Embracing AI in Contemporary Software Systems Engineering

Guilherme Horta Travassos
PESC/COPPE
Federal University of Rio de Janeiro
CNPq researcher, Faperj CNE, ISERN member
ght@cos.ufrj.br
orcid: 0000-0002-4258-0424

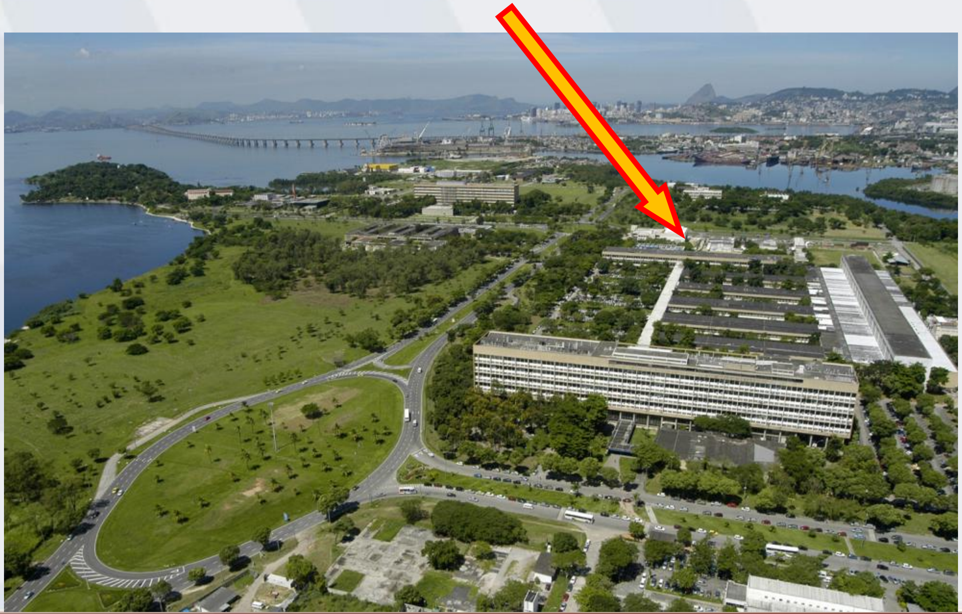
October 13, 2025

Rio de Janeiro City (picture taken near my home…)

About me

- Full professor of Software Engineering at PESC/Coppe/UFRJ, CNPq Researcher, CNE FAPERJ, IST assoc. editor, SBC and ACM Professional Member, and an ISERN member.
- Coppe/UFRJ is located at the Fundão Island closed to the International Airport in Rio de Janeiro, and the main highways approaching Rio de Janeiro....

- Electrical Engineer (UFJF,85), M.Sc.(COPPE/UFRJ, 90) and D.Sc.(COPPE/UFRJ, 94).
- Post-doc at UMCP/USA and NASA/SEL (00)
- Used to work in different academic areas (administration, research, education) and in collaboration with the software industry
- Current research interests include experimental software engineering and the engineering of contemporary software systems.
- Further information at http://www.cos.ufrj.br/~ght and https://scholar.google.com.br/citations?hl=pt-BR&user=hn4LDmkAAAAJ&oi=sra



What is software?

“**Software** is non-tangible and non-physical, but often intends to manage tangible and physical. It has a hierarchical structure of interconnected components having different purposes, which are expected to require modification/evolution. Its analysis and verification are universal underlying needs”

Adapted from Osterweil, L. J. (2018). What is Software? The Role of Empirical Methods in Answering the Question. https://link.springer.com/content/pdf/10.1007%2F978-3-319-73897-0_4.pdf

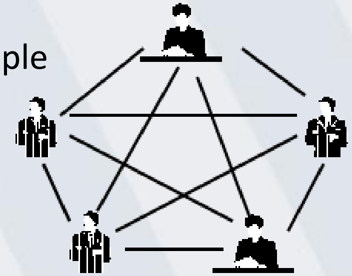
SOFTWARE IS EVERYWHERE!

Engineering of (traditional) Software Systems…

Software systems construction usually does not follow a smooth pathway…

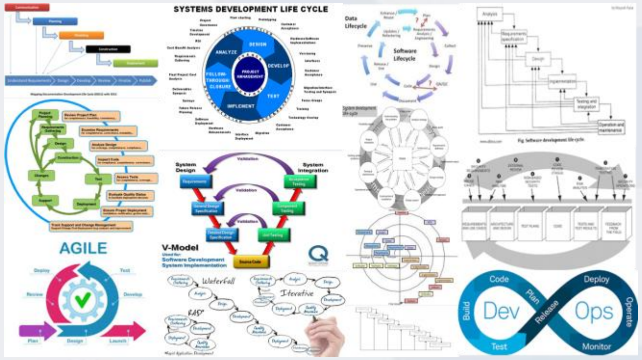
Peopleware (human factors)

N people



N(N-1)/2 communication lines
$\mathbf{2^N-1}$ possible working groups

Processes



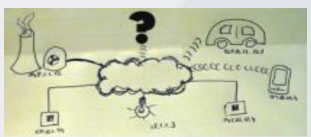
Software Technologies

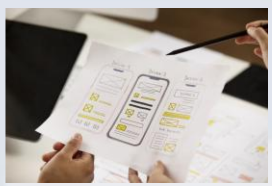
Cost Estimation Models
Measures and Metrics
Design Models
Software Product Lines Testing Techniques
Requirements Elicitation Techniques
Model Based Testing Techniques
Software Process Capability/Maturity Models
Testing Stop Criteria
Software Testing Effort Estimation
Inspection Techniques
Distributed Software Development Models
Agile Characteristics and Agile Practices
and many others…

(traditional) Software Systems Construction

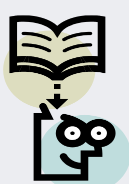
REQUIREMENTS

ByeByeSmoke Software System April, 2025
Overview
IDT's mission includes the continuous improvement of healthcare systems. Among its many research and knowledge development initiatives, one of the most prominent is its anti-smoking program (targeting both traditional cigarettes and VAPE devices), which involves monitoring smokers who seek to improve their quality of life by quitting smoking.
To support this, patients undergo regular clinical follow-ups, during which individual health markers are collected, and anamnesis is performed. This process updates the patient's medical records (stored in individual files) and gathers data that helps track the progress of treatment.
It has been observed that treatment success is influenced by the frequency of patient visits to the clinic and the motivational support provided by the medical team. However, maintaining constant contact with each patient is challenging. For this reason, IDT decided to develop a mobile application for Android (called "Goodbye Cigarette") designed to support smokers in permanently quitting cigarettes. This app is currently under development and allows the collection of relevant information for the clinical monitoring of each individual. However, this data must be transmitted and stored in a database located at IDT, where it can be accessed and used.





Tacit requirements

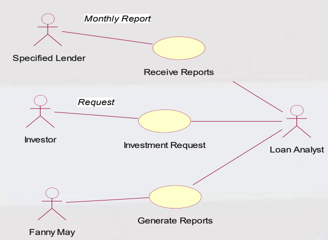


AD-HOC

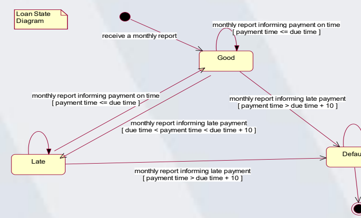
Problem Domain

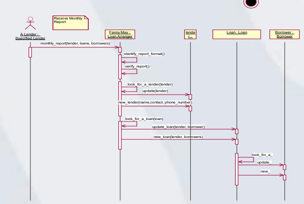
FORMAL

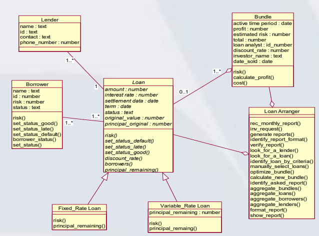
Scalene Triangle:
{<x,y,z>: (x != y) ^ (x != z) ^ (y != z)}



Solution Domain







TEST CASES			
CLASS	X	Y	Z
Scalene	3	4	5
Isosceles	5	5	8
Isosceles	3	4	3
Isosceles	4	7	7
Eqüilateral	2	2	2
No-triangle	1	2	3
No-triangle	5	1	4
No-triangle	3	5	2

Computer Domain

SOURCE CODE

Travassos, G.H. (2014). **Software Defects: Stay Away from Them. Do Inspections**!. In: 9th International Conference on the Quality of Information and Communications Technology (QUATIC) . Guimarães, Portugal.
http://dx.doi.org/10.1109/QUATIC.2014.8

Software Systems Persistent Problems

The Untamed Software

External Quality

DEFECTS
(observed by users)

Internal Quality

TECHNICAL DEBT
(perceived by developers)

Product Recall
Capital Loss

Jeronimo Junior, H.; Travassos, G. H. (2022). **Consolidating a Common Perspective on Technical Debt and its Management Through a Tertiary Study**. INFORMATION AND SOFTWARE TECHNOLOGY, v. 149, p. 106964. http://dx.doi.org/10.1016/j.infsof.2022.106964

High-quality *software* *is vital* to the global competitiveness, innovation, and national security of countries like Brazil, France, and others. *It also ensures* **society's** *modern standard of living and enables continued advances in defense, infrastructure, healthcare, commerce, education, and entertainment.*

Borrowed and adapted from **Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development.** (2021) Software Engineering Institute, Carnegie Mellon University https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=741193

Engineering of (traditional) Software Systems…

Quality is the <u>degree</u> to which a <u>set of inherent characteristics</u> of an <u>object</u> fulfils **requirements**.
ISO 9000:2015 clause 3.6.2

"It is achieved by <u>conformance to all requirements</u> regardless of what characteristic is specified or how requirements are grouped or named."
ISO/IEC TR 19759:2015 (en)

https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

Software Quality Assurance

https://www.iso.org/obp/ui/#iso:std:iso-iec:tr:19759:ed-2:v2:en

Software **Q**uality **A**ssurance is a means and practice of monitoring the software engineering processes and methods used in a project to ensure proper software quality (SWEBOK). It may include ensuring compliance with standards or models, such as ISO 25010, CMMI, MPS-SW, and others.

(traditional) Software Systems SQA

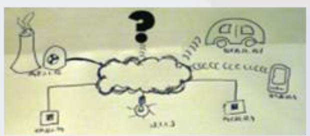
REQUIREMENTS

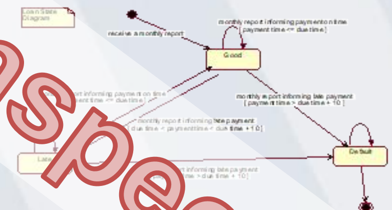
ByeBye Smoke Software System April, 2025
Overview
IDT's mission includes the continuous improvement of healthcare systems. Among its many research and knowledge development initiatives, one of the most prominent is its anti-smoking program (targeting both traditional cigarettes and VAPE devices), which involves monitoring smokers who seek to improve their quality of life by quitting smoking.
To support this, patients undergo regular clinical follow-ups, during which individual health markers are collected, and anamnesis is performed. This process updates the patient's medical records (stored in individual files) and gathers data that helps track the progress of treatment.
It has been observed that treatment success is influenced by the frequency of patient visits to the clinic and the motivational support provided by the medical team. However, maintaining constant contact with each patient is challenging. For this reason, IDT decided to develop a mobile application for Android (called "Goodbye Cigarette") designed to support smokers in permanently quitting cigarettes. This app is currently under development and allows the collection of relevant information for the clinical monitoring of each individual. However, this data must be transmitted and stored in a database located at IDT, where it can be accessed and used.

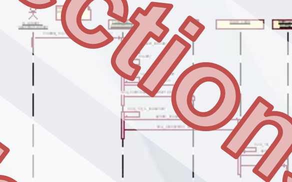
FORMAL

Scalene Triangle:
{<x,y,z>: (x != y) ^ (x != z) ^ (y != z)}

TEST CASES			
CLASS	X	Y	Z
Scalene	3	4	5
Isosceles	5	5	8
Isosceles	3	4	3
Isosceles	4	7	7
Eqüilateral	2	2	2
No-triangle	1	2	3
No-triangle	5	1	4
No-triangle	3	5	2







Solution Domain

Problem Domain

Computer Domain

Reviews and Inspections

Software Testing

Tacit requirements

AD-HOC

SOURCE CODE

Conte, T. ; Franco, N. ; Prikiladnicki, R. ; Rocha, A. R. ; Santos, G. ; Travassos, G. H. ; Weber, K.C. . **Towards Successful Software Process Improvement Initiatives: Experiences from the Battlefield.** In: 21st Americas Conference on Information Systems, 2015, Porto Rico. Proceedings of ICTS In Global Development (SIG GLOBDEV), 2015. v. 1. p. 1-12.

(traditional) Software Systems SQA

We have built, evaluated, and deployed MANY traditional software systems.

Conte, T. ; Franco, N. ; Prikiladnicki, R. ; Rocha, A. R. ; Santos, G. ; Travassos, G. H. ; Weber, K.C. . **Towards Successful Software Process Improvement Initiatives: Experiences from the Battlefield.** In: 21st Americas Conference on Information Systems, 2015, Porto Rico. Proceedings of ICTS In Global Development (SIG GLOBDEV), 2015. v. 1. p. 1-12.

Software Systems Engineering Evolution

However…

Software Systems Engineering Evolution

Early years
Custom Software
Standalone
Batch

Second Stage
Multi-user
Real-time
Database
Product Software

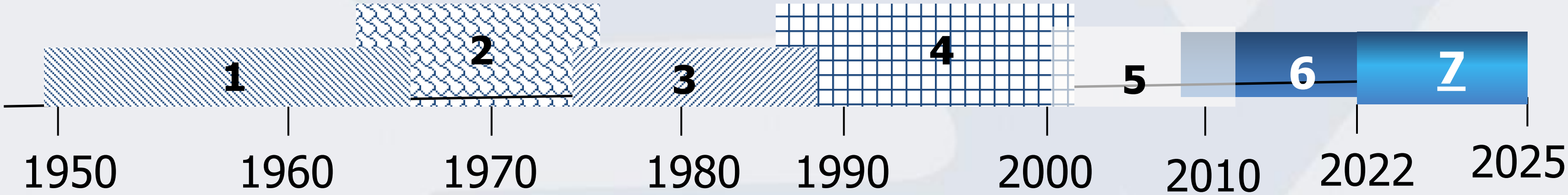
Third Stage
Distributed Systems
Embedded "intelligence"
Low-cost hardware
Consumer Impact

Fourth Stage
Powerful desk-top systems
Object-oriented technologies
Expert systems
Artificial neural networks
Parallel computing
Network computers

Fifth Stage
Multi-skilled, geographically distributed development
Componentry (reuse and recycling)
Development and evolution models, including biological analogies
Interdependence among design, business, and evaluation
Agile software manufacture
Empowering the domain expert (vs. maintaining integrity)
Non-scripting development languages

Sixth Stage
mobile apps
e-science with intensive use of e-infrastructure
Ubiquitous Systems (systems of systems)

Seventh Stage
Internet of (every)thing
AI-Driven Software Systems
Digital Twins
Industry 5.0
Quantum Computing



Evolved from PRESSMAN, R. S. , 1994, "*Software Engineering: A Practitioner's Approach*", *European Edition, McGraw-Hill.*

Computing Paradigms

1st Paradigm (stages 1, 2, and 3)

- One computer, many users
- Mainframes

2nd Paradigm (stages 4 and 5)

- One computer, one user
- Personal Computers, Desktops

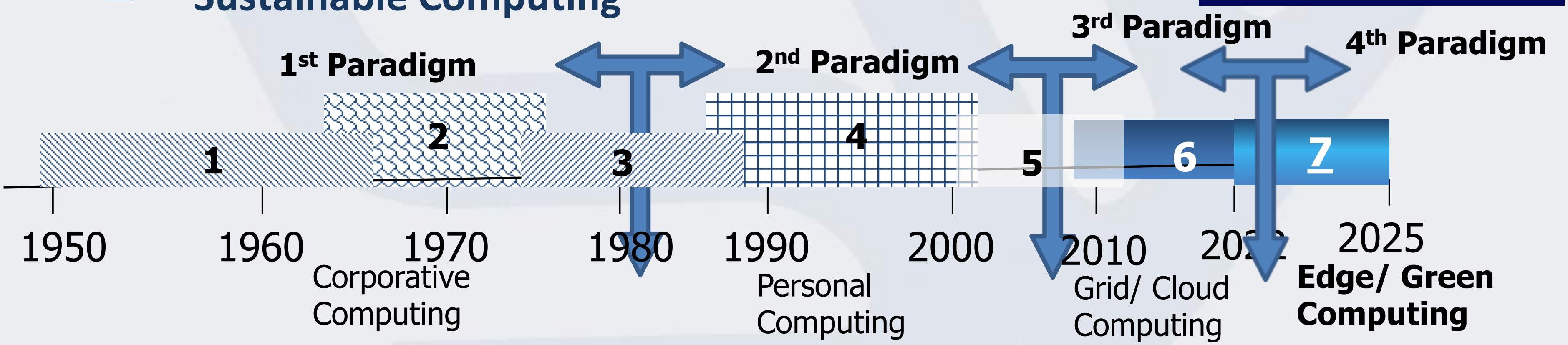
3rd Paradigm (stage 6)

- Many computers for one user
- E-infrastructure and ubiquitous systems (*systems of systems)*

4th Paradigm (stage 7)

- **Many computers, many users**
- **Pervasive Sensoring**
- **Sustainable Computing**

Contemporary Software Systems (CSS)



Contemporary Software Systems (CSS)

- A conten[illegible] software sys[illegible]efers to any modern, co[illegible]ex software solution designed to meet cu[illegible]t technological, business, o[illegible]er needs.
  - The[illegible]ms are typically built [illegible]p-to-date con[illegible] platforms, frameworks, [illegible]chitectural patterns, and they often integrate with cloud services, mobile plat[illegible]s, and data analytics t[illegible]s.

Contemporary Software Systems (CSS)

- CSS can interact with different systems and devices to complete their tasks and act according to the context, regardless of their development and organizational differences.

  - They demand the interoperability of devices and communications technologies

"ambient intelligence" OR "assisted living" OR "multiagent systems" OR "systems of systems" OR "internet of things" OR "Cyber Physical Systems" OR "Industry 4" OR “Industry 5" OR "web of things" OR "Internet of Everything" OR "smart manufacturing" OR digitalization OR digitization OR "digital transformation" OR "smart cit*" OR "smart building" OR "smart health" OR "smart environment“ OR “Autonomous Vehicle” OR…

R.C. Motta, K.M. de Oliveira, G.H. Travassos, **A conceptual perspective on interoperability in context-aware software systems**, Inf. Softw. Technol. 114 (2019) 231–257. https://doi.org/10.1016/j.infsof.2019.07.001.

Contemporary Software Systems (CSS)

Pervasive or Ubiquitous Computing

Context-Aware Systems

Ambient Intelligence

Network of Things

Social IoT

Internet of People

Human-Computer Interaction

Internet of Objects

4th Industrial Revolution!

DIGITAL TRANSFORMATION!

Internet of Computers

Web of Things

Micro-electro-mechanical Systems

Machine-to-Machine Interaction

Intranet/Extranet of Things

Cyber-Physical Systems (CPS)

Industry 5.0

Motta, R. C.; da Silva, V. M.; Travassos, G. H. (2019) **. Towards a more in-depth understanding of the IoT Paradigm and its challenges.** Journal of Software Engineering Research and Development, 2019, 6:3, https://doi.org/10.5753/jserd.2019.14

CSS under the IoT paradigm

Context-awareness

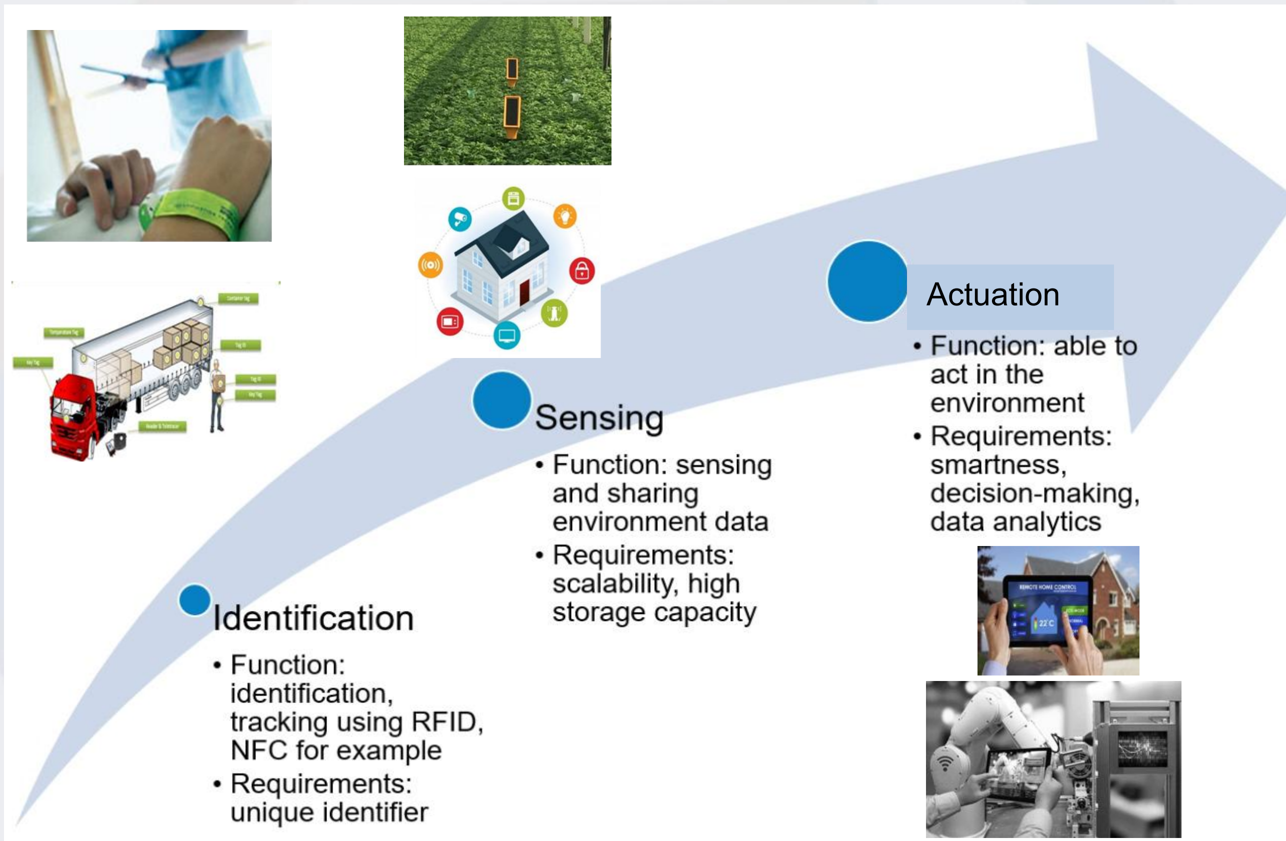
Security

Heterogeneity

Interoperability

Mobility

Addressability



Unique ID

Object Autonomy

Autonomy

(<u>no human intervention</u>)

Maia, V.C.; Oliveira, K. M.; Kolski, C.; Travassos, G. H. (2023) . **Using RFID in the Engineering of Interactive Software Systems: A Systematic Mapping.** Proceedings of the ACM on Human-Computer Interaction, Volume 7, Issue EICS Article No.: 183, Pages 1 – 37 https://doi.org/10.1145/35932

Challenges in CSS Engineering

IoT CONCERNS

Context-awareness

Security

Heterogeneity

Interoperabilit

Accuracy

Ubiquity

Scalability

Recall that “Software quality is achieved by <u>conformance to all requirements </u>regardless of what characteristic is specified or how requirements are grouped or named.”
ISO/IEC TR 19759:2015 (en)

Sustainability

Traceability

Trust

Visibility

Unique ID

Object Autono

Autonomy
(no human intervention)

Motta, R. C.; da Silva, V. M.; Travassos, G. H. (2019) . **Towards a more in-depth understanding of the IoT Paradigm and its challenges.** Journal of Software Engineering Research and Development, 2019, 6:3, https://doi.org/10.5753/jserd.2019.14

The Untamed CSS

2 Killed in Driverless Tesla Car Crash, Officials Say

"No one was driving the vehicle" when the car crashed and burst into flames, killing two men, a constable said.

https://www.nytimes.com/2021/04/18/business/tesla-fatal-crash-texas.html

NASA delays Mars helicopter Ingenuity's 1st flight to April 14

By Meghan Bartels 3 days ago

The little chopper was grounded after a test ended early.

https://www.space.com/nasa-mars-helicopter-flight-delay

The Failure of Uber's Self-Driving Car, Polestar's Debut, and More Car News This Week

Plus, a call to require helmets for cyclists, Paris battles e-scooters, and more.

https://www.wired.com/story/uber-self-driving-crash-volvo-polestar-1-roundup/

The world's first robot car death was the result of human error — and it can happen again

The damage from the Uber crash will have far-reaching consequences

By Andrew J. Hawkins | @andyjayhawk | Nov 20, 2019, 2:23pm EST

https://www.theverge.com/2019/11/20/20973971/uber-self-driving-car-crash-investigation-human-error-results

Tesla's 'shatterproof' window a metaphor for self-driving tech industry

NOVEMBER 22, 2019 • Tyson Fisher

https://landline.media/teslas-shatterproof-window-a-metaphor-for-self-driving-tech-industry/

Exclusivo: GM volta a vender Onix Plus após atualizar software do motor

No entanto, fabricante pede que donos não utilizem o carro antes de passar pelo recall. Seis casos de falha foram registrados até agora, todos no NE

Por Henrique Rodriguez, Leonardo Felix, Waldez Amorim e Zeca Chaves Atualizado em 11 nov 2019, 13h00 - Publicado em 11 nov 2019, 11h35

https://quatrorodas.abril.com.br/noticias/exclusivo-gm-volta-a-vender-onix-plus-apos-atualizar-software-do-motor/

The first Boeing 737 Max crash was 2 years ago today. Here's the complete history of the plane that's been grounded since 2 crashes killed 346 people 5 months apart.

David Slotnick Oct 29, 2020, 2:55 PM

https://www.businessinsider.com/boeing-737-max-timeline-history-full-details-2019-9

Microsoft: 30% of IoT projects fail in the proof-of-concept stage

Kyle Wiggers @Kyle_L_Wiggers July 30, 2019 8:00 AM Cloud

https://venturebeat.com/2019/07/30/microsoft-30-of-iot-projects-fail-in-the-proof-of-concept-stage/

Chess robot grabs and breaks finger of seven-year-old opponent

https://www.theguardian.com/sport/2022/jul/24/chess-robot-grabs-and-breaks-finger-of-seven-year-old-opponent-moscow

Tesla recalls almost 12,000 vehicles

https://www.testdevlab.com/blog/2021/12/27/10-biggest-software-bugs-and-tech-fails-of-2021/

The Untamed CSS

LACK OF SOFTWARE QUALITY

We struggle to build reliable software — and even more to maintain it. Poor design, limited resources, and lack of engineering expertise threaten everything we've already built.

The Tamed CSS

Software Engineering Determines Software Quality.

"<u>Software failures are a direct reflection of inadequacies in how software is developed and maintained.</u> That is, <u>poor quality software is the direct result of the current state of the art and practice in software engineering.</u> Some effects are highly visible, such as the lives lost due to the loss of control of physical objects. Other effects are less visible, such as when vehicle emissions systems perform poorly or cell phone apps collect and share data without permission from the user."

Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development. (2021) Software Engineering Institute, Carnegie Mellon University
https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=741193

"Immortality is a by-product of good work." - Mel Brooks

Software Systems Engineering Evolution

Assumption:

The state of Software Engineering practices and resources is not enough to tackle CSS challenges!

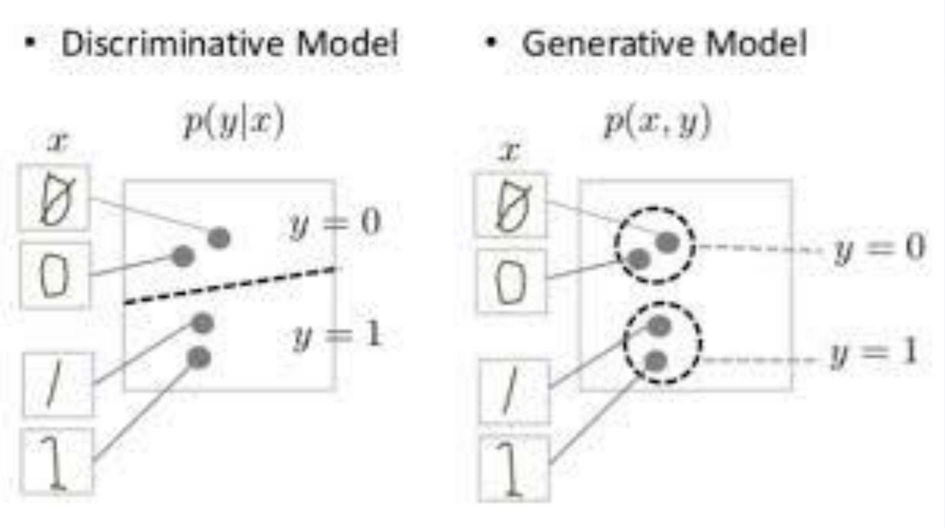
Conjecture:

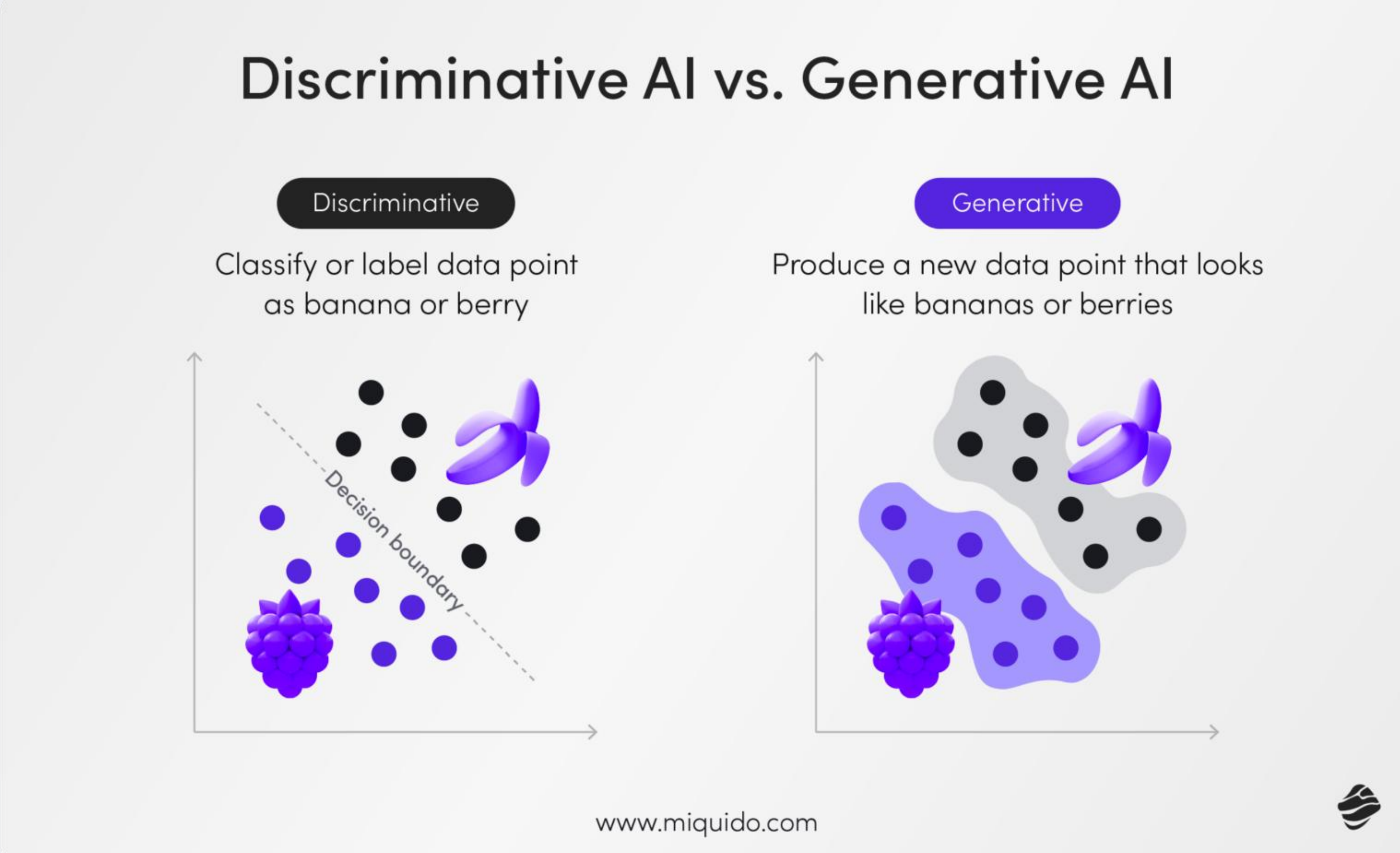
Embracing AI in Contemporary Software Systems Engineering can help address the lack of practices and resources to tackle CSS challenges.

AI Models

- AI models are mathematical constructs designed to perform specific tasks by learning from data.
  - They are the foundation of machine learning and artificial intelligence, enabling systems to recognize patterns, make predictions, and even generate new content

- Two common models:
  - Discriminative
  - Generative



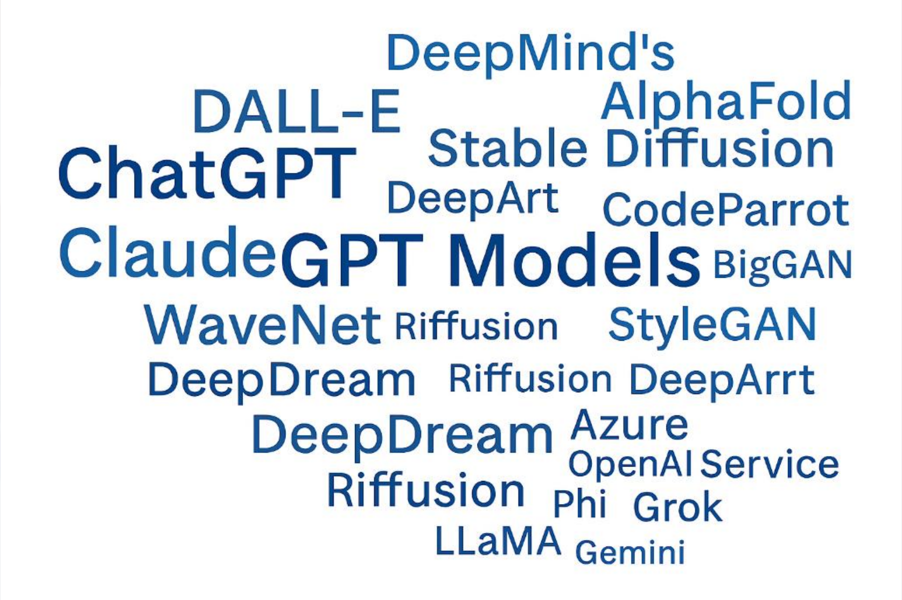


Discriminative AI models focus on classifying existing data by learning the decision boundary between classes, while Generative AI models aim to learn the underlying data distribution to generate new data instances.

Discriminative x Generative Models

Model	Discriminative AI	Generative AI
Purpose	Classifies data into predefined categories	Creates new data instances using generative and discriminative models
Example	Distinguishing between cat and dog images	Generating a new image of a cat using generative models
Common Algorithms	Logistic Regression, SVMs, Neural Networks	Generative adversarial networks (GANs), VAEs, Transformer models
Application Areas	Image classification, NLP, fraud detection	AI-generated art, text generation, synthetic data creation

Some Use Cases and Models for Generative AI



Software Engineering

Requirements Engineering, Software Design, Software Construction, Software Evolution, Software Project Management, Software Quality Assurance, Experimentation

Simple Example of AI in SE: Llama 3

C:\Users\ghtra>**ollama run llama3**
>>> Hello
Hello! It's nice to meet you. Can I help you with something, or would you like to chat?
>>> You are a software engineer. I need software to calculate the power of an electric circuit. Could you describe its requirements?

prompt

Requirements Drafting

As a software engineer, I'd be happy to help you design software that calculates the power of an electric circuit.
Here's a list of requirements we can consider:
1. **Input Requirements**:
    * The software should accept input values for the following parameters:
        + Voltage (V): the voltage across the circuit
        + Current (I): the current flowing through the circuit
        + Resistance (R): the resistance of the circuit's components
        + Frequency (f): the frequency of the AC circuit (if applicable)
2. **Power Calculation**:
    * The software should calculate the power (P) in watts (W) using the following formulas:
        + For DC circuits: P = V x I
        + For AC circuits: P = Vrms x Irms, where Vrms is the root mean square voltage and Irms is the root mean square current.
3. **Unit Conversions**:
    * The software should allow for unit conversions between different units of measurement (e.g., volts to amperes, watts to kilowatts).
4. **Error Handling**:
    * The software should handle invalid or out-of-range input values by displaying an error message and preventing the calculation from being performed.
5. **User Interface**:
    * The software should have a user-friendly interface that allows users to enter input values and view the calculated power value.
6. **Additional Features (optional)**:
    * Displaying the calculated power in different units of measurement (e.g., kilowatts, horsepower).
    * Providing a summary or report of the calculation, including any relevant formulas or equations used.

These are some of the key requirements for a software that calculates the power of an electric circuit. Let me know if you have any specific requirements or additional features in mind!

Test Cases Design

>>> Thanks, could you give me test cases for this software?

Here are some test cases to help ensure your software is functioning correctly:

Test Case 1: Simple DC Circuit
* Input values:
    + Voltage (V): 12
    + Current (I): 2
    + Resistance (R): Not applicable (DC circuit)
* Expected result:
    + Power (P): 24 W

Test Case 2: Simple AC Circuit
* Input values:
    + Voltage (V): 120
    + Current (I): 5
    + Frequency (f): 60 Hz
    + Resistance (R): Not applicable (AC circuit)
* Expected result:
    + Power (P): approximately 300 W (rms value)

Test Case 3: DC Circuit with Resistance
* Input values:
    + Voltage (V): 9
    + Current (I): 1.5
    + Resistance (R): 10 ohms
* Expected result:
    + Power (P): 13.5 W

Test Case 4: AC Circuit with Resistance and Frequency
* Input values:
    + Voltage (V): 230
    + Current (I): 3
    + Frequency (f): 50 Hz
    + Resistance (R): 20 ohms
* Expected result:
    + Power (P): approximately 345 W (rms value)

Test Case 5: Invalid Input Values
* Input values:
    + Voltage (V): -1
    + Current (I): 2
    + Resistance (R): Not applicable (DC or AC circuit)
* Expected result:
    + Error message indicating invalid input values

Test Case 6: Out-of-Range Input Values
* Input values:
    + Voltage (V): 1000
    + Current (I): 5
    + Resistance (R): Not applicable (DC or AC circuit)
* Expected result:
    + Error message indicating out-of-range input values

Test Case 7: Unit Conversions
* Input values:
    + Voltage (V): 12 V (volts)
    + Current (I): 2 A (amperes)
    + Resistance (R): Not applicable (DC or AC circuit)
* Expected result:
    + Power (P) displayed in watts (W)

These test cases cover various scenarios and edge cases to help ensure that your software functions correctly. You may want to add more test cases depending on the specific requirements of your project.

Pros of Generative AI

- **Creativity and Novelty**
  - It enables the creation of new and unique content, whether it's images, music, or text.
  - It can generate innovative and original outputs that may not have been created otherwise.
- **Automation and Efficiency**
  - It automates content creation, saving time and resources.
  - It can generate large volumes of content quickly and efficiently, assisting in tasks like data augmentation, content generation, and design exploration.
- **Personalization and Customization**
  - Generative models can be trained on specific data or preferences, allowing personalized recommendations, tailored content, and customized user experiences.
- **Exploration and Inspiration**
  - It can inspire artists, designers, and writers by generating diverse variations, exploring creative possibilities, and serving as a starting point for further creative exploration.

Cons of Generative AI

Ethical Concerns

It raises ethical concerns, particularly regarding the misuse of synthetic media, deepfakes, and potential infringement of intellectual property rights.

It requires careful consideration and responsible usage to avoid malicious or deceptive applications.

Lack of Control

Generative models can produce outputs that are difficult to control or fine-tune to specific requirements.

The generated content may not always meet the desired expectations or adhere to specific guidelines.

Dataset Bias and Generalization

Generative models heavily rely on the training data they are exposed to.

If the training data is biased or limited, the generated outputs may inherit those biases or struggle with generalizing to unseen scenarios.

Ethical considerations around AI-generated content, such as deepfakes and intellectual property, will become increasingly important, driving the development of new regulations and standards.

Cons of Generative AI

Computational Resources and Complexity

Training and deploying generative models can be computationally intensive, requiring significant resources, including high-performance hardware and substantial training time.
Implementing and maintaining these models can be complex and resource-demanding.

Quality and Coherence

While generative models have made significant progress, they may still struggle with producing outputs that consistently exhibit high quality, coherence, and contextual relevance.
Fine-tuning and careful model selection may be necessary to achieve desired results.

Hallucination!

Software Engineering Evolution

Using AI in SE leads to the incorporate of new software artifacts into the development process:

The Prompts

What is a Prompt?

- Basically, a prompt is a command!
  - It represents a communication channel with a Language Model

What is a prompt?

A prompt is essentially the input or command you give to an AI model to get a desired response. It can be a question, statement, instruction, or even a combination of these. Think of it as the conversation starter or the task-setter for the AI—it's how you guide the model to understand what you're looking for.
For example:
Prompts can range from straightforward to complex depending on the task. The better the prompt, the more tailored and useful the AI's response will be. Want to explore ways to craft perfect prompts? Let me know!

Prompt Elements

- A prompt usually contains some or all of these basic elements:
  - **Instruction**
    - a specific task or instruction you want the model to perform
  - **Context**
    - a specific task or instruction you want the model to perform
  - **Input Data**
    - the input or question that we are interested in finding a response to
  - **Output Indicator**
    - The type or format of the output.

The elements of a prompt are the components that guide an AI to produce effective and accurate responses.

PROMPT "ENGINEERING"

- It represents a relatively new discipline for developing and optimizing prompts to efficiently use language models (LMs) for various applications and research topics.
- Prompts improve the capacity of LMs on a wide range of common and complex tasks such as question answering and arithmetic reasoning.

Asking to MSCopilot:

What is prompt engineering?

prompt

Prompt engineering is the art and science of crafting effective inputs (or "prompts") for AI models to achieve specific outputs. Essentially, it's about figuring out how to phrase questions, statements, or instructions in a way that maximizes an AI's ability to understand and respond accurately or creatively.
For example, with a language model like me, prompt engineering involves tweaking your phrasing, providing context, or adding examples to get better or more tailored responses. It's used across various fields, including AI-powered writing, code generation, image creation, and even chatbot design.
If you're curious, I could explain more about techniques or share examples. Let me know!

When an AI model doesn't produce the desired response, prompt engineering allows us to iterate and adjust the prompt to optimize the output.

PROMPT "ENGINEERING"

- **Importance**
  - It bridges the gap between vague, general queries and specific, actionable results.
  - It helps mitigate errors, such as generating irrelevant content or incorrect responses.
  - It ensures that the AI can handle tasks like creative writing, image generation, or even code development with minimal post-processing needed.
  - It is particularly useful for overcoming limitations of generative models, such as logical errors or insufficient context in responses

The quality of the AI's response depends directly on how clear, detailed, and structured the prompt is.

Prompting Techniques

- **Zero-Shot**: No examples given, just the task.
- **One-Shot**: One example provided.
- **Few-Shot**: A handful of examples to guide the model.
- **Chain-of-Thought**: Encourages step-by-step reasoning.
- **ReAct**: Combines reasoning and acting with tools.
- **Self-Ask**: Breaks down complex questions into sub-questions.
- **Tree-of-Thought**: Explores multiple reasoning paths.
- **Toolformer**: Uses tools dynamically during generation.

Embracing AI in Contemporary Software Systems Engineering

A Metaprotocol For a Family of Rapid Multivocal Reviews of Generative AI in the Software Industry

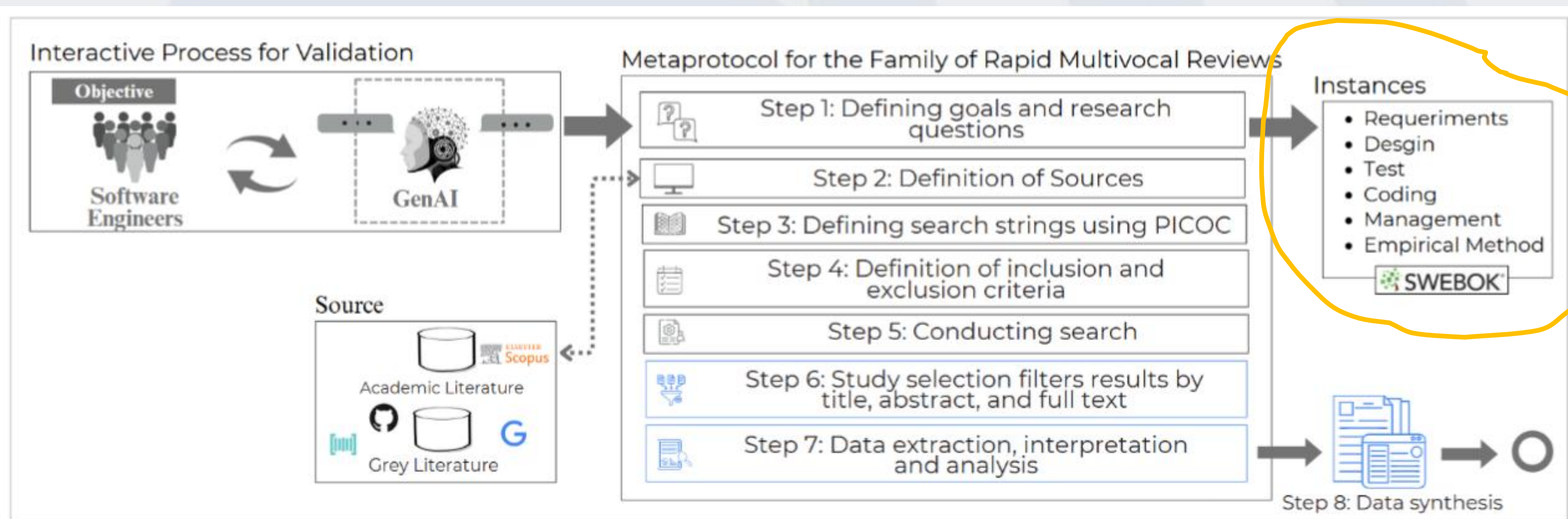
Sabrina Rocha
Program Systems Engineering and Computer Science
Federal University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
sabrinarocha@cos.ufrj.br

Rodrigo Feitosa
Program Systems Engineering and Computer Science
Federal University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
rfeitosa@cos.ufrj.br

Larissa Galeno
Program Systems Engineering and Computer Science
Federal University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
galeno@cos.ufrj.br

Guilherme H. Travassos
Program Systems Engineering and Computer Science
Federal University of Rio de Janeiro
Rio de Janeiro, RJ - Brazil
ght@cos.ufrj.br

Published at **2025 Brazilian Symposium on Software Engineering (SBES)**.
Available at
https://doi.org/10.5753/sbes.2025.11577



Embracing AI in Contemporary Software Systems Engineering

“Is it possible to develop a software system using only generative AI–based tools across the different phases of the development cycle?"

A case of observation:

“*Bye Bye Smoke*” Software System

Lessons Learned from the Use of Generative AI in Engineering and Quality Assurance of a WEB System for Healthcare

Guilherme H. Travassos
ght@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Sabrina Rocha
sabrinarocha@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Rodrigo Feitosa
rfeitosa@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Felipe Assis
fassis@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Patrícia Gonçalves
patriciaamaralgurgel@ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

André Gheventer
gheventer@ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Larissa Galeno
galeno@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Arthur Sasse
artsasse@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brazil

Júlio César Guimarães
jcguimaraes@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

Carlos Brito
carloshenriquefbf@poli.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

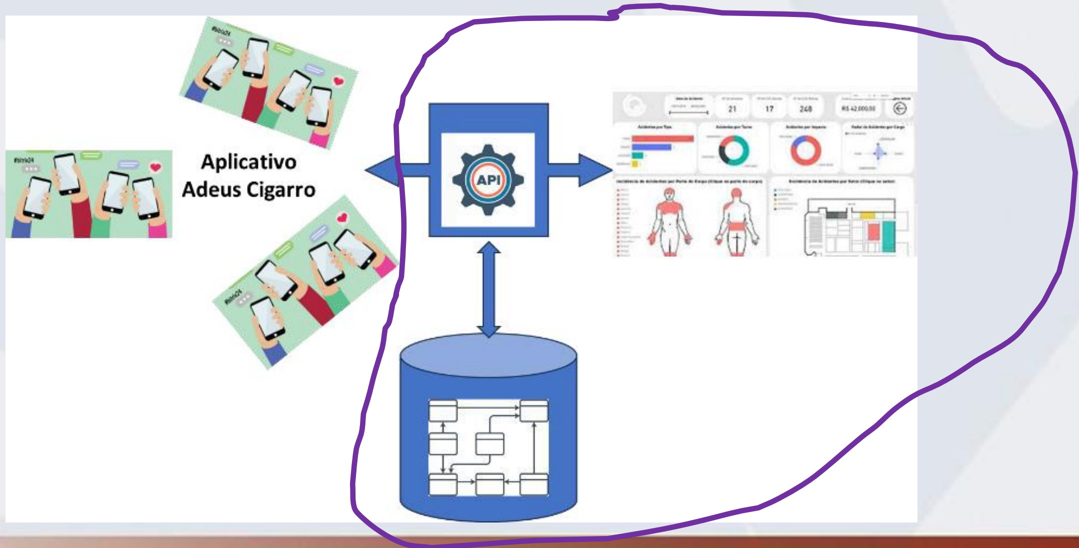
João Pedro Wieland
jpvbwieland@cos.ufrj.br
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ, Brasil

To appear at the **2025 Brazilian Software Quality Symposium (SBQS)**. Available at the **S**BC**O**PEN**L**IB - https://sol.sbc.org.br

ByeBye Smoke Software System

The mission of IDT/UFRJ is to promote the continuous improvement of healthcare systems. Among its various areas of activity, its work on smoking cessation stands out, addressing both conventional cigarettes and vaping, with a focus on monitoring patients who seek to improve their quality of life by quitting smoking. To this end, patients undergo periodic clinical follow-ups, during which individual markers are collected and anamnesis is performed to update medical records (maintained in separate files) and to obtain data to monitor treatment progression.

The web-based software system was conceived to replace the manual process of recording clinical information, which had previously been carried out using physical documents for consultation notes and the progress of smoking cessation treatment at IDT/UFRJ.



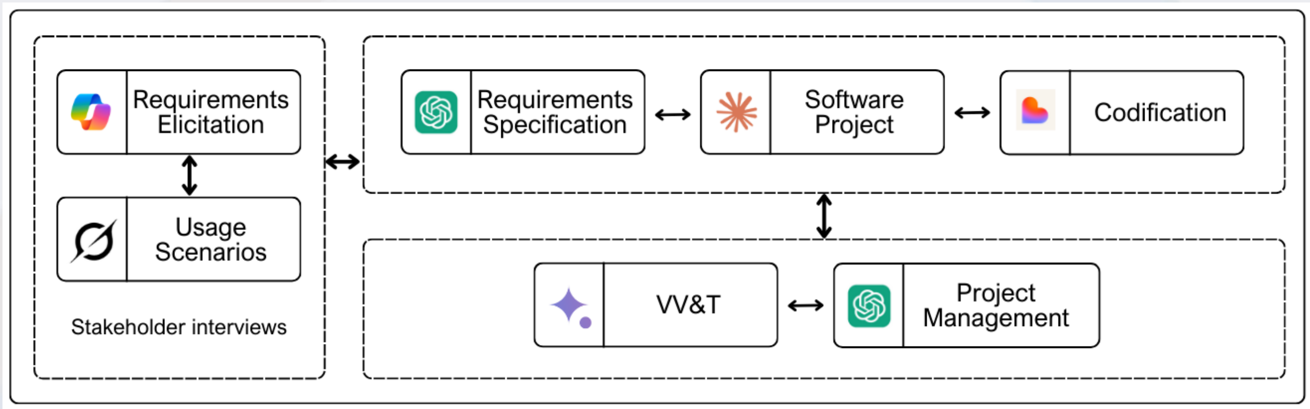
ByeBye Smoke Development Team

Developer	Area of Expertise	Professional Level
Dev1	Software Engineering	Master (Leader)
Dev2	Requirements Engineering	Mid-level
Dev3	Requirements Engineering	Mid-level
Dev4	Requirements Specification	Mid-level
Dev5	Requirements Specification	Mid-level
Dev6	Design	Mid-level
Dev7	Design	Mid-level
Dev8	Project Management	Senior
Dev9	Project Management	Junior
Dev10	Coding	Senior
Dev11	Coding	Mid-level
Dev12	Software Quality	Senior
Dev13	Software Quality	Senior
Dev14	Software Quality	Mid-level
Dev15	Design	Junior

Five Working Teams:
Requirements, Design, Software Quality, Management, and Coding

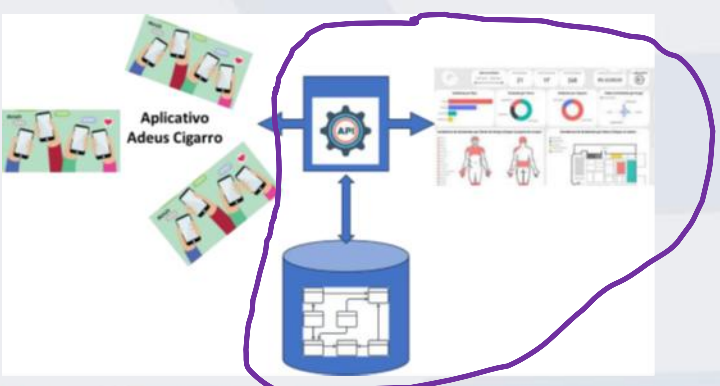
ByeBye Smoke Development Process

SWEBOK 4.0 based



Project Started by March/2025

Product deployed by June/2025



https://git-lab.cos.ufrj.br/cos827/byebyefumo

ByeBye Smoke Development Process

Example of Use

1)) Input Artifacts

Vision Document · Usage Scenarios · Clinical History Form

2) Prompt Engineering

Structured Prompts

#Persona
#Task
Context
Exemple
#Output Format

3) Requirements Generated by Generative Artificial Intelligence and Inspected by Developers

V1

FR01	The system shall allow the registration of a new patient, including personal information (name, gender, date of birth, contact details, address, profession, etc.).	Essential

DEV

V9

FR01	The system shall allow the registration of a patient according to the Identification section of the Clinical History Form.	Essential

4) Use Cases Generated by Generative Artificial Intelligence and Inspected by Developers

V1

Item	Description
Objective	Register a new patient who is a smoker.
Actors	Pneumologista (primário).
Preconditions	Authenticated professional with an authorized role.
Input Condition	Patients Menu → New.
Main Flow	1) The system displays the form. 2) The user fills in the mandatory fields. 3) The user selects Save.4) The system validates the data, saves it, and 5) displays a success message
Alternate Flows	A1 – Cancel Registration: At any point, the user clicks Cancel ⇒ the system discards the data and returns to the list.
Exception Flows	E1 – Mandatory Fields Empty: The system highlights the missing fields. E2 – Database Failure: The system displays the error message: *"Unable to save."*
Postconditions	The patient is saved in the database and listed in 'Patients'.

DEV

V8

UC1 — Patient Identification Registration	
Requirement: FR01	Essential
Objective	Allow the registration of patient identification.
Actors	Healthcare professional.
Preconditions	Authenticated Professional.
Input Condition	Access to the 'New Patient' functionality.
Main Flow	The professional accesses the registration module. The system displays the form: (Anonymized for confidentiality) The professional fills in the data. The professional clicks **"Save."** The system stores the information.bThe system confirms success.
Alternate Flows	Mandatory fields not filled: error message.
Exception Flows	Storage failure: error displayed.
Postconditions	Data saved; patient listed
Form Section	Identification

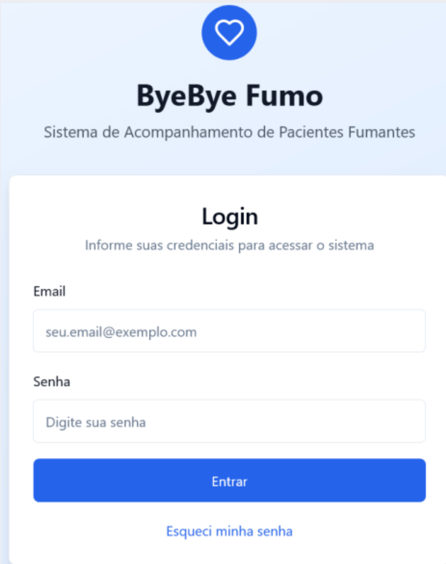
ByeBye Smoke Development Process

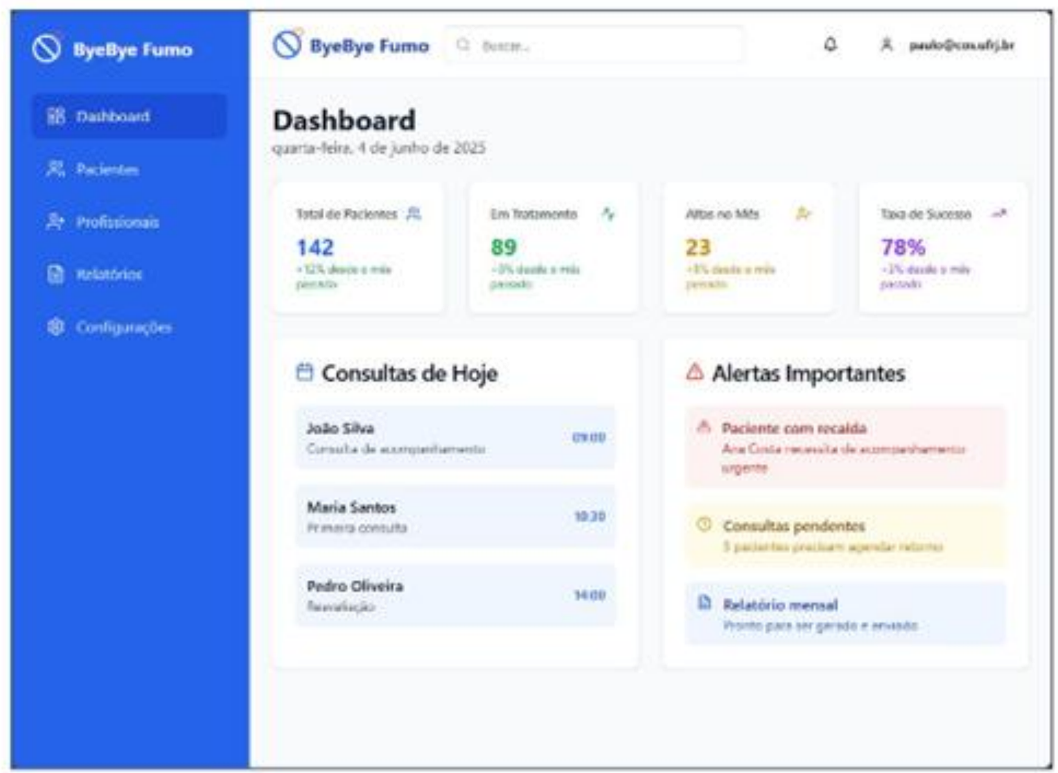
Phase	Input	Tool or Model Used	Outputs
Requirements Elicitation	Stakeholder interviews	**MS CoPilot (https://copilot.microsoft.com/)**	Vision document
Initial Working Scenarios	Scenarios Vision document	**Grok (4.0)** (https://grok.com/)	Document with eight use-case scenarios
Requirements	document, clinical history form, use cases	**Gemini 2.5 Flash** https://gemini.google.com/	Document with Test Plan and test case suggestions
Requirements Specification	Vision document, use-case scenarios, clinical history form	**ChatGPT-4o** (https://chatgpt.com/)	Document with 41 requirements, including functional (28) and nonfunctional (13)
Use Cases	Requirements document, clinical history form	**ChatGPT-4.1** (https://chatgpt.com/)	Use Case document with main, alternative, and exception flows
Project Management	Vision document and use-case scenarios	**ChatGPT-o4-mini** (https://chatgpt.com/)	Project Plan (classical model)
Design	Requirements document, clinical history form	**Claude-3.5** (https://claude.ai/)	Architecture diagram, Data model
Coding	Requirements document, clinical history form, use cases	**Lovable** https://lovable.dev	Backend, frontend

ByeBye Smoke Development Process

 # Persona
You are a senior software requirements engineer.
 # Task
You have been assigned to construct functional and non-functional software requirements based on the context provided below. You must also prioritize the requirements within the categories [essential, important, desirable].
 # Context
The Institute of Thoracic Diseases at the Federal University of Rio de Janeiro (UFRJ) has a mission to improve healthcare systems continually. Among its many research and knowledge advancement fronts, the anti-smoking initiative (covering both conventional cigarettes and Vapes) stands out, focusing on monitoring smoking patients who seek to improve their quality of life by quitting. To this end, patients undergo periodic clinical follow-ups, during which individual markers are collected and anamnesis is performed, to update their medical records (stored in separate files) and obtain data that allows monitoring the progress of treatment. This information must be transmitted and stored in a database located at IDT/UFRJ, where it can be accessed and utilized. The proposal document presents different scenarios. The focus at this moment is on scenario 1: **1. Patient Registration and Profile** - A new smoking patient arrives at the clinic. The pulmonologist needs to record detailed information for follow-up. **System use:** - Registration of personal data (name, age, contact information, etc.). - Recording smoking history: number of cigarettes per day, years of smoking, calculation of pack-years. However, other scenarios must also be considered, taking into account the system's evolution and dependencies between requirements. In summary, the system's goal is to provide a web-based solution that enables healthcare professionals to register and monitor their patients. Additionally, the system must allow healthcare professionals to monitor patients registered in the associated mobile application.
 # Examples
Below are some preliminary examples of how functional requirements should be described: - The system must register a new patient through a form, as described in the *Markdown* file. - The system must allow the registration of visits for already registered patients. - The system must present a list of registered patients, displaying their names and status. Below are some preliminary examples of how non-functional requirements should be described: - The system must be adaptable to different screen sizes. - The system must be compatible with the browsers available on the market.
 # Output Format
I want you to generate one table for functional requirements (FR) and another for non-functional requirements (NFR) in the following format: Requirement ID | Description | Priority FR 01 | The system must... | Essential

ByeBye Smoke Software System





Produced with Lovable

https://lovable.dev

Lessons Learned

Observations on embracing AI in the engineering of *ByeByeSmoke*:

Requirements Specification
Software Design
Project Management
Coding
Software Quality

Common Strengths

- **Acceleration of Tasks**
  - Generative AI consistently helped speed up repetitive, operational, and early-stage tasks—like drafting requirements, structuring test scenarios, and generating initial diagrams.
- **Prompt Engineering Matters**
  - Across every area, well-crafted, detailed, and contextualized prompts led to significantly better results. Markdown formatting, visual references, and examples were especially helpful.
- **Modular and Structured Development**
  - Dividing work by use cases or components improved modularity and manageability, especially in coding and design.

Common Strengths

- **Supportive Role in Creativity and Planning**
  - AI tools served as valuable assistants—enhancing productivity, aiding visualization, and inspiring creative solutions.

- **Human Oversight Is Essential**
  - Manual reviews, inspections, and expert supervision were critical to ensuring the quality, coherence, and correctness of outputs.

Common Limitations

- **Prompt Sensitivity and Fragility**
  - Vague or generic prompts led to poor results—simplified outputs, missing details, and even risky suggestions (e.g., weakened security policies).

- **Shallow or Incomplete Outputs**
  - Whether in use cases, test plans, or complex functionalities, the AI frequently missed edge cases, exceptions, or domain-specific logic.

- **Inconsistency Across Iterations**
  - The model often failed to maintain continuity, disregarding previously validated elements or introducing unrequested changes.

Common Limitations

- **Context Management Challenges**
  - AI struggled to retain and apply context across stages, requiring constant re-prompting and parallel documentation.

- **Security and Ethical Risks**
  - Poorly defined prompts could introduce vulnerabilities or biases, especially in QA and requirements stages.

- **Dependence on Experts**
  - Despite its capabilities, generative AI could not replace skilled professionals. Their involvement was crucial for validation, adaptation, and ethical oversight.

Embracing AI in Contemporary Software Systems Engineering

"Is it possible to develop a software system using only generative AI–based tools across the different phases of the development cycle?"

NOT YET!

Without Peopleware, no software!

Conjecture:

Embracing AI in Contemporary Software Systems Engineering can help address the lack of practices and resources to tackle CSS challenges.

YES!

If used with caution and applying an engineering perspective!

"We cannot ignore the opportunities that lie ahead. Nor should we disregard the concerns associated with them. Specifically, we must exercise caution ***against over-reliance on AI****. While the next generations of software engineers should be trained in prompt engineering and AI, this should not overshadow the necessity of core software engineering knowledge.* ***Human judgment remains indispensable for critically assessing AI-generated artifacts.*** *It is crucial to emphasize again that <u>AI serves as a tool to enhance developers' productivity and cannot (in the near future) replace humans.</u>*

<u>Putting too much trust in the software artifacts generated by AI can have serious repercussions on the quality and safety of our software systems</u>*."*

Terragni, V.; Vella; A.; Roop, P.; Blincoe, K. (2025) **The Future of AI-Driven Software Engineering.** ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3715003

"***Our research community must acknowledge and address the opportunities and challenges that arise from the use of AI in software engineering***. *Concerns persist regarding the quality of AI-generated code, with notable issues regarding security and privacy*."

"*Our research community stands at the forefront of this revolution*; ***we need to tempestively address the challenges of the symbiotic partnership between human developers and AI.***"

Terragni, V.; Vella; A.; Roop, P.; Blincoe, K. (2025) **The Future of AI-Driven Software Engineering.** ACM Transactions on Software Engineering and Methodology. https://doi.org/10.1145/3715003

Software has sometimes been compared to air: It's invisible and everywhere, and everyone and everything needs it. This feeling can lead to two different ways of considering software and, hence, software engineering: (1) letting it remain invisible and taking it for granted, or (2) nurturing it, caring for it, protecting it, and improving it.

Architecting the Future of Software Engineering: A National Agenda for Software Engineering Research & Development. (2021) Software Engineering Institute, Carnegie Mellon University
https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=741193

It is up to us to decide on letting it untamed or persist in taming it.

“You don't learn well except through experience.”
Francis Bacon

MERCI!

Guilherme Horta Travassos

PESC/COPPE

Federal University of Rio de Janeiro

CNPq researcher, CNE Faperj, ISERN member

ght@cos.ufrj.br

orcid: 0000-0002-4258-0424