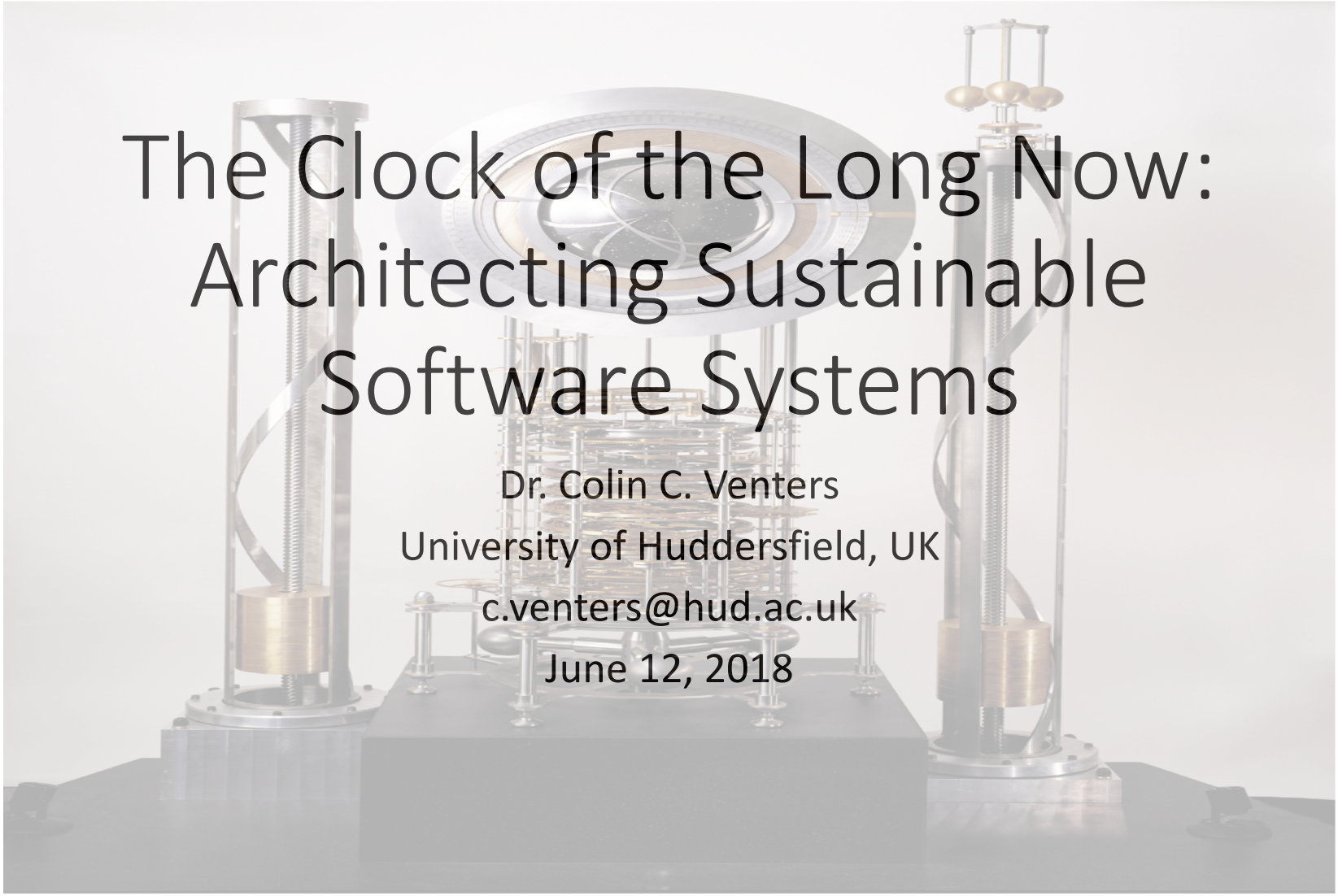
https://www.youtube.com/watch?v=VSEYmszKpBs

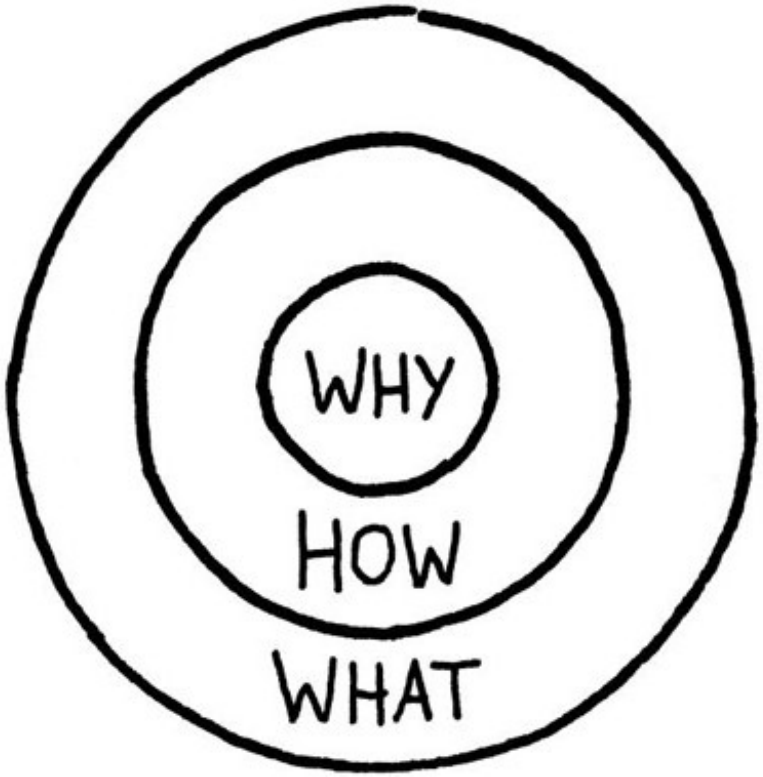


The Clock of the Long Now: Architecting Sustainable Software Systems

Dr. Colin C. Venters
University of Huddersfield, UK
c.venters@hud.ac.uk

June 12, 2018

Software Sustainability



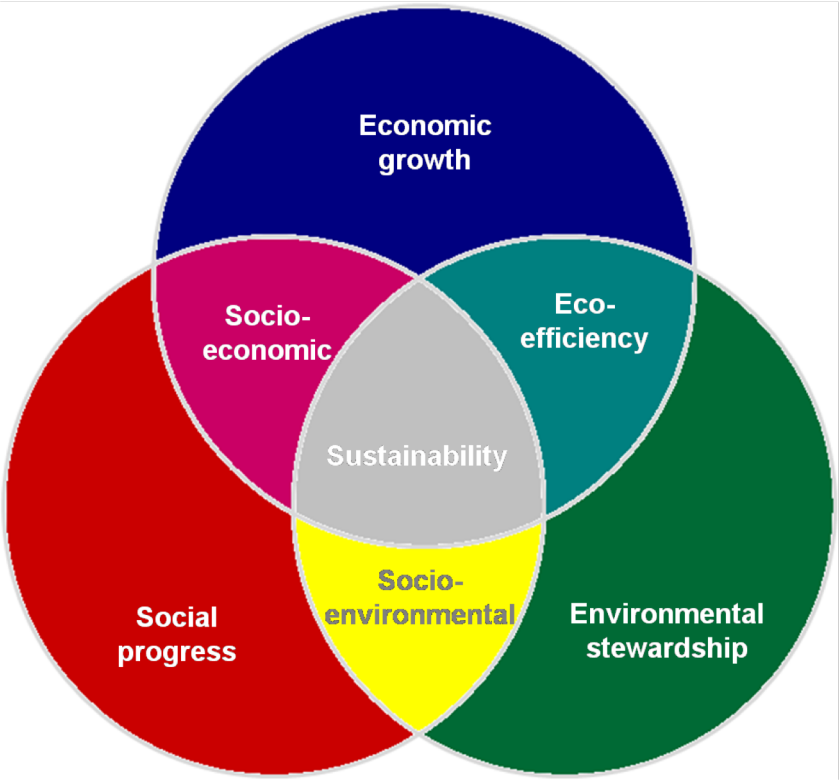

SAY
WHAAAT

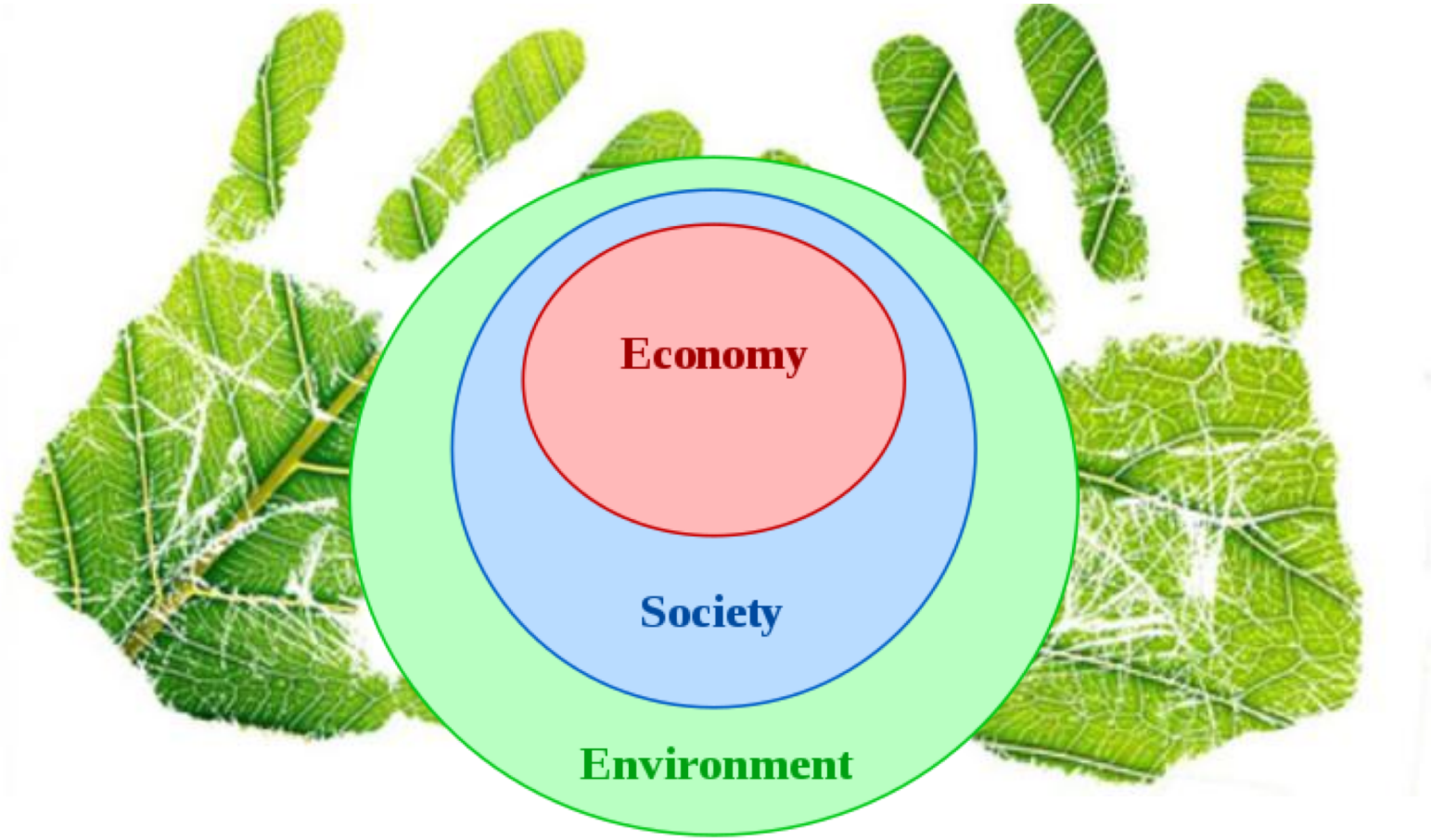

WHAT DOES
SUSTAINABILITY
MEAN TO YOU?


consumption
renewable energy
earth
environmental friendly
clean
operation
sources
education
negative
nature
water sustainably
solar
impact
planet
tree
power
business
natural
plant
industrial
management
human rights
conservation
environment
sustainable business
green business
needs
government
recyclable
society
environmental
minimal negative
water
economy
minimal
industry
responsibility
sustainable distribution
friendly
electricity
respect
resource
participatory
organization
production
healthy
standards
sustainability
global
waste
fuels
solution
alternative
natural environment
technology
organic
ecological
use
globalization
ethics
wind
green
innovation
windmill
economic
efficient
modern
future
recycle
agriculture
sustainable
sort
multiple cropping
growth
geothermal
social
patterns
mixed farming
world
road-map
action plan
eco
development
policy
renewable
life
plants
sustain
progressive
reduce
change
energy
ecology
engineering
pollution
climate
agroforestry
human
permaculture
population
efficiency
crop rotation
recycling

A long time ago in a galaxy far, far away…

- Environmental sustainability is concerned with [minimizing] the [impact] on the environment and natural resources.
- Economic sustainability is concerned with economic [growth] and its [impact] on social or natural resources.
- Social sustainability is concerned with building [social equity].




Economy
Society
Environment

“You keep using <u>that</u> Word…”

https://www.youtube.com/watch?v=G2y8Sx4B2Sk

바벨탑

Position Paper: RE4SuSy: Third International Workshop on Requirements Engineering for Sustainable Systems

Software Sustainability: The Modern Tower of Babel

C. C. Venters, [2]C. Jay, [3]L. M. S, Lau, [4]M. K. Griffiths, [1]V. Holmes, [1]R. R. Ward, [5]J. Austin, [6]C. E. Dibsdale, and [3]J. Xu

[1]University of Huddersfield
School of Computing & Engineering
Huddersfield, UK
{v.holmes; r.r.ward}@hud.ac.uk

[2] University of Manchester
School of Computer Science
Manchester, UK
caroline.jay@cs.manchester.ac.uk

[3]University of Leeds
School of Computing & Informatics
Leeds, UK
{l.m.s.lau; j.xu}@leeds.ac.uk

[4]University of Sheffield
Sheffield, UK
m.griffiths@sheffield.ac.uk

[5]University of York
Department of Computer Science
York, UK
austin@cs.york.ac.uk

[6]Optimized Systems and Solutions Ltd,
Derby, UK
charlie.e.dibsdale@o-sys.com

***Abstract*— The development of sustainable software has been identified as one of the key challenges in the field of computational science and engineering. However, there is currently no agreed definition of the concept. Current definitions range from a composite, non-functional requirement to simply an emergent property. This lack of clarity leads to confusion, and potentially to ineffective and inefficient efforts to develop sustainable software systems. The aim of this paper is to explore the emerging definitions of software sustainability from the field of software engineering in order to contribute to the question, what is software sustainability? The preliminary analysis suggests that the concept of software sustainability is complex and multifaceted with any consensus towards a shared definition within the field of software engineering yet to be achieved.**

***Index Terms*— Non-functional requirements, quality attributes, software engineering, software sustainability, sustainability**

I. Introduction

The concept of sustainability has principally been associated with ecology, and the relationship between humans and planet Earth [1]. In recent years, software sustainability has emerged as an area of research in the field of software engineering and has been identified as an important future topic as new approaches to research become increasingly dependent on complex software systems, which operate in evolving, distributed e-infrastructure eco-systems [2].

Its importance has been further underlined by recent funding initiatives from the National Science Foundation [3] in the US and the Engineering and Physical Sciences Research Council [4] in the UK, combined with the establishment of the Software Sustainability Institute [5]. In addition, a number of workshops have emerged which are dedicated to exploring the topic of sustainable software and systems from a range of different perspectives [6-7].

Fundamental to the advancement of software sustainability as a field of research requires an understanding of the concept [8]. However, there is no agreed definition of software sustainability. While there have been a number of contributions to formalize a definition of software sustainability, the concept remains an elusive and ambiguous term with individual's, groups and organizations holding diametrically opposed views [9]. However, this is not a problem unique to the field of software engineering [10-11].

The narrative of the 'Tower of Babel' provides a useful analogy to describe the current understanding of the concept of software sustainability both within and outside the community. While there is no divine intervention at its source, there is a considerable amount of confusion and divergence regarding what software sustainability means, how it can be measured or demonstrated, or how to train and educate the broad spectrum of domain scientists and advance the skills of software engineers to develop sustainable software [2, 12-13]. The principal aim of this paper is to explore the definitions that have emerged from the field of software engineering in order to address the question, *what is software sustainability*? Section 2 examines the concept of software sustainability from a software artifact perspective. Section 3, examines definitions which focus on the software development process. Section 4 examines software sustainability as a non-functional requirement. Section 5 examines the use of software sustainability frameworks for exploring sustainability. Section 6 considers whether software sustainability is an emergent property. In Section 7, conclusions are drawn and future directions are outlined.

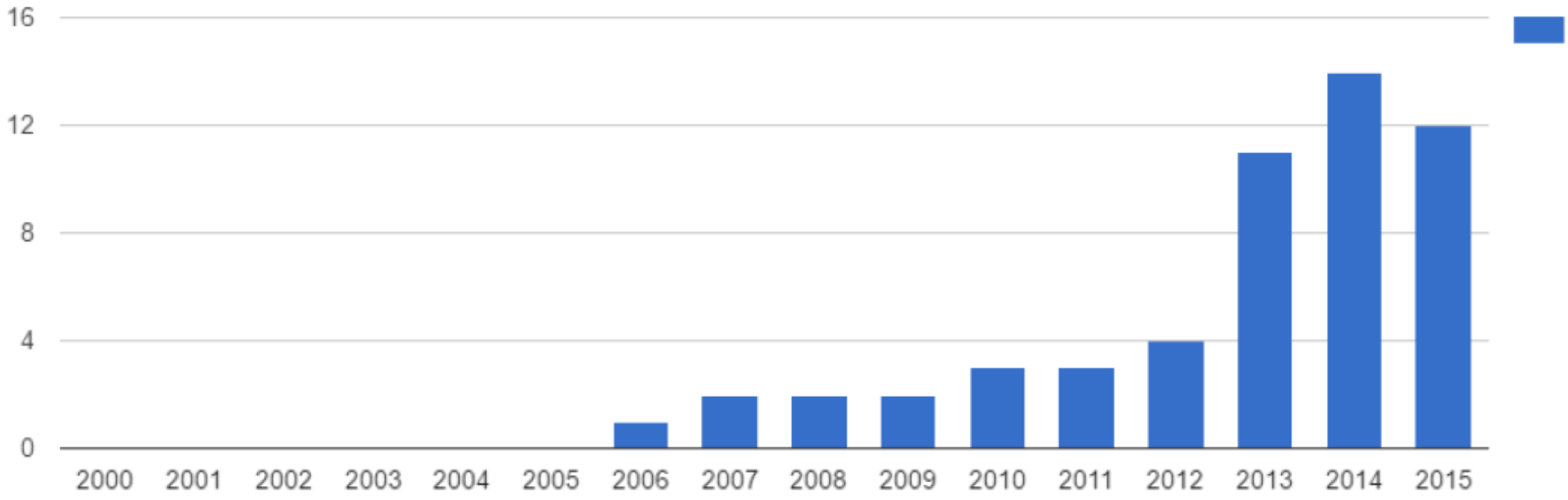
II. Software Sustainability

The word sustainability is derived from the Latin *sustinere*. The Oxford English Dictionary [14] defines sustainability as '*the quality of being sustained*', where sustained can be defined as '*capable of being endured*' and '*capable of being 'maintained*'. Endured is defined as '*continuing to exist*' and maintained as '*being supported*' [14]. This suggests that time or longevity and maintenance are important factors in understanding sustainability. The most widely adopted definition of sustainability is that proposed by the Brundtland commission which defined sustainability as '*meeting the needs of the present without compromising the ability of future generations to meet their needs*' [15]. However, this definition is rather broad and difficult to understand and apply in any meaningful way. In recent years, a triple bottom line perspective of sustainability has been adopted which considers sustainability to include three components: environment, society and economy [10]. It is argued that by incorporating the three dimensions it leads to more sustainable outcomes [16].

A number of definitions have emerged from the field of software engineering, which focuses on the sustainability of the

Venters, Colin, Jay, Caroline, Lau, Lydia, Griffiths, Michael K., Holmes, Violeta, Ward, Rupert, Austin, Jim, Dibsdale, Charlie E. and Xu, Jie (2014) Software Sustainability: The Modern Tower of Babel. Proceedings of the Third International Workshop on Requirements Engineering for Sustainable Systems

Sustainability Requirements?



Venters, Colin, Seyff, Norbert Becker, Christoph, Betz, Stefanie, Chitchyan, Ruzanna, Duboc, Leticia, McIntyre, Dan, and Penzenstadler, Birgit (2017). Characterising Sustainability Requirements: A New Species, Red Herring or Odd Fish? In: ICSE'17: 39th International Conference on Software Engineering, Buenos Aires, Argentina, May 20-28, 2017

Sustainability Requirements?

- How does current research construct the notion of sustainability requirements through published work?

Characterising Sustainability Requirements:
A New Species, Red Herring, or Just an Odd Fish?

Colin C. Venters
University of Huddersfield
Huddersfield, UK
c.venters@hud.ac.uk

Norbert Seyff
University of Zurich
Zurich, Switzerland
seyff@ifi.uzh.ch

Christoph Becker
University of Toronto
Toronto, ON, Canada
christoph.becker@utoronto.ca

Stefanie Betz
Karlsruhe Insitute of Technology
Karlsruhe, Germany
stefanie.betz@kit.edu

Ruzanna Chitchyan
University of Leicester
Leicester, UK
rc256@leicester.ac.uk

Leticia Duboc
State Univ. of Rio de Janeiro
Rio de Janeiro, Brazil
leticia@ime.uerj.br

Dan McIntyre
University of Huddersfield
Huddersfield, UK
d.mcintyre@hud.ac.uk

Birgit Penzenstadler
CSU Long Beach
Long Beach, CA, USA
birgit.penzenstadler@csulb.edu

***Abstract*—Requirements articulating the needs of stakeholders are critical to successful system development and key to influencing their long-term effects. As the concept of sustainability has entered the discourse of a number of software-related computing fields, so has the term 'sustainability requirement'. However, it is unclear whether sustainability requirements are and should be different from how we already understand software requirements. This paper presents the results of a corpus-assisted discourse analysis study that explored the concept of sustainability requirements in order to understand how the term is being used in software and requirements engineering and related fields. The results of this study reveal that the term 'sustainability requirement' is generally used ambiguously and reveals significant segmentation across different fields. Our detailed analysis of selected influential papers highlights the segmented use of the term and suggests key focus questions that need to be addressed to establish a shared operative understanding of the term.**

I. Introduction

Modern societies are increasingly dependent on complex software and software systems, which underlie almost every aspect of day to day living from transportation, finance, education, retail, communication, governance, healthcare and fitness, entertainment and leisure, defense and security etc. [1].

Requirements are the foundation of all software products [2]. Failure to produce a set of software requirements that satisfies the primary needs of its users and other stakeholders can result in significant economic consequences [3]. As a result, the field of Requirements Engineering (RE) plays a critical role in software system development and is considered to be the key leverage point for practitioners who want to design sustainable software-intensive systems [4].

There are many ways to characterise sustainability [5], but it is generally defined as 'the capacity to endure' [6]. To increase the understanding and tangibility of this abstract concept, Tainter [7] suggests reflecting on four points when thinking about sustainability: What should be sustained? For whom? For how long? At what cost? As part of the concept of sustainable development, a widely adopted characterization proposed by Brundtland [8] emphasizes its focus on 'meeting the needs of the present without compromising the ability of future generations to meet their own needs'. This perspective emphasizes the view that development has effects that lie outside the system to be developed. The word 'need' is central to this definition and includes a dimension of time, present and future.

While sustainability is difficult to define as it needs context and (social) structure [9], it has become clear that the concept of sustainability requires simultaneous consideration of several interrelated dimensions [4]. Consequently, we characterize it using the following five dimensions: (1) *Environmental* refers to the responsible use and stewardship of natural resources. (2) *Economic* focuses on assets, capital and added value, which includes wealth creation, prosperity, profitability, capital investment, income, etc. (3) *Individual* covers individual freedom and agency. (4) *Social* is concerned with societal communities (groups of people, organizations) and the factors that erode trust in society. (5) *Technical* relates to the ability to maintain and evolve artificial systems over time.

The concept of sustainability has emerged as a topic of interest in different areas of computing such as artificial intelligence, high-performance computing, human-computer interaction, software engineering (SE), and requirements engineering. While there have been attempts to understand how sustainability is perceived in the practice of software engineering and how sustainability can become an inherent part of software engineering practice [10], consensus on what sustainability means in the field of software and requirements engineering is still emerging [11], [12].

The emergence of sustainability included the introduction of the term 'sustainability requirement' in software and requirements engineering literature with a number of approaches being proposed for eliciting, modelling, managing, and capturing reusable knowledge with regards to sustainability requirements [13], [14], [15], [16], [17].

However, it is unclear how this term is being defined,

Venters, Colin, Seyff, Norbert Becker, Christoph, Betz, Stefanie, Chitchyan, Ruzanna, Duboc, Leticia, McIntyre, Dan, and Penzenstadler, Birgit (2017). Characterising Sustainability Requirements: A New Species, Red Herring or Odd Fish? In: ICSE'17: 39th International Conference on Software Engineering, Buenos Aires, Argentina, May 20-28, 2017

Sustainability Collocates

Rank	Freq.	Freq (L)	Freq. (R)	MI	Token
1	20	17	3	6.59	theme
2	123	70	53	6.24	dimension
3	163	73	90	6.14	dimensions
4	30	20	10	5.91	improving
5	103	12	91	5.83	debt
6	34	31	3	5.78	integrate
7	70	58	12	5.78	generic
8	230	195	35	5.72	social
9	21	21	0	5.61	integrating
10	22	20	2	5.58	corporate

Sustainability Requirement* Collocates

Rank	Freq.	Freq(L)	Freq(R)	Score	Collocate
1	48	0	48	4.54	requirement
2	247	13	234	4.19	requirements
3	22	3	19	3.11	patterns
4	13	12	1	1.98	technical
5	11	11	0	1.34	social
6	11	9	2	0.76	environmental
7	19	7	12	0.73	we
8	59	37	22	0.54	for
9	10	5	5	0.52	engineering
10	18	4	14	0.31	software

Is Sustainability a…?

Is Sustainability…

The First Rule of Software Sustainability?

Karlskrona Consortium #icse15

www.sustainabilitydesign.org

Karlskrona Manifesto for Sustainability Design

- Sustainability is a concern independent of the purpose of the system.
  - Sustainability has to be considered even if the primary focus of the system under design is not sustainability.

Sustainability Design and Software:
The Karlskrona Manifesto

Christoph Becker
Faculty of Information
University of Toronto
Toronto, ON, Canada
christoph.becker@utoronto.ca

Ruzanna Chitchyan
Dept of Computer Science
University of Leicester
Leicester, UK
rc256@leicester.ac.uk

Leticia Duboc
Dept of Inf. & Computer Science
State Univ. of Rio de Janeiro
Rio de Janeiro, Brazil
leticia@ime.uerj.br

Steve Easterbrook
Dept of Computer Science
University of Toronto
Toronto, ON, Canada
sme@cs.utoronto.ca

Birgit Penzenstadler
Institute for Software Research
University of California, Irvine
Irvine, California, US
bpenzens@uci.edu

Norbert Seyff
Dept of Informatics
University of Zurich
Zurich, Switzerland
seyff@ifi.uzh.ch

Colin C. Venters
School of Computing & Engineering
University of Huddersfield
Huddersfield, UK
c.venters@hud.ac.uk

***Abstract*—Sustainability has emerged as a broad concern for society. Many engineering disciplines have been grappling with challenges in how we sustain technical, social and ecological systems. In the software engineering community, for example, maintainability has been a concern for a long time. But too often, these issues are treated in isolation from one another. Misperceptions among practitioners and research communities persist, rooted in a lack of coherent understanding of sustainability, and how it relates to software systems research and practice. This article presents a cross-disciplinary initiative to create a common ground and a point of reference for the global community of research and practice in software and sustainability, to be used for effectively communicating key issues, goals, values and principles of sustainability design for software-intensive systems. The centrepiece of this effort is the *Karlskrona Manifesto for Sustainability Design*, a vehicle for a much needed conversation about sustainability within and beyond the software community, and an articulation of the fundamental principles underpinning design choices that affect sustainability. We describe the motivation for developing this manifesto, including some considerations of the genre of the manifesto as well as the dynamics of its creation. We illustrate the collaborative reflective writing process and present the current edition of the manifesto itself. We assess immediate implications and applications of the articulated principles, compare these to current practice, and suggest future steps.**

I. Introduction

It is clear that society is facing major sustainability challenges that require long-term, joined-up thinking. How do we sustain our technical infrastructures, given how much we rely on them for everything from communication and navigation through to storing health records, identifying security threats, and keeping the lights on? How do we sustain prosperity in society, given the erosion of trust in our political institutions and a growing inequality in ownership of resources? And, above all, how do we sustain the planetary systems that support life on earth, in the face of accumulation of pollutants, species loss, and accelerating climate change?

The discipline of Software Engineering (SE) has a major role to play in sustainability, because of the extent to which software systems mediate so many aspects of our lives. However, software practice has a tendency to focus only on the immediate effects and tangible benefits of software products and platforms. SE research has, for the most part, focused on increasing the reliability, efficiency and cost-benefit relation of software products for their owners, through a focus on processes, methods, models and techniques to create, verify and validate software systems and keep them operational.

The lack of long-term thinking in software engineering research and practice has been critiqued throughout the history of the discipline. For example, software maintenance and evolution were raised as concerns even at the very first software engineering conference [1]. Since then, efforts to increase the maintainability of software products and facilitate their evolution have often focused on improving architecture, decreasing lifecycle costs and managing technical debt [2]. Neumann has criticized the lack of long-term thinking over security considerations in SE [3]. For our digital information assets, some now speak of a 'digital dark age' [4], where, having discarded analog media in preference for digital, we now find that many of these assets become unreadable, due, in part, to the rapid lifecycles of software technology.

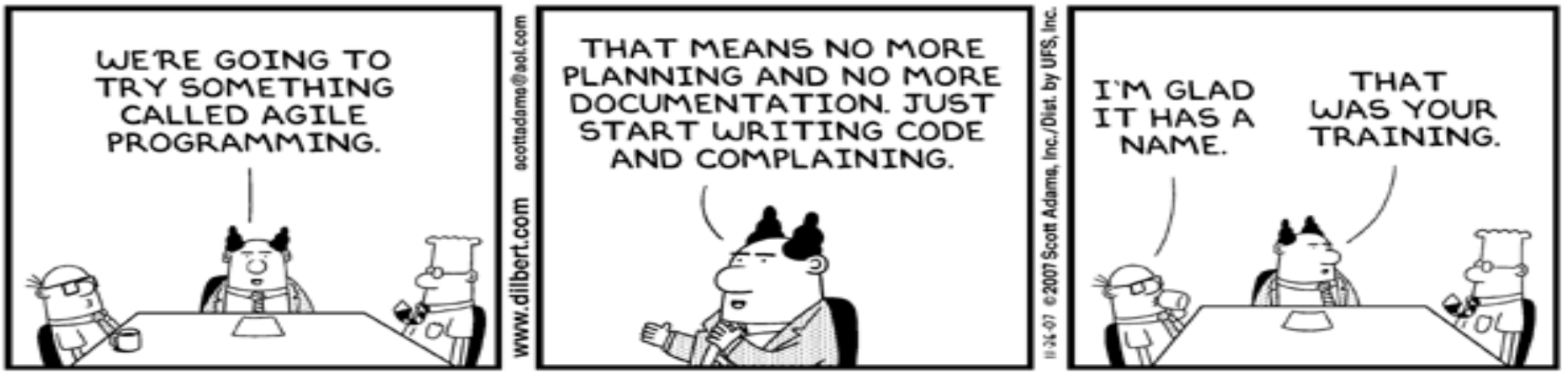
While progress has been made on design for maintainability of software *per se*, considerations that extend beyond immediate software product qualities and user benefits are generally treated as secondary concerns, optional qualities to be addressed only after the system under design has been shown to be a success in terms of technical and/or marketing criteria. The larger impact of software artefacts on society and the natural environment is not routinely analyzed. But by trading off longer-term sustainability questions for shorter-term success criteria, we accumulate threats to sustainability. We argue that this trade-off itself is unnecessary. As Neumann

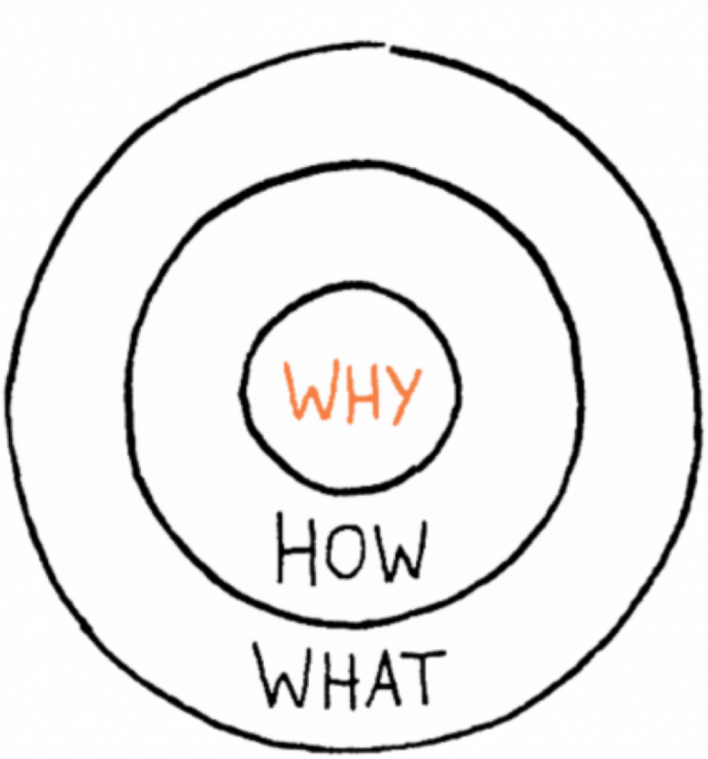
http://sustainabilitydesign.org/karlskrona-manifesto/

Do we need a [another] Manifesto?

Becker, Christoph, Chitchyan, Ruzanna, Duboc, Leticia, Easterbrook, Steve, Penzenstadler, Birgit, Seyff, Norbert and Venters, Colin (2015) Sustainability Design and Software: The Karlskrona Manifesto. In: 37th International Conference on Software Engineering, 16th - 24th May 2015, Florence, Italy.

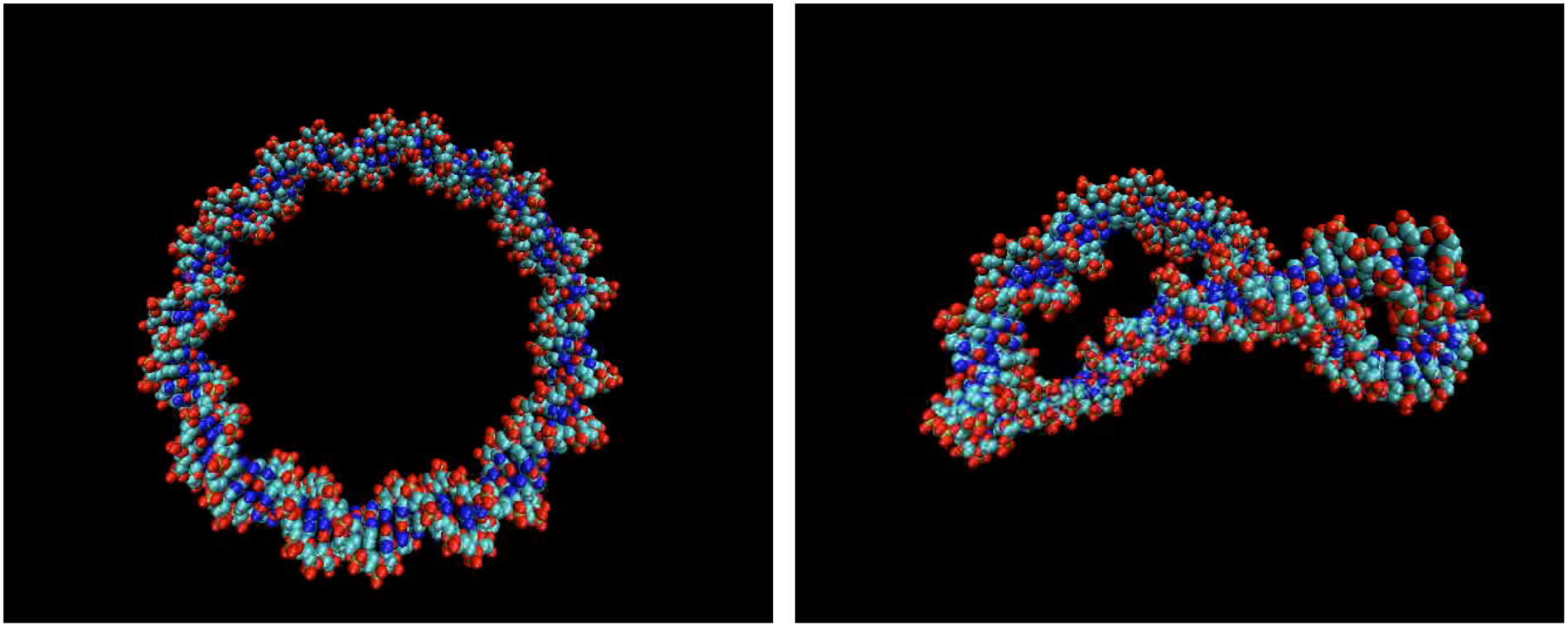
What is Software Engineering Anyway?




WHY
HOW
WHAT

Ubiquitous Software Systems

Computational Science

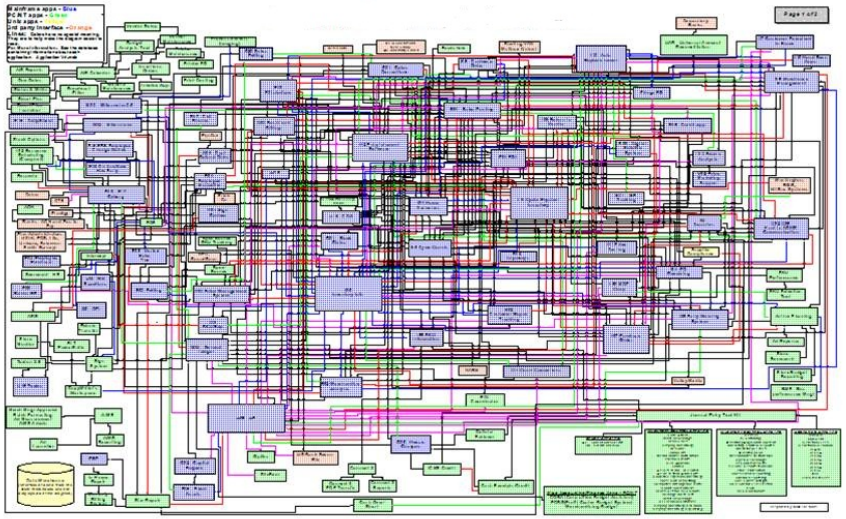


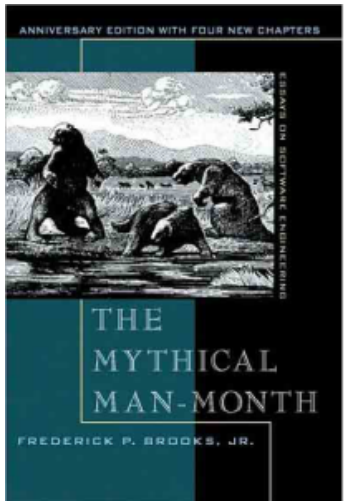
Harris, S. A., Laughton, C. A., and Liverpool, T. B. (2008). Mapping the phase diagram of the writhe of DNA nanocircles using atomistic molecular dynamics simulations. Nucleic Acids Research, 36 (1), pp. 21-29

Computational Engineering

Trent 980 with 80,000 lbf (360 kN) of thrust.

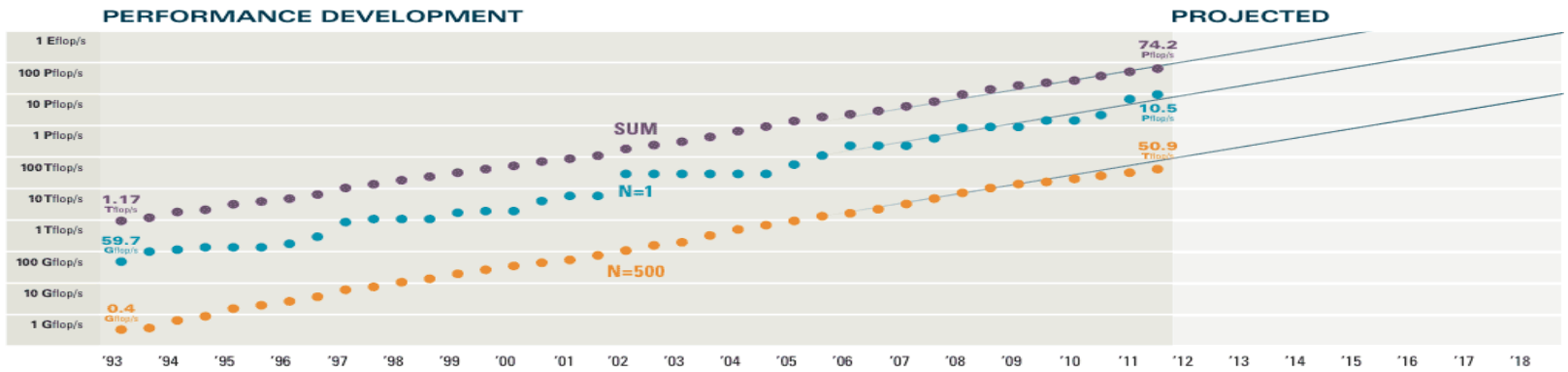
Complex Systems

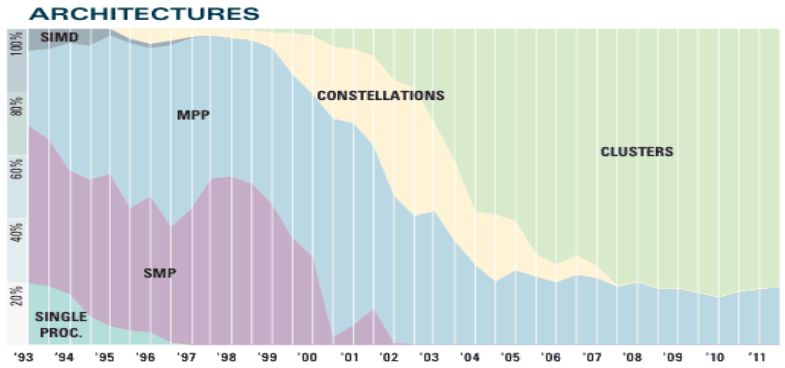


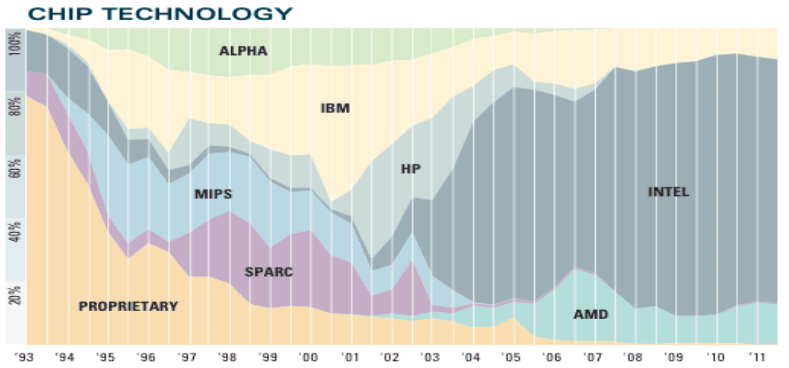


$$P = \frac{\text{content} * \text{quality}}{\text{resources} * \text{time}}$$

Evolving Compute Infrastructures

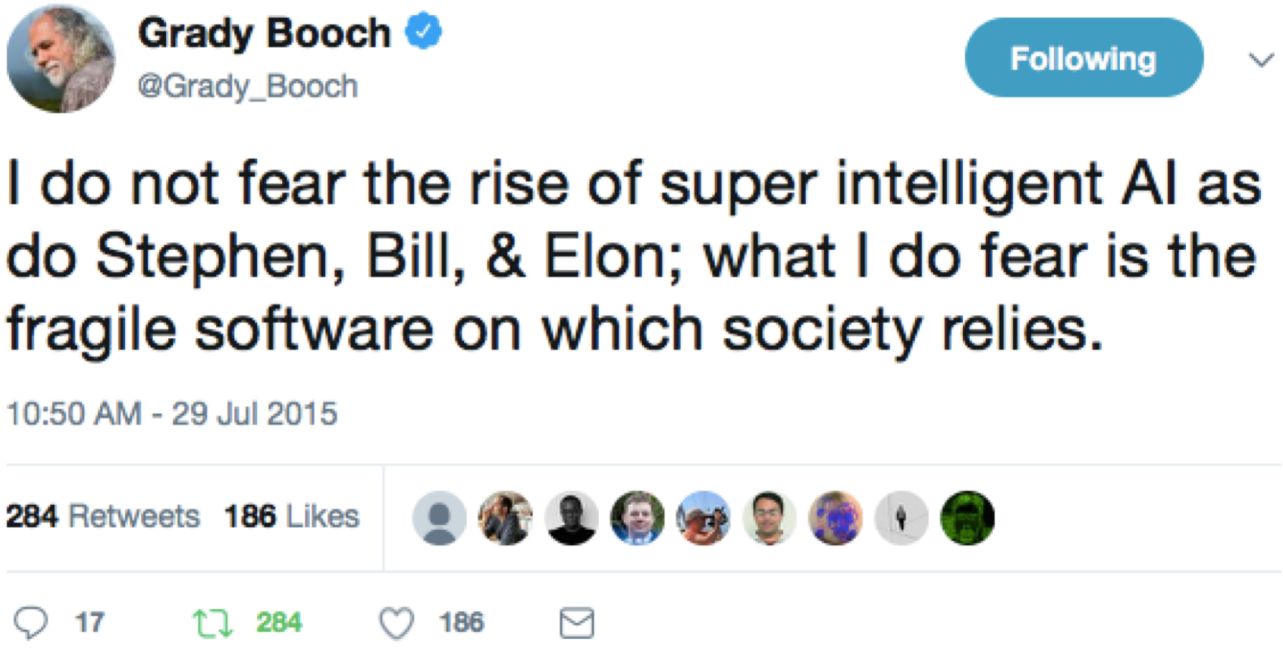






https://www.top500.org/news/summit-up-and-running-at-oak-ridge-claims-first-exascale-application/

Software Crisis?



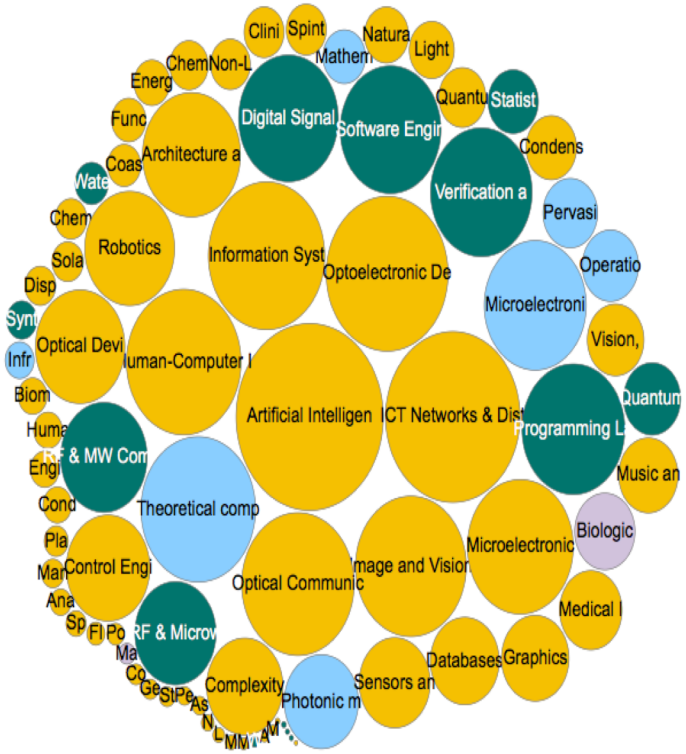
[Scientific] Software Crisis?

- 92% of academics use research software
- 69% say that their research would not be practical without it
- 56% of UK researchers develop their own research software.
  - 21% of those have no training in software development;
  - 15% self taught;
  - 40% had received some form of taught course.
- 70% of male researchers develop their own software
  - 30% of female researchers
- 4% of jobs advertised in UK universities were software related.

http://www.software.ac.uk/blog/2014-12-04-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers

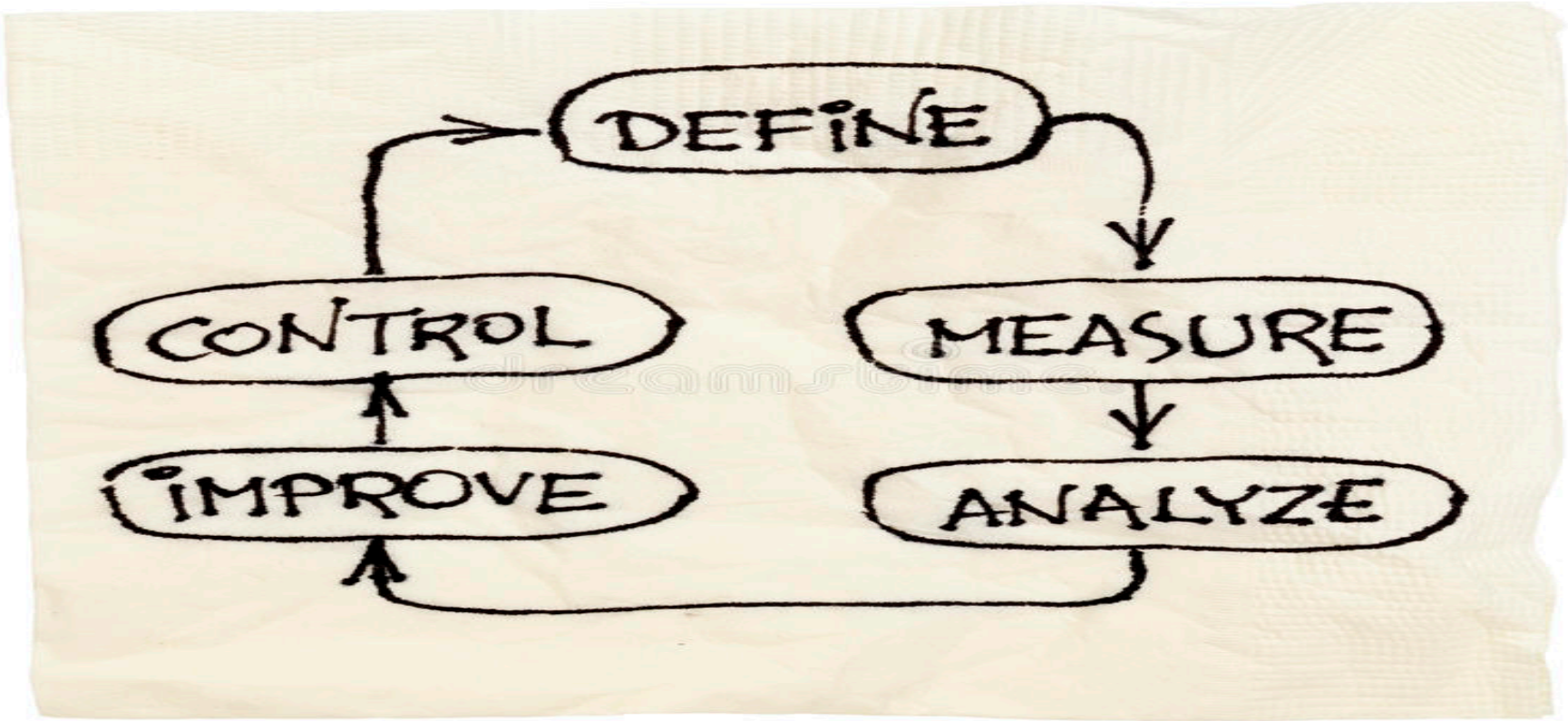
Better?

https://www.software.ac.uk



https://www.epsrc.ac.uk/research/ourportfolio/themes/ict/

If You Cannot Define it…



How do you measure Reasonable?

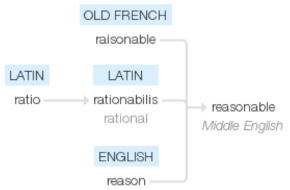
reasonable

/ˈriːz(ə)nəb(ə)l/

adjective
adjective: **reasonable**

1. having sound judgement; fair and sensible.
   "no reasonable person could have objected"
   synonyms: sensible, rational, open to reason, full of common sense, logical, fair, fair-minded, just, equitable, decent; More
   antonyms: unreasonable, illogical
   - based on good sense.
     "it seems a reasonable enough request"
   - *archaic*
     able to reason logically.
     "man is by nature reasonable"
2. as much as is appropriate or fair; moderate.
   "a police officer may use reasonable force to gain entry"
   synonyms: within reason, practicable, sensible; More
   antonyms: excessive, obsessional
   - fairly good; average.
     "the carpet is in reasonable condition"
     synonyms: fairly good, acceptable, satisfactory, average, adequate, respectable, fair, decent, all right, not bad, tolerable, passable; More
     antonyms: bad, poor
   - (of a price or product) not too expensive.
     "a restaurant serving excellent food at reasonable prices"
     synonyms: inexpensive, moderate, low, low-cost, low-priced, modest, within one's means, economical, cheap, budget, bargain; More
     antonyms: expensive, inflated

Origin



Middle English: from Old French *raisonable*, suggested by Latin *rationabilis* 'rational', from *ratio* (see reason).

http://webhotel.bth.se/re14/pages/conference/keynote_speakers/

Is Sustainability a Non-Functional Requirement?

C. C. Venters et. al., “The blind men and the elephant: Towards an empirical evaluation framework for software sustainability,” Journal of Open Research Software. 2(1):e8, pp: 1-6, 2014.

Non-Functional Requirement!

- Software [technical] sustainability is a composite, non-functional requirement which is ‘a measure of a systems:
  - Extensibility;
  - Interoperability;
  - Maintainability;
  - Portability;
  - Reusability;
  - Scalability;
  - Usability.

C. C. Venters et. al., “The blind men and the elephant: Towards an empirical evaluation framework for software sustainability,” Journal of Open Research Software. 2(1):e8, pp: 1-6, 2014.

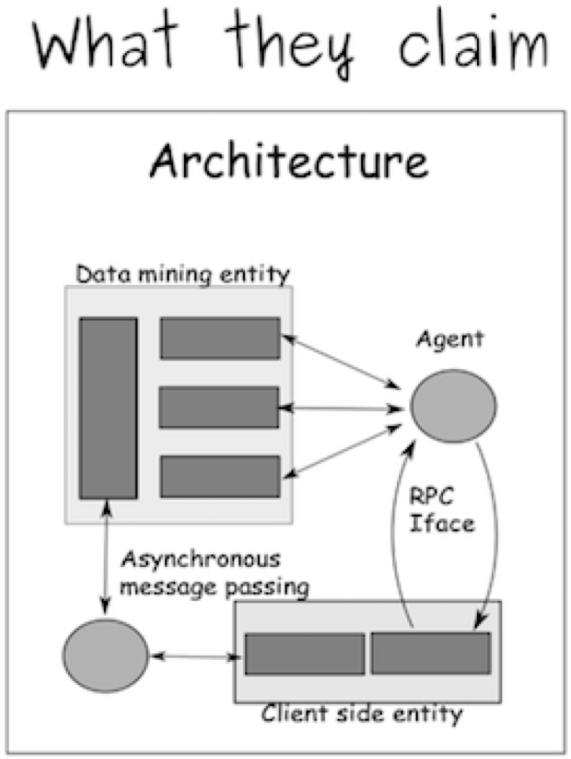

HOW?

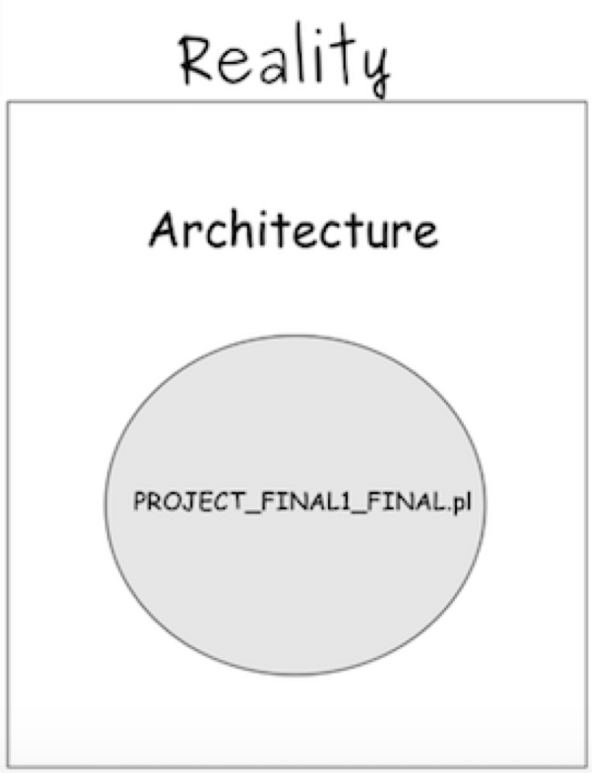
[Sustainable] Software Architectures

- Software architectures are fundamental to the development of technically sustainable software
  - Primary carrier of system qualities (NFR) i.e. pre-system understanding.
  - Influence how developers are able to understand, analyze, extend, test and maintain a software system i.e. post-deployment system understanding.
- Address the sustainability of software architectures to endure different types of change and evolution i.e. Architectural drift and erosion, architectural knowledge vaporization.

Venters, C. C., et. al., (2014). "The Nebuchadnezzar Effect: Dreaming of Sustainable Software through Sustainable Software Architectures," WSSSPE'2, Second Workshop on Sustainable Software for Science: Practice and Experiences, New Orleans, LA, USA, 16 November 2014. DOI: http://dx.doi.org/10.6084/m9.figshare.1112484

Software Architectures: WTFs/Minute



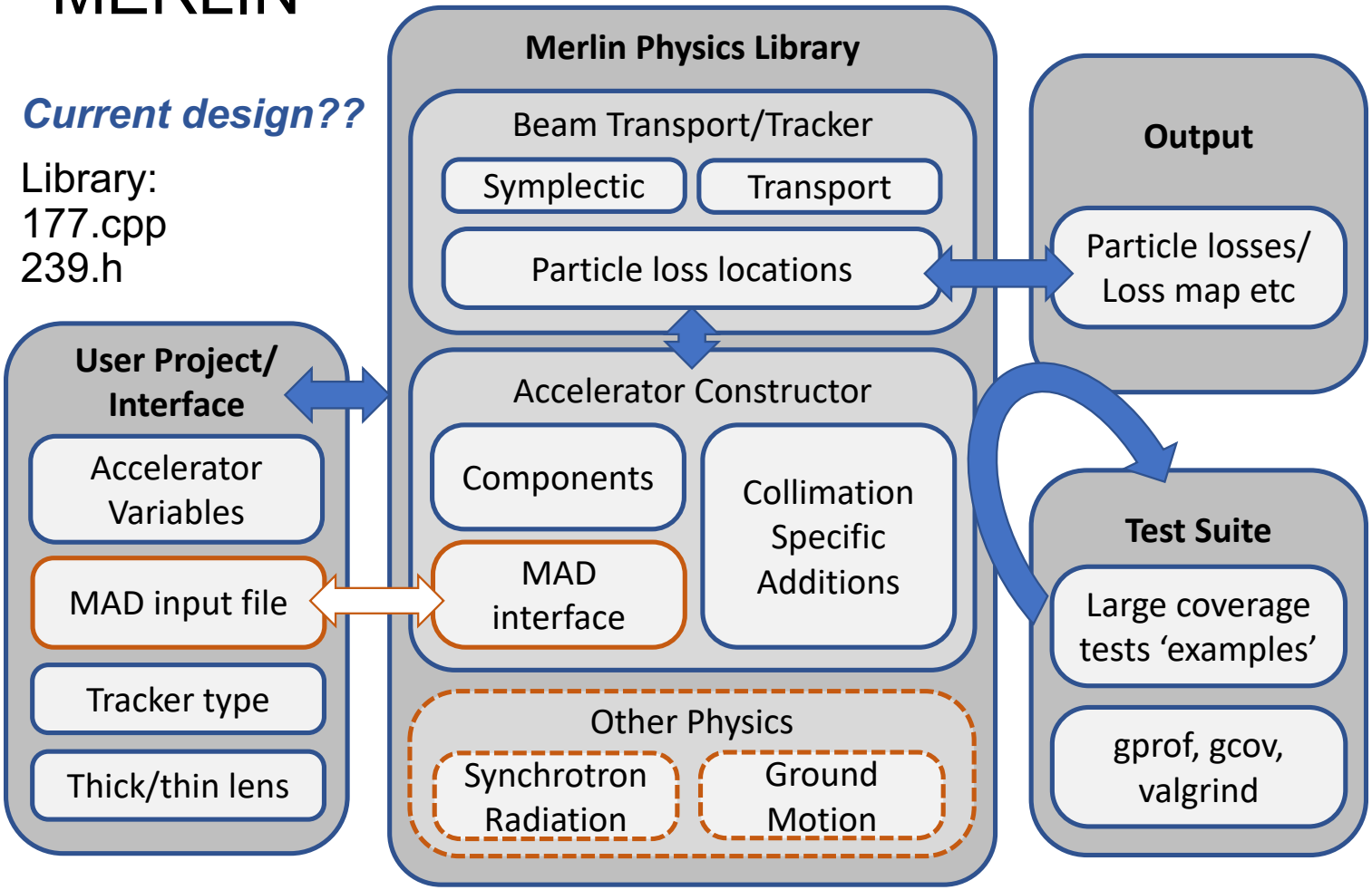


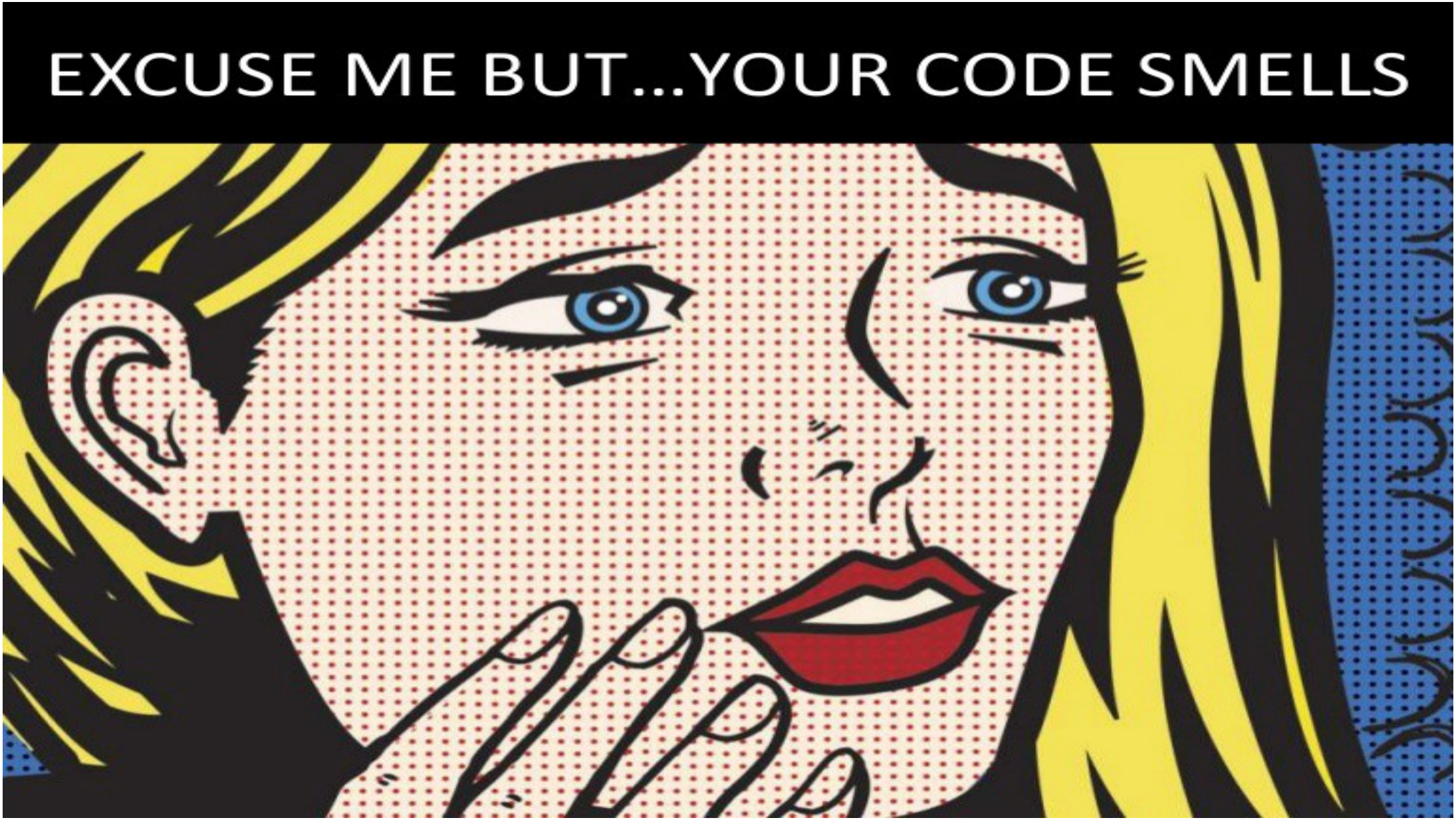
Dr. Colin C. Venters (c.venters@hud.ac.uk), University of Huddersfield, UK
May 10, 2017

MERLIN

Current design??

Library:
177.cpp
239.h




EXCUSE ME BUT...YOUR CODE SMELLS

Colin C. Venters @ccv1968 · Jun 1

If we don't focus on architecture we will drown in a sea of technical debt

#ICSE2018 #icseSEIP

1

Future Research Directions

http://sustainabilitydesign.org/


PRAY