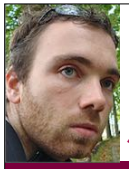
McMaster University
ENGINEERING
Computing & Software

50 shades of Scalability

Sébastien Mosser
May 30th, 2022
CSER Spring Meeting Keynote

McSCert

CC BY NC SA



Sébastien Mosser

***Snowboarding** since 1995. **Composing** things since 2007.*

McSCert

- 22-...: Associate Professor, McMaster Univ.
- 19-21: "*Professeur*", UQAM
- 12-18: "*Maître de Conférences*", Univ. Côte d'Azur
- 11-12: Research Scientist, SINTEF
- 10-11: Postdoc, Inria Lille Nord-Europe
- 07-10: PhD student, Université de Nice

The ultimate question

Does it scale*?

** please stop asking this question during PhD or MSc defenses, it's a terrible one.*

Scalability?

scalable adjective

Save Word

scal·able | \ ˈskā-lə-bəl \

Definition of *scalable*

1 : capable of being scaled

2 : capable of being easily expanded or upgraded on demand
// a *scalable* computer network

Merriam-Webster

scalability

Cambridge Dictionary

noun [U] • BUSINESS • specialized

UK /ˌskeɪ.ləˈbɪl.ə.ti/ US /ˌskeɪ.ləˈbɪl.ə.ţi/

the ability of a business or system to grow larger:

- *There are doubts about the profitability and the scalability of the company's web business.*

So Scalability derives from "Growth"

growth noun

Save Word

Merriam-Webster

\ ˈgrōth \

Definition of *growth*

1 a (1) : a stage in the process of growing : SIZE
(2) : full growth
b : the process of growing
c : progressive development : EVOLUTION
d : INCREASE, EXPANSION
// the *growth* of the oil industry

Evolution of Size

What does "size" mean in Soft. Eng?

Kollektivet: Music Video - ÆØÅ (Size Matters)

In this sketch, Norway competes with the USA, and eventually concludes that they have the biggest alphabet (with Æ, Ø, Å)

This talk is not all about reading the dictionnary!

1 Composition in a nutshell

2 Composition at scale

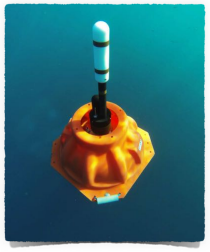
3 Scaling Composition

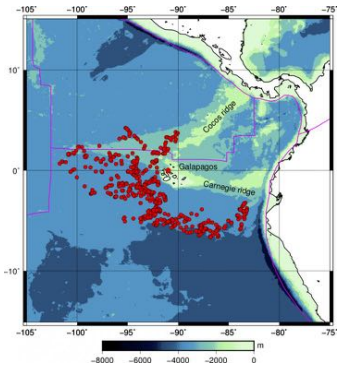
Composition in a nutshell

Like Mr Jourdain, you're composing without knowing it

1

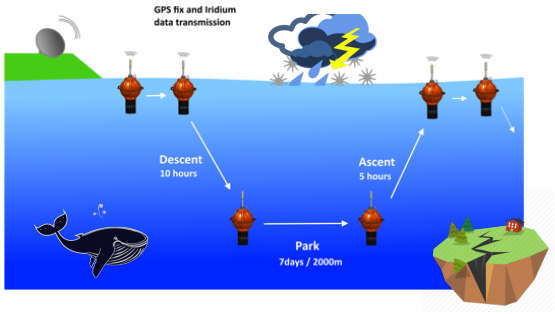
Example of Industrial Collaboration: MERMAID

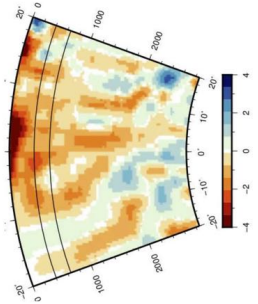




- Project **started in 2002** by Guust Nolet
- **Acoustic passive monitoring** of oceans
- Now **3rd generation** buoys deployed

Running Software, 20,000 leagues under the seas





Ok, but where is the composition? Where is the "scale"?

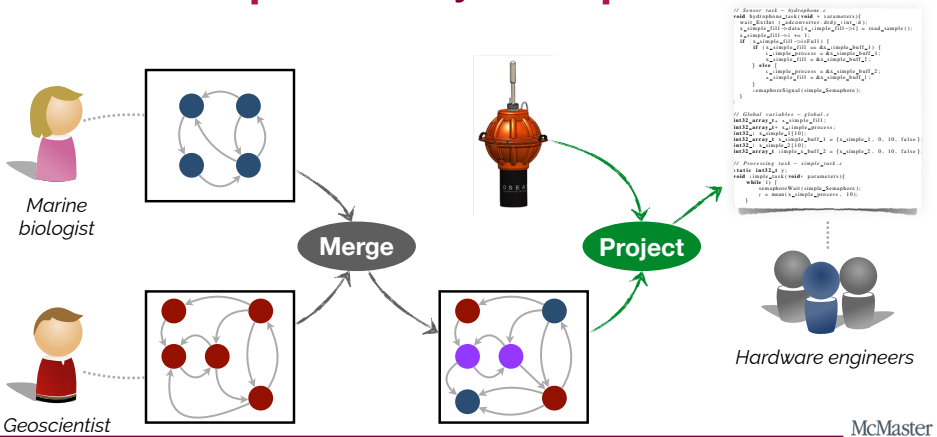
Running Software, 20,000 leagues under the seas

- **MERMAIDs** are ***expensive*** and often ***idle***
  - *Oceans are full of challenges* *(80% unexplored)*
    - *Monitoring whales, plastic pollution, salinity, …*
  - ***Compose*** *"data collection campaigns"*
- **A MERMAID is not a smartphone**
  - *Legacy ad hoc code (no operating system)*
  - *Hostile environment (salt, pressure), energy, …*

"Can you build an AppStore for our MERMAIDs?"

- G. Nolet, 2016

Software Composition by example



Software Composition matters! (2010-…)

- Like *Mr Jourdain*, **you're composing without knowing it**:
  - ***Modularizing** code (e.g., packages, functions)*
  - ***Configuring** the Linux kernel*
  - ***Weaving** persistance into a Spring application*
  - ***Pulling** code from a Git repository*
  - ***Deploying** & **invoking** micro-services*
  - …

But the "size" depends of the application domain!

Intermediate conclusions #1

Divide to conquer
Compose to vanquish

2

Composition at Scale

Ok, composition is banal. But "does it scale"?

Requiem for Aspect-Oriented Programming

> My reservation of Aspect-Oriented Programming is that we already have fairly good techniques for separating aspects of programs, and we don't use them. I don't think the real problem will be solved by making better techniques for separating aspects. We don't know what should be the aspects that need separating, and we don't know when it is worth separating them and when it is not.

*[Fowler (quoting Johnson), **2003**, IEEE Software]*

- AOP was supposed to "solve" the separation of concerns issues
- Scaling the integration of concerns
  - *Automating the weaving process*
  - *AspectJ is an engineering jewel*
- **This was not what needed to scale!**
  - **Conflict** & **Design** were the real issues!

Let's look at the Linux kernel

- A modern version of the kernel contains ~**1,500 features**
  - Combinatorial worst case: $2^{1,500} (\approx 10^{452})$
- There are 10^{80} atoms in the visible universe

```
Last login: Thu May 26 19:22:19 on ttys000
mosser@loki ~ % python
Python 3.10.2 (main, Feb 22 2022, 10:51:53) [Clang 13.0.0
(clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> "{:e}".format(2**1500)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OverflowError: int too large to convert to float
>>>
```

> *If you're an astrophysicist, you're dealing with objects that are way simpler than the ones software engineers use on a daily basis* — *J.-M. Jézéquel*

- *Example of linux engineering at scale: Coccinelle (rewriting rules)*

Program rewriting in a nutshell

- Code rewriting **in a nutshell**:
  - $Rule : Program \rightarrow Program$
  - With ρ a rule and P a program, $P' = \rho(P)$ the rewritten one
- **Rules are composable** by definition:
  - $\rho_{12}(P) = (\rho_1 \bullet \rho_2)(P) = \rho_1(\rho_2(P))$
- **Size matters**:
  - Linux ≈ 20 millions of lines of codes
  - 7k commits /month

Exemple of rule: kmalloc to kzalloc

Coccinelle used separation of concerns to separate each rewriting rule

```
@@
type T;
expression x, E, E1,E2;
@@
- x = kmalloc(E1,E2);
+ x = kzalloc(E1,E2);
... when != \( x[...]=E; \| x=E; \)
- memset((T) x, 0, E1);
```

R_i

```
...
x = kmalloc(sizeof(*a), 0);
memset(a, 0, sizeof(*a))
...
```

P

$R_i(P)$

```
...
x = kzalloc(sizeof(*a), 0);
...
```

P'

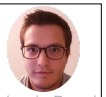
Scaling issue: Rule composition

- Running coccinelle on 20M LoCs is surprisingly fast
  - `make coccicheck` takes 190 minutes in average (*35 **sequential** rewritings*)
  - We're talking about graph pattern matching at scale!

$$59! \approx 10^{80}$$

- Software engineering is all about trade-offs:
  - Coccinelle does not look for fixed-points invariants, or **conflicts**.
- **No guarantee that rules commute**: $R_1(R_2(P)) =^{?} R_2(R_1(P))$

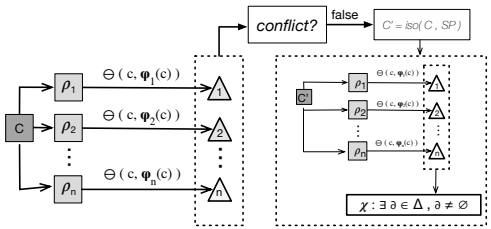
Composition at scale: a new operator!



Benjamin Benni
PhD (2019)
MSc (2016)

- Two objectives
  - Stay in the 190 minutes execution window
  - Identify composition conflict

Validated on 19 versions of Linux
Quicker than sequential (in average)
Identified 2 conflicting rules



[ICSR, 2018]

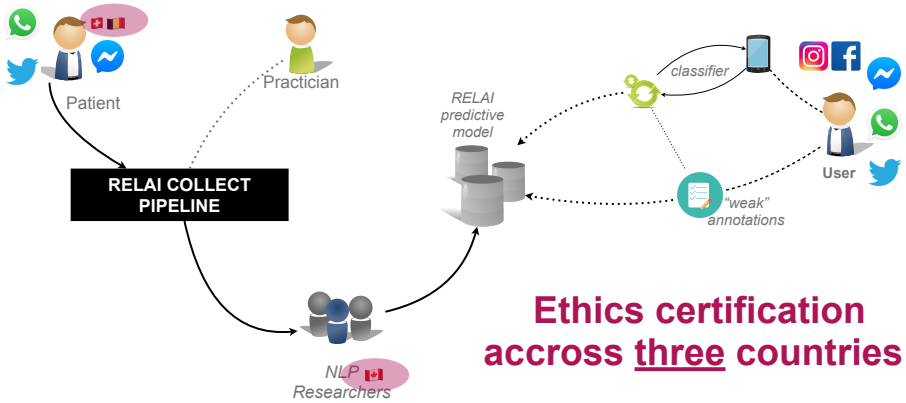
Another example: the RELAI Project

- **R**espectful and **E**xp**L**ainable **AI** to support struggling people and mental health practitionners
  - Mental illness is a leading cause of disabilities in Canada
  - **4.9M of Canadians** (15yo+) needed mental health care (2015)
  - 33% of Ontario's students (12-17) reported a need to talk
  - And COVID-19 did not help…
- **Objectives of the project**:
  - ***Collect an annotated corpus*** of textual conversations from patients received by ER (Switwerland, Belgium)
  - Train an NLP model to ***support suicidal risk detection***

New Frontiers in Research Fund
Fonds Nouvelles frontières en recherche

Dr M.-J. Meurs, PI

Where is the scale issue here?

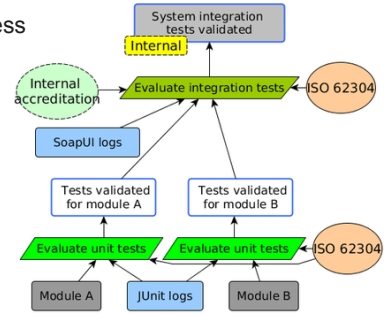


Ethics certification accross <u>three</u> countries!

Convincing ethics committees!

C. Pulgar
MASc (21-…)

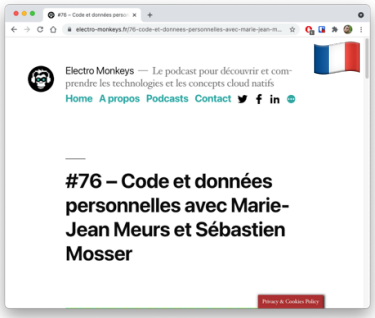
- Need to obtain **three different ethics committees certifications**
  - (**Super**) Long & (**Super**) tedious process (*15 days meeting in , early 2020*).
- Looks like a **classical "certification"** process
  - **Justification diagrams**!
- **Insights (ongoing)**:
  - Strengthen the DMP
  - Give confidence to defend it
  - Even if not executable!



Supporting developers!

```
public void method store(Set<Entry> dataset) {
  Database db = RemoteStorage("...");
  for(Entry e: dataset) {
    db.save(e)
  }
}
```

Error: Entry 'e' is not anonymized



*Enhance tooltips and error handlers in IDEs with **composable requirements** that are driven by external & evolving concerns*

Intermediate conclusions #2

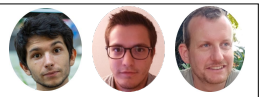
Scaling is not always about BIG numbers

Scaling Composition

Composition at scale is not enough!

3

Legacy needs to scale



[JOT, 2020]

- **Performance** related issues:

  In Familiar, $\underbrace{\mu(A,B)}_{t_{AB}} \equiv \underbrace{\mu(B,A)}_{t_{BA}}$, but if $|A| > |B|$, then $t_{AB} \gg t_{BA}$

- Operators can be **legacy code**: Familiar is 241k LoCs of Java.
  - *Git merge is another "black box" legacy operator (1 function, 4K LoCs of C).*
- **Algebraic properties** of composition operators helps to scale
  - Commutativity allows us to swap arguments and **save hours of computation** (🐶).
  - Idempotency avoid useless composition, …

Scaling composition: “History”

```
/* Property to be associated to an operator */
property associative(op: Operator, =: Relation, {a,b,c}):
    op(op(a,b), c) = op(a, op(b,c))

/* Property to be associated to a relation */
property transitive(=: Relation, {a,b,c}):
    a = b && b = c && a = c
```

Properties as assertions

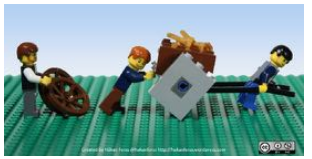
```
import stdlib.ace
model fm

trait NamedElement         { required name: String     }
concept FM <: NamedElement { required contents: String }

relation equivalence: FM x FM
operator union: FM x FM -> FM

declare FM::union as associative with equivalence
declare FM::union as commutative with equivalence
```

Operator modelling

Property-based testing approach

Validated on Familiar and Git

Stop reinventing the wheel



- Remember the **Linux rewriting** study?
  - Coccinelle is not the only rewriting tool out there!
- Examples of **other modern rewriting** aproaches:
  - Fixing energy-efficiency anti-patterns in Android apps [ICSR, 2018]
  - Fixing good pratices violations in Docker [SAC, 2018]
- Application domains differs
  - But in essence, **composition verification** and **order-indepence** do not change

Scaling Composition: Reusability

2 x [SAC, 2018]

“The devil is (still) in the details”

- Sort the **wheat** from the **chaff**:
  - What is domain-specific? What is a composition-specific?
- **Current work**:
  - Reverse-engineering micro-services architectures & the LLVM compiler

[ICSOC, 2020], [SPLC, 2020]

$$\text{Let } \rho = (\varphi, \chi) \in (\Phi \times X) = P, \quad (\varphi : AST \to AST) \in \Phi$$
$$\chi : AST \times AST \to \mathbb{B} \in X, \qquad \forall p \in AST,\ \chi(p, \varphi(p))$$

$$iso : AST \times P^n \to AST$$
$$p, \{\rho_1, \ldots, \rho_n\} \mapsto p_{iso} = p \oplus (\ominus_{i=1}^{n} (\varphi_i(p) \ominus p)), \quad \wedge_{i=1}^{n} \chi_i(p, p_{iso})$$

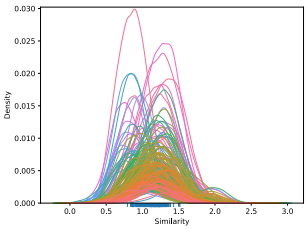
$$apply : AST \times P_<^n \to AST$$
$$p, [\rho_1, \ldots, \rho_n] \mapsto \text{Let } p_{2..n} = (\bullet_{i=2}^{n} \varphi_i)(p),\ p' = \varphi_1(p_{2..n}), \quad \chi_1(p_{2..n}, p')$$

This is not about cheating to increase your h-index, or a hammer-nail situation. This is about finding the right abstraction, and demonstrating its rightness

Scale as first-class citizen for operators

[JOT, 2022]

- Context: **identifying similiarites** inside agile product backlogs
  - *“As a <…>, I want to <…> so that <…>”*
  - Reference dataset (Dalpiano, 2018): 1,681 stories for 22 products
- Scale issues: combinatorial similarities (+ NLP analysis)
  - *1,671* stories, *2,973* elements, *4M+* measurements
  - Computation: 122 CPU hours, 40 CPUs



Need for an operator able to support that scale

Scaling Composition: Tech. Blinders

- **Lots of reserarch** is done in the "**Java**" ecosystem
- Needed technology **outside of our comfort zone**
  - *Neo4J did not scale, switch to Python / NetworkX*
  - *NLP Dev. expertise is in Python*

Obstacle (Sedano et al. 2019)	Validation Scenario
O_1: Preconceiving Problems	Σ_4, Σ_5
O_2: Preconceiving Solutions	Σ_3
O_3: Pressure to Converge	Σ_1
O_4: Ambiguity	Σ_4, Σ_5
O_5: Time Pressure	Σ_2, Σ_3
O_6: Blocking Access to Users	*N/A*

- Actually, **legacy operators that scale are not written in Java** (neither in Python)
  - *Have you tried reverse engineering C/C++ code?* 🔥
  - *The SE community is missing lots of opportunities here.*

Intermediate conclusions #3

Composition is an application domain by itself

Conclusions

The real ones!

4

Thanks for your attention!

- **Software Composition** is everywhewre
  - *It solves "programming" scalability (divide & conquer)*
  - *It requires **expertise*** to scale.*
- **What can we do as Soft. Eng. researchers?**
  - **Reify** composition operators when we encouter them
    - *(Stop putting half-baked and non reusable code on Github)*
  - **Reify** what "scale" means in different contexts.

**Maybe I should have sticked to snowboarding*

This work is a team effort


Selena


Jinx


Dumpling


Alexandra Lapointe
PhD, 01/2021-…


Corinne Pulgar
MASc, 09/2021-…


Sébastien Bonnieux
PhD (2020)


Sami Lazreg
PhD (2020)


And all co-authors & colleagues…


Benjamin Benni
PhD (2019), MSc (2016)


Günter Jungbluth
MSc (2018)


Cyril Cecchinel
PhD (2017), MSc (2014)


Ivan Logre
PhD (2017), MSc (2013)


Alexandre Feugas
PhD (2014)


Eirik Brandtzæg
MSc (2011)

Blatant advertisement

- **Research lab**: McMaster Centre for Software Certification
  - *And more generally the Computing and Software (CAS) department*
- **Current openings**: *Faculty positions, PDF, PhD, MASc, Summer internships …*
  - **Long-lasting tradition of industrial collaboration** (*e.g.*, Cubic, Stellantis, TELUS)
- Join us! For example: "***Safety Over-the-Air software Updates***"
  - Topics: Change impact analysis, Safety insurance & tooling, automation, …
  - Other opportunities: *https://www.mcscert.ca/recruiting/*
  - Contact: Dr **Vera Pantelic** (pantelv@mcmaster.ca)

McSCert

McMaster University

ENGINEERING
Computing & Software

Dr. Sébastien Mosser
Associate Professor

mossers@mcmaster.ca
https://mosser.github.io