Software Development: Art or Engineering?
Software Engineering @ PESC


Prof. Ana Regina
- Software Processes & Quality
- Maturity Models (MPS.Br)

Prof. Claudia Werner
- Software Reuse
- Software Ecosystems

Prof. Guilherme Horta Travassos
- Empirical Software Engineering
- Evidence-based Software Engineering

Prof. Toacy C. de Oliveira
- Software Reuse Processes
- Software Process Representation and Analysis
Prisma@COPPE/UFRJ

Prisma is a Research Group aiming to:
◦ Research and understand the underlying principles of Software Engineering.
◦ Engineer tools and techniques to aid how software systems are created and maintained.

Current Topics
◦ Software Process Representation and Analysis
  ◦ Software Process Representation with BPMN
  ◦ Tailoring in Software Process
  ◦ Software Process Composition
  ◦ Process Mining
◦ Software Reuse
  ◦ Reuse Process Specification and Analysis
  ◦ MDE as a Service
  ◦ Reuse in MDE

http://prisma.cos.ufrj.br/
Software Engineering

SCENARIO & DEFINITIONS
Increasing Demand

Software Systems are reshaping society.....

People...
- Are highly connected
- Take care of their own stuff
  - Buy tickets, do banking, get paid, find places, book hotels
- Demand more and more information (awareness, transparency)

Organizations (Companies, Governments, NGOs,..)...
- Are also highly connected
- Need to do more with less (be efficient)
- Need to be transparent and responsible

Both people and organizations generate tons of information!
Things need software 😊
In practice, Software Development practitioners battle between the "Real World" and the "Automated World" to create bespoke (semi-)automated systems that usually mimic real world procedures.
Art or Engineering.... Or just bakery !!!!!
Complexity!
+ Complexity

- LOT’S OF PEOPLE
- LOT’S OF INGREDIENTS
- LOT’S OF MONEY
Computer Science 101

How to deal with complexity?

• Rule 1: Divide and conquer

• Rule 2: Abstraction
Software Engineering – Definition

According to IEEE's definition, software engineering can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.
Art

According to Wikipedia 😊, Art is a diverse range of human activities in creating visual, auditory or performing artifacts (artworks), expressing the author's imaginative or technical skill, intended to be appreciated for their beauty or emotional power.

According to Apple Art is the expression or application of human creative skill and imagination, typically in a visual form such as painting or sculpture, producing works to be appreciated primarily for their beauty or emotional power.
Art vs Engineering

LET’S VOTE!!!!!!
Why SD is still a Joke?
And Still a Problem!!!
Software Engineering – Definition

According to IEEE's definition, software engineering can be defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software.

From the dictionary...

- **systematic**: def. done or acting according to a fixed plan.
- **disciplined**: def. showing a controlled form of behavior or way of working.
- **quantifiable**: def. express or measure the quantity.
Engineering can’t be that hard!!!!

- Plan,
- Control
- Quantify

Let’s try!!!!!
Materializing SE Concepts....

Software Processes?  Software Projects
Software Development Processes

A software process is a set of interrelated **activities** and **tasks** that transform **input work products** into **output work products**. At minimum, the **description** of a software process includes required inputs, transforming work activities, and outputs generated.

-Swebok, 2014

Software Development Processes (SDPc) attempt to represent common and successful practices that together facilitate how software systems are built or maintained. Processes typically come from independent "entities" such as ISO, CMM, MPS.Br and SCRUM Manifesto or from companies such as the Rational (Rational Unified Process).
Example – A Process
Software Development Projects

Software Development Projects (SDPj) using Software Engineering principles and concepts:

- Leverage on plans (to be systematic)
- Measure (time, budget, satisfaction, errors, etc.) what is happening
- Control (execution, implementation,...) what is happening

...and...

- Gather volatile and unstable information from users and the like (manuals, procedures, laws, etc.)
- Convert informal information to formal representations (NL => UML, BPMN, Java,)
- Add technology to the mix (web, mobile, tv, cloud, etc.)
Example – A Plan in SDPjs
Will they ever meet?

Software Processes come in several flavors:
  ◦ Methodology, Framework, Manifesto, Model, Rules

Materializing such concepts in projects is hard:
  ◦ Projects are unique by nature (particular timeframe, specific budget, unique set of requirements, etc.)

Supporting Tools (Trackers, Project Management) seem oblivious to processes, or not adopted at all (PSEEs)
Do they need to meet?

Well.....

Software Engineering => Concepts based on research and best-practices.


Projects => Instantiate and Execute Processes.

According to Fuggeta, Booch and “Bakers” ....a software process largely influences the software system under development (it is kind of obvious!!)
How to “impose” sequence?
- For some projects, Test A comes after Create A.

How to assure timely artifact creation?
- For some projects, a UserStory or Scenario must be created before coding.

How to avoid missing important events?
- Daily meetings are a practice for some projects.

How to gather relevant information?
- Quality may be measured in terms of rework.
SE Concepts Improvement

As with any other engineering discipline, SE has Concepts that will improve over time.

- Gather experiences from executing several software development project plans.
- Create conceptual Models.
- Compile such experiences/models in a body of knowledge such as UP, CMM, MPS, ISO, SWEBOK, books,..
- Replicate successful experiences.
What is a Process?

(Florac e Carleton. 1999)
The Process Improvement Lifecycle

The Software Process Improvement Lifecycle materializes the evolution of Software Engineering concepts and best-practices over time.

- PDCA (plan–do–check–act or plan–do–check–adjust) is an iterative four-step management method used in business for the control and continuous improvement of processes and products.
The cycle

- **PLAN** => Establish the objectives and **processes** necessary to deliver results in accordance with the expected output (the target or goals). By establishing output expectations, the completeness and accuracy of the spec is also a part of the targeted improvement. When possible start on a small scale to test possible effects.

- **DO** => Implement the plan, **execute the process**, make the product. **Collect data** for charting and analysis in the following "CHECK" and "ACT" steps.

- **CHECK** => Study the actual results (**measured and collected in "DO"** above) and compare against the expected results (targets or goals from the "PLAN") to ascertain any differences. Look for deviation in implementation from the plan and also look for the appropriateness and completeness of the plan to enable the execution, i.e., "Do". Charting data can make this much easier to see trends over several PDCA cycles and in order to convert the collected data into information. Information is what you need for the next step "ACT".

- **ACT** => If the **CHECK** shows that the PLAN that was implemented in DO is an improvement to the prior standard (baseline), then that becomes the new standard (baseline) for how the organization should ACT going forward (new standards are enACTed). If the **CHECK** shows that the PLAN that was implemented in DO is not an improvement, then the existing standard (baseline) will remain in place. In either case, if the **CHECK** showed something different than expected (whether better or worse), **then there is some more learning to be done**... and that will suggest potential future PDCA cycles.
Reifying the Halo – An Example

Every SD Process refers to a unique scenario, thus needing a bespoke solution.

On the right we have an example where processes P1, P4 are reified from the Halo and after a set of operations (tailoring, merge, instantiate), they originate P7, which is then executed as P8.

The PDCA Cycle is:
P = (Re+Ta+Me+In)
D = (En)
C/A = (Im)
The Halo 😊

The Software Process Halo contains a set of definitions based on extensive research and best-practices on Software Development.

The definitions found in the Halo are typically represented using natural language or a type of non-formal representations. Therefore their reification procedure (to become a process) requires experts in the “Halo”

It is important to stress the definitions found in the halo are extremely important to disseminate best-practices in Software Development!!!!!
Halo - Example

Agile Manifesto 12 Principles...

1. Customer satisfaction by early and continuous delivery of useful software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Self-organizing teams
12. Regular adaptation to changing circumstance

http://www.agilemanifesto.org/principles.html

MPS.Br

Programa MPS.Br – Melhoria de Processo do Software Brasileiro, é uma iniciativa para melhorar a capacidade de desenvolvimento de software nas empresas brasileiras.

- AP 2.1 O processo é gerenciado
  - RAP 2. Existe uma política organizacional estabelecida e mantida para o processo.
  - RAP 3. A execução do processo é planejada.
  - RAP 4. (Para o nível G) A execução do processo é monitorada e ajustes são realizados.
  - RAP 4. (A partir do nível F) Medidas são planejadas e coletadas para monitoração da execução do processo e ajustes são realizados.
  - RAP 5. As informações e os recursos necessários para a execução do processo são identificados e disponibilizados.

http://www.softex.br/mpsbr/mps/mps-br-em-numeros/
Reifying the Halo – The Reification Procedure

The "reification procedure" (Re) is typically executed by Halo experts to materialize concepts from the Halo so that PDCA can be in place.

- Hard to predict: What concepts best fit a given software development scenario?

![Diagram showing the reification process with labels P1, P2, P3, P4, Re, and Ta.]
Reifying the Halo – The Tailoring Procedure

A Reified Process such as P1, is not adequate for all scenarios and needs adaptations.

P1 needs to take into consideration: team expertise, time, budget, technology, business regulations, etc.

The tailoring procedure (Ta) allows adding, adapting and removing reified halo concepts (process elements)
Tailoring Example
Reifying the Halo – The Merging Procedure

Sometimes reified halo concepts need to be merged (ex. To incorporate a technology such as Spring, Jhipster, Business Regulations or Javascript that have their own ways).

*Merging* (Me) is about composing and "aligning namespaces"
- Role Tester vs Role Software Tester
- Parallel/Sequential/Inline composition
Merging Example
Reifying the Halo – The Instantiation Procedure

*Instantiation* (In) can also be defined as planning where the process concepts are deeply "integrated" with the underlying execution context.

- Break the work in several Iterations (It).
- How many "Define UseCase" activities are executed? When? By Whom?

By the way..this may be a big difference between BPs and SPs. In SPs there maybe no single execution trace but a series of iteration traces that should be gathered.
Instantiation Example
Reifying the Halo – The Enacting Procedure

*Enacting* (En) a Software Process is where the action takes place.

Several tools allow controlling the flow (Gantt?)

Some process elements are modified on the fly (activities skipped, redone or added; artifacts deferred)
Enacting Example

https://processhub.owse.com.br

https://university.atlassian.com
Improving (Im) a Software Process deals with modifying processes (or the Halo) based on lessons learned, typically after enacting.

Some process elements are modified during the project and such modifications may deliver better results.
Improvement Example
Software Processes as Knowledge Intensive Processes
Software Processes

“A series of actions that produce something or that lead to a particular result."

- Merrian

"A process is a collection of interrelated work tasks initiated in response to an event that achieves a specific result for the customer of the process."

- CBOK 2009

"A software process is a set of interrelated activities and tasks that transform input work products into output work products. At minimum, the description of a software process includes required inputs, transforming work activities, and outputs generated."

- Swebok, 2014.
KiPs

Knowledge-intensive Processes (KiPs): processes whose conduct and execution are heavily dependent on knowledge workers performing various interconnected knowledge intensive decision making tasks. KiPs are genuinely knowledge, information and data centric and require substantial flexibility at design- and run-time.
Conclusion

SOFTWARE DEVELOPMENT IS STILL A MIX OF ART AND ENGINEERING AS LONG AS PEOPLE PLAY THE CENTRAL ROLE....

......AND THIS IS THE REASON WE WILL NEVER BECOME UNEMPLOYED 😊😊😊😊
Questions??