Language-Level Support for Multiple Versions for Software Evolution

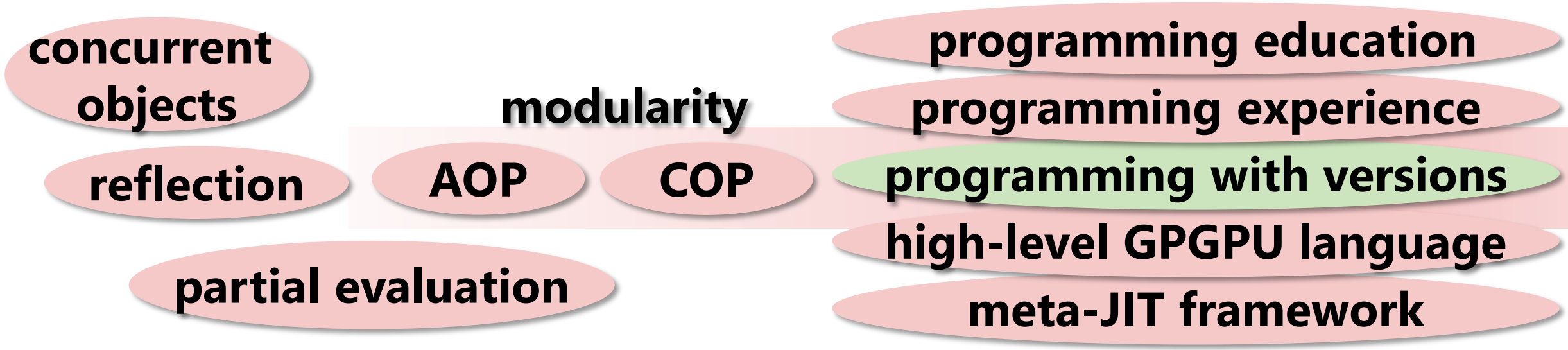
Tomoyuki Aotani (Sanyo-Onoda City University)
Satsuki Kasuya
Lubis Luthfan Anshar
Hidehiko Masuhara
Yudai Tanabe

Institute of SCIENCE TOKYO

About me: Hidehiko Masuhara

Institute of SCIENCE TOKYO

- at Science Tokyo (née Tokyo Tech)
  and Hasso Plattner Institute, U. Potsdam (Jun.-Nov. 2025)
- working on programming language design & impl.
  and software development environments



Talk Overview: Versions!

Institute of SCIENCE TOKYO

- **Our "Programming with Versions"**
  - **Versions are great**
  - **Versions are not so great**
  - **Our work**
- **Software evolution issues and versions** (food for thought)
  - **architecture, management and compatibility**
  - **continuous, gradual, incremental software evolution**
  - **library migration**

Institute of SCIENCE TOKYO

Programming with Versions

Versioned Packages (yeah, we all know)

Institute of SCIENCE TOKYO

- **e.g., NumPy 2.3.4, async 3.2.6, …**
- **a unit of deployment/development/reasoning**

- **outside of PL: managed by a package manager**

  (eg. npm, gradle, pyPI, RubyGems, Cargo,…)

  - **managed with version numbers**
- **inside of PL: associated to modules**

  (classes, functions, types, variables, etc)

Versioned Packages Help Software Evolution

Institute of SCIENCE TOKYO

by

- enabling independent development
  - provided & required
- allowing existence of multiple versions
- reasoning about compatibility
  - “require NumPy 2.2 or later, but not 3.x”
  - package managers can resolve combinations of versions

Versioned Packages **Can't** Help Software Evolution

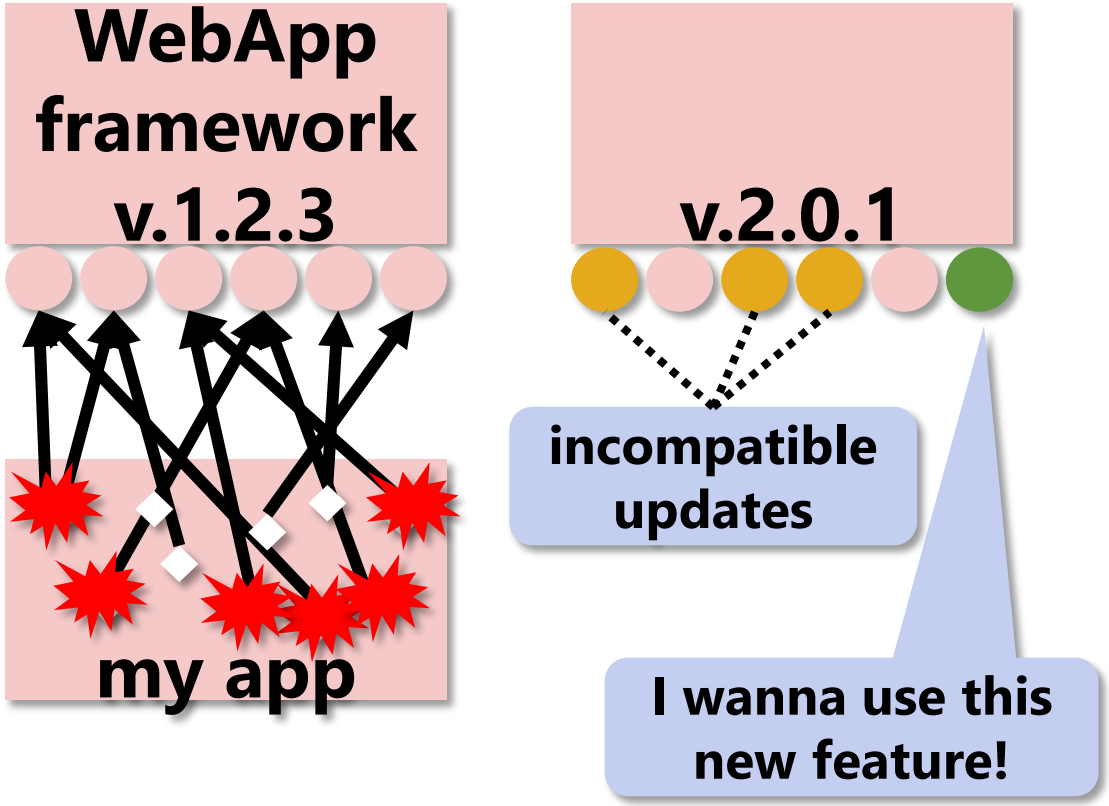
Institute of SCIENCE TOKYO

- **Painful package updates**
  - **due to breaking changes**
- **Conflicting version requirements**
  - **sometimes happens indirectly**

(Versioned packages are merely an eruption point of many software evolution problems.)

Package Updating can Require Mass Changes

- Even for receiving
  a small improvement
  (or a security fix)
- Due to incompatible updates

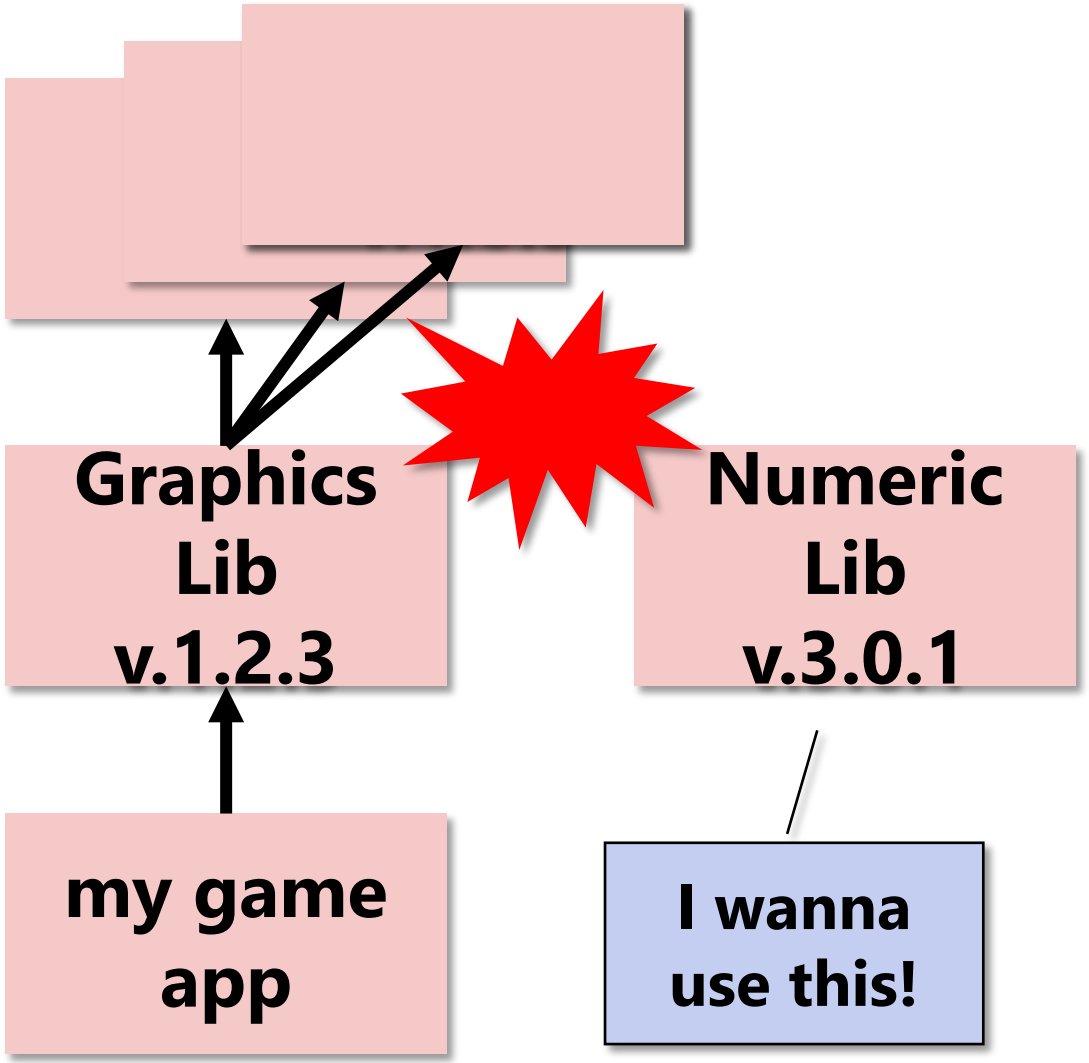


Consequences:

- unwanted extra work
- procrastination

Indirect Version Conflicts a.k.a. “Dependency Hell”

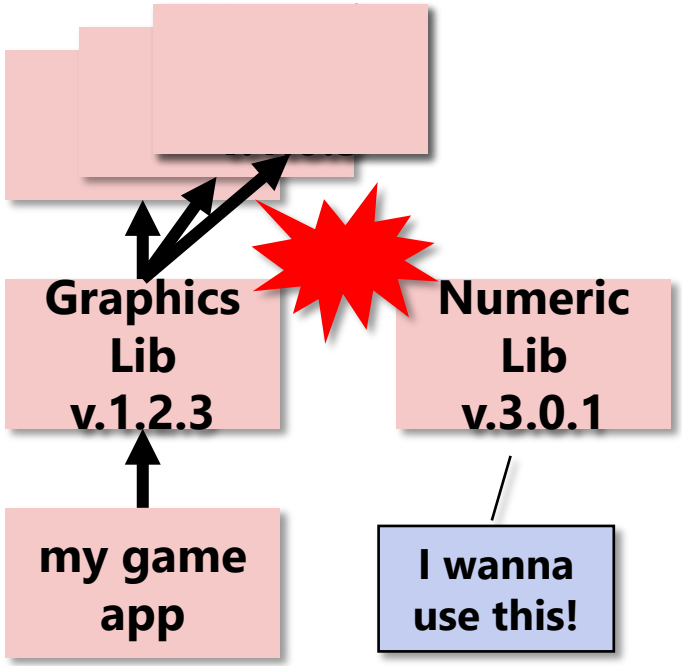
- Conflicting version requirement
  - Package managers can only report but not resolve
- Difficult to predict and resolve due to indirection



Existing Research: What can We Do?

Institute of SCIENCE TOKYO

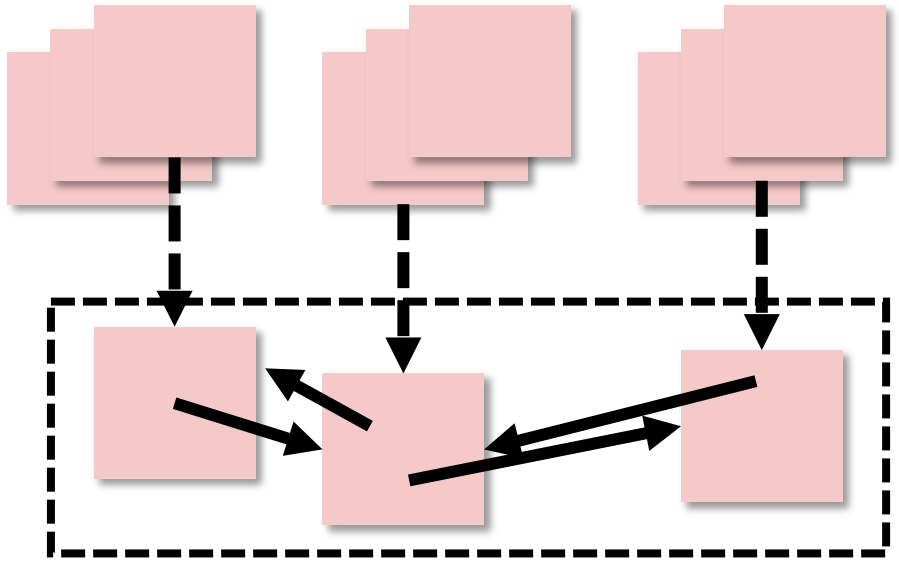
- Make “compatibility” more precise
  - based on program analysis, etc.
  - to alleviate version conflicts
  - and link to version numbers (SemVer)
- Adapt new version to old (or vice versa)
- Migrate old programs to a new library
  - with automated program refactoring/rewriting



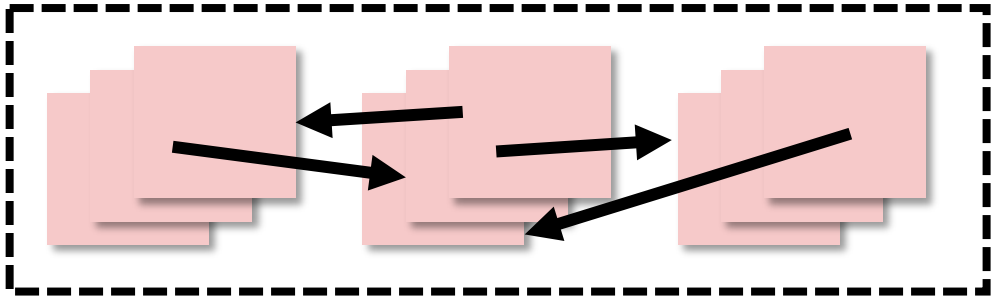
are they all superficial solutions!?

Source of the Problem: One-Version-at-a-Time Principle

- in most programming languages
- hence package managers select versions beforehand
- hence mass changes, conflicts, ...

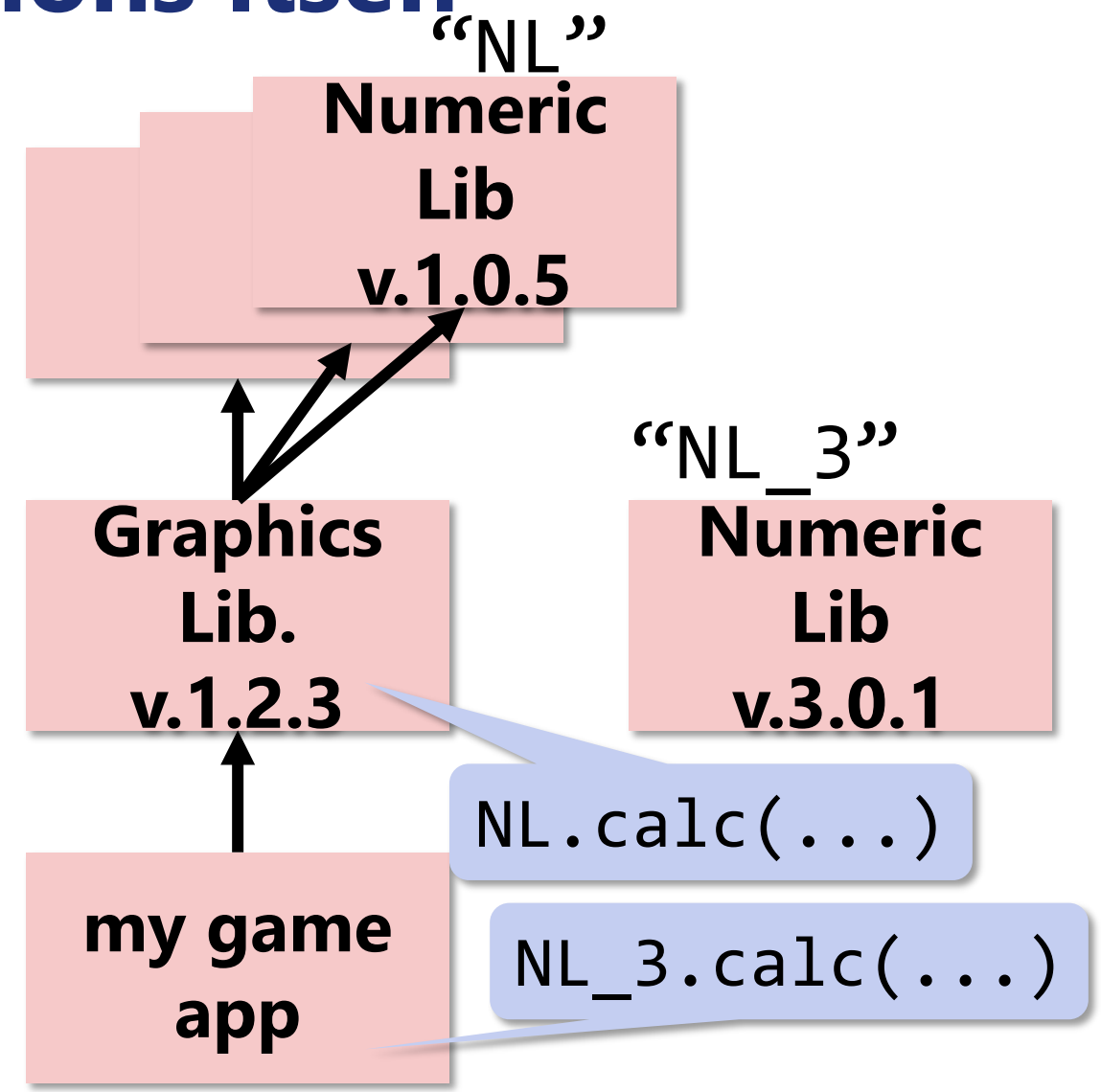


Imagine all the versions living together in peace!



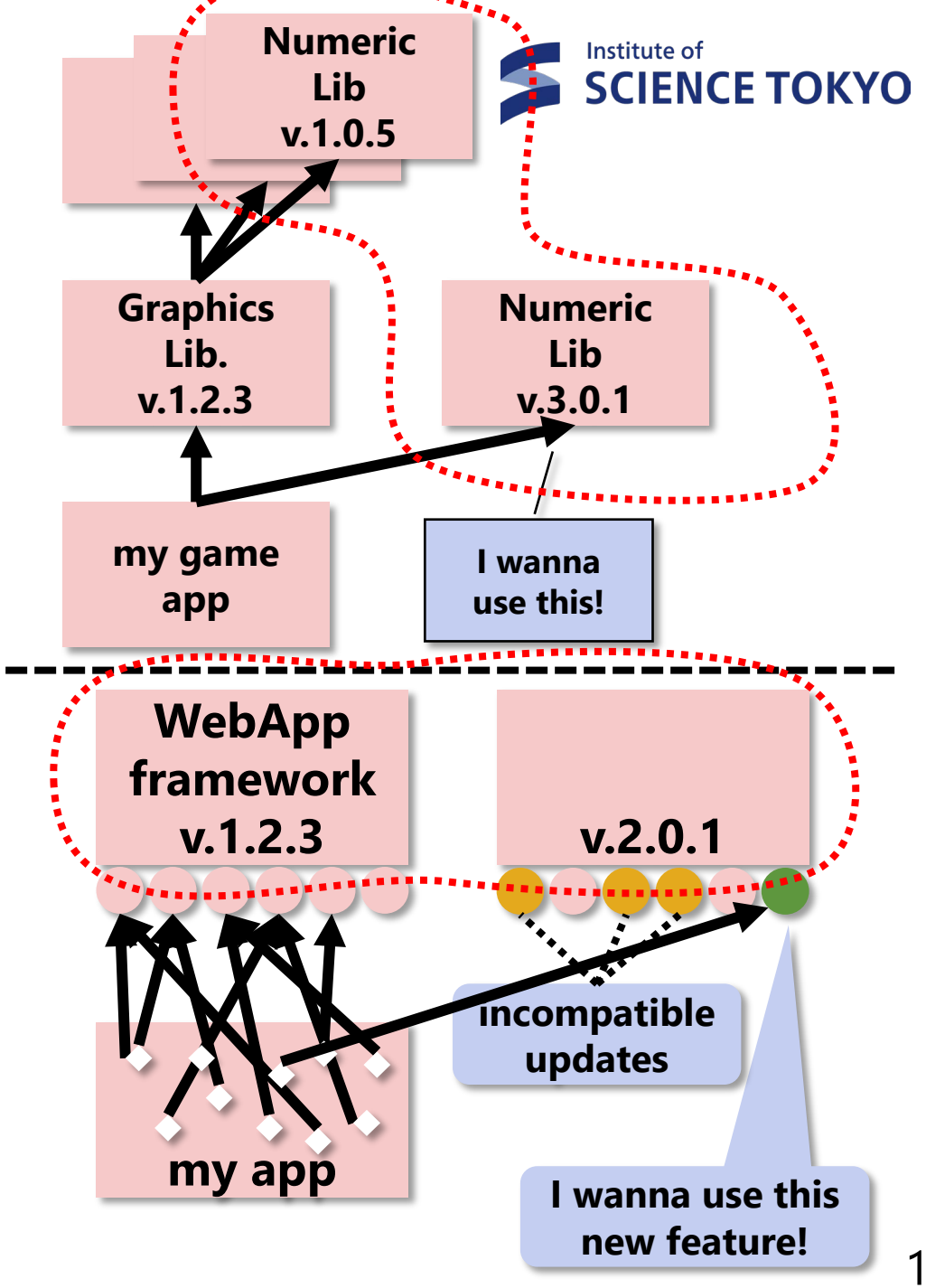
Supporting Multiple Versions Itself is not Difficult

- by just renaming modules (so called “name mangling”)
- through package managers i.e., without extending programming languages
  - at package level
  - eg. cargo/Rust, OSGi/Java



Programming with Versions

- our proposed paradigm
  ‹Programming›'21
- to support multiple versions of a module in a program
  - at different granuality
- with challenges (or non-)
  - co-existence of two versions
  - prevention of inconsistent usage
  - software development process



Our PwV Languages Designs

Institute of SCIENCE TOKYO

- Ver.1: **VL, a functional language** ‹P›'21,APLAS'23
  - function-level: `f(x)` where `f` has multiple versions
- Ver.2: **BatakJava, class-based language** SLE'22
  - function&data, i.e., a class has multiple impls.
- Ver.3: **Vython, dynamically-typed language** JIP'25
  - dynamic conflict detection (vs. type-based)
- Ver.4: **another class-based language** *ongoing*
  - dynamic, i.e., an object can be of multiple versions

Data Consistency in VL ‹P›'21,APLAS'23

Institute of SCIENCE TOKYO

- **support: allow a function to have multiple versions**
  - **version selection by data consistency (or annotation)**

```
image(w,h) = ...
save(img) = ...
```

ver.1.1

```
image(w,h) = ...
save(img) = ...
save_png(img) = ...
save_jpg(img) = ...
```

ver.2.0

two versions

Data created in a version must be fed to the same version

```
icon = image(30,30)
photo = image(1000,750)
save(icon)
save_jpg(photo)
...
save(photo)
```

switch to ver.2.0

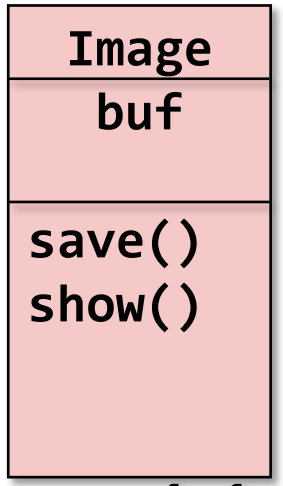
detect conflict

  - **type-based data analysis (similar to information flow analysis)**

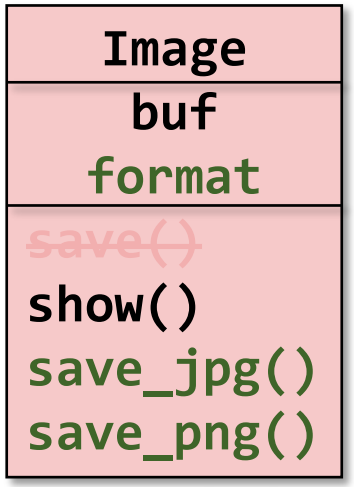
Data&Behavior Consistency in BatakJava SLE'22

Institute of SCIENCE TOKYO

- one class, multiple versions
- version selection
  - by usage
  - at instantiation
- version polymorphism
- inheritance



ver.1.1



ver.2.0

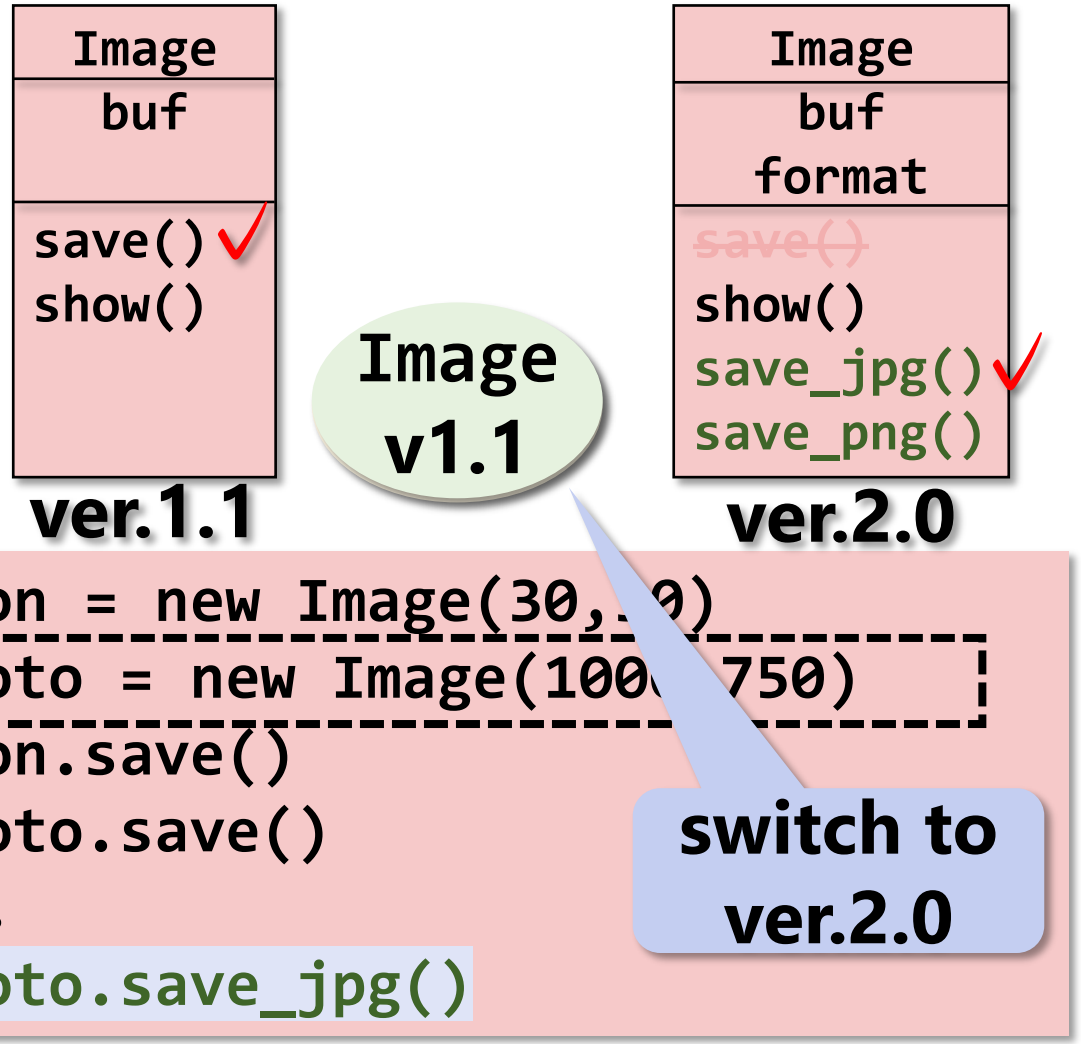
```
icon = new Image(30,30)
photo = new Image(1000,750)
icon.save()
photo.save_jpg()
...
...im=icon...im=photo...
im.show()
```

switch to ver.2.0

Dynamic Version Switching (ongoing)

Institute of SCIENCE TOKYO

- one object, multiple versions
  - switch representation versions on demand
  - state transformation is not trivial



cf. multi-version schema migration in DB

But Do They Really Work?

Institute of SCIENCE TOKYO

- Performance overhead?
  - Almost zero to 3x.
- Doesn't PwV delay critical updates?
  - Yes and no.
- What about different yet compatible versions?
- Can multiple versions be used w/o conflicts?
- How library developers update their code?
  - Need more assistance: compatibility checking, caller-side migration support

development process

Institute of SCIENCE TOKYO

Let's Talk About Versions and Software Evolution!

Software Evolution Issues from the Versions' Perspectives

Institute of SCIENCE TOKYO

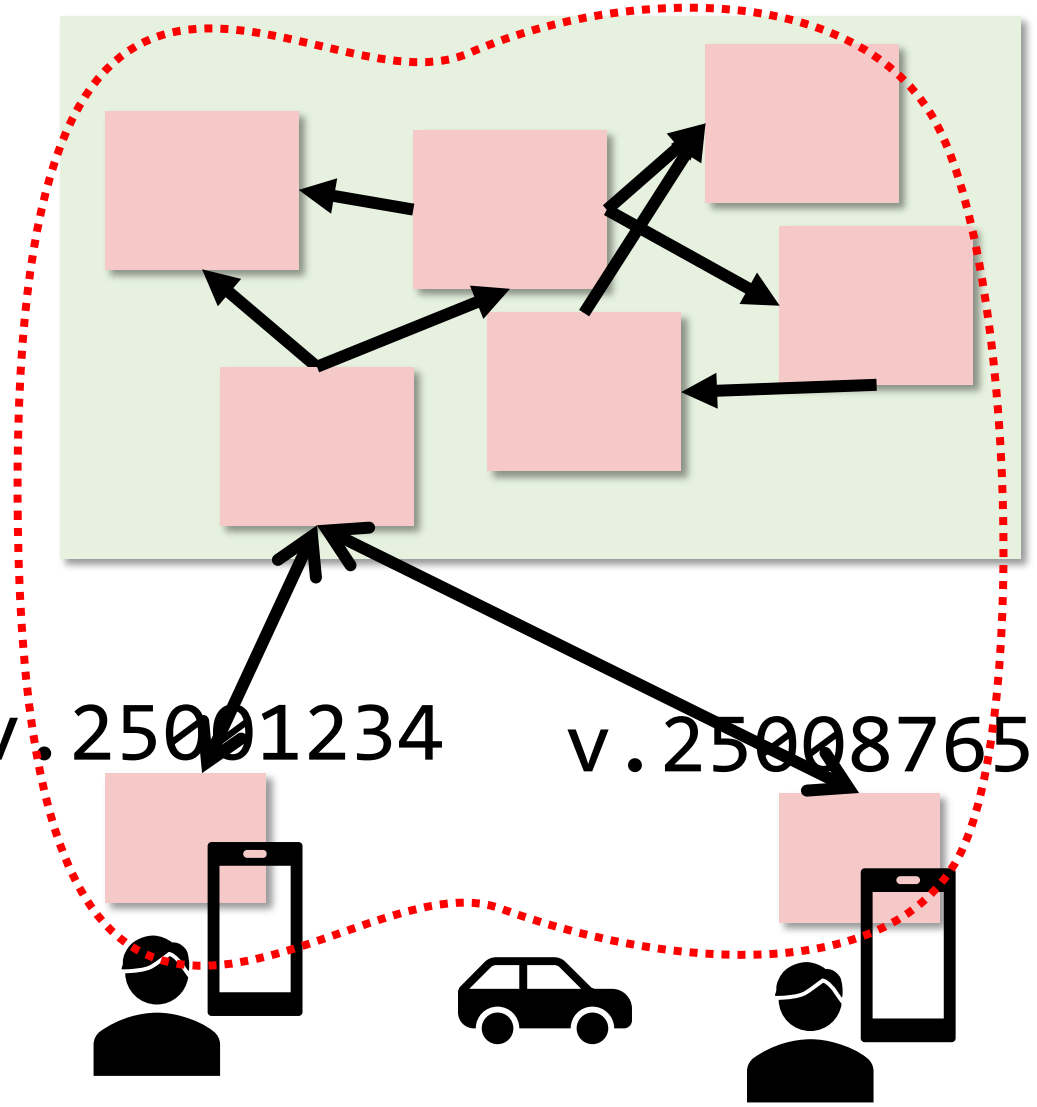
- **Multi-version software in the wild**
- **Design principle of package managers**
- **Versions and compatibility**
- **Continuous/gradual/incremental software updating**
- **Library migrations**
- **...and more?**

  visualization, AI generated code, mining, CI tools, testing, modeling, ...

Multi-Version Software in the Wild

- There are many (I believe)
- Unknown architecture (to me)
  - microservices?
- Best practices? / Issues?

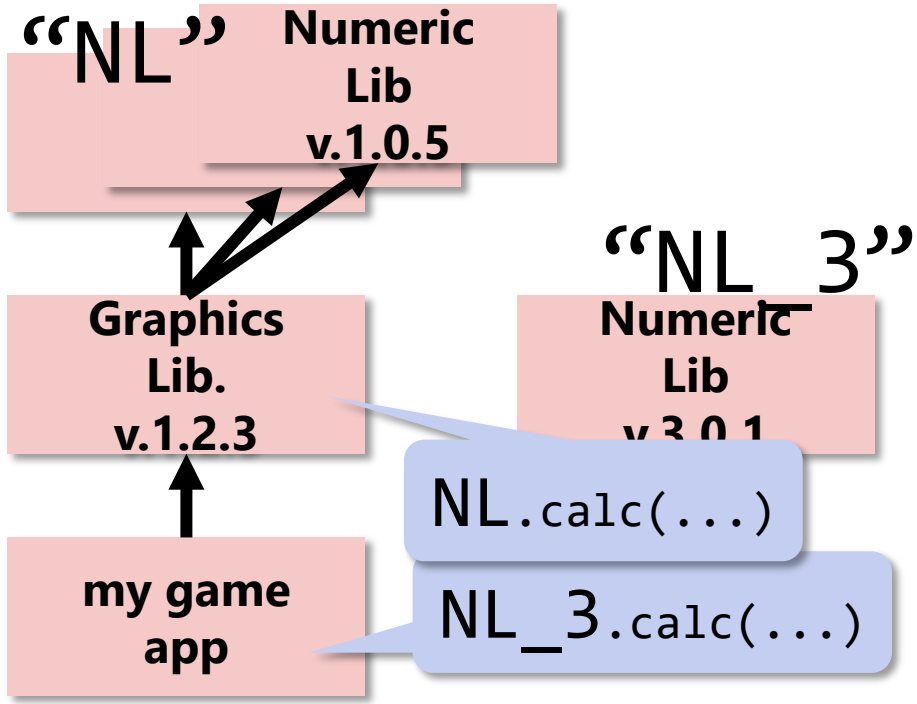
- (Can PwV help?)



Design Principles of Package Managers

Institute of SCIENCE TOKYO

- Is name mangling good?
  - rename packages when they conflict
  - some package managers employ this strategy (e.g., cargo, OSGi)
  - can have side-effects to common types
  - leading errors after version resolution
  - How serious is this problem in practice?
- No common ground for package manager designs!?



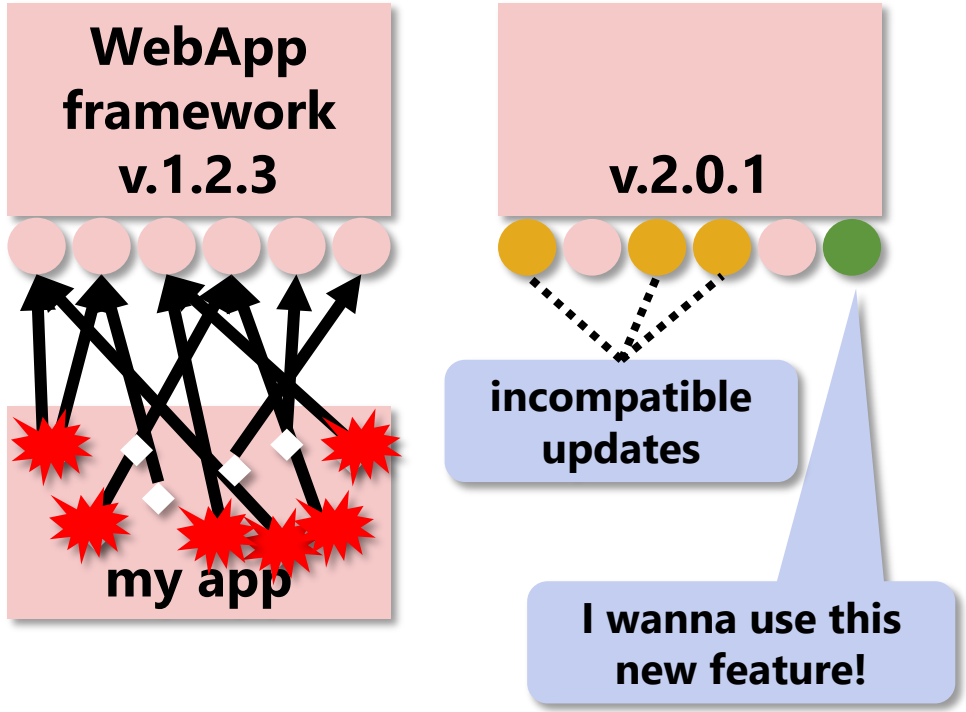
Versions and Compatibility

Institute of SCIENCE TOKYO

- Different versions can be incompatible (but not always)
  - Semantic Versioning https://semver.org/
    - use version numbers to distinguish compatibility
    - many violations in practice
  - package level compatibility
- Need to reconsider versions with PwV
  - to what unit versions should be assigned?
  - what compatibility means there?

Continuous/Gradual/Incremental Updating

- We eventually have to update all call sites
  - but PwV allows to do it gradually
  - can test intermediate states

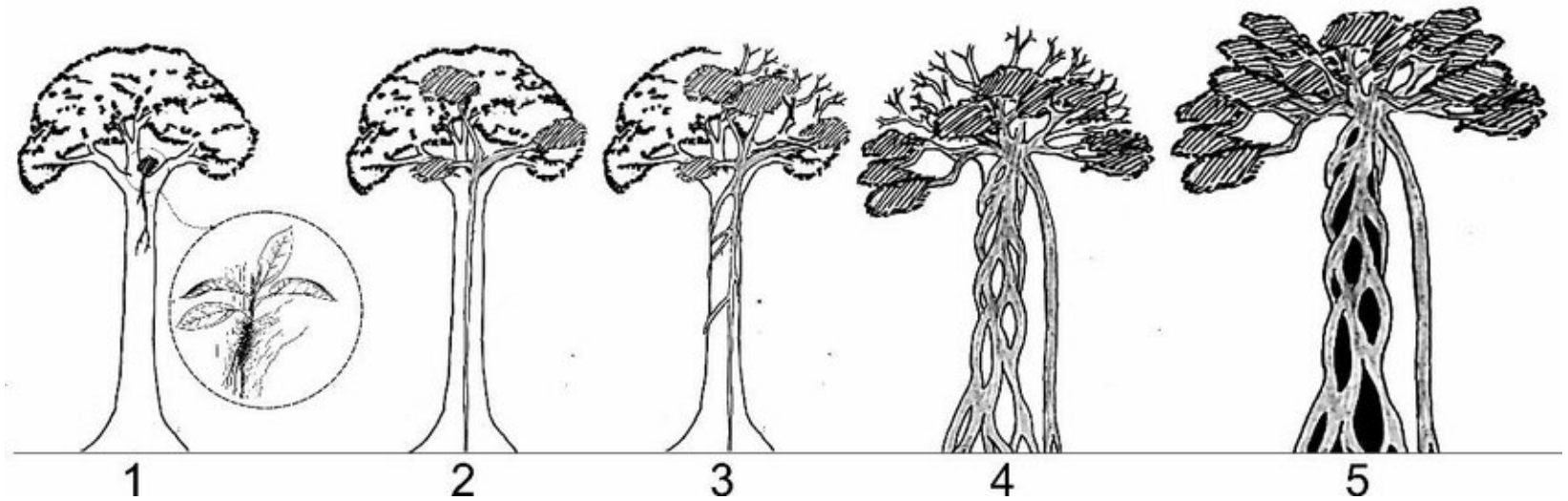


- Challenges: development process
  - How can we test?
  - In what order should we update?

Strangler Fig for Modernization [Fowler]

Institute of SCIENCE TOKYO

...do a gradual process of modernization. Like the fig, it begins with small additions, often new features, that are built on top of, yet separate to the legacy code base. ...



Kurniawan et al. DOI: 10.29244/medkon.27.3.83-90

...there's considerable work in figuring out how to break it down into manageable pieces. ...

https://martinfowler.com/bliki/StranglerFigApplication.html

Can PwV help?

Library Migration

- **Updating from one library to another**
  - **e.g., from TensorFlow to PyTorch**
  - **compatible to some degree**
  - **similar to version updates**

Common ground for
version update & library migration?

Final Words: Versions are Interesting

Institute of SCIENCE TOKYO

- **“Programming with Versions”**
  - **allowing multiple versions in a language**
  - **just one possible language design**
- **Versions and Software Evolution**
  - **many interesting challenges (even w/o PwV)**
  - **your thoughts?**
  - **Shonan Meeting, Dagsthul Seminar, workshops?**