There Is Only One Time In Soft. & Sys. Engineering!

Towards a *Continuous (model-based) Engineering*

Prof. Benoit Combemale
University of Rennes
IRISA & Inria, DiverSE team

benoit.combemale@irisa.fr
http://combemale.fr
@bcombemale

Thanks to my students and all the colleagues from DiverSE, the Bellairs and WMM workshop series, the Inria/CWI Associate Team ALE, and the MDEnet International group (esp., AE group)

One Time

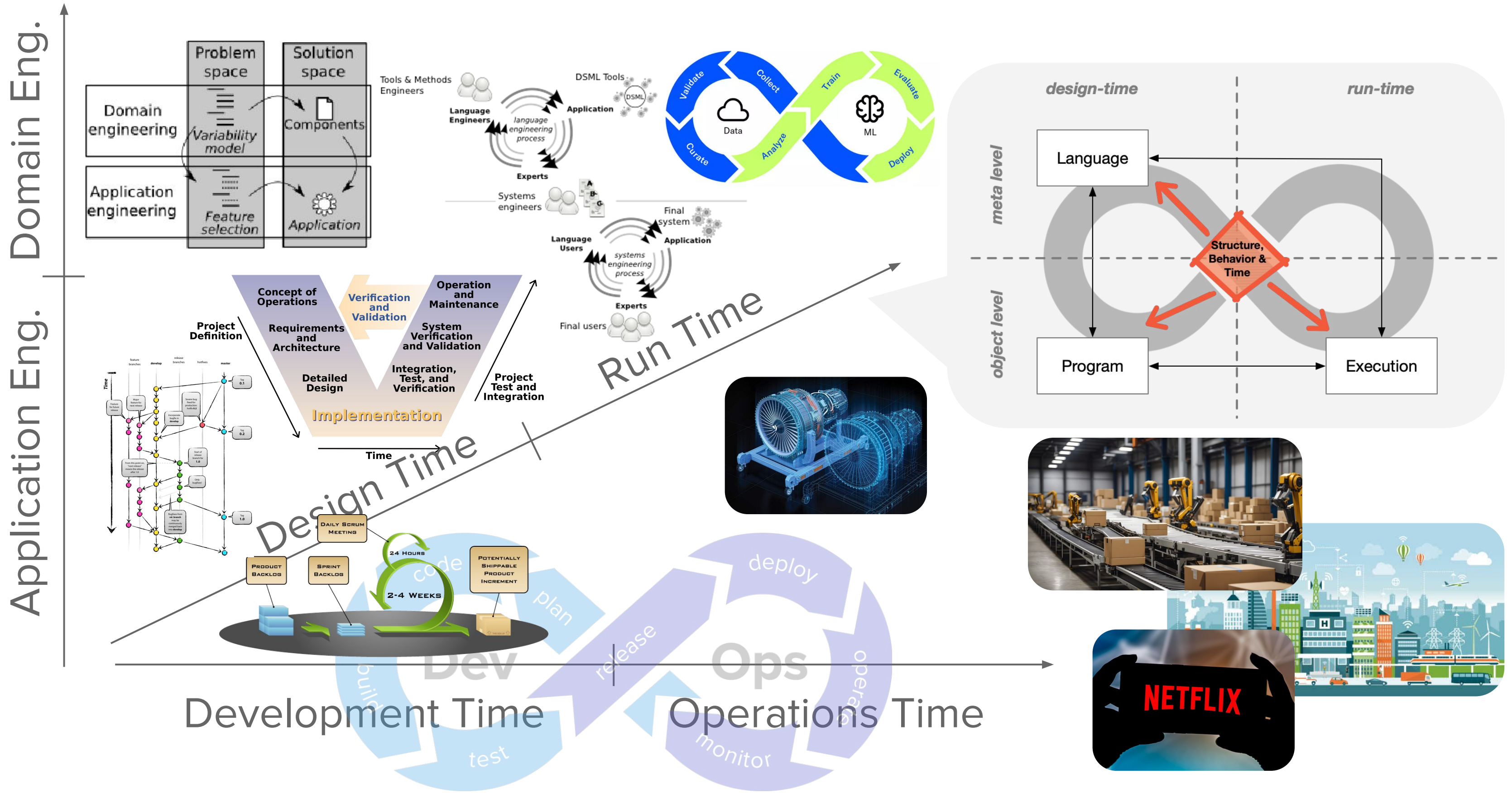
Wait! What?

Design time
Development time
Training time
Compile time
Just-in-time
Run-time
Operation time
…

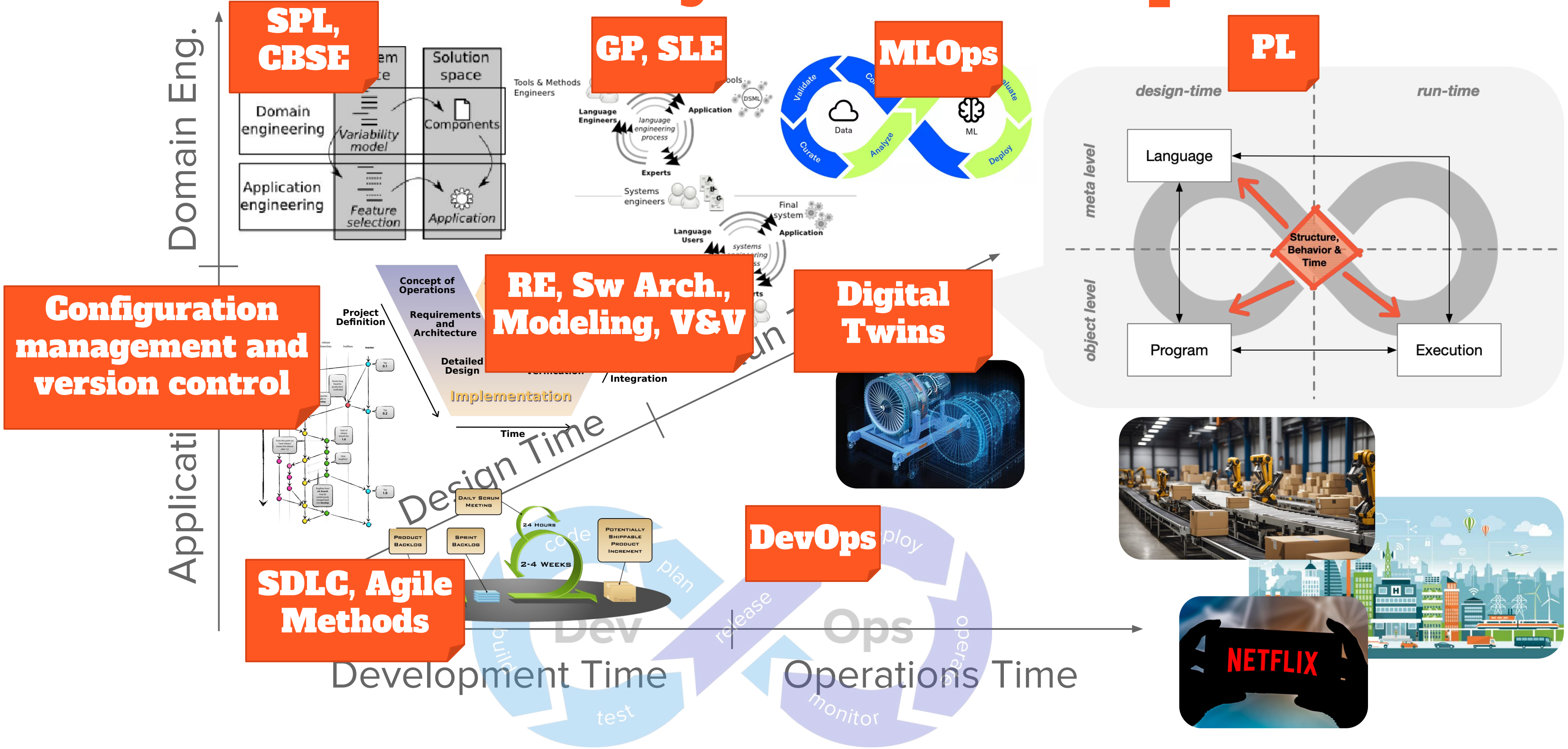
To what refer these “times” in SE?

A tool? an activity? a moment in the life cycle?

Focus on Software Systems Development



Focus on Software Systems Development



Dogma of (Traditional) Software Engineering

“Organizations which design systems (in the broad sense used here) are constrained to produce designs which are copies of the communication structures of these organizations.” —Melvin E. Conway, How Do Committees Invent?

⇒ **Conway’s law applied!**

“The wealth of methods and tools that are used at development-time to forge software have no more use when the software enters the run-time stage” — Luciano Baresi and Carlo Ghezzi. The disappearing boundary between development-time and run-time. In Future of software engineering research (FoSER '10).

S,D ⊨ R, where D might continuously change, due to mobility,
but also more recently to socio interactions, wicked phenomena to consider, etc.

In the context of S&S Eng. this prevents both

- a **seamless and continuous cross-fertilization** over the engineering processes, and
- to **explore new scenarios** beyond the ones captured in the established engineering processes

Taming Software Hyper Agility

- **Software systems development belongs to a multi-dimensional space**: nb **function points**, nb **concerns**, **configuration** space, **release** frequency, nb **execution platforms**, **correctness** space & guarantee…
- **Software systems must adapt** not only to a fixed space of variable requirements, but also **to an emerging chain of changing requirements**, often **driven by incoming input data**

➢ **Software Engineering must embrace this new temporal adaptability over a multi-dimensional space!**

⇒ Design-space exploration, trade-off analysis & decision making all along the life cycle

Towards a Continuous (model-driven) Software Engineering

Deep Variability

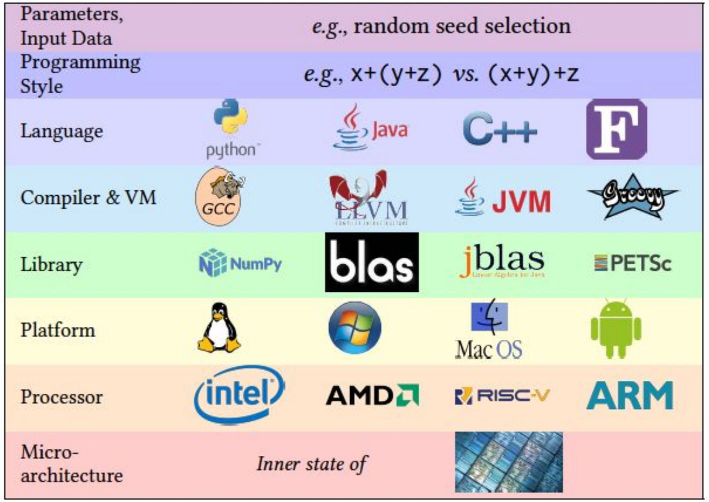
Variability occurs on all concerns

Variability showcases interdependencies

Variability impacts soft/sys properties

=> combinatorial explosion of the epistemic and ontological variability

Deep Variability *refer to the interaction of all concerns modifying the behavior (including both functional and nonfunctional properties) of a software system*



Evidences of deep variability:

- Climate model
- Machine learning
- Neuroimaging
- Bluff-body aerodynamics
- Performance modeling of software
- Reproducible builds
- etc.

Our Vision

Embrace deep variability!

Explicit modeling of the variability points and their relationships, such as:

1. Get insights into the variability concerns and their possible interactions
2. Capture and document configurations for the sake of **reproducibility**
3. Explore diverse configurations to **replicate**, and hence optimize, validate, increase the robustness, or provide better resilience

Embracing Deep Variability For Reproducibility & Replicability

Mathieu Acher, Benoit Combemale, Georges Aaron Randrianaina, Jean-Marc Jézéquel
IRISA, Université de Rennes
Rennes, France

ABSTRACT

Reproducibility (*a.k.a.*, determinism in some cases) constitutes a fundamental aspect in various fields of computer science, such as floating-point computations in numerical analysis and simulation, concurrency models in parallelism, reproducible builds for third parties integration and packaging, and containerization for execution environments. These concepts, while pervasive across diverse concerns, often exhibit intricate inter-dependencies, making it challenging to achieve a comprehensive understanding. In this short and vision paper we delve into the application of software engineering techniques, specifically variability management, to systematically identify and explicit points of variability that may give rise to reproducibility issues (*e.g.*, language, libraries, compiler, virtual machine, OS, environment variables, *etc.*). The primary objectives are: i) gaining insights into the variability layers and their possible interactions, ii) capturing and documenting configurations for the sake of reproducibility, and iii) exploring diverse configurations to replicate, and hence validate and ensure the robustness of results. By adopting these methodologies, we aim to address the complexities associated with reproducibility and replicability in modern software systems and environments, facilitating a more comprehensive and nuanced perspective on these critical aspects.

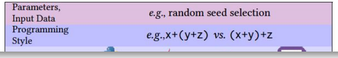
In this paper we propose to characterize both intended and unintended variability of any software-intensive system in order to support reproducibility and replicability, and eventually estimate its robustness, uncertainty profile, and explore different hypotheses.

2 DEEP SOFTWARE VARIABILITY

Uncertainty in informatics comes from many different origins [16, 36], either ontological (*i.e.*, inherent unpredictability, *e.g.*, aleatory) or epistemic (*i.e.*, due to insufficient knowledge).

Ontological causes include noise in the input data of a program, its memory layout, network delays, the internal state of the processor, the ambient temperature and even the age of the processor[1].

Epistemic causes include misunderstanding of the user's needs, variable behavior of conceptually similar resolution methods, choice of threshold parameters, unexpected behavior of APIs, variable behavior among functionally similar libraries, or subtle differences in the semantics of programming languages (*e.g.*, $-3\%2$ evaluates to -1 in Java but to 1 in Python), or even inside the same programming language (for instance $x/0$ is an undefined behavior in C).



ACM REP 2024

⇒ We aim to **address the complexities associated with reproducibility and replicability in modern software systems and environments**, facilitating a more comprehensive and nuanced perspective on these critical concerns.

Reproducibility in Software Engineering

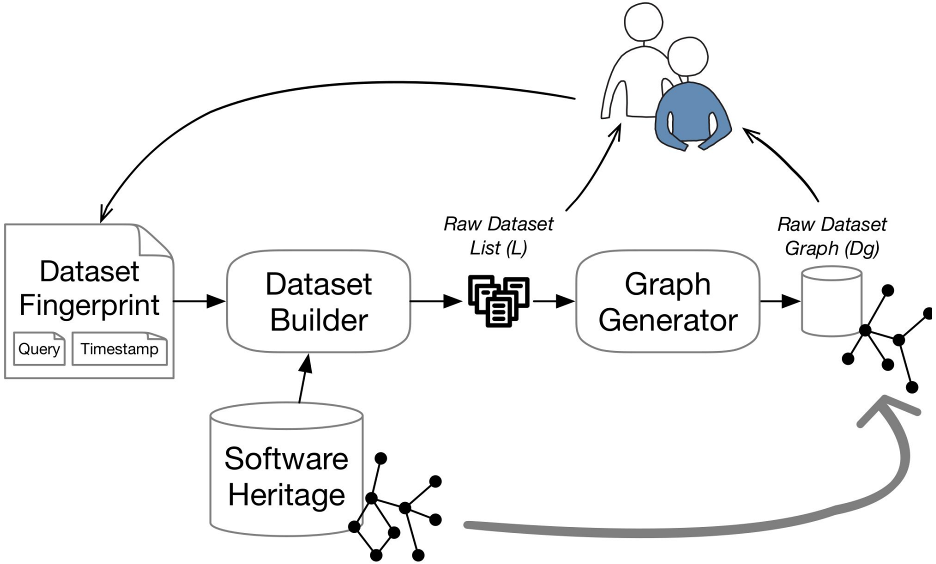
- Reproducibility (a.k.a., *determinism* in some cases) constitutes a fundamental aspect in various fields of computer science

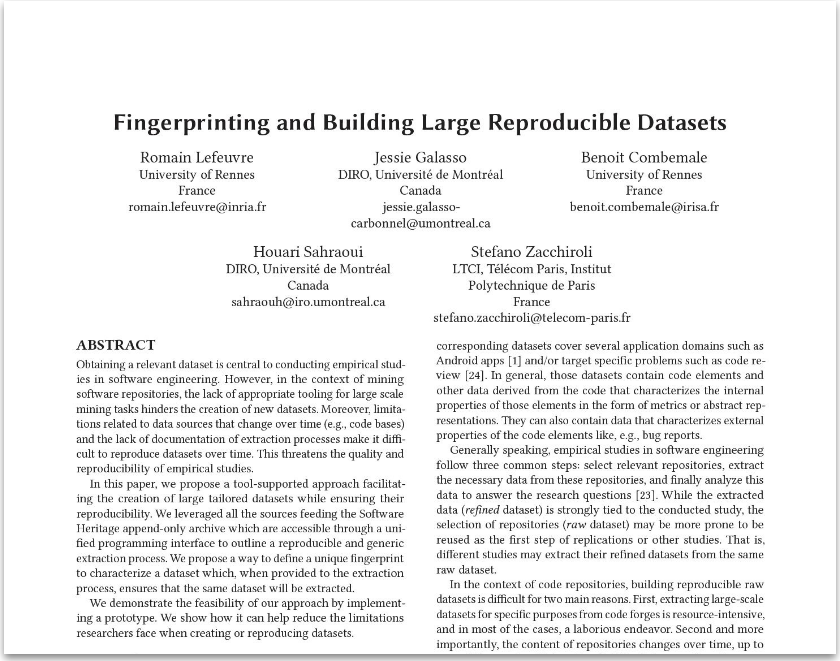
    e.g., floating-point computations in numerical analysis and simulation, concurrency models in parallelism, reproducible builds for third parties integration and packaging, and containerization for execution environments.

- These concepts, while pervasive across diverse concerns, often exhibit intricate inter-dependencies, making it challenging to achieve a comprehensive understanding

- Ongoing initiatives try to fix *(part of)* the configuration, e.g., datasets, sbom, builds, runtime environments, IaC, etc.

Reproducibility in Empirical Studies: The Case of Datasets





Fingerprinting and Building Large Reproducible Datasets

Romain Lefeuvre
University of Rennes
France
romain.lefeuvre@inria.fr

Jessie Galasso
DIRO, Université de Montréal
Canada
jessie.galasso-carbonnel@umontreal.ca

Benoit Combemale
University of Rennes
France
benoit.combemale@irisa.fr

Houari Sahraoui
DIRO, Université de Montréal
Canada
sahraouh@iro.umontreal.ca

Stefano Zacchiroli
LTCI, Télécom Paris, Institut Polytechnique de Paris
France
stefano.zacchiroli@telecom-paris.fr

ABSTRACT

Obtaining a relevant dataset is central to conducting empirical studies in software engineering. However, in the context of mining software repositories, the lack of appropriate tooling for large scale mining tasks hinders the creation of new datasets. Moreover, limitations related to data sources that change over time (e.g., code bases) and the lack of documentation of extraction processes make it difficult to reproduce datasets over time. This threatens the quality and reproducibility of empirical studies.

In this paper, we propose a tool-supported approach facilitating the creation of large tailored datasets while ensuring their reproducibility. We leveraged all the sources feeding the Software Heritage append-only archive which are accessible through a unified programming interface to outline a reproducible and generic extraction process. We propose a way to define a unique fingerprint to characterize a dataset which, when provided to the extraction process, ensures that the same dataset will be extracted.

We demonstrate the feasibility of our approach by implementing a prototype. We show how it can help reduce the limitations researchers face when creating or reproducing datasets.

corresponding datasets cover several application domains such as Android apps [1] and/or target specific problems such as code review [24]. In general, those datasets contain code elements and other data derived from the code that characterizes the internal properties of those elements in the form of metrics or abstract representations. They can also contain data that characterizes external properties of the code elements like, e.g., bug reports.

Generally speaking, empirical studies in software engineering follow three common steps: select relevant repositories, extract the necessary data from these repositories, and finally analyze this data to answer the research questions [23]. While the extracted data (*refined* dataset) is strongly tied to the conducted study, the selection of repositories (*raw* dataset) may be more prone to be reused as the first step of replications or other studies. That is, different studies may extract their refined datasets from the same raw dataset.

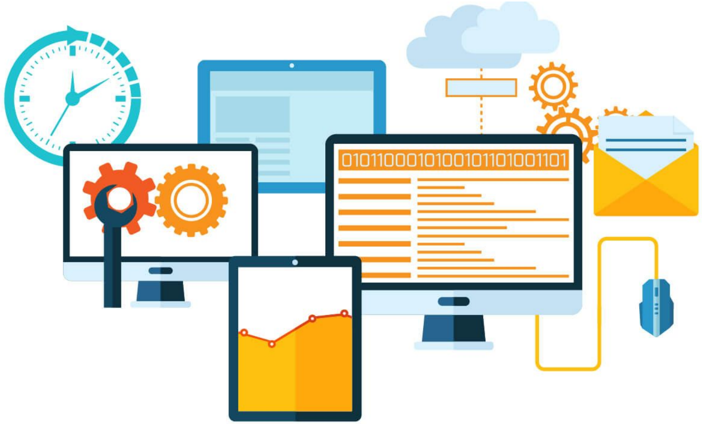
In the context of code repositories, building reproducible raw datasets is difficult for two main reasons. First, extracting large-scale datasets for specific purposes from code forges is resource-intensive, and in most of the cases, a laborious endeavor. Second and more importantly, the content of repositories changes over time, up to

ACM REP 2023, cf. https://hal.science/hal-04132604

What about Replicability?

Exploring various configurations:

- Make more **robust** scientific findings
- Define and assess the **validity** enveloppe
- Enable **exploration** and **optimization**
- Ensure a better **resilience**



⇒ We propose to embrace deep variability to face software hyper agility, for the sake of **replicability modulo heuristics** (i.e., kpi, mco, quality attributes…)

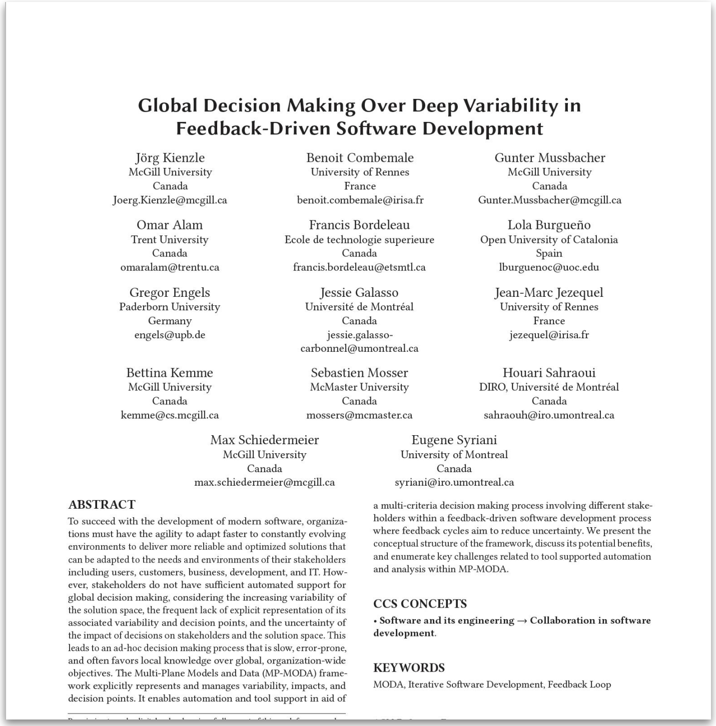
Feedback-Driven Software Development

Deep Software Variability needs decision-making support

- variability all along the technological stack
- various stakeholders
- inter-dependencies between concerns
- decision making is de facto iterative

The MultiPlane MODA Framework (Bellairs'22)

- encapsulate the variability and impact intra-/inter- plane
- **"Decision Space"** that derives from the dependencies in individual variability models and impact models
- **global feedback loop**



Global Decision Making Over Deep Variability in Feedback-Driven Software Development

Jörg Kienzle, McGill University, Canada, Joerg.Kienzle@mcgill.ca
Benoit Combemale, University of Rennes, France, benoit.combemale@irisa.fr
Gunter Mussbacher, McGill University, Canada, Gunter.Mussbacher@mcgill.ca
Omar Alam, Trent University, Canada, omaralam@trentu.ca
Francis Bordeleau, Ecole de technologie superieure, Canada, francis.bordeleau@etsmtl.ca
Lola Burgueño, Open University of Catalonia, Spain, lburguenoc@uoc.edu
Gregor Engels, Paderborn University, Germany, engels@upb.de
Jessie Galasso, Université de Montréal, Canada, jessie.galasso-carbonnel@umontreal.ca
Jean-Marc Jezequel, University of Rennes, France, jezequel@irisa.fr
Bettina Kemme, McGill University, Canada, kemme@cs.mcgill.ca
Sebastien Mosser, McMaster University, Canada, mossers@mcmaster.ca
Houari Sahraoui, DIRO, Université de Montréal, Canada, sahraouh@iro.umontreal.ca
Max Schiedermeier, McGill University, Canada, max.schiedermeier@mcgill.ca
Eugene Syriani, University of Montreal, Canada, syriani@iro.umontreal.ca

ABSTRACT

To succeed with the development of modern software, organizations must have the agility to adapt faster to constantly evolving environments to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, business, development, and IT. However, stakeholders do not have sufficient automated support for global decision making, considering the increasing variability of the solution space, the frequent lack of explicit representation of its associated variability and decision points, and the uncertainty of the impact of decisions on stakeholders and the solution space. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. The Multi-Plane Models and Data (MP-MODA) framework explicitly represents and manages variability, impacts, and decision points. It enables automation and tool support in aid of a multi-criteria decision making process involving different stakeholders within a feedback-driven software development process where feedback cycles aim to reduce uncertainty. We present the conceptual structure of the framework, discuss its potential benefits, and enumerate key challenges related to tool supported automation and analysis within MP-MODA.

CCS CONCEPTS

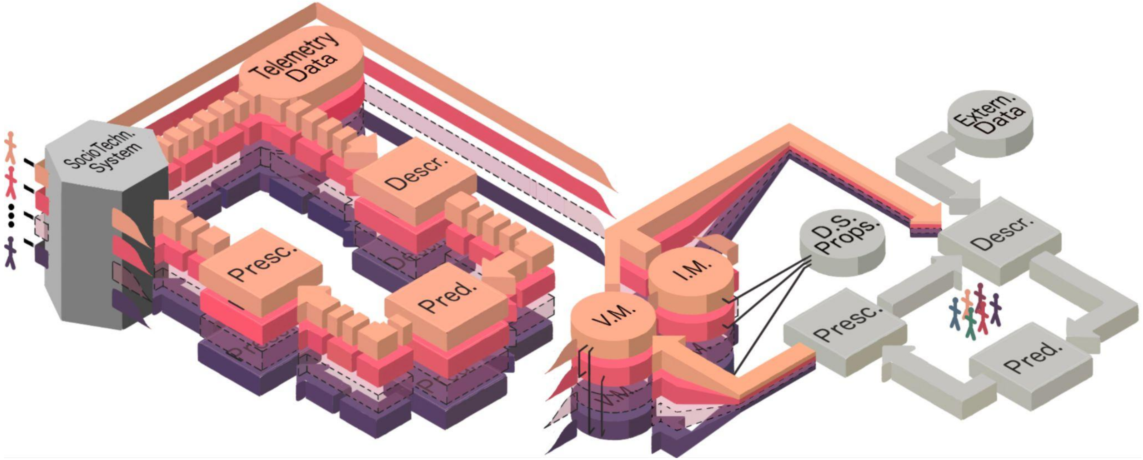
• **Software and its engineering → Collaboration in software development**.

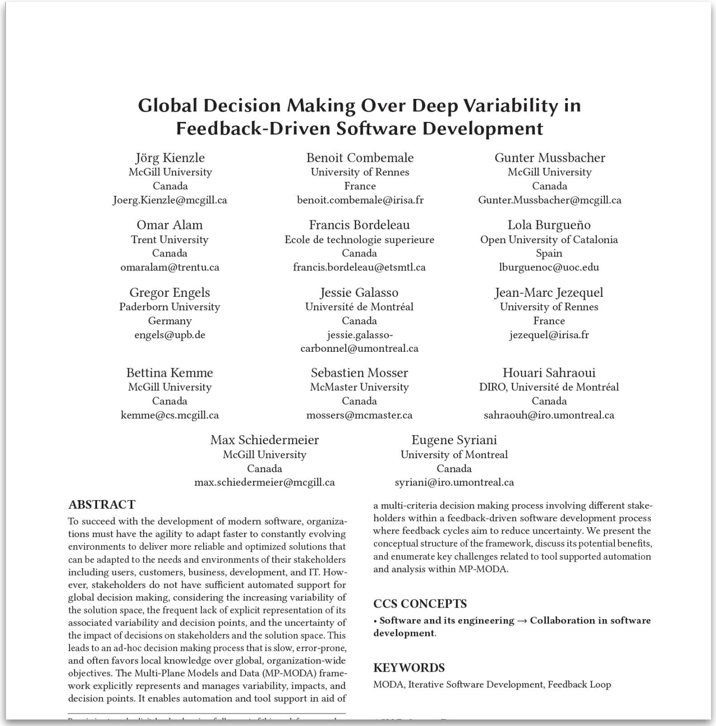
KEYWORDS

MODA, Iterative Software Development, Feedback Loop

ASE 2022, cf. https://hal.inria.fr/hal-03770004

Feedback-Driven Software Development





Global Decision Making Over Deep Variability in Feedback-Driven Software Development

Jörg Kienzle, McGill University, Canada, Joerg.Kienzle@mcgill.ca
Benoit Combemale, University of Rennes, France, benoit.combemale@irisa.fr
Gunter Mussbacher, McGill University, Canada, Gunter.Mussbacher@mcgill.ca
Omar Alam, Trent University, Canada, omaralam@trentu.ca
Francis Bordeleau, Ecole de technologie superieure, Canada, francis.bordeleau@etsmtl.ca
Lola Burgueño, Open University of Catalonia, Spain, lburguenoc@uoc.edu
Gregor Engels, Paderborn University, Germany, engels@upb.de
Jessie Galasso, Université de Montréal, Canada, jessie.galasso-carbonnel@umontreal.ca
Jean-Marc Jezequel, University of Rennes, France, jezequel@irisa.fr
Bettina Kemme, McGill University, Canada, kemme@cs.mcgill.ca
Sebastien Mosser, McMaster University, Canada, mossers@mcmaster.ca
Houari Sahraoui, DIRO, Université de Montréal, Canada, sahraouh@iro.umontreal.ca
Max Schiedermeier, McGill University, Canada, max.schiedermeier@mcgill.ca
Eugene Syriani, University of Montreal, Canada, syriani@iro.umontreal.ca

ABSTRACT

To succeed with the development of modern software, organizations must have the agility to adapt faster to constantly evolving environments to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, business, development, and IT. However, stakeholders do not have sufficient automated support for global decision making, considering the increasing variability of the solution space, the frequent lack of explicit representation of its associated variability and decision points, and the uncertainty of the impact of decisions on stakeholders and the solution space. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. The Multi-Plane Models and Data (MP-MODA) framework explicitly represents and manages variability, impacts, and decision points. It enables automation and tool support in aid of a multi-criteria decision making process involving different stakeholders within a feedback-driven software development process where feedback cycles aim to reduce uncertainty. We present the conceptual structure of the framework, discuss its potential benefits, and enumerate key challenges related to tool supported automation and analysis within MP-MODA.

CCS CONCEPTS

• **Software and its engineering → Collaboration in software development.**

KEYWORDS

MODA, Iterative Software Development, Feedback Loop

ASE 2022, cf. https://hal.inria.fr/hal-03770004

RE for Cyber-Physical Systems Development

1. There is a clear need for advanced global decision support that is **cross-discipline** and reduces information overload while prioritizing uncertain or hard areas

2. It is a challenge to **reduce information overload** while making balanced decisions that address uncertainty and maintain ecosystem equilibrium to achieve **continuous decision making**

Global Decision Making Support for Complex System Development

Lola Burgueño
ITIS Software
University of Malaga
Malaga, Spain
lolaburgueno@uma.es

Damien Foures
DDMS
Airbus Group
Toulouse, France
damien.da.foures@airbus.com

Benoit Combemale
ESIR & IRISA
University of Rennes
Rennes, France
benoit.combemale@irisa.fr

Jörg Kienzle
ITIS Software / School of Computer Science
University of Málaga / McGill University
Málaga, Spain / Montreal, Canada
joerg.kienzle@uma.es / mcgill.ca

Gunter Mussbacher
Department of Electrical and Computer Engineering
McGill University / INRIA
Montreal, Canada / Rennes, France
gunter.mussbacher@mcgill.ca

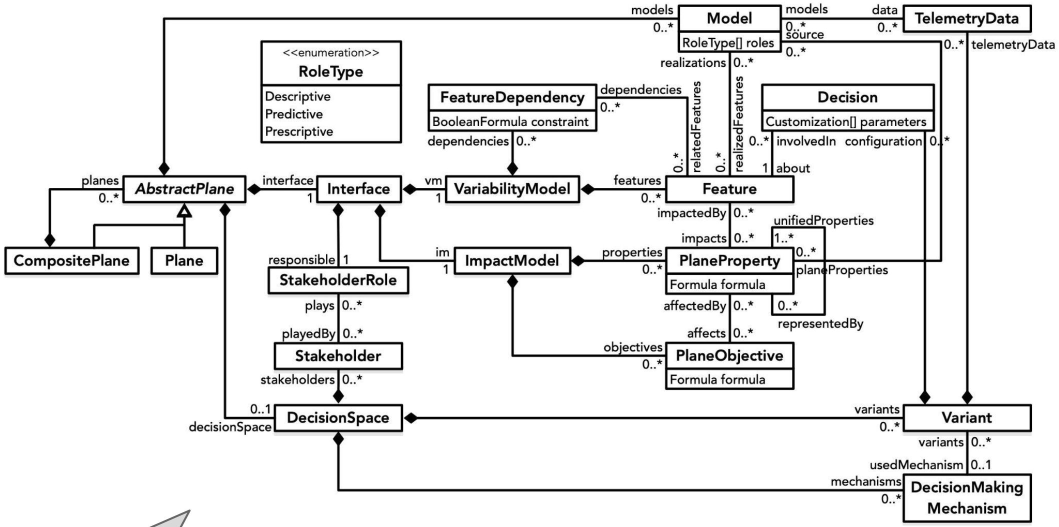
***Abstract*—To succeed with the development of modern and complex systems (e.g., aircrafts or production systems), organizations must have the agility to adapt faster to constantly evolving requirements in order to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, suppliers, and partners. However, stakeholders do not have sufficiently explicit and systematic support for global decision making, considering the vast decision space and complex inter-relationships. This decision space is characterized by increasing yet inadequately represented variability and the uncertainty of the impact of decisions on stakeholders and the solution space. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. As a result, one team's design decisions may impose too restrictive requirements on another team. In this paper, we evaluate our understanding of global decision making in the context of complex system development based on a conceptual model which explicitly represents and manages decision spaces including variability and impacts. We have conducted our evaluation by means of a case study where we interviewed domain experts with an average of 20 years of experience in complex system industries and report the key findings and remaining challenges. In the future, we aim at providing explicit and systematic tool-supported approaches for global decision making support for complex systems.**

***Index Terms*—Global Decision Making, Multi-Stakeholder, Variability, Impact, Requirements, Design.**

the organization from reaching a better global result. Global decision making that considers not only the solution space of each specialized team, but also the overall solution space of the organization is required. Furthermore, organizations must have the agility to adapt faster to constantly evolving requirements to succeed with complex system development by delivering more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, suppliers, and partners. Without automated support, teams have to revert to an ad-hoc decision making process for requirements and design that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. The vast decision space with ever increasing, but inadequately represented variability, and the uncertainty of the impact of decisions on stakeholders and the solution space further exacerbate this decision making problem. While there is a growing body of knowledge around variability management and decision making (see related work in Section VI), the challenges of global decision making in complex system development in an industrial context are not yet well understood.

Based on an initial conceptual model, we evaluate our understanding of global decision making by means of a case study in the context of complex system development and report our findings in this paper. The conceptual model explicitly

RE for Complex System Development



continuous decision making support?

uncertainty management?

Global Decision Making Support for Complex System Development

Lola Burgueño
ITIS Software
University of Malaga
Malaga, Spain
lolaburgueno@uma.es

Damien Foures
DDMS
Airbus Group
Toulouse, France
damien.da.foures@airbus.com

Benoit Combemale
ESIR & IRISA
University of Rennes
Rennes, France
benoit.combemale@irisa.fr

Jörg Kienzle
ITIS Software / School of Computer Science
University of Málaga / McGill University
Málaga, Spain / Montreal, Canada
joerg.kienzle@uma.es / mcgill.ca

Gunter Mussbacher
Department of Electrical and Computer Engineering
McGill University / INRIA
Montreal, Canada / Rennes, France
gunter.mussbacher@mcgill.ca

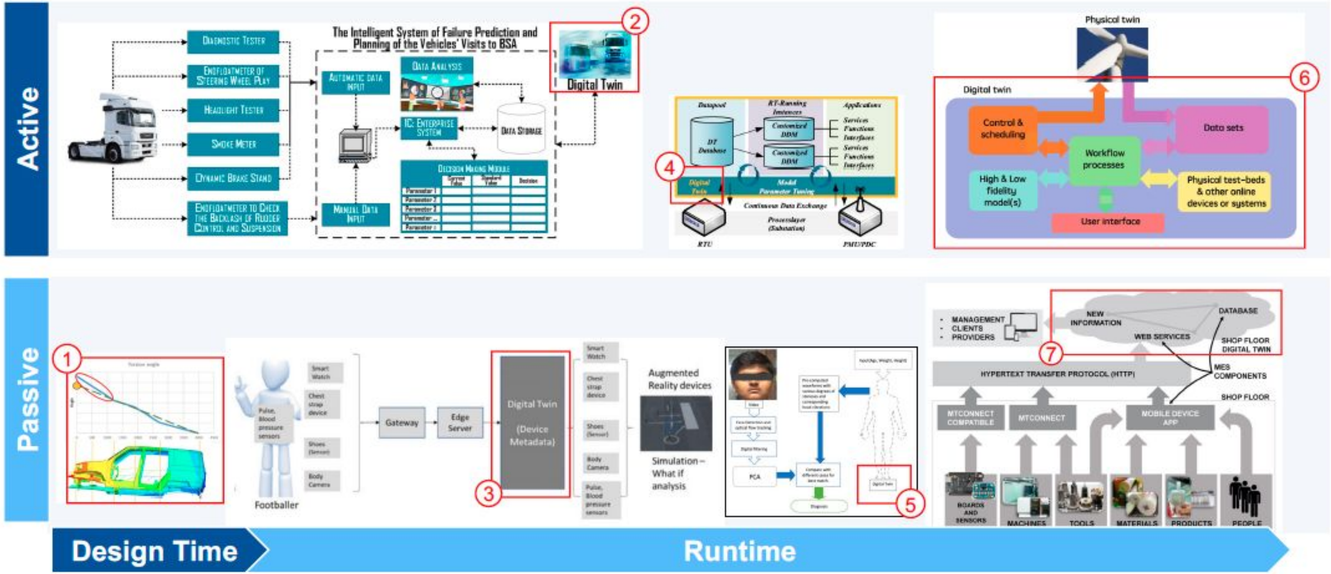
***Abstract*—To succeed with the development of modern and complex systems (e.g., aircrafts or production systems), organizations must have the agility to adapt faster to constantly evolving requirements in order to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, suppliers, and partners. However, stakeholders do not have sufficiently explicit and systematic support for global decision making, considering the vast decision space and complex inter-relationships. This decision space is characterized by increasing yet inadequately represented variability and the uncertainty of the impact of decisions on stakeholders and the solution space. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. As a result, one team's design decisions may impose too restrictive requirements on another team. In this paper, we evaluate our understanding of global decision making in the context of complex system development based on a conceptual model which explicitly represents and manages decision spaces including variability and impacts. We have conducted our evaluation by means of a case study where we interviewed domain experts with an average of 20 years of experience in complex system industries and report the key findings and remaining challenges. In the future, we aim at providing explicit and systematic tool-supported approaches for global decision making support for complex systems.**

***Index Terms*—Global Decision Making, Multi-Stakeholder, Variability, Impact, Requirements, Design.**

the organization from reaching a better global result. Global decision making that considers not only the solution space of each specialized team, but also the overall solution space of the organization is required. Furthermore, organizations must have the agility to adapt faster to constantly evolving requirements to succeed with complex system development by delivering more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, suppliers, and partners. Without automated support, teams have to revert to an ad-hoc decision making process for requirements and design that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. The vast decision space with ever increasing, but inadequately represented variability, and the uncertainty of the impact of decisions on stakeholders and the solution space further exacerbate this decision making problem. While there is a growing body of knowledge around variability management and decision making (see related work in Section VI), the challenges of global decision making in complex system development in an industrial context are not yet well understood.

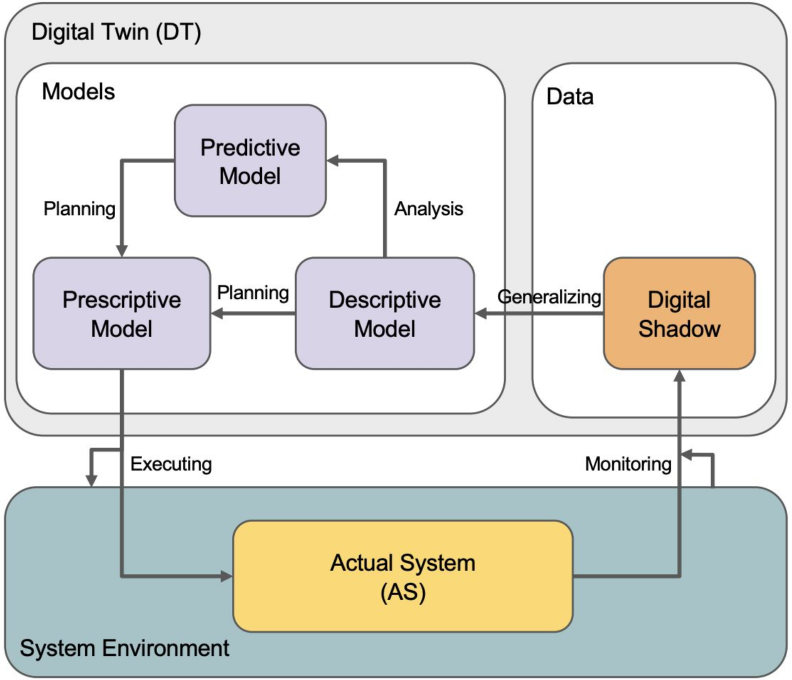
Based on an initial conceptual model, we evaluate our understanding of global decision making by means of a case study in the context of complex system development and report our findings in this paper. The conceptual model explicitly

Digital Twin: Seamless Continuum over Engineering Processes



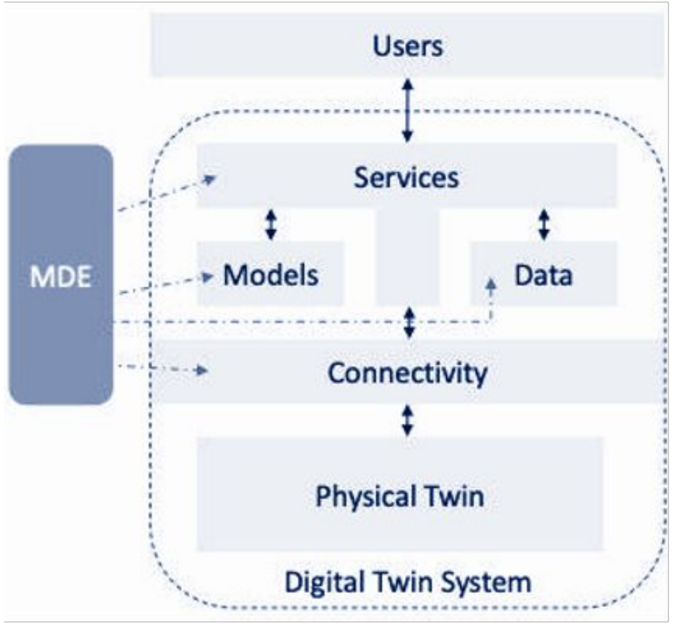
Dalibor, Jansen, Rumpe, Schmalzing, Wortmann: *A Cross-Domain Systematic Mapping Study on Software Engineering for Digital Twins.* In: Journal of Systems and Software, 2022.

Digital Twin: The Role of Models and Data



Conceptualizing Digital Twins. Romina Eramo, Francis Bordeleau, Benoit Combemale, et al.. IEEE Software, March-April 2022, pp. 39-46, vol. 39.

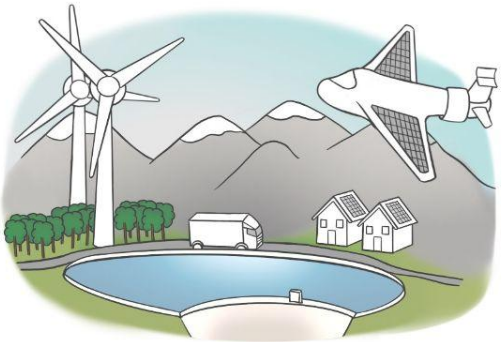
Digital Twin: The Role of MDE



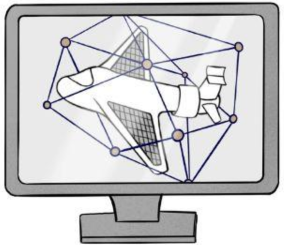
Towards Model-Driven Digital Twin Engineering: Current Opportunities and Future Challenges. Francis Bordeleau, Benoit Combemale, Romina Eramo, et al.. ICSMM 2020.

Model-Driven Engineering of Digital Twins.
Dagstuhl Seminar #22362, 2022.
https://www.dagstuhl.de/22362

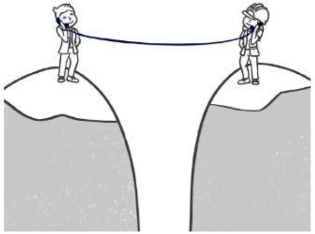
Nowadays products are gaining in **complexity**, operated in **various environments** with increasing interactions and **multiple** use cases.



Our multi-field adaptive modeling technology offers you an **innovative digital representation** of your product



Comprehensive view from design to maintenance



Efficient collaboration between expertise fields

Efforts focused in the right place

Our digital twin solution is made of **open-source** modules **compatible** with your existing tools.

Quicker and smarter design

Reduced operating costs

Assessed maintenance costs

Developed by a **highly skilled team** led by :

Dr. Guy DE SPIEGELEER, CEO
guy.de-spiegeleer@twiinit.com
Aerospace design, system engineering

Eng. Adrien DELSALLE, CTO
adrien.delsalle@twiinit.com
Computer science & modeling

Scientifics **advisors** from Inria

Prof. Benoit COMBEMALE
CSA
Systems eng., Open Source Software

Prof. Olivier BARAIS
CTA
Web dev., DevOps

twiinIT
Collaborative Design Software

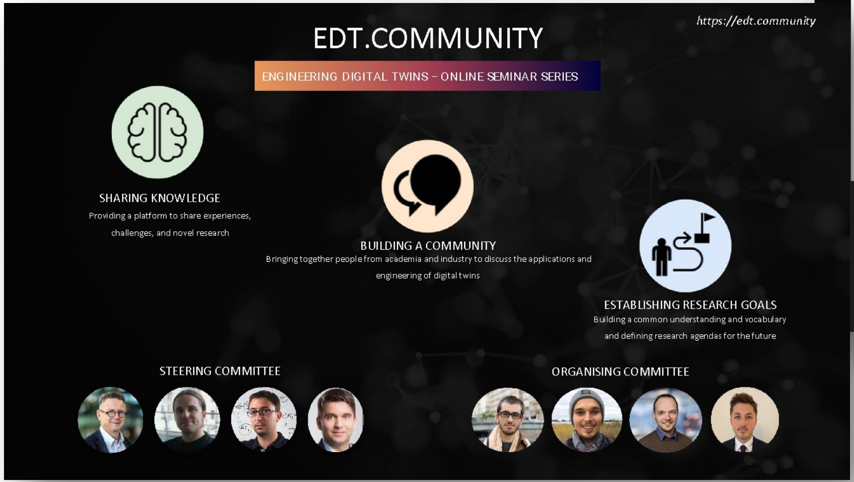
www.twiinit.com

SAFRAN SAINT-GOBAIN Inria StartupStudio ASTech bpifrance LES DEEP TECH HEC PARIS CHALLENGE+

This document contains proprietary, confidential, and copyrighted materials of twiinIT©
Any use or disclosure in whole or in part of licensed information without the express written permission of twiinIT© is prohibited

2022 - SAS RCS Nanterre registered - SIRET : 913 780 482

Engineering Digital Twins (EDT): An International Community



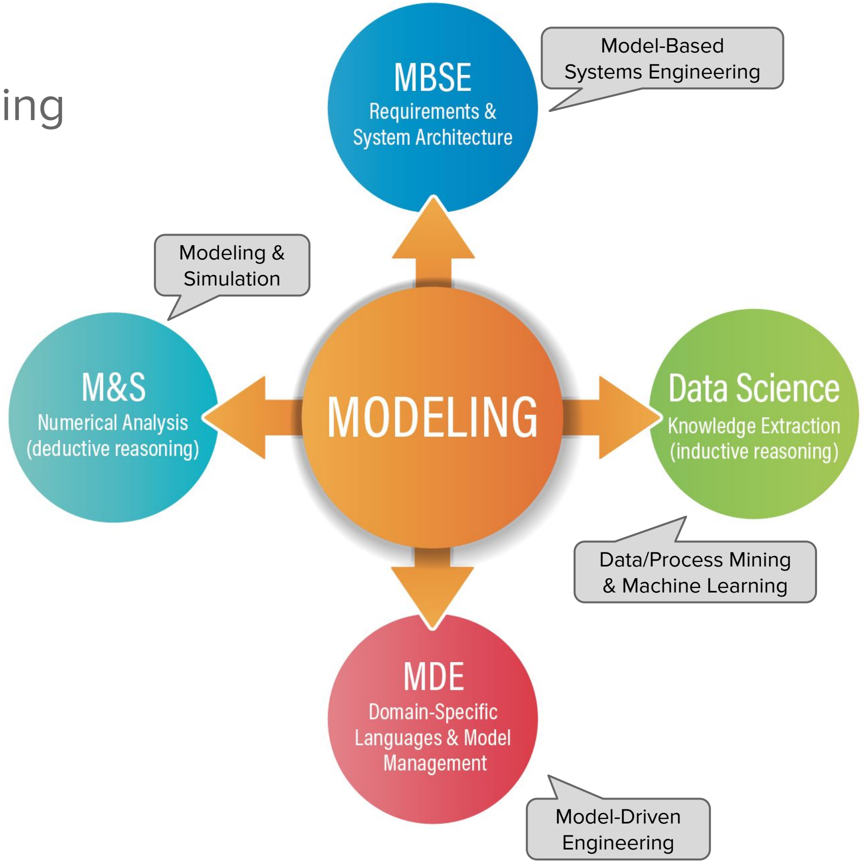
Scaled up to the new EDT conference series => EDT conf 2024

Challenge: Model Hybridization

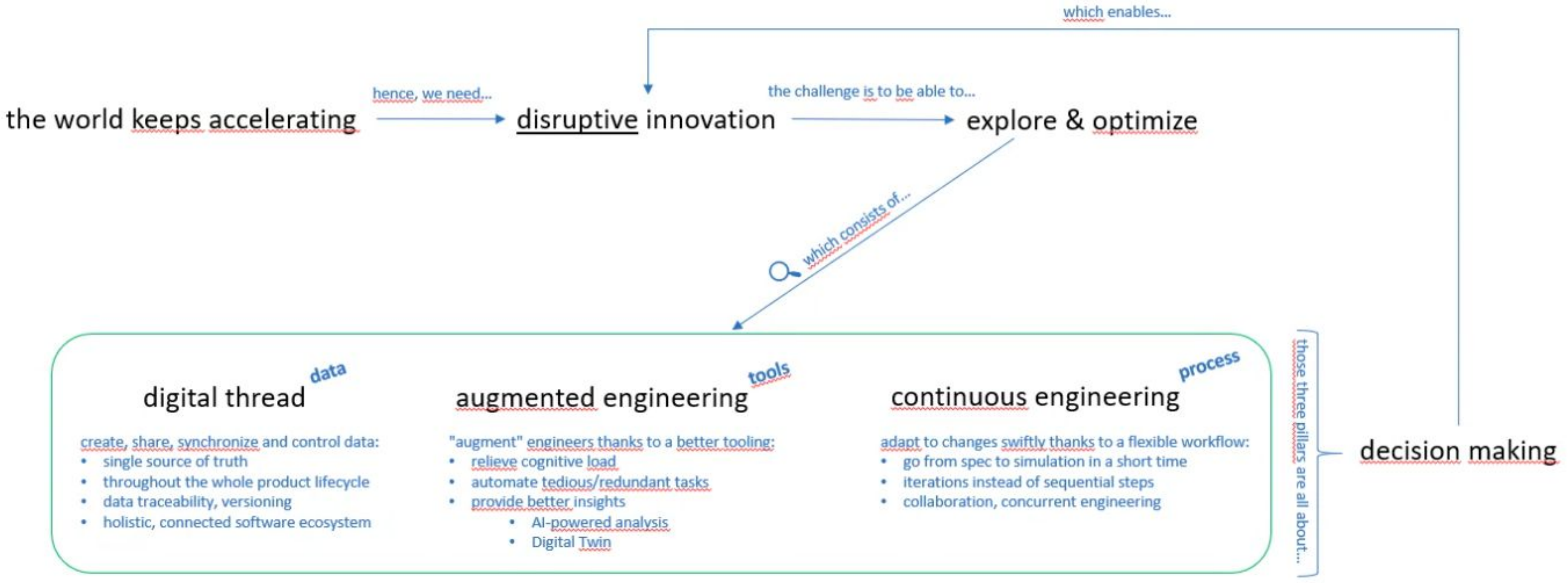
Man-made and inferred abstraction engineering

Towards a unifying theory for inductive and deductive reasoning

- **Hybrid modeling**
  - coordinated use of heterogeneous predictive models
- **Adaptive modeling**
  - model adaptation (inference/refinement/configuration)

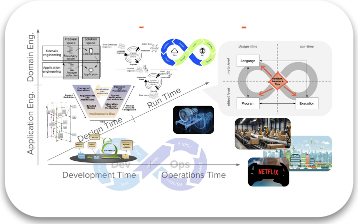


Challenge: Tool Support

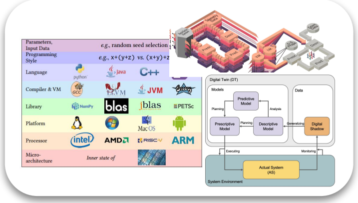


➢ **Towards a continuous (model-based) software engineering!**

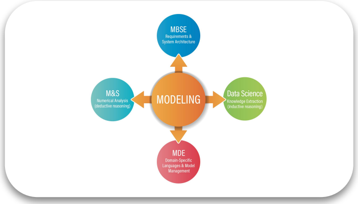
Take Away Messages



- **There is only one time to tame *Software Hyper-Agility***
  - **Innovation = Exploration & Optimization**
    - Breakthrough over incremental innovation
  - **New temporal adaptability**
    - Dynamic environment



- **Towards a *Continuous (model-driven) Software Engineering***
  - Deep Variability
  - Feedback-driven Software Development
  - Digital twins



- **Open challenges**
  - **Foundations**: abstraction engineering (e.g. model hybridization, language engineering) ; DT modularization, interoperability and composition ; uncertainty management…
  - **Technologies**: context-aware dev tool, DT engineering…
  - **Businesses**: IP management, standards, patents…

There Is Only One Time In Soft. & Sys. Engineering!

Towards a Continuous (Model-Based) Software Engineering

Software and systems engineering is a complex endeavor that encompass various socio-technical activities. These activities are traditionally orchestrated over a development life cycle from development time to operation time, and applying engineering processes both at design and run times, and at the application and domain levels. This organization of the activities led to well defined life cycles (V-model, Scrum, DevOps, language-oriented programming, etc.) to cope with the complexity of the engineering of software-intensive systems. This organization also structures the available tools and methods we use, and even the various communities among the software and systems engineering one (i.e., The Conway's law applied to our own discipline!). While such an organization was important at the inception of the discipline (divide and conquer!), I argue during this talk this is now hurting the high degree of adaptability we need in software and systems engineering to face what I call the ***software hyper agility****. In particular, modern systems are evolving at an accelerating pace, operating in increasingly dynamic environments and contending with ever-increasing uncertainty. This requires a* ***continuous (model-based) engineering*** *of such complex cyber-physical, socio-technical, ecosystems. In this context, I will discuss challenges related to variability management and abstraction engineering to better support a feedback-driven software development process, and explore the concepts of engineering forge and digital twins as key enablers.*