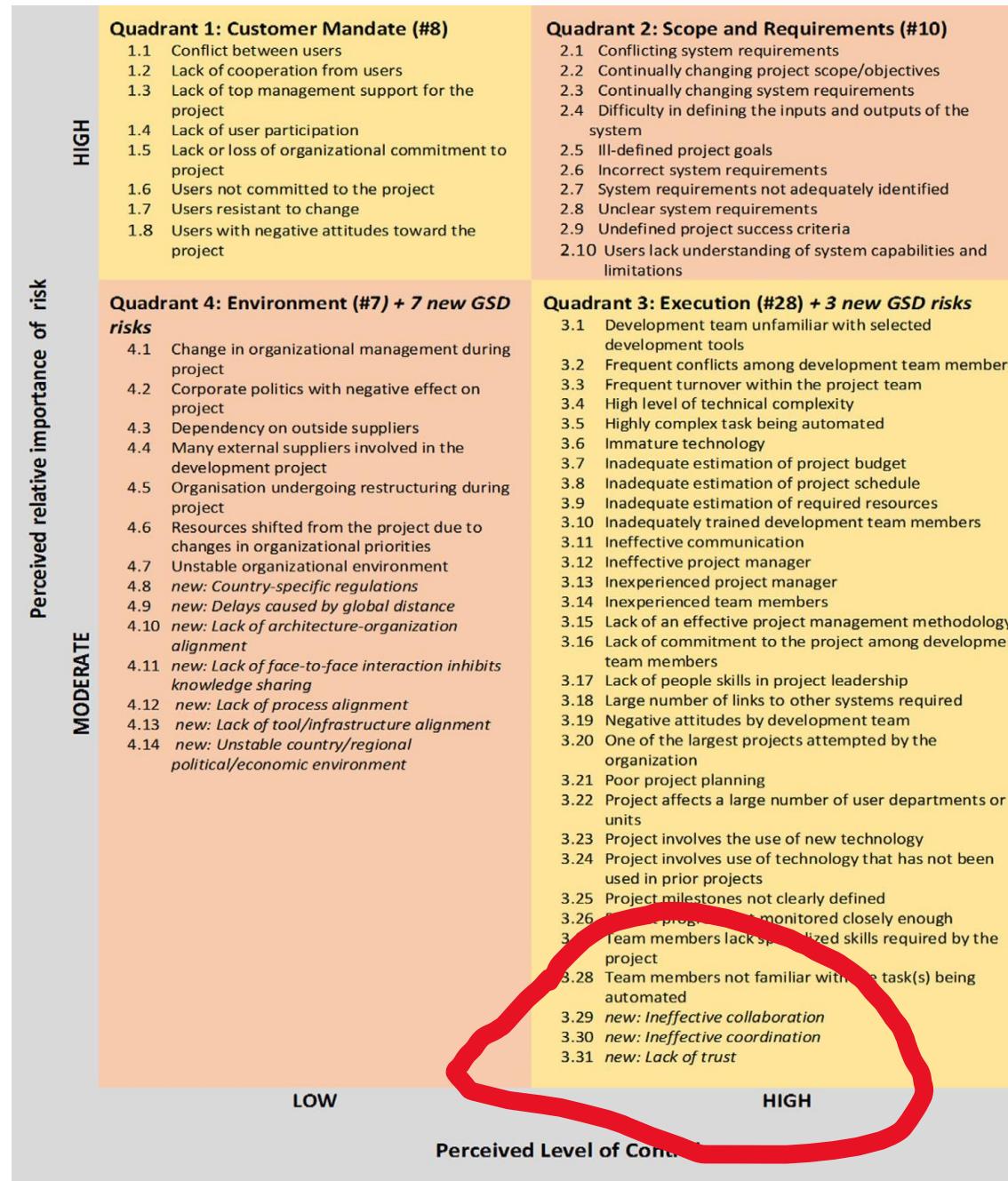


Fig. 2. GSD Risk Catalog derived from Wallace and Keil (2004) and Verner et al. (2014).

Verner, J. M.,
Brereton, O. P.,
Kitchenham, B.
A., Turner, M.,
& Niazi, M.
(2014). Risks
and risk
mitigation in
global
software
development:
A tertiary
study.
*Information
and Software
Technology*,
56(1), 54-78.



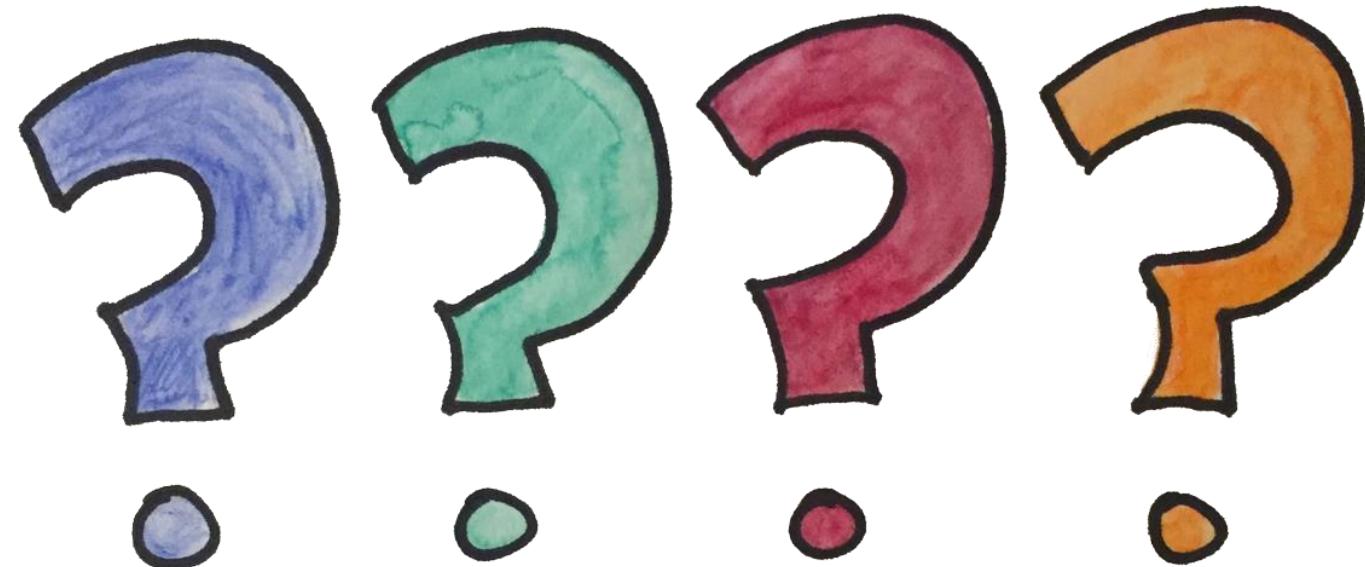
Utility and Academic Impact?

- **Google Scholar Citations [19/05/2022 – 2021 paper]**

- 1) Evaluating the role of scrum methodology for **risk management** in information technology enterprises
- 2) Adoption of Large-Scale Scrum Practices through the Use of Management 3.0
- 3) Agile **transformation**, from classical-to agile project management in a multidisciplinary production environment, a case study
- 4) Tools Engineers Need to **Minimize Risk around CI/CD Pipelines** in the Cloud
- 5) Servitization **Program Management Process Based On Scaled Agility**: A Facilitating Framework
- 6) Analyzing **SAFe Practices with Respect to Quality Requirements**: Findings from a Qualitative Study
- 7) Customer Controlled Managed Services Processes for Productivity Gains without Breaking Contractual Obligations: An Exploratory Study
- 8) Evaluating AGILE adoption in software delivery organizations
- 9) Analyzing **SAFe Practices with Respect to Quality Requirements**: Findings from a Qualitative Study
- 10) Toward Unveiling How **SAFe Framework Supports Agile** in Global Software Development
- 11) An effective agile development process by a hybrid **intelligent effort estimation protocol**
- 12) Agile and generic work values of British vs Indian IT workers: a **culture-clash** case
- 13) Systematic Literature Review: **Causes of Rework** in GSD
- 14) Från kaffeautomaten till digitala mötesrum: Mjukvaruutvecklare upplevelser av att arbeta med agila utvecklingsmetoder under rådande pandemic *From the coffee machine to digital meeting rooms: Software developers' experiences of working with agile development methods during the current pandemic*
- 15) Modelo de evaluación de metodologías de desarrollo de software web *Evaluation model of web software development methodologies*



- Nga mihi
- And
- Questions?



Agile is eating the World



Why and What does it Even Mean ?

Jan 2, 2018, 07:35am EST

Forbes magazine

Why Agile Is Eating The World



Steve Denning Senior Contributor

Leadership Strategy

I write about 21st century leadership, Agile, innovation & narrative.

The Wall Street Journal

Why Software Is Eating The World

By Marc Andreessen

August 20, 2011

The meaning has got confused...



70 different Agile practices

Mob programming

DevSecOps

Fast Agile

ShapeUp!

<https://craigsmith.id.au/2015/12/03/yow-2015-40-agile-methods-in-40-minutes/>

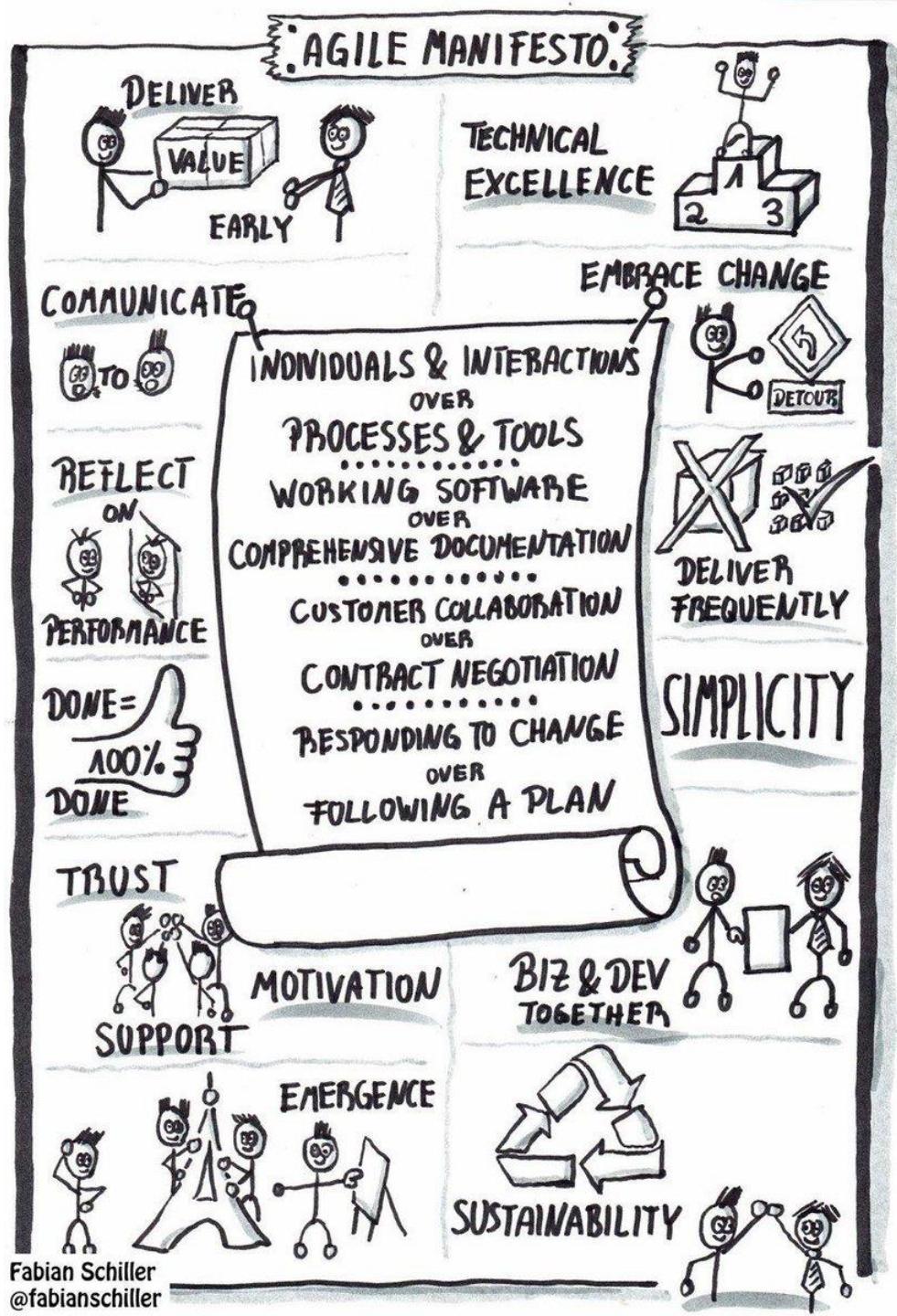
Agile Manifesto

This describes a set of values, beliefs and principles

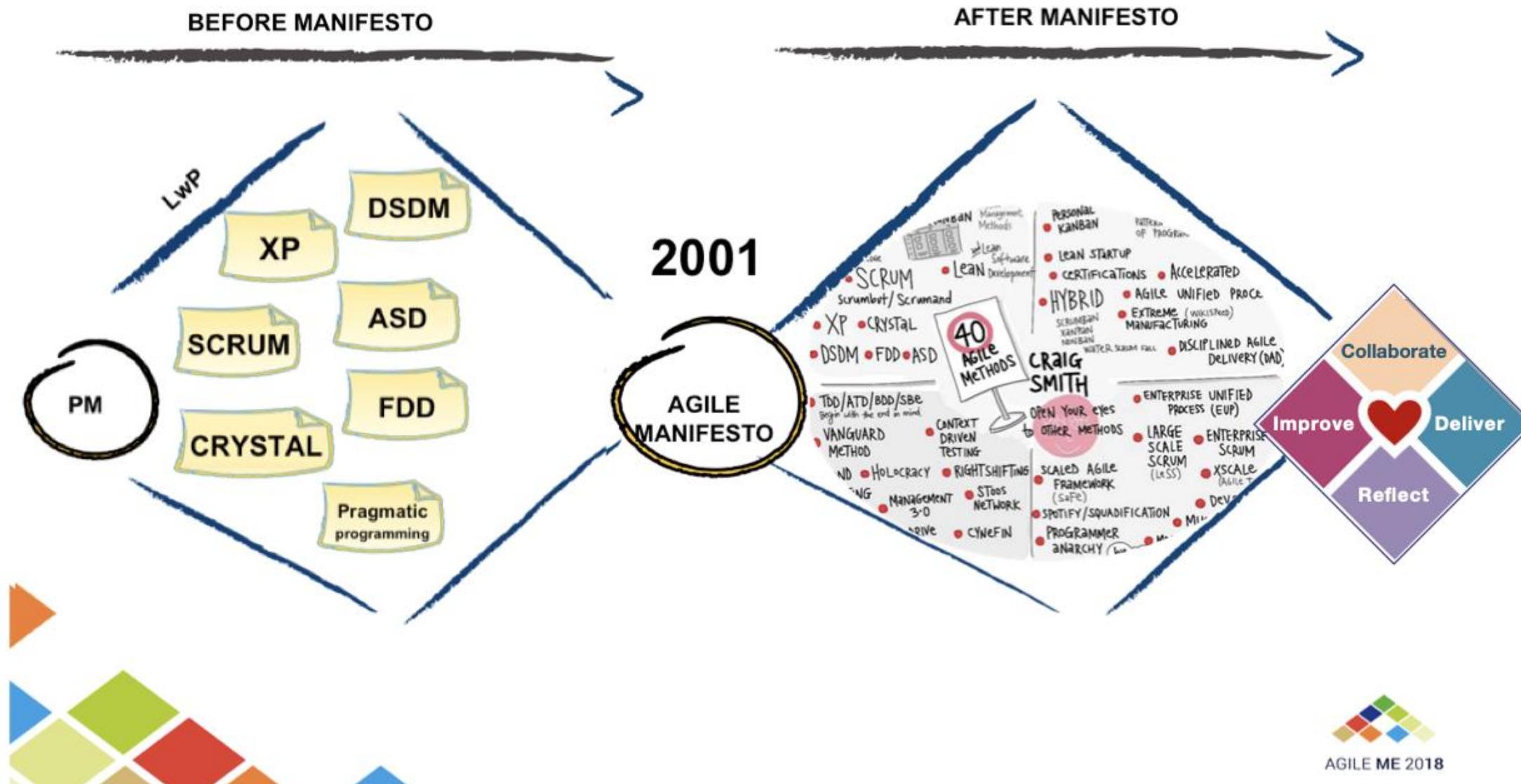
A Mindset....

That guides how to behave, interact and design processes to do work

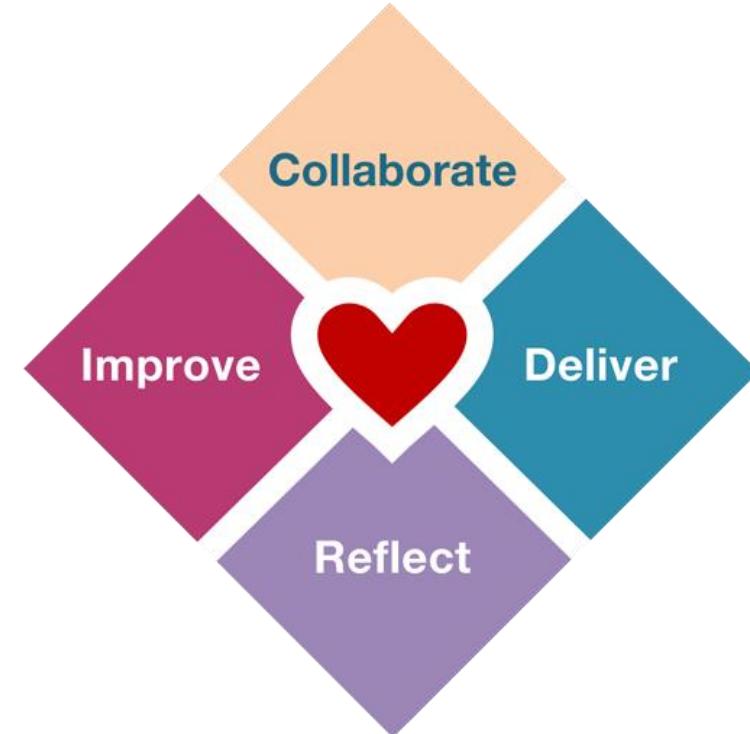
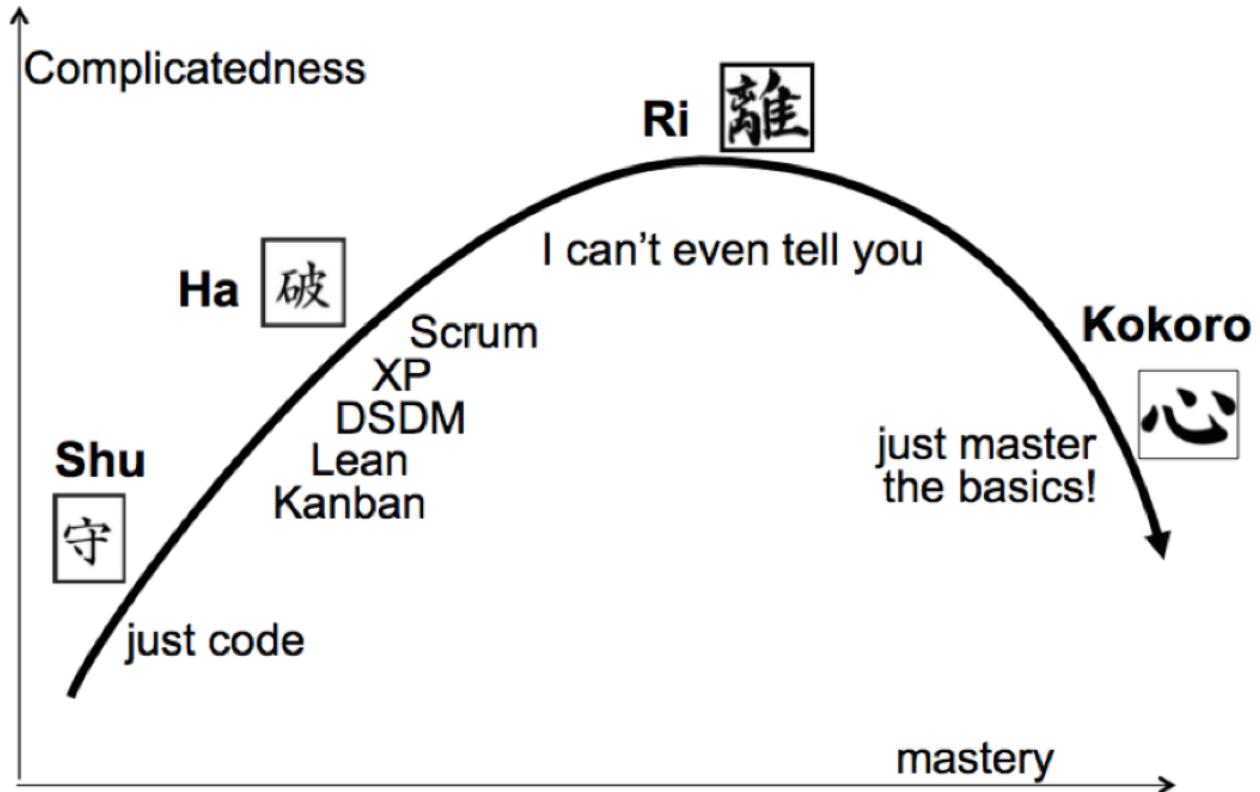
Focus on Software Development Work



Pierre Hervouet's recap of history:



Let's simplify it....the heart of Agile



<https://heartofagile.com>

Figure 1. The Shu-Ha-Ri-Kokoro progression.

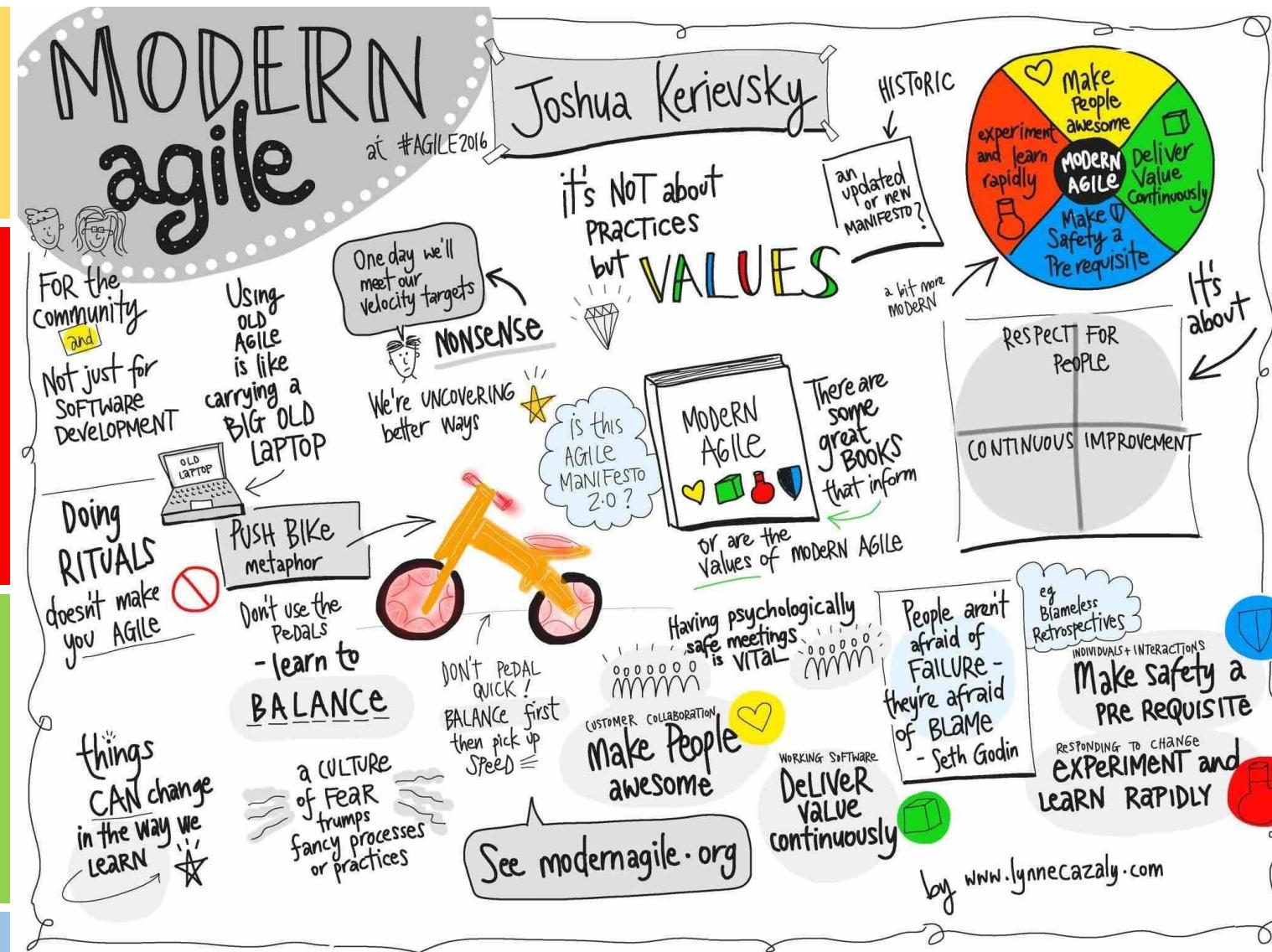
<https://alistair.cockburn.us/wp-content/uploads/2018/02/The-Heart-of-Agile-Technical-Report.pdf>

We learn their context and pain points, what holds them back and what they aspire to achieve. How can we make them awesome?

We learn rapidly by experimenting frequently. We make our experiments "safe to fail" so we are not afraid to conduct more experiments. When we get stuck or aren't learning enough, we take it as a sign that we need to learn more by running more experiments.

In modern agile we ask ourselves, "How could valuable work be delivered faster?" Delivering value continuously requires us to divide larger amounts of value into smaller pieces that may be delivered safely now rather than later.

We protect people's time, information, reputation, money, health and relationships. And we endeavor to make our collaborations, resilient and safe.



The goal of Agile mindset, values, principles, practices

“Making money” is not the goal

“Being agile” is not the goal.

“Working software” is not the goal.

Agile & Scrum & working software are means to achieving the goal.

Everyone must focus on the goal

What are your values related to why and how you work and interact and collaborate? What is the GOAL?

To create product or service that delights your customers

What will make it delight them?

It's of value to your clients

What is the value?

The product or service

...makes the world your client lives in and the way they work and interact easier/faster/competitive

Delight a customer =

“Providing a continuous stream of additional value to customers and delivering it sooner”

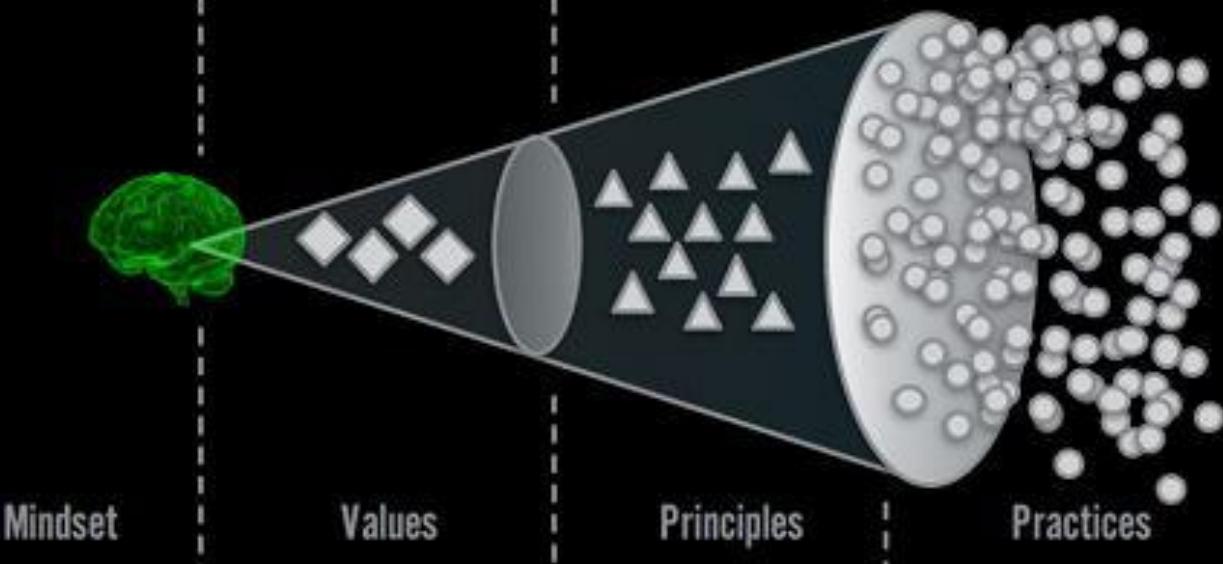
(Denning)

What is the value you create for your customers and do you care?

From mind set to process

WHAT IS AGILE?

AGILE IS A MINDSET DESCRIBED BY 4 VALUES DEFINED BY 12 PRINCIPLES MANIFESTED THROUGH AN UNLIMITED NUMBER OF PRACTICES

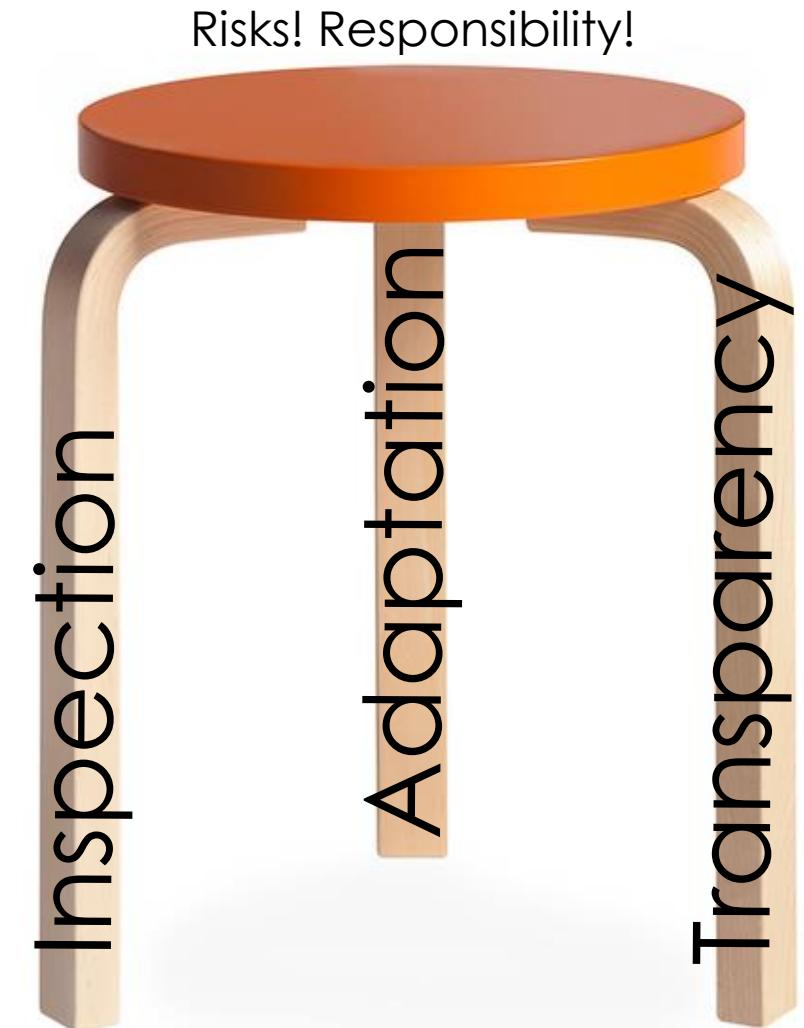


Scrum Values

There is a focus on understanding and making progress towards goals

Product Goal -> Sprint Goals that are about customer value/needs

Agile values lead to an empirical process control control system – Scrum is such a system

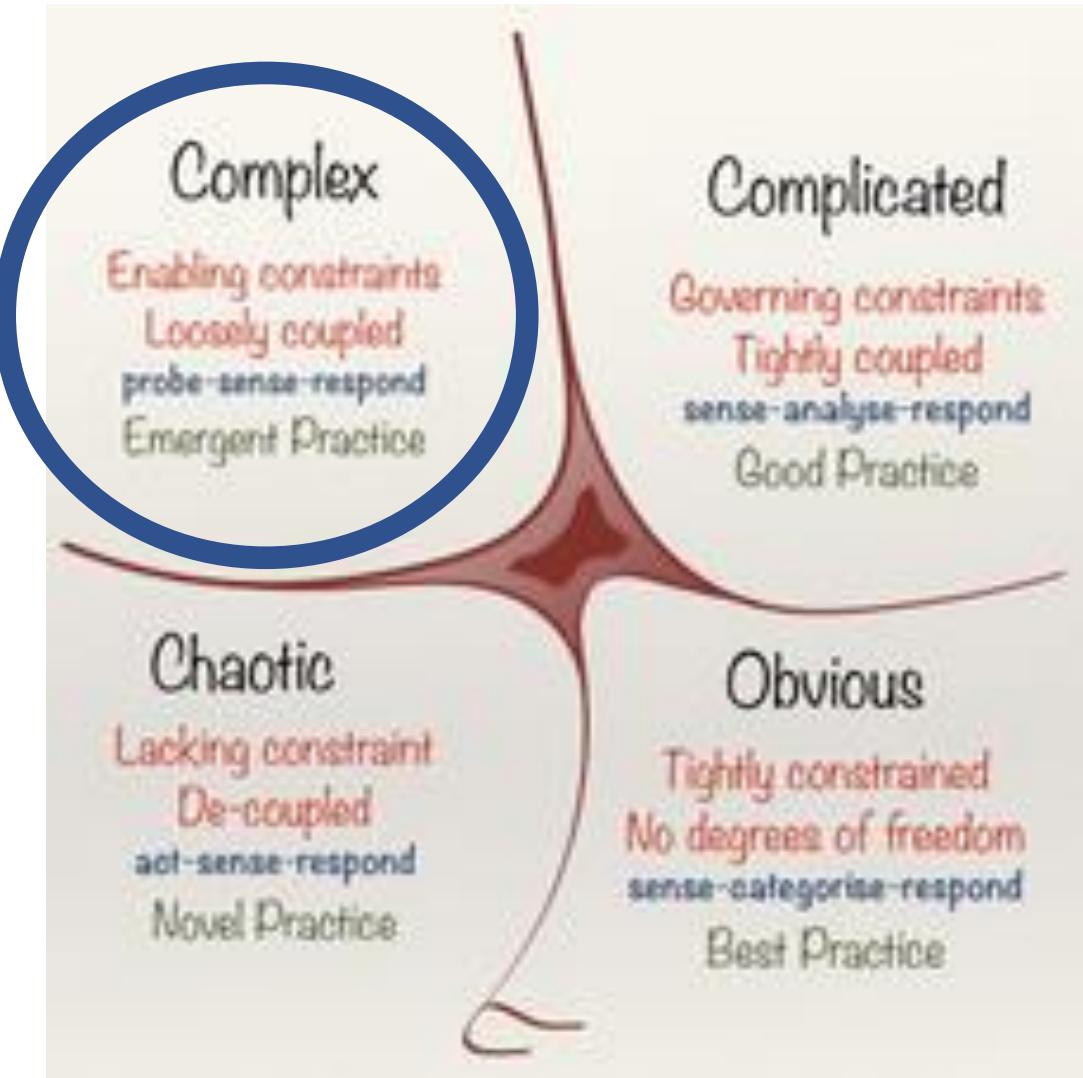


Why the need to be Agile?

Person
Team
Organisation

Work together in a way that aligns with Agile values and principles

Complex work does not have best practice



1 The accelerating pace of change ...

<https://medium.com/ml-everything/ai-optimists-vs-pessimists-and-why-the-singularity-isnt-near-5d3a614dbd45>



2045 Surpasses brainpower equivalent to that of all human brains combined

2 ... and exponential growth in computing power ...

Computer technology, shown here climbing dramatically by powers of 10, is now progressing more each hour than it did in its entire first 90 years



Colossus

The electronic computer, with 1,500 vacuum tubes, helped the British crack German codes during WW II



UNIVAC I

The first commercially marketed computer, used to tabulate the U.S. Census, occupied 943 cu. ft.



Apple II

At a price of \$1,298, the compact machine was one of the first massively popular personal computers



Power Mac G4

The first personal computer to deliver more than 1 billion floating-point operations per second

COMPUTER RANKINGS

By calculations per second per \$1,000



Analytical engine

Never fully built, Charles Babbage's invention was designed to solve computational and logical problems



Hollerith Tabulator

IBM Tabulator

National Ellis 3000

Zuse 2

Zuse 3

ENIAC

IBM SSEC

EDVAC

Datematic 1000

IBM 1620

DEC PDP-10

Whirlwind

DEC PDP-4

IBM 1130

Data General Nova

Intellic-8

Compaq Deskpro 386

IBM PC

Pentium PC

Pentium II PC

Dell Dimension 8400

Mac Pro

Nvidia Tesla GPU & PC

ELECTROMECHANICAL

RELAYS

VACUUM TUBES

TRANSISTORS

INTEGRATED CIRCUITS

0.00001

2045

3 ... will lead to the Singularity

Surpasses brainpower of human in 2023



Surpasses brainpower of mouse in 2015



Break free from Method prison



#noprojects – Optimize continuous delivery, flow, improvements and benefits rather than resources and time



Put effort into delivering benefits rather than meeting triple constraints



When will the software (work) deliver value next rather than when will the project be finished



Stream of value with uncertain stop date rather than a project handover
(projects end - software continues to evolve until retired)



Late requirements accepted rather than considered costly and undesirable



Keep learning from experience and experiments (double loop learning)

What does it mean to be Agile?

Person
Team
Organisation

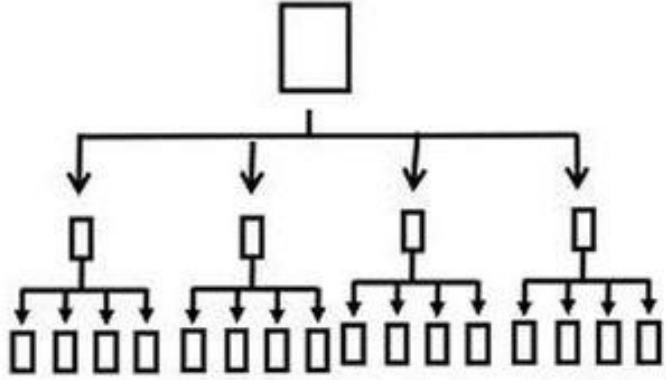
The Law of the Customer

The Law of the Team

The law of the Network

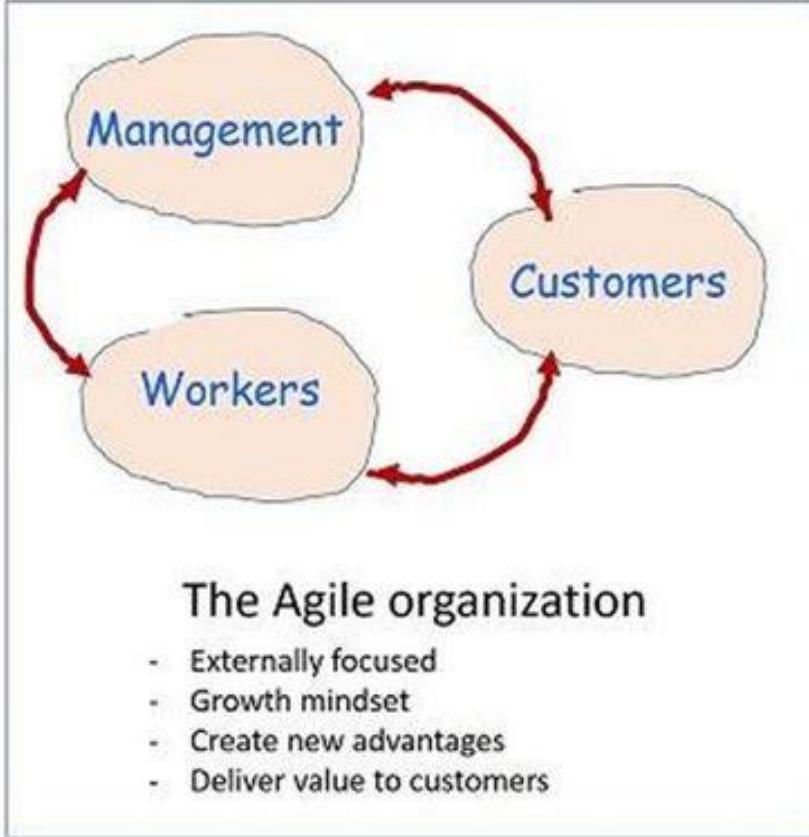
The Age of Agile
Denning

The law of the customer



The bureaucratic organization

- Internally focused
- Fixed mindset
- Defend existing advantages
- Make money for shareholders



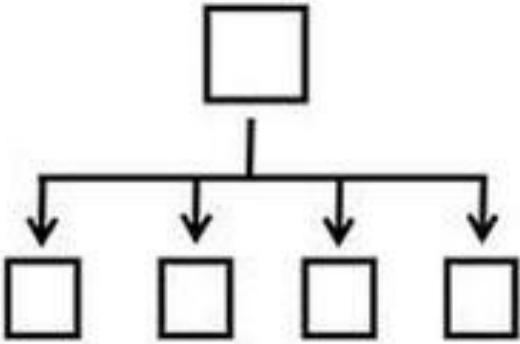
The Agile organization

- Externally focused
- Growth mindset
- Create new advantages
- Deliver value to customers

Agile practitioners are obsessed with delivering value to customers.

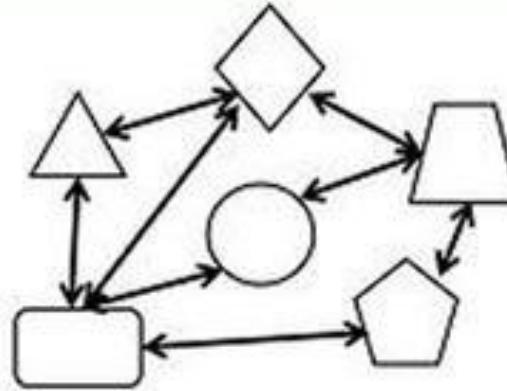
Everyone in the organization has a clear line of sight to the ultimate customer and can see how their work is adding value to that customer—or not. If their work isn't adding value to any customer or user, then an immediate question arises as to why the work is being done at all. The firm adjusts everything—goals, values, principles, processes, systems, practices, data structures, incentives—to generate continuous new value for customers and ruthlessly eliminate anything that doesn't contribute.

The law of teams



The bureaucratic team

- Top down
- Individual responsibilities
- Little interaction

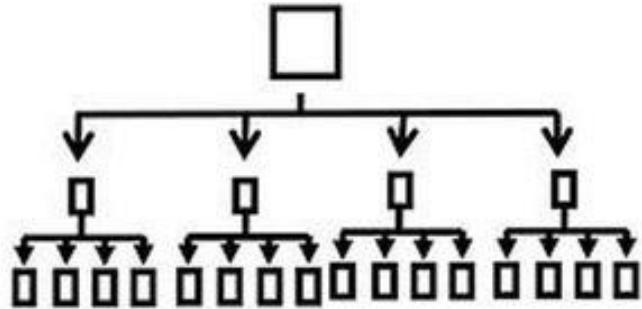


The Agile team

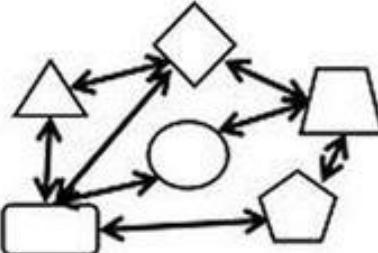
- Autonomous
- Cross-functional
- Much interaction

Agile practitioners share a mindset that work should in principle be done in small autonomous cross-functional teams working in short cycles on relatively small tasks and getting continuous feedback from the ultimate customer or end user.

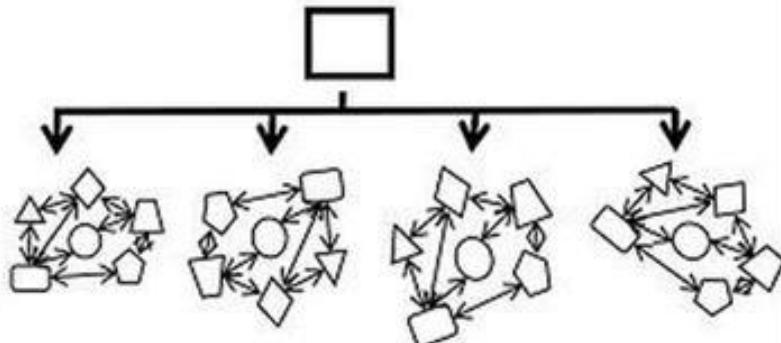
The law of networks



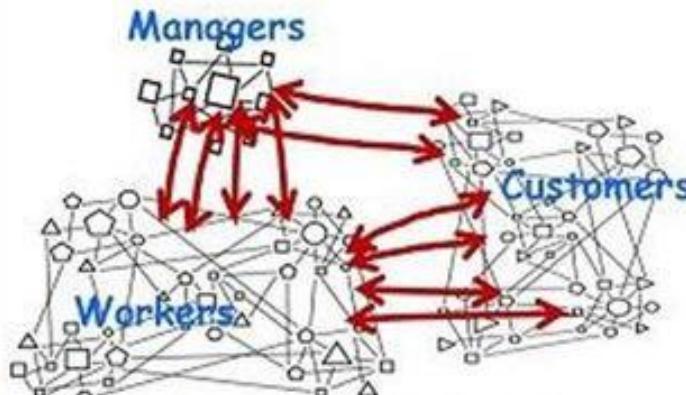
1. Bureaucracy



2. Agile team



3. Agile teams in a bureaucracy



4. Agile network

When Agile teams are housed within a bureaucracy, collaboration between teams can be just as much a problem as it is between silos in a pure bureaucracy.

Scrum Guide Revision 10 (2020)

//The Key Changes



1. Artifacts & Commitments
2. Accountabilities not roles
3. Simplification of Language for a Wider Audience - removed references to IT work
4. From self-organising to self-managing
5. Stuff removed

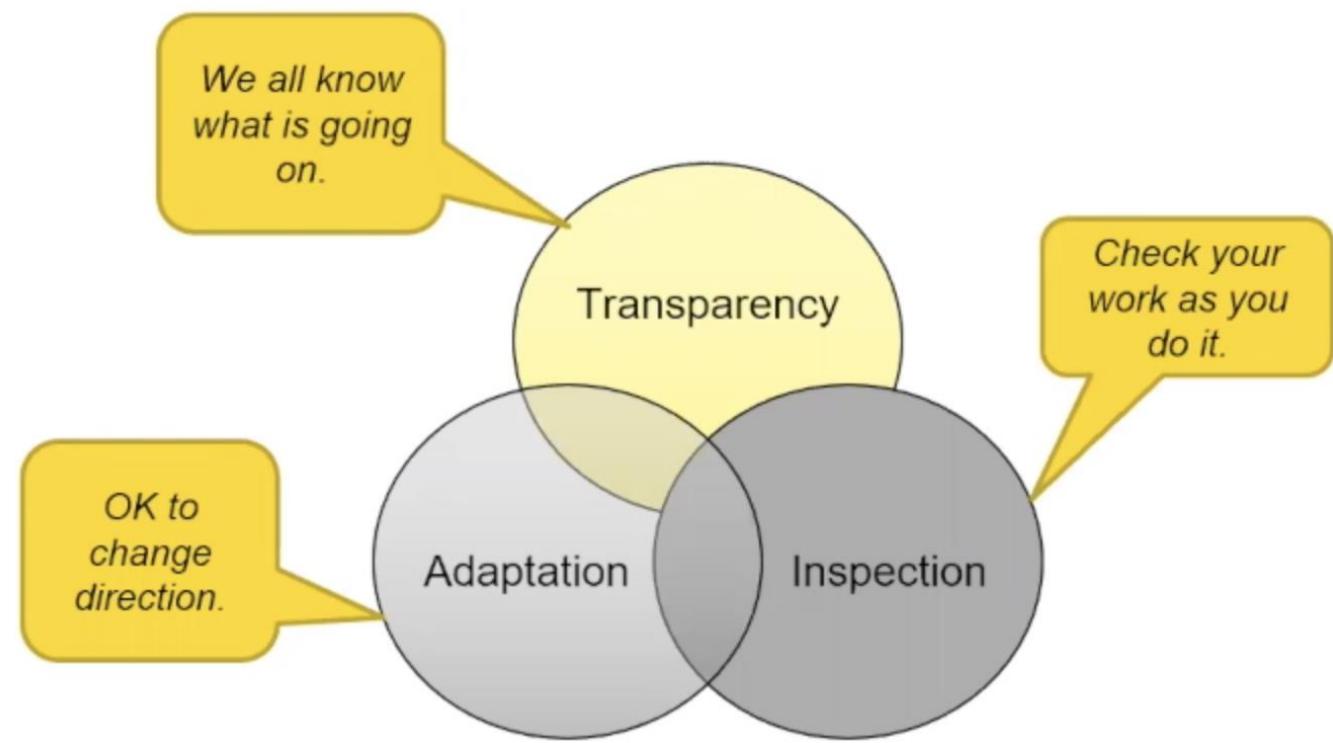
//Scrum is built on emergence & empiricism



Emergence – in Complex Adaptive Systems, solutions emerge from the interplay of the components. We *discover* the right solution.



Empiricism – the use of evidence in the formation of ideas. We have a theory, we run an experiment to test & learn, we pivot.



//Lack of emergence & empiricism in product devel



- A lot of agile product development lacks emergence & empiricism
- Two clear symptoms
 1. **Mechanical Scrum** – outputs not outcomes
 2. **Lack of Done**

//Mechanical Scrum



Too much focus on

- the framework itself, rather than to where it's supposed to take you
- delivering more outputs, not better and more valuable outcomes.

Not enough focus on using Scrum creatively to solve problems.

- Product Backlogs typically a list of stuff to do
- Teams typically an execution unit, not a creative problem solving unit.

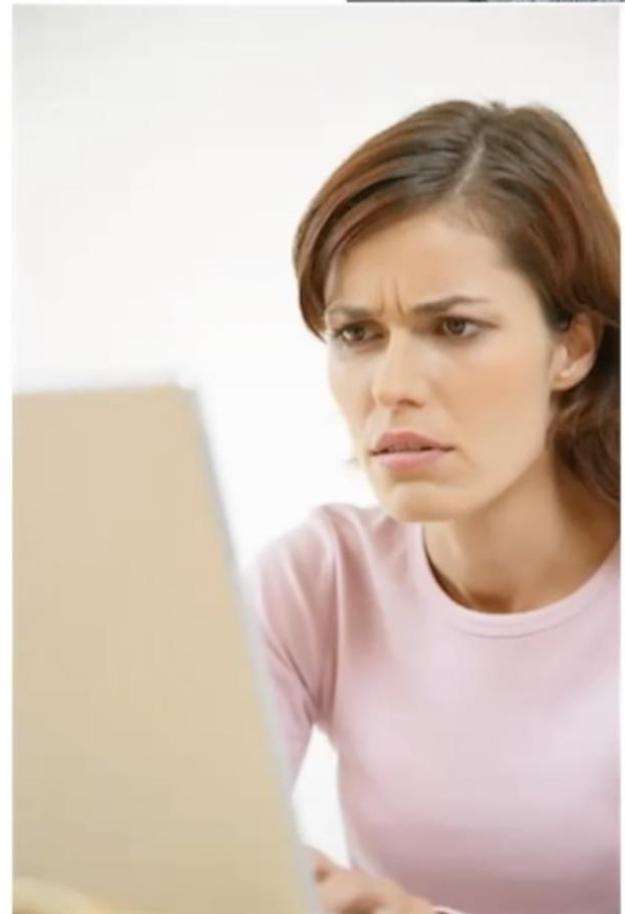
But...aren't complex problems best solved by the interplay of the "agents" in the system – emergence & empiricism?





//Lack of “Done”

- The entire point of Scrum is to create Done Increments.
 - Done increments maximise transparency. If we can
 - use the Increment (transparently inspect it)
 - validate hypotheses & assumptions
 - give feedback and adapt.
 - Without this, empiricism is limited
 - Being Done is utterly vital...
- } This is the very heart of Scrum!
- But the Definition of Done has never been a core part of the Scrum framework!
 - Few teams can reliably deliver a Done increment, at least every Sprint





//Change 1: Each Artifact now has a “Commitment*

Artifact	Commitment	Explanation
Sprint Backlog	Sprint Goal	<ul style="list-style-type: none">Provides context to the Sprint - why are we doing this Sprint?Is part of the Sprint BacklogThe rest of the Sprint Backlog emerges to define what will fulfil the Sprint Goal.
Product Backlog	Product Goal	<ul style="list-style-type: none">Provides context to the Product - why are we making this Product?Is part of the Product BacklogThe rest of the Product Backlog emerges to define what will fulfil the Product Goal.
Increment	Definition of Done	<ul style="list-style-type: none">A formal description of the state of the Increment when it meets the required quality.Creates transparency by providing everyone a shared understanding of what work was completed.

//The Product Goal



- A single, high-level, long-term product objective
- Describes a future state of the product (a target) for the Scrum Team to plan against.
- The Product Backlog emerges to define “what” will fulfil the Product Goal.
- Product Owner accountable for developing and explicitly communicating it
- How it is created and communicated is up to you.



*“Be stubborn on the vision
and flexible on the details!”*

- jeff bezos

//Challenge: Roles



Edwin Dando

Earlier versions referred to two teams

- Scrum Team
- Development Team

This led to “us and them”:

- The Product Owner defines the direction
- the Development Team execute – “we just take work off the Product Backlog and deliver it”

Fails to leverage collective intelligence

Over the years, the 3 roles slowly turned into common job titles

- This person does this
- The other one does that...

Scrum Master Anti-Patterns FROM JOB ADS



©Stefan Wolpers 2021 · Berlin Product People GmbH

//Change 2: People based changes



1. **Accountabilities not roles.** Ignore your job title and focus on the necessary accountabilities.
2. **One team** - the Scrum Team, focused on the same objective. There is no longer a Development Team, just Developers

Comment:

- Will require people to have meaningful discussions and agree accountabilities. Scrum doesn't care about your job title, just the minimal set of accountabilities.
- It will provide much more flexibility re how the accountabilities are filled – e.g. Scrum Master accountabilities could be fulfilled by a Developer

//Change 3. Language



Why: Recipe's are poor approach to solving Complex problems. However, unfortunately many people apply Scrum as a recipe, and misinterpret it. Language is very important.

- Simpler, clearer, less prescriptive
- Targeted at a wider audience

The Scrum framework is purposefully incomplete, only defining the parts required to implement Scrum theory. Scrum is built upon by the collective intelligence of the people using it. Rather than provide people with detailed instructions, the rules of Scrum guide their relationships and interactions.

IT-specific language removed

//Change 4. From self-organising to self-managing



Why this update was made:

- Some teams view self-organizing as meaning they can do whatever they want. Self-organization occurs when an intelligent system finds the best way in an environment to achieve a goal.
- The 2020 version uses the term self-managing to bring clarity to the concept and stop the weaponizing of self-organization as an excuse.

Previous version
<p><i>Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team.</i></p>

2020 version
<p><i>They are self-managing, meaning they internally decide who does what, when, and how.</i></p>



//Change 5 – Stuff Removed

Removed	Reason
The Daily Scrum three questions	Because they led people to believe the Daily Scrum was a status meeting, rather than an empirical daily planning meeting. Instead think <ul style="list-style-type: none">1. Where are we now?2. Based on that, what is our next optimal move for the next 24 hours?3. What is/might stop us? <p><i>“The purpose of the Daily Scrum is to inspect progress toward the Sprint Goal and adapt the Sprint Backlog as necessary, adjusting the upcoming planned work”.</i></p>
The word ‘Roles’ when describing The Scrum Master, Product Owner, and Developers	Already covered
A Sprint Review led only by The Product Owner	Now led by the Scrum Team (in order to make the Sprint Review as inclusive as possible).
No Team-Within-A-Team	Already covered
IT-specific language removed	Already covered – make it more accessible to non-IT people.

Latest views on ‘agile’ from Scott Ambler

<https://www.linkedin.com/pulse/agile-community-shat-bed-scott-ambler-1npwc/>

<https://www.linkedin.com/pulse/how-agilists-can-move-forward-after-shutting-bed-scott-ambler-fatpc/>

<https://www.linkedin.com/pulse/future-agile-isnt-shit-scott-ambler-np7vc/>

AUT

COMP 501 – Part Two



Course lecturer –
Assoc Prof Tony Clear



COMP 501 – Part B Requirements and Modelling

Course lecturer – Assoc Prof Tony Clear

Computing Technology in Society

Paper Overview





Systems Analysis and Design

System Development

- **Systems Analysis and Design** [*may have other names*]
 - Step-by-step process for developing high-quality information systems
 - **Information systems:** Combination of technology, people, and data to perform certain business functions
- **What Does a ‘Systems Analyst’ Do?** [*may have many other names*]
 - Plans, develops, and maintains information systems
 - Manages IT projects, including tasks, resources, schedules, and costs
 - Conducts meetings, delivers presentations, and writes memos, reports, and documentation

Systems Analysis and Design

- Step-by-step process for developing high-quality Enterprise Systems (ES)
- Who is responsible?
 - Organizations may have an IT department
 - IT department may have a software engineering unit
 - Software engineering units consist of development team (s) (systems analysts/designers/developers), a quality assurance team (testers) and a documentation team (technical writers).
 - Systems Analysts focus on the analysis/design work
 - They also plan, help develop/test, and maintain information systems

Techniques to Understand the Business to Design the System

Business Process modeling (BPM)

- BPM is carried out by business analysts and managers
- BPM is the activity of representing processes (operations) & information needs of a business
- Business process modeling notation (BPMN)

Business Profile

- Mission statement
- **Organizational structure**
- Business processes
- Strengths Markets
- IT infrastructure
- Customers

Business Models

- Business model - graphically displays one or more **business process**
- **Business process** - is a specific set of transactions, events and results that can be described and documented

Systems Development Skills: Systems Analyst

- Investigates, analyzes, designs, develops, installs, evaluates, and maintains a company's information systems
- Constantly interacts with users and managers within and outside the organization
- **Knowledge, Skills, and Education**
 - Technical knowledge
 - Communication and business skills
 - **Critical thinking skills**
 - Education - A college degree in information systems, science, or business
 - Some IT experience is required
 - Certification
 - Helps IT professionals learn new skills and gain recognition for their efforts

Systems Development Skills: Systems Analyst

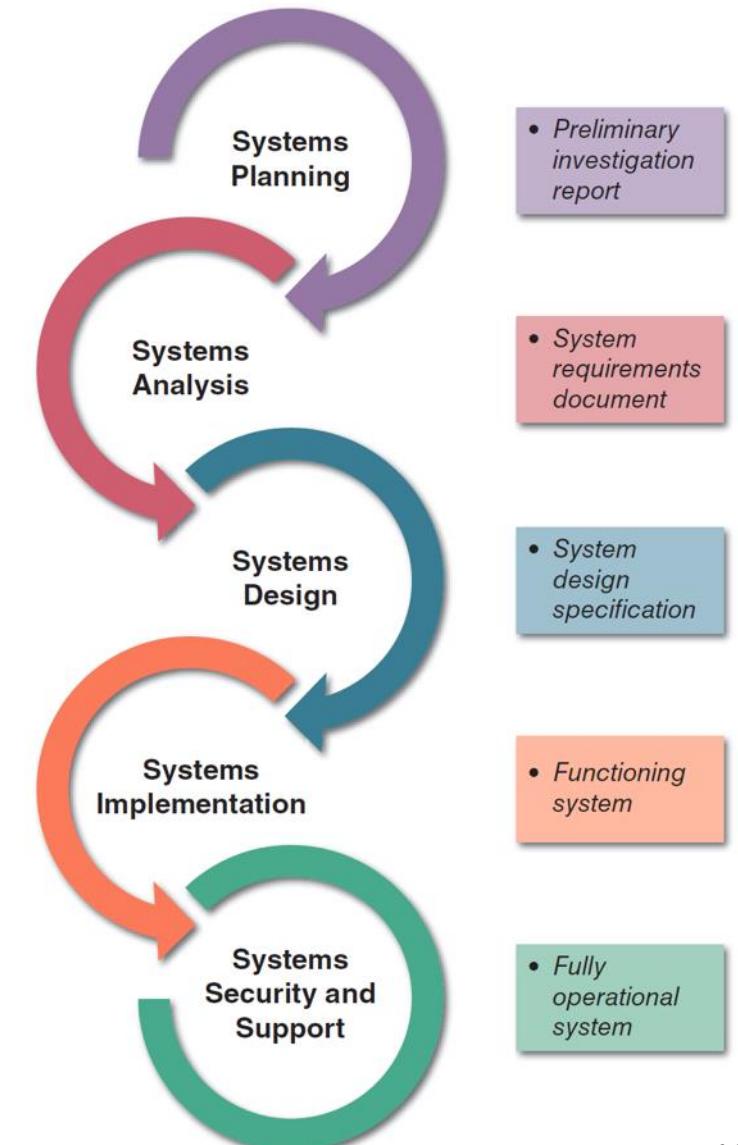
Technically qualified	Business oriented	Highly motivated	Communication skills (teamwork)	Strong analytical and critical thinking skills
<ul style="list-style-type: none">• Degree in IT, CS, SE or IS• Certification course	<ul style="list-style-type: none">• Understand about business processes or operations, customers, organizational structures, values, culture• Know more about the IT needs of the business than managers would know themselves• Provide business solutions (ES) based on business needs rather than the technology trend.	<ul style="list-style-type: none">• You have firm belief and courage• Have a positive attitude; do job with pride and passion• Show respect; recognize other people's good qualities or achievements.• Have empathy & sincerity	<ul style="list-style-type: none">• Understand problems• exchanging information all the time• build relationships & connections• communicate negative or difficult messages without creating conflict or destroying trust	<ul style="list-style-type: none">• break a problem down in parts• provide solution to each part• Suggest new ideas

Systems Analyst: The Role

- **Role**
 - Acts a translators to managers and programmers
 - A company's best line of defense in an IT disaster
 - Most valuable skill - The ability to listen
 - Seeks feedback from users to ensure that systems do not deviate from accomplishing set objectives
- **Tasks systems analyst does to help develop Enterprise Systems**
 - Translate business requirements into IT projects
 - Plan projects; develop schedules & estimate's cost
 - Document business profile, (review/document business processes), create models (DFDs)
 - Help to create various designs
 - Select hardware & select software packages
 - Conduct meetings, deliver presentations, write memos, produce reports and other documentation in regards to project
 - Test
 - Train users

Systems Development Method for Structured Analysis

- The SDLC model usually includes five steps
 - Systems planning
 - Systems analysis
 - Systems design
 - Systems implementation
 - Systems support and security

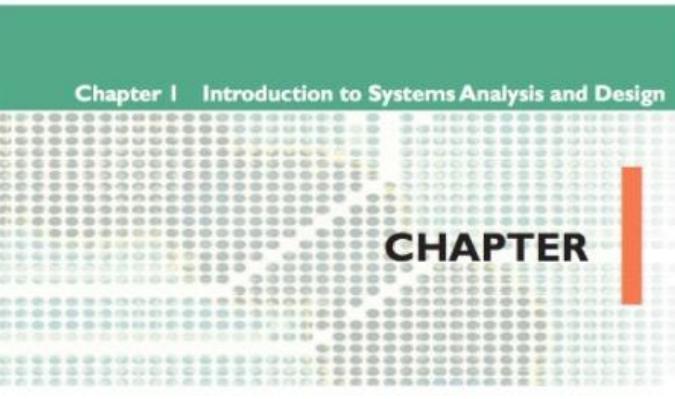


Related Chapter in eBook

Table Of Contents

The screenshot shows a eBook's Table of Contents. At the top is a logo for 'CP1117: INFS500 Enterprise Systems (Custom)' with a sun icon. Below it is a section titled 'Tiller'. The table of contents lists several sections: 'Title page', 'Copyright page', 'Table of Contents', 'Extract 1 Introduction to Systems Analysis and Design' (which is highlighted with a blue vertical bar), 'Extract 2 Analysing the Business Case', 'Extract 3 Data and Process Modeling', 'Extract 4 Development Strategies', 'Extract 5 Managing Systems Implementation', 'Extract 6 Managing Systems Support and Security', 'Extract 7 Enterprise Systems', 'Extract 8 Information and Decision Support Systems', and 'Extract 9 Knowledge Management and Specialised...'. Page numbers are listed to the left of each extract.

	Section	Page Number
ii	Title page	
iii	Copyright page	
iv	Table of Contents	
1	Extract 1 Introduction to Systems Analysis and Design	
40	Extract 2 Analysing the Business Case	
70	Extract 3 Data and Process Modeling	
106	Extract 4 Development Strategies	
138	Extract 5 Managing Systems Implementation	
186	Extract 6 Managing Systems Support and Security	
239	Extract 7 Enterprise Systems	
277	Extract 8 Information and Decision Support Systems	
331	Extract 9 Knowledge Management and Specialised...	



Chapter 1 is the first of three chapters in the systems planning phase. This chapter describes the role of information technology in today's dynamic business environment. This chapter describes the development of information systems, systems analysis and design concepts, and various systems development methods. This chapter also describes the role of the information technology department and its people.

LEARNING OBJECTIVES

When you finish this chapter, you should be able to:

- Describe the impact of information technology
- Define systems analysis and design and the role of a systems analyst
- Define an information system and describe its components
- Explain how to use business profiles and models
- Explain Internet business strategies and relationships, including B2C and B2B
- Identify various types of information systems and explain who uses them
- Distinguish among structured analysis, object-oriented analysis, and agile methods

Introduction to Systems Analysis and Design

The chapter includes four "Case in Point" discussion questions to help contextualize the concepts described in the text. The "Question of Ethics" invites examination of the ACM's code of ethics and those of a developing systems analyst.

CHAPTER CONTENTS

- 1.1 Introduction
- 1.2 What Is Information Technology? Case in Point 1.1: Cloud Nine Financial Advisors
- 1.3 Information System Components
- 1.4 Business Today
- 1.5 Modeling Business Operations
- 1.6 Business Information Systems
- 1.7 What Information Do Users Need?
- 1.8 Systems Development Tools
- 1.9 Systems Development Methods
- 1.10 The Information Technology Department Case in Point 1.2: Global Hotels and Momma's Motels
- Case in Point 1.3: What Should Lisa

Modelling Enterprise Systems

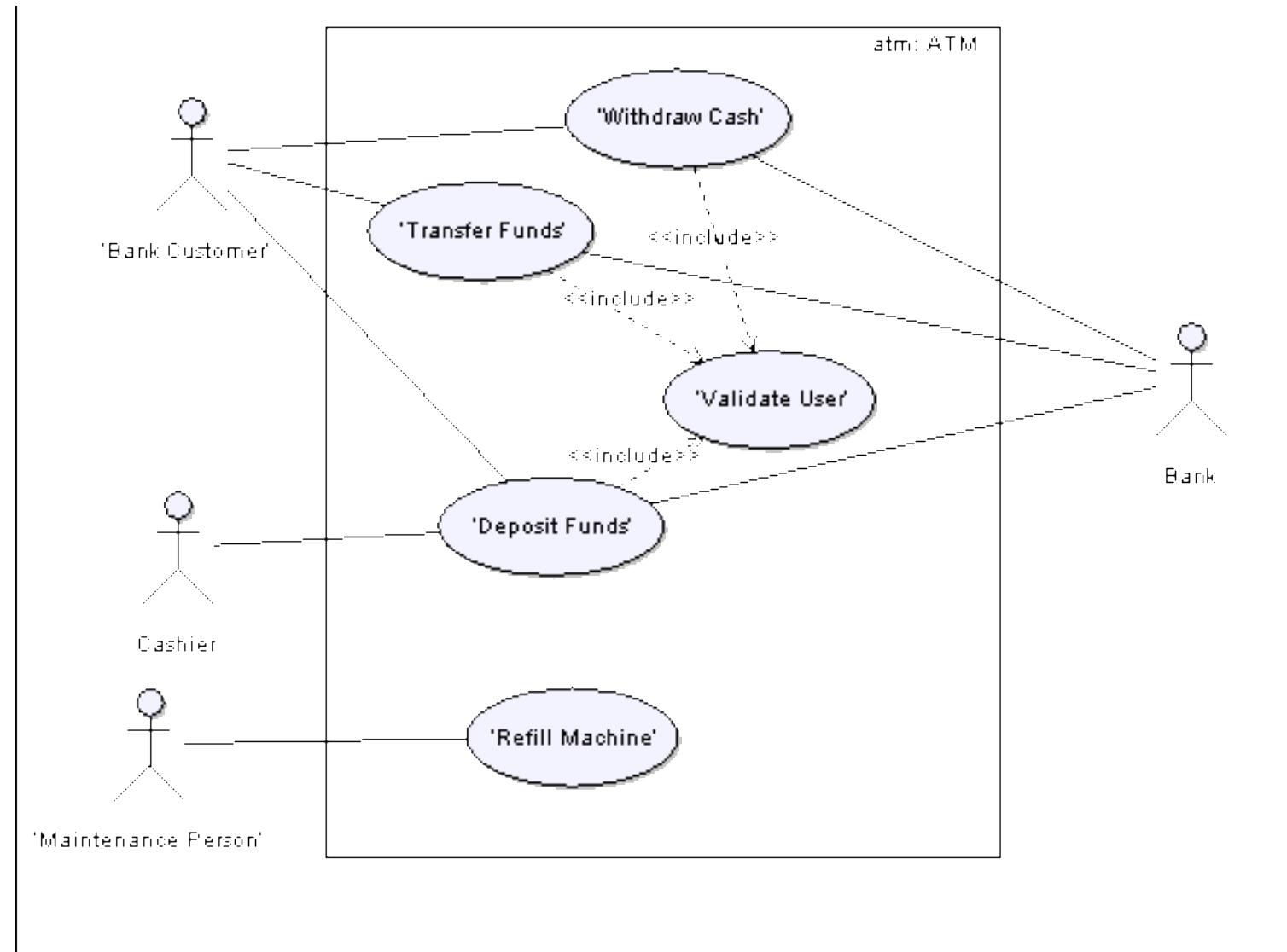
Modelling

- Analysts and Developers will use various models to learn, get feedback, and communicate what needs to be implemented and how it needs to be implemented
 - Requirements model
 - Describes the features that a system must provide,
 - **Data Flow Diagrams (DFD)**
 - Object model (O/O analysis/design)
 - Describes objects, which combine data and processes
 - Data model
 - Describes data structures and design
 - Network model
 - Describes the design and protocols of telecommunications links
 - Process model
 - Describes the logic of a process that programmers use to write code.

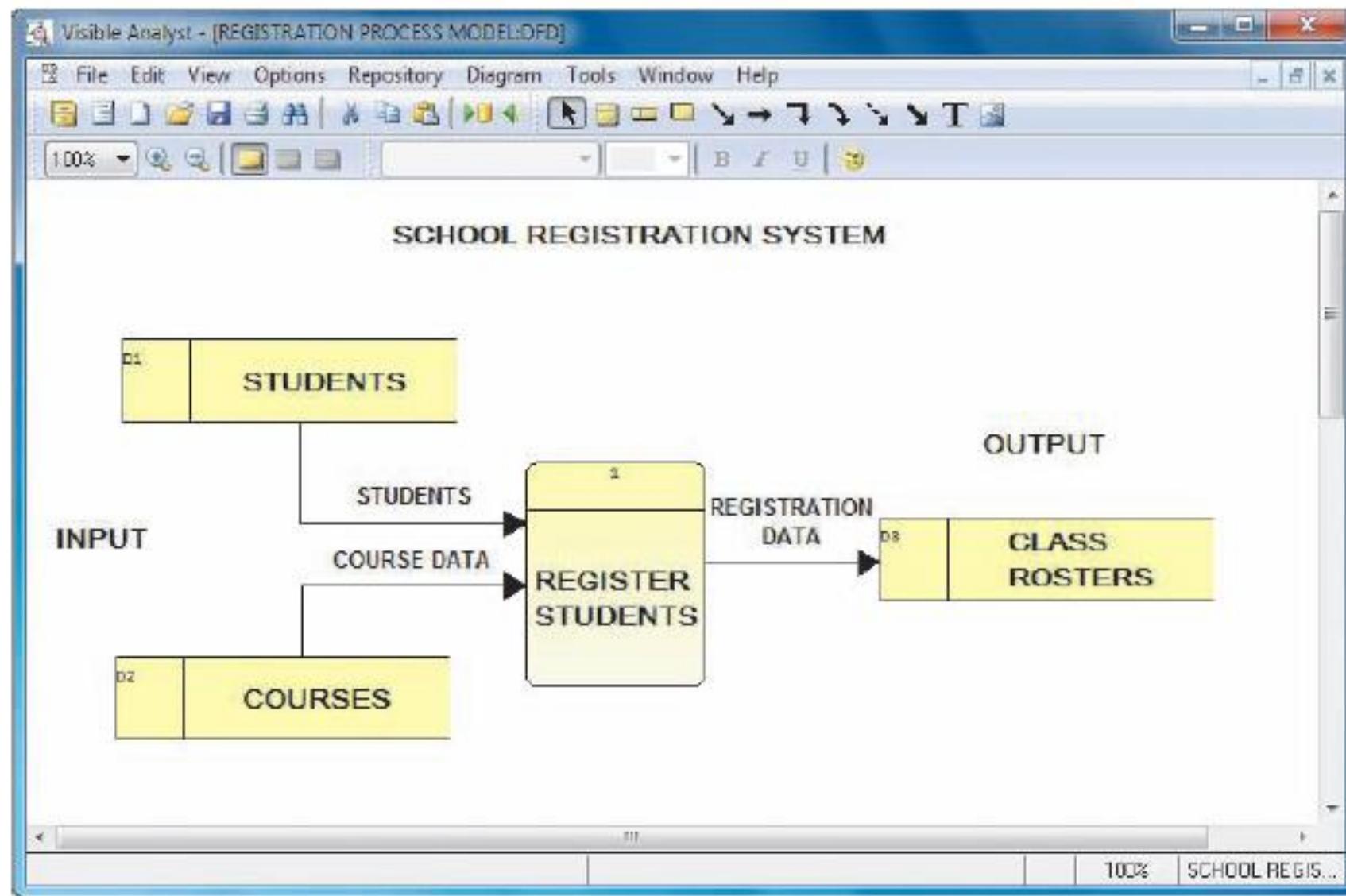


We will learn DFDs in labs

Use case diagram / model

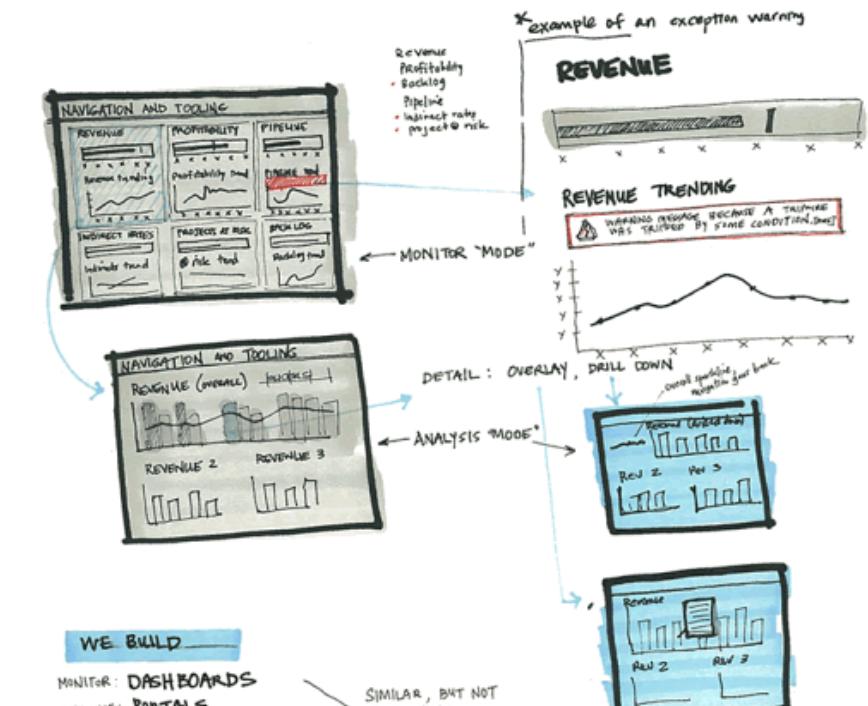
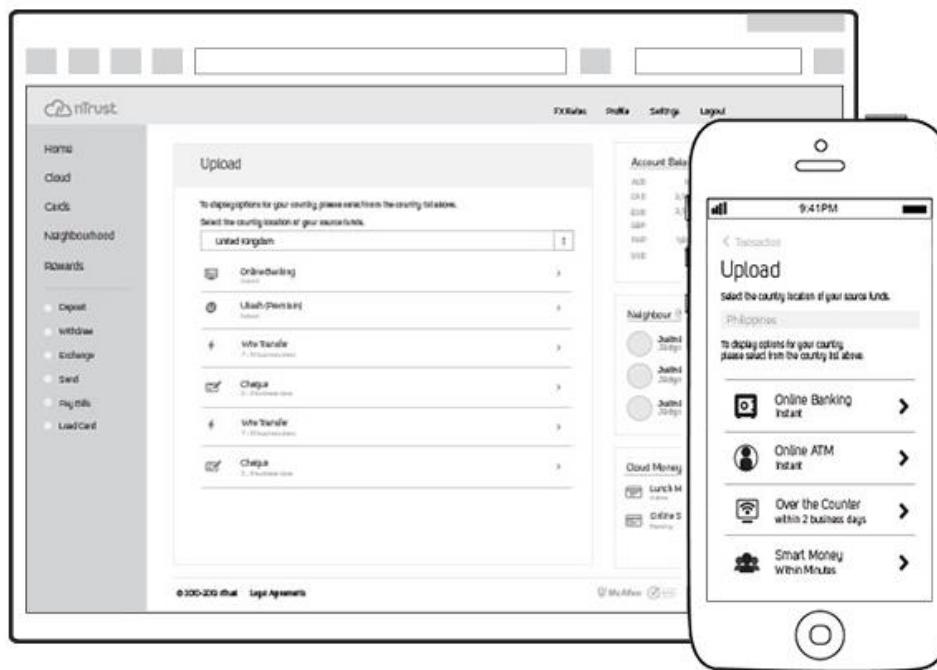


Data flow model example



Systems Development Technique

- Prototyping
 - Allows to identify, test, and get a feedback on features
 - It is a working version of a system but not fully functional system
 - Test input, output and user interface before making a final implementation decision



Systems Development Tools

- A confusing plethora of tools to help automate parts of the development lifecycle

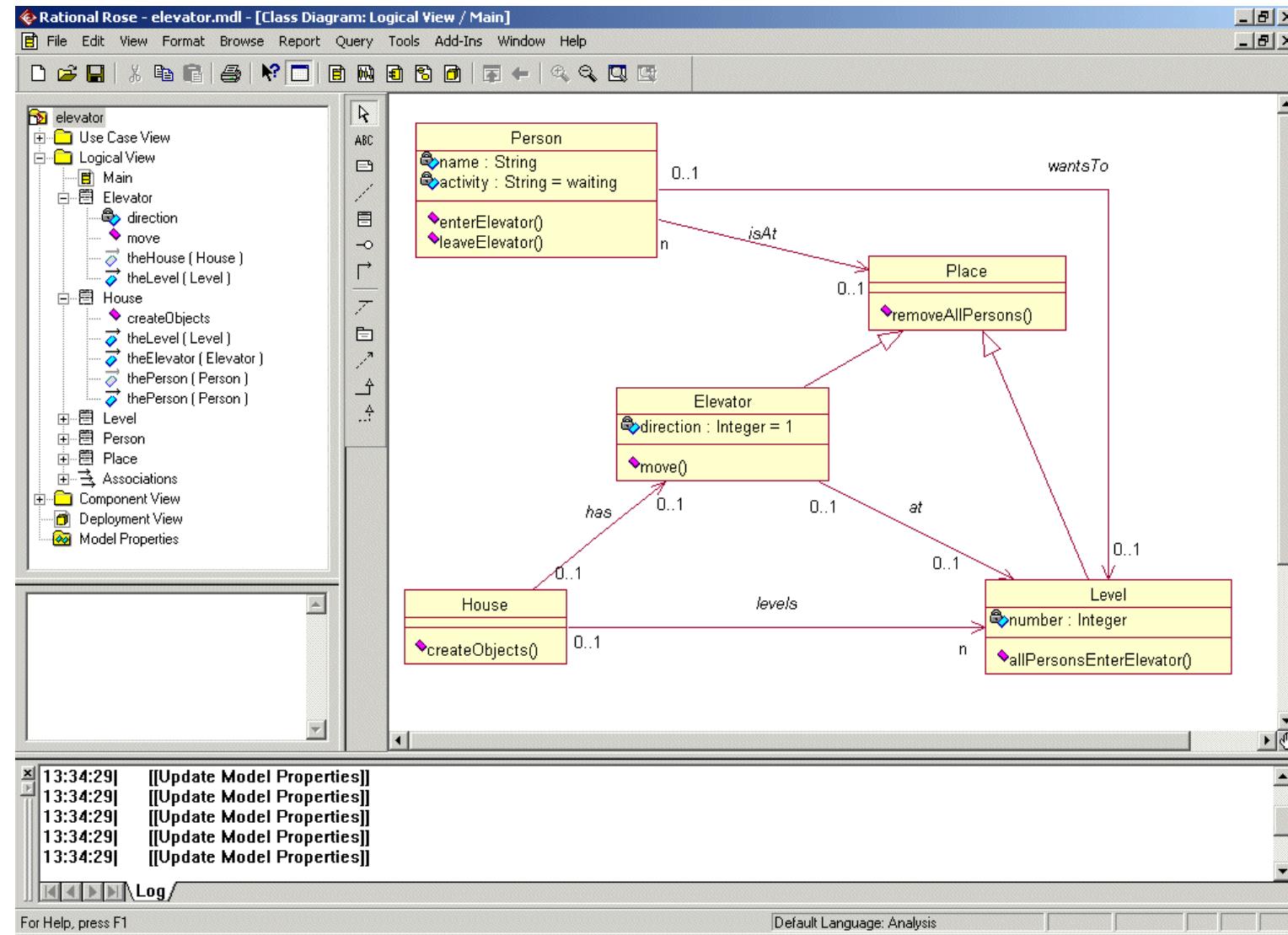
https://digital.ai/sites/default/files/pictures/2020-06/Digital.ai_Periodic-Table-of-DevOps.pdf

PERIODIC TABLE OF DEVOPS TOOLS (V2)																			
1	Fm	Gh	Github	Os	Os	4	Pd	Gt	Dm	Git	DBmaestro	Fr	Free	Scm	Scm	Build	Ch	En	
3														Fr	Repo Mgmt	Puppet	Pu	En	
5														Fm	Deployment	Ansible	An	Os	
7														Pd	Config / Provisioning	Sl	Dk	Os	
9														En	Release Mgmt	Docker	Az	Pd	
11	Fm	12	Os	Bb	Lb	Bitbucket	Liquibase	13	Os	14	En	15	Os	16	Fr	17	Os	18	En
19	Os	20	En	21	Os	22	Os	23	Os	24	Os	25	Fr	26	Os	27	Fr	28	Os
Gl	Rg	Mv	Gr	At	Fn	Se	Ga	Dh	Jn	Ba	Tr	Gd	Sf	Cn	Bc	Mo	Rs		
GitLab	Redgate	Maven	Gradle	ANT	FitNesse	Selenium	Gatling	Docker Hub	Jenkins	Bamboo	Travis CI	Deployment Manager	SmartFrog	Consul	Bcfg2	Mesos	Rackspace		
37	Os	38	En	39	Os	40	Os	41	Os	42	Fr	43	Os	44	Fr	45	Os	46	Fm
Sv	Dt	Gt	Gp	Br	Cu	Cj	Qu	Npm	Cs	Vs	Cr	Cp	Ju	Rd	Cf	Ds	Op		
Subversion	Datical	Grunt	Gulp	Broccoli	Cucumber	Cucumber.js	Qunit	npm	Codeship	Visual Studio	CircleCI	Capistrano	Juju	Rundeck	CFEngine	Swarm	OpenStack		
55	Os	56	En	57	Fr	58	Os	59	Os	60	Fr	61	Fr	62	Fr	63	Os	64	Fm
Hg	Dp	Sb	Mk	Ck	Ju	Jm	Tn	Ay	Tc	Sh	Cc	Ry	Cy	Oc	No	Kb	Hr		
Mercurial	Delphix	sbt	Make	CMake	JUNIT	JMeter	TestNG	Artifactory	TeamCity	Shipable	CruiseControl	RapidDeploy	CodeDeploy	Octopus Deploy	CA Nolio	Kubernetes	Heroku		
73	En	74	En	75	Os	76	Os	77	Fr	78	Os	79	En	80	Os	81	Os	82	Os
Cw	Id	Msb	Rk	Pk	Mc	Xltv	Jm	Nx	Co	Ca	So	Xld	EB	Dp	UD	Nm	Os	90	En
ISPW	Idera	MSBuild	Rake	Packer	Mocha	XL TestView	Jasmine	Nexus	Continuum	Continua CI	Solano CI	XL Deploy	ElectricBox	Deploybot	UrbanCode Deploy	Nomad	OpenShift		
91	En	92	En	93	En	94	En	95	En	96	En	97	En	98	Pd	99	Fm	10	Pd
Xlr	Ur	Bm	Hp	Au	Pl	Sr	Tfs	Tr	Jr	Rf	Sl	Fd	Pv	Sn					
XL Release	UrbanCode Release	BMC Release Process	HP Cedar	Atomic	Plutora Release	Serena Release	Team Foundation	Trello	Jira	HipChat	Slack	Flowdock	Pivotal Tracker	ServiceNow					
106	Os	107	Fm	108	Os	109	Os	110	En	111	Os	112	Os	113	En	114	Fm	115	Fm
Ki	Nr	Ni	Zb	Dd	EI	Ss	Sp	Le	SI	Ls	Gr	Fd	Pv	Sn	Tr	Ff			
Kibana	New Relic	Nagios	Zabbix	Datalog	Elasticsearch	StackState	Splunk	Logentries	Sumo Logic	Logstash	Graylog	Flowdock	Pivotal Tracker	ServiceNow	Tripwire	Fortify			

XebiaLabs
Deliver Faster

Follow @xebialabs

CASE tool – Rational Rose





Agile Models Distilled: Potential Artifacts for Agile Modeling

[Home](#)[Start Here](#)[Core Practices](#)[Disciplines](#)[Artifacts](#)[Resources](#)[Contact Me](#) Search

Choose Your WoW!

To be effective, the principle **Multiple Models** tells us that agile modelers should know a wide variety of modeling techniques so that they have the skills and knowledge to **apply the right artifact(s)** for the situation at hand. Unfortunately this is easier said than done. This page links to summary descriptions of a wide variety of modeling artifacts. Each page describes the artifact, provides an example or two, and provides links to suggested resources.

Some, but not all, of the potential models that you may want to create on a software development project include:

AGILE Modelling – Potential Models

- Acceptance Test
- Business Rule (Template)
- Change Case (Template)
- Class Responsibility Collaborator (CRC) model
- Constraint
- Contract model (Template)
- Data Flow Diagram (DFD)
- Domain Model
- Essential/Abstract Use Case (Template)
- Essential/Abstract User Interface Prototype
- Feature
- Free-Form Diagrams
- Flow Chart
- Glossary
- Logical Data Model (LDM)
- Mind Map
- Network Diagram
- Object Role Model (ORM) Diagram
- Personas
- Physical Data Model (PDM)
- Robustness Diagram
- Security Threat Model
- System Use Case (Template)
- Technical Requirement
- UML Activity Diagram
- UML Class Diagram



We will learn DFDs in labs

<http://agilemodeling.com/artifacts/>

Tutorial Session 6: Modelling Dataflows

1. Modelling dataflows and why?

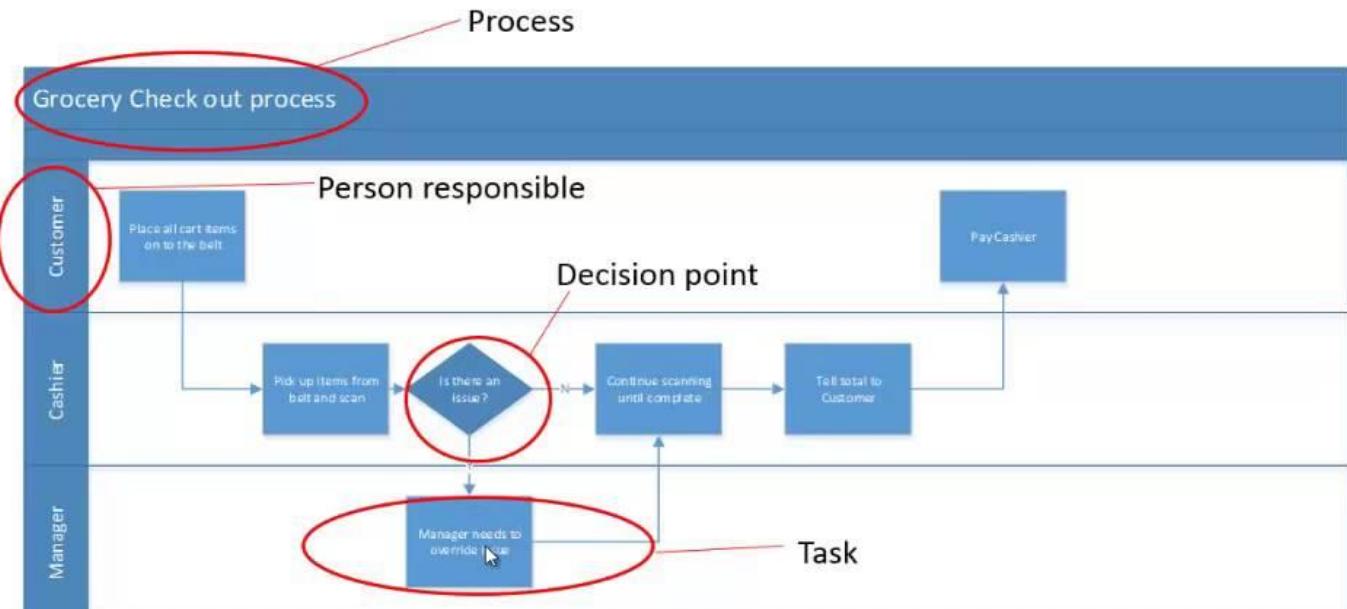
2. Blog Case – Digital Government: Remodelling a Process
The Service: Applying for NZ Superannuation
<https://www.digital.govt.nz/blog/user-focussed-content-equals-good-business/>

3. DFD's

Modelling Dataflows – Swimlane Process Maps

<https://www.youtube.com/watch?v=wQxnzLu7TqU>

Swim Lane Process Map



Modelling Dataflows – Swimlane Process Maps

1. Modelling Current and Future States

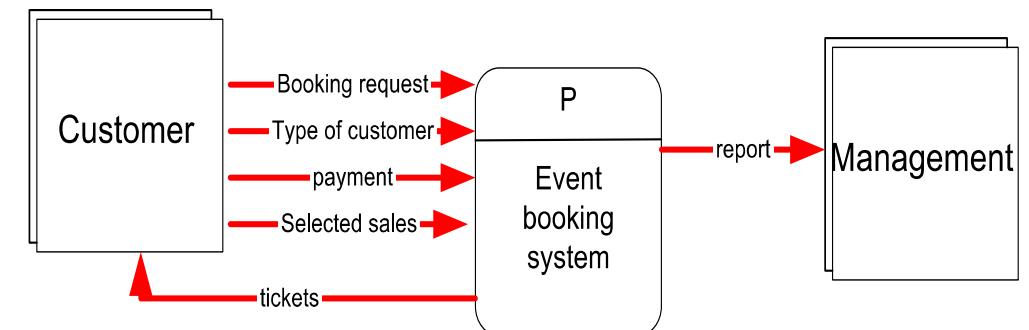
- Current State Modelling
 - <https://www.youtube.com/watch?v=lmjvmcuhaKY>
- Future State Modelling and Process Improvement Design
 - <https://www.youtube.com/watch?v=cAkPmSmK6ZM>

The background of the slide features a large, abstract graphic on the left side composed of numerous small, semi-transparent triangles. These triangles are arranged in a way that creates a sense of depth and movement, transitioning from dark red at the bottom to bright orange at the top. The right side of the slide is a plain white space.

Data Flow Diagrams (DFDs)

Objectives of this section

- Describe data and process modeling concepts and tools, including data flow diagrams, and process descriptions
- Describe the symbols used in data flow diagrams and explain the rules for their use
- Exposure to drawing **context diagram**



Process Modelling using DFDs

- DFDs show how the system transforms input data into useful information
- Technique for organising and documenting a system's **processes, inputs, outputs and data stores (storage)**
- Also includes the **external entities**.
- Process modelling is structured analysis tool which deals with a business process from the systems owners' and systems users' point of view
- What the system does or must do
- We use DFD to capture system's components (features) and external entities

Process modeling: Data Flow Diagrams (DFD)

- A data flow diagram (DFD) shows how **data moves** through an information system but does **not show program logic or processing steps**
- **Levels of modelling**
 - Context Diagram
 - High level (little detail, completeness of structure)
 - Low level (more detail, decomposed to smaller parts)



System Outline

- The first step towards identifying the components for the process driven model is to complete a System Outline.
- A System Outline identifies the system **inputs, outputs, processes, files (data storage) and external entities**. The processes are the events that occur in the system.
- The details from which the system outline can be derived, will be obtained by interviewing the users throughout the analysis and design phases of the **Systems Development Life Cycle**.

System Outline

Sample system outline. A large collection of these will be created to meet the needs of a complex system.

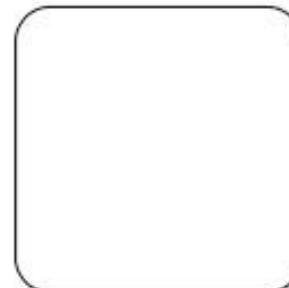
System Outline

Title	System	Document	Name	Sheet
Input				Processes
Files (Datastores)				Outputs
External Entities				
Author		Date		

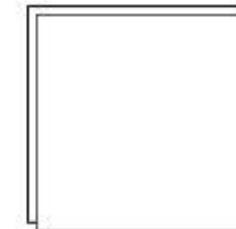
DFD Notations

Yourdon and
DeMarco

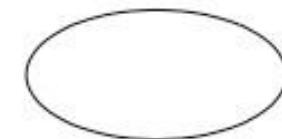
External Source
or destination of
data



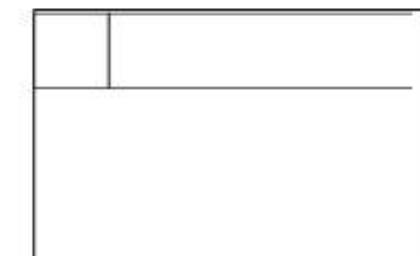
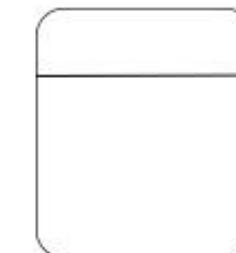
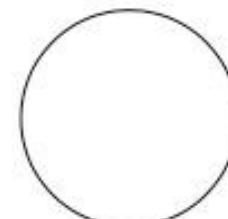
Gane and Sarson



SSADM



Process that
transforms the
data



Dataflow



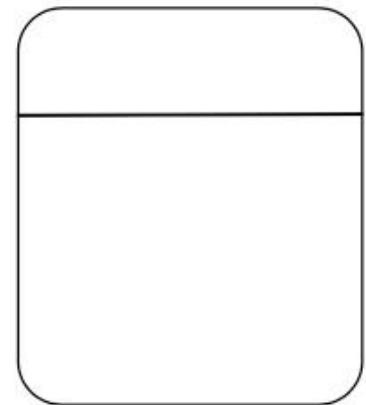
Datastore



Figure 2.5: Various DFD notations

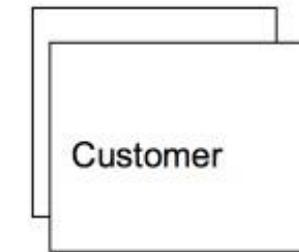
Data Flow Diagrams(DFDs) – Process Symbol

- Drawn as **rectangle with round corners**. Each process must be numbered
- Name of the process starts with a verb followed by a singular object e.g. calculate gross pay, produce invoice, validate customer
- Receives input data and produces output that has a different content, form, or both
- Contain the business logic, also called business rules
- Referred to as a black box



Data Flow Diagrams(DFDs) - External Entities

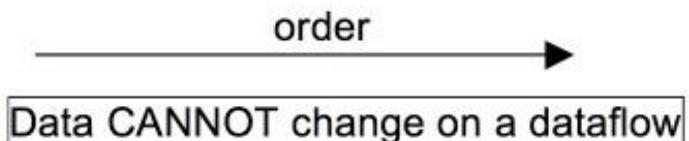
- Provide input or receive output from the system
- **Source** (provides data [input] to the system) or a **Sink** (gets the data [output] from the system)
- Drawn as a **square with another square behind it**
- E.g. Customer, Supplier, IRD, Employee etc.



(a) **External Entity** is a source or sink for data. This is drawn as a square with another square behind it. Examples: Customer, IRD, Employee, Supplier

Data Flow Diagrams(DFDs) – Data Flow

- Drawn as an **arrow**. Arrow head indicates the direction of the flow
- Dataflow name is singular, written on top of the arrow
- E.g. Invoice, payslip, timesheet, order



Data CANNOT change on a dataflow

(b) **Dataflow**. Drawn as an arrow. The arrowhead indicates the direction of the flow. The dataflow name is usually singular. Examples: invoice, payslip, timesheet, order. We are only concerned with the data content of the flow — NOT with the number of copies of the data.

Data Flow Diagrams(DFDs)

- **Data store (storage)**

- Drawn as an **open-ended rectangle**
- Data is stored for use by a process at a later time
- Data storage also provides input into another process
- **Name of data store is usually plural**
- Each data store is usually numbered for reference purpose e.g. D1
- **Customer master file named as customers, employee master file named as employees**

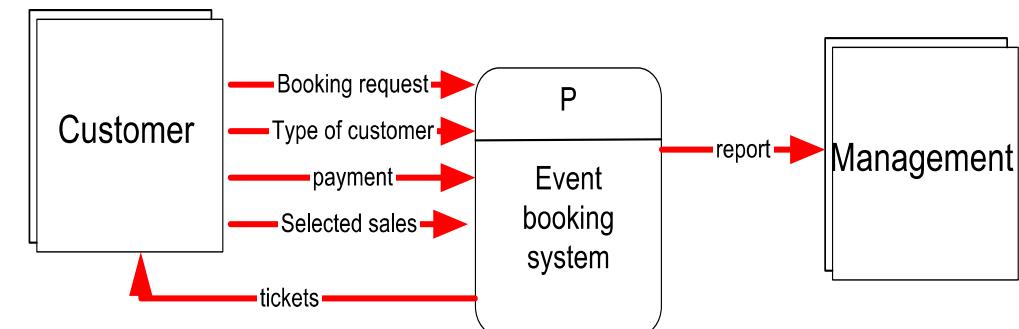
- **Duplicates**

- Sometimes it is necessary to repeat the a symbol on a DFD in order to avoid crossing lines



Drawing context diagram

- Draw the context diagram so that it fits on one page
- Has a **single process** representing the entire system. The name of the process is name of the system
- Identify **all the external entities** to go with the process.
- Shows the system as a single process with external entities
- External entities show all the input and output they provide and receive from the system
- Context diagram helps the analyst to gain an overall view of the system he/she is investigating



DFD Modelling

1. DFD's

Enrolling in the University

<http://agilemodeling.com/artifacts/dataFlowDiagram.htm>

2. Blog Case – Digital Government

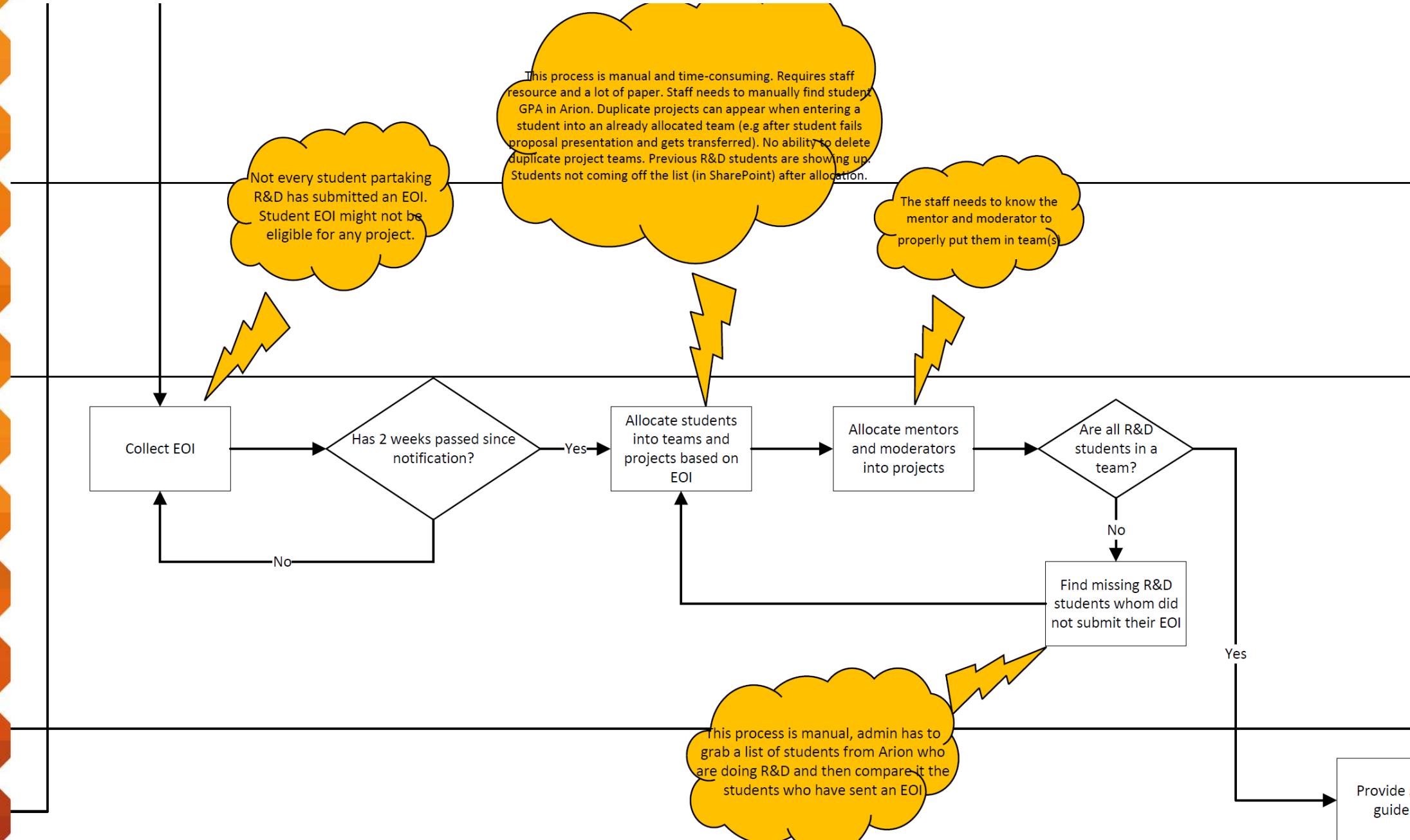
The Service: Applying for NZ Superannuation

<https://www.digital.govt.nz/blog/user-focussed-content-equals-good-business/>

DFD Modelling – BCIS R&D Project - S2/ 2021

1. Consultation Report
2. Pain points – to see next slide
3. Discussion and issues
4. Options and Implications

DFD Modelling – BCIS R&D Project - Pain Points



Tutorial Session 6: Modelling Dataflows (1)

THE STATIONERY STORES SYSTEM

The Stationery Stores System is a case study used by Park Place Training (see References) to teach the general principles of structured analysis and design. For the purpose of illustrating the approach to Mk II function point counting, we will only use a small part of the overall case study. This part concerns the processes from the time when the Stores sees the need to re-order some stationery, through the selection of possible suppliers based on past performance, the preparation of requests for quote, and their issue to suppliers, and the receipt and registration of quotes from suppliers.

In the case study, we are at a point where Data Flow Diagrams ("DFD's") have been produced for the required processes (referred to as a 'First Cut Functional Design'), data entity analysis has been carried out, and the

Tutorial Session 6: Modelling Dataflows (2)

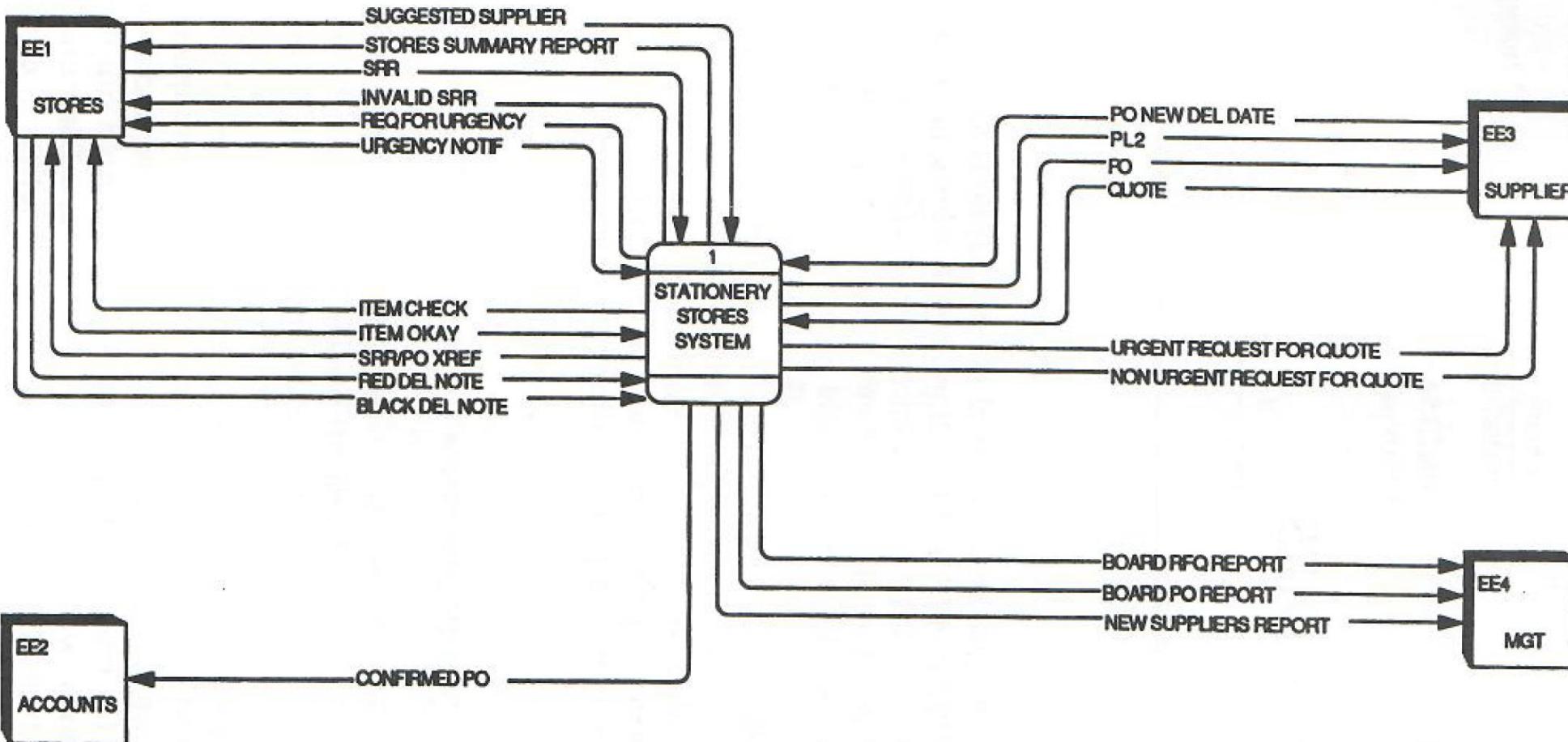


Figure 26 Stationery Stores First Cut Functional Design—Context

Case

Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.

Tutorial Session 6: Modelling Dataflows (3)

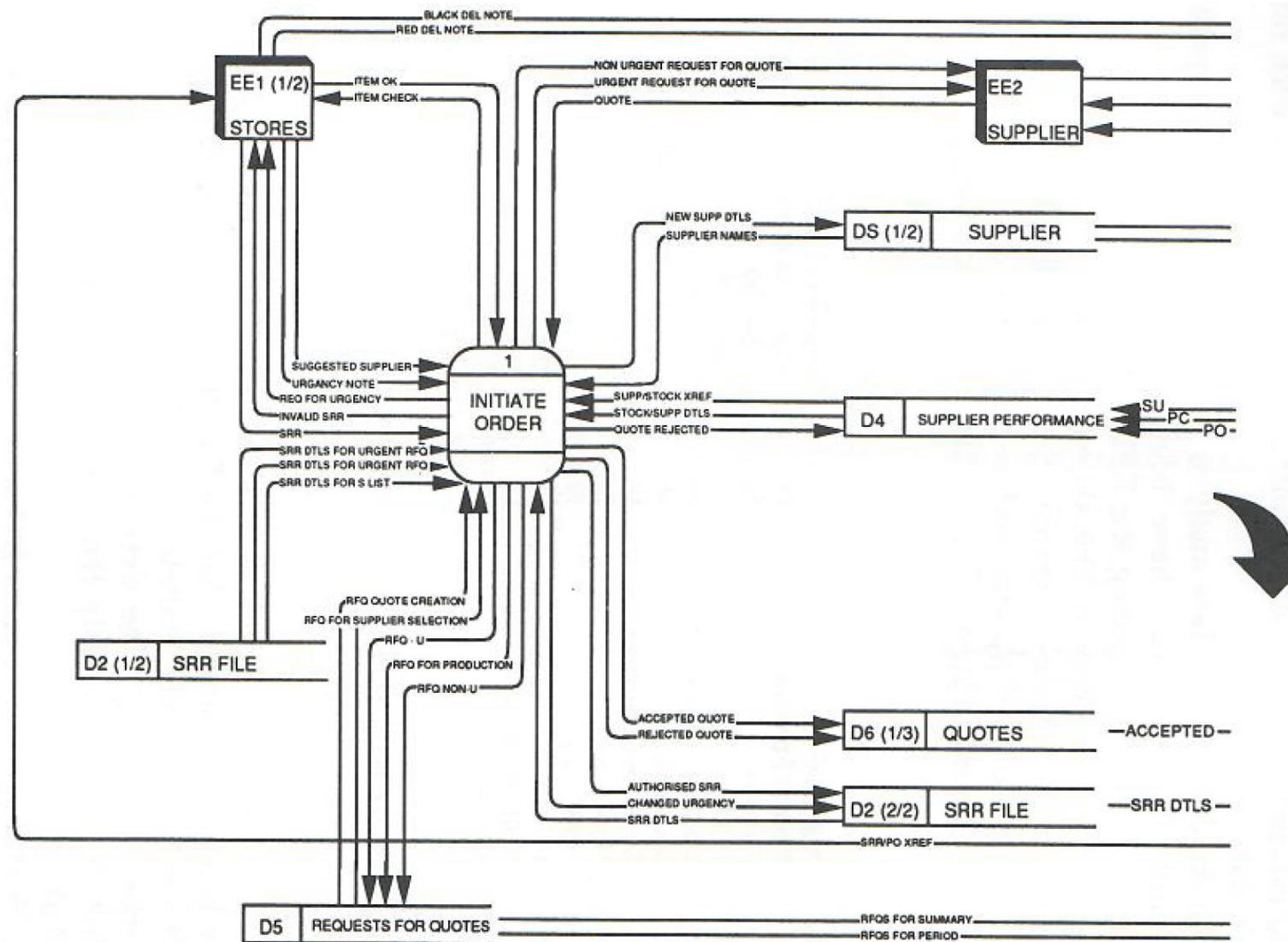


Figure 28(a) First Cut Functional Design—Level 0

Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.

A large, abstract graphic on the left side of the slide features a repeating pattern of orange triangles. The triangles are arranged in a way that creates a sense of depth and movement, resembling a stylized flame or a rising sun. The colors range from bright orange at the top to darker shades of orange and red towards the bottom.

Extend your knowledge

1. Watch a video on – what is an Information System
<https://youtu.be/Qujsd4vkqFI>
2. Explore job information about Business / System Analyst
<http://tinyurl.com/ztzulel>

References

- Baecker, R. M. (2019). *Computers and society: Modern perspectives*: Oxford University Press, USA.
- Kaczmarczyk, L. C. (2016). *Computers and society: computing for good*: CRC Press.
- Stair, R., & Reynolds, G. (2020). *Principles of information systems*: Cengage Learning.
- Litchfield, A. (2017). *INFS500 Enterprise Systems - Bachelor of Computer and Information Science: Process Modelling Workbook*. Auckland: Auckland University of Technology.
- Symons, C. R. (1991). *Software sizing and estimating: Mk II FPA (function point analysis)*: John Wiley & Sons, Inc.