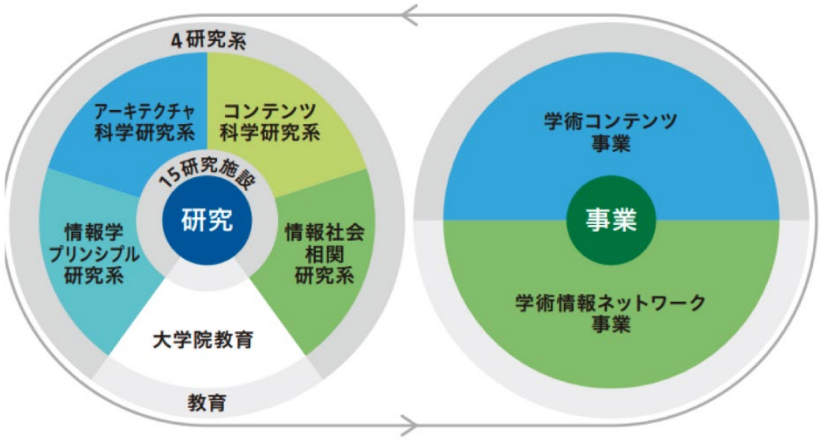
10年後、同じデータを出せますか？

データ駆動科学の再現性・再利用性を考える

藤原 一毅
国立情報学研究所 アーキテクチャ科学研究系 准教授

自己紹介

- 藤原一毅　*FUJIWARA, Ikki*
  - 国立情報学研究所 准教授
    アーキテクチャ科学研究系／
    クラウド基盤研究開発センター
- 仕事
  - NII研究データ基盤 (NII RDC) の開発・運用
- 趣味
  - キーボードの文字配列の最適化[2]
  - オンロード車で林道を走る



[1]

▲NIIは研究と事業とを両輪とする「大学共同利用機関」

[1] https://www.nii.ac.jp/today/101/1.html
[2] https://mobitan.hateblo.jp/

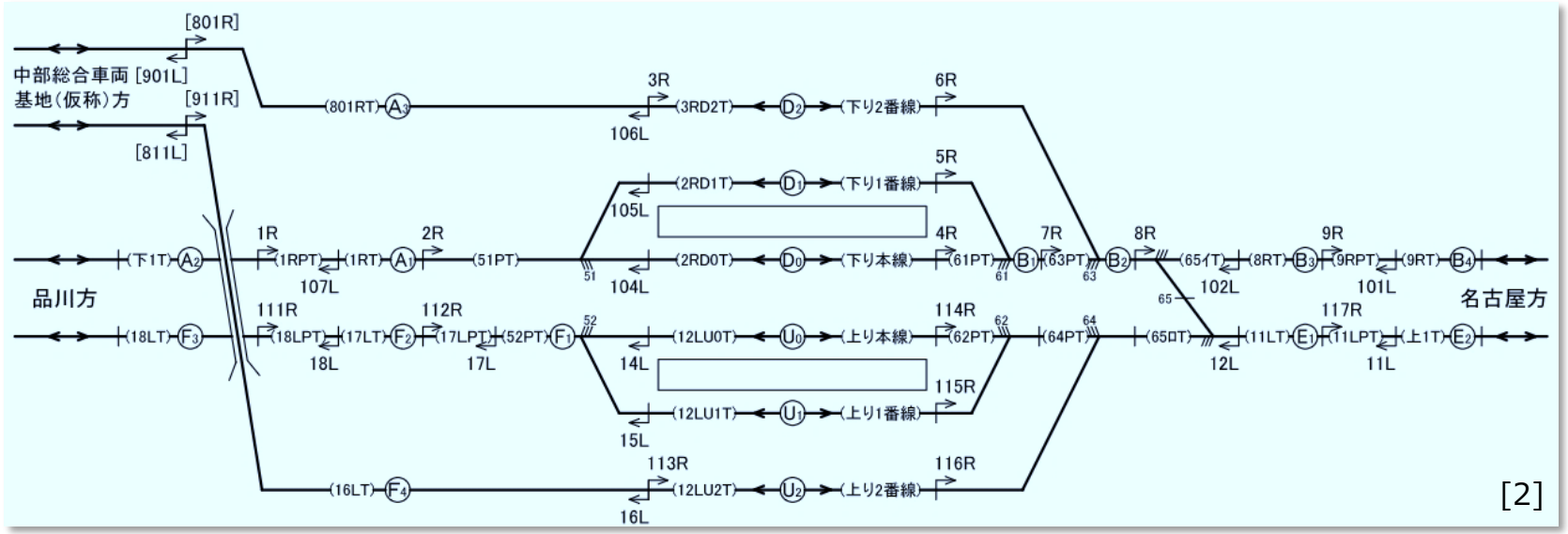
はじめに

- データ駆動科学の再現性・再利用性を考える ~~講座~~ 放談会
- 研究者に限った話ではありません
  - プログラマ、システムエンジニア、経営者を含め、コンピュータでものづくりをするすべての人に問題意識を持ってほしい
- 明日から役立つ耳寄り情報は出てきません
  - 社内／所内で取り組みを始めるきっかけにしていただければ嬉しい
  - NIIで一緒に取り組んでくれる人が見つかればもっと嬉しい

ソフトウェア
~~データ駆動科学~~の再現性・再利用性を考える

H社での思い出（～2008）

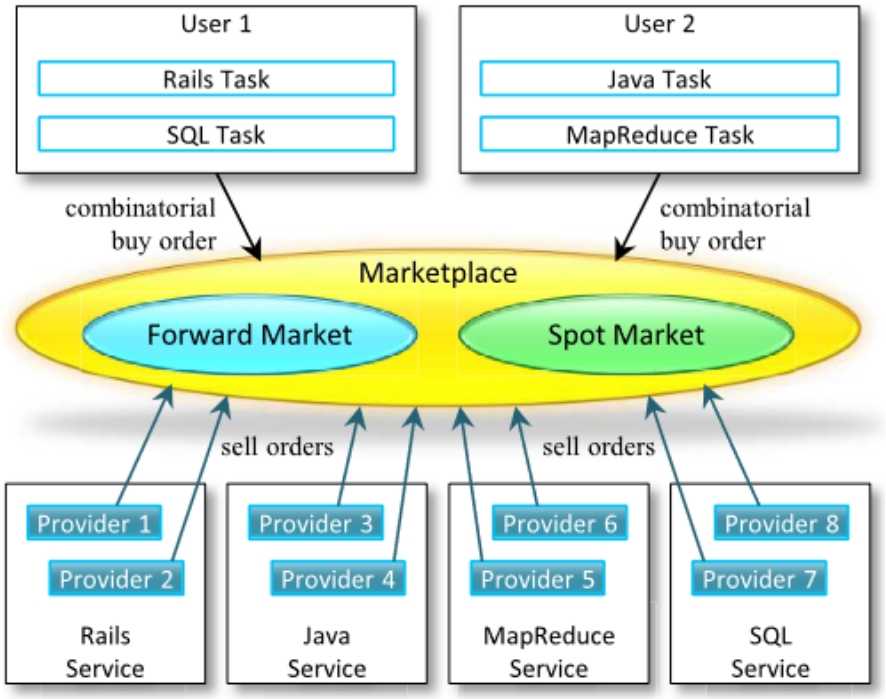
- とあるミッションクリティカルシステムの開発
- 図面や仕様書がクラリスインパクトで作られていた
- 2007年、クラリスインパクト（の後継である AppleWorks）が販売終了
- 手作業で Microsoft Visio へ転記した
  - 同じ Apple 製の iWork は「建前上、AppleWorks 形式のファイル群と互換性があることにはなっているが、互換性はないに等しい」[1]



[1] https://ja.wikipedia.org/wiki/AppleWorks
[2] JR東海「中央新幹線品川・名古屋間の工事実施計画（その3）及び変更の認可申請について」(2023) https://company.jr-central.co.jp/chuoshinkansen/procedure/construction3/_pdf/03_9-2-6.pdf

A研究室での思い出（～2012）

- クラウド資源の市場取引の研究
- 組合せオークション市場のシミュレータを開発
- ドイツのカールスルーエ工科大学などが実施した「CATNETS プロジェクト」の成果である Java ソフトウェアをベースに改造
  - すべての成果物はプロジェクトサイト[1]で公開されていた
- 2024年現在、公式サイトが消滅
  - ソースコードは SourceForge [2]にまだあるが、検索困難
- 改造したソースコードは Google Code で管理していた
- 2015年、Google Code がサービス終了
  - 手作業で GitHub へ移行した

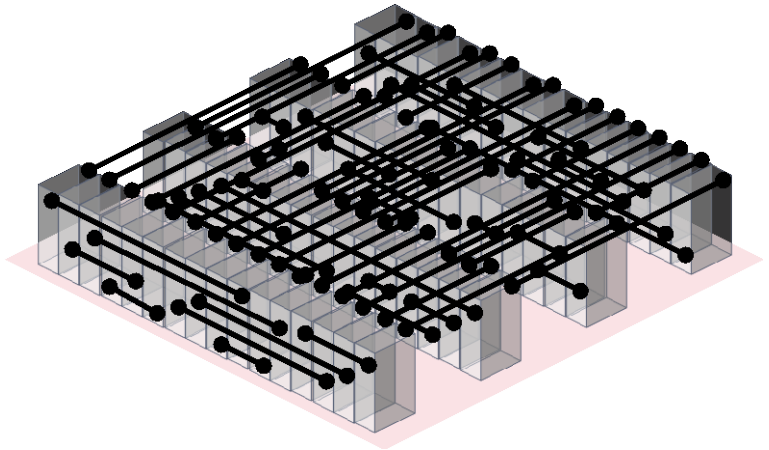


▲クラウド資源マーケットの概念

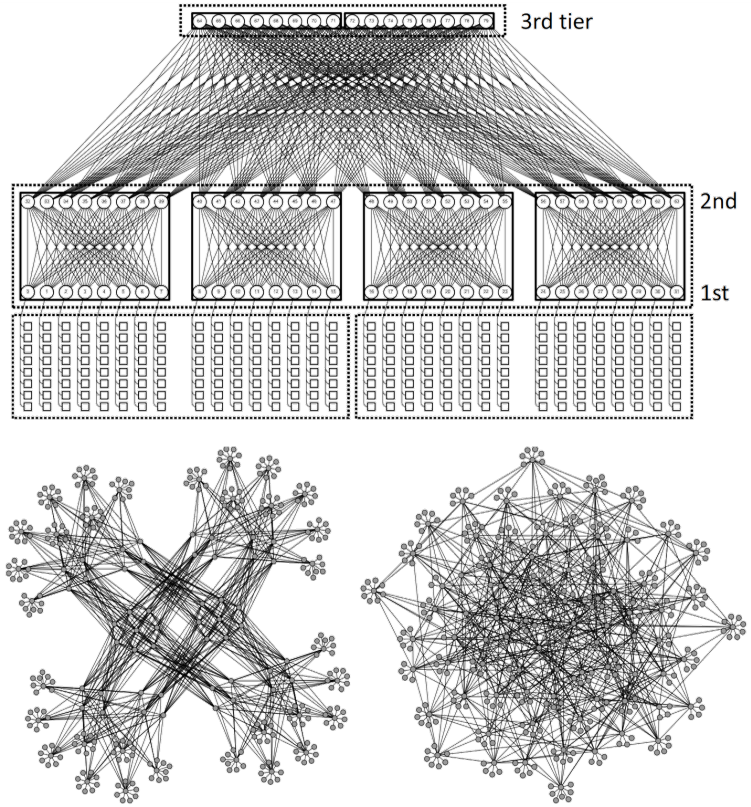
[1] http://www.catnets.org/
[2] https://sourceforge.net/projects/catnets/

K研究室での思い出（～2016）

- スパコンのネットワークトポロジ＝グラフ理論の研究
- R言語のパッケージ「igraph」を用いてグラフを生成・解析
- 2015年、igraph のバージョンアップで、頂点ID／辺IDが0オリジンから1オリジンへ変更
- 手作業で Python へ移植した
  - Rに嫌気が差した



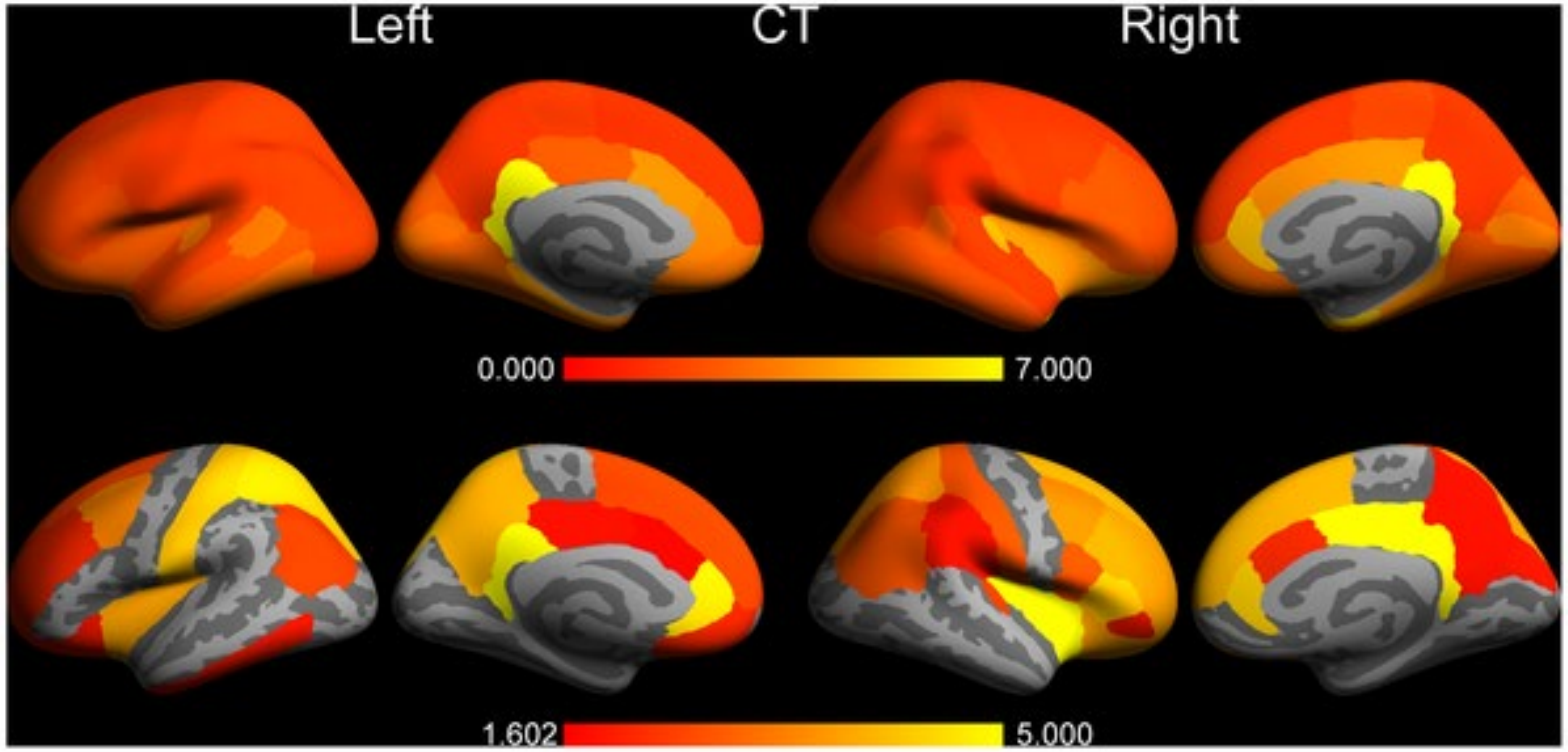
▲ケーブルの短いランダムネットワーク



▲従来の規則的なネットワーク構造（上・下左）と、ランダム化したネットワーク構造（下右）

コンピュータシミュレーションは関係ないのでは？

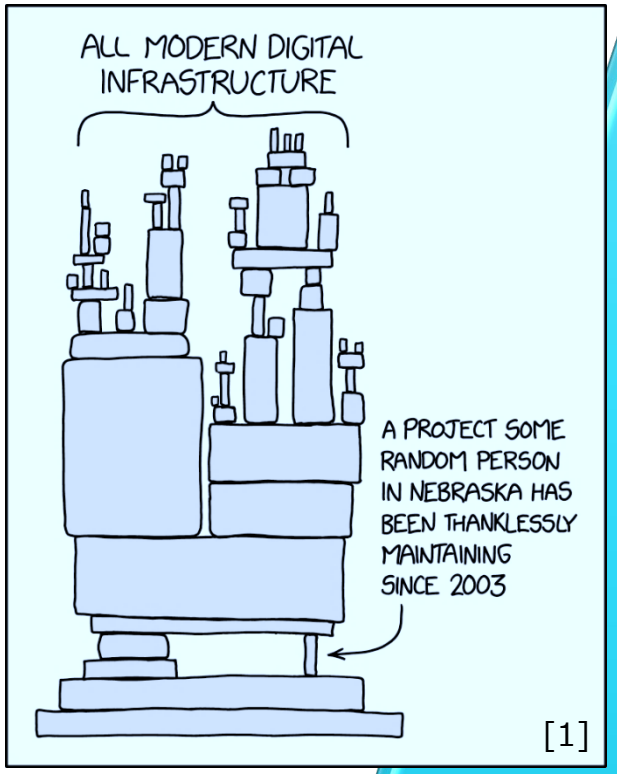
- MRI画像の自動解析ツール「FreeSurfer」で、脳の皮質の厚さを測定
- FreeSurfer v4.3.1 と v5.0.0 で計算結果が異なる
  - 脳の体積で平均 8.8 ±6.6%（範囲 1.3～64.0%）
  - 皮質厚測定値で平均 2.8 ±1.3%（範囲 1.1～7.7%）



[1]

[1] Gronenschild EHBM, et al. "The Effects of FreeSurfer Version, Workstation Type, and Macintosh Operating System Version on Anatomical Volume and Cortical Thickness Measurements". PLOS ONE 7(6) e38234 (2012) https://doi.org/10.1371/journal.pone.0038234

ソフトウェアの再現性・再利用性を阻む問題は？



- ソースコード
- ライブラリ
- 言語処理系
- 実行環境
- OS／ドライバ
- ハードウェア
- ネットワーク？

技術的な問題

- 問題意識
- ドキュメント
- 人材
- メンテナンス体制
- 予算
- ライセンス
- 組織間連携

人的・社会的な問題

[1] https://xkcd.com/2347/

技術的な問題: ハードウェア

起こりうる問題▶ 経年劣化、アーキテクチャの変化、製造中止、修理不能

解決策▼

- 仮想マシン／エミュレータを作る
  - ハードウェアの問題をソフトウェアの問題に移し替える
    - cf. VMWare 値上げ問題
  - 新規開発とあまり変わらない場合も…
- 代替ハードウェアを開発する
- ハードウェアを保存する
  - Living Computers: Museum + Labs[1]（閉館）
  - IPSJ コンピュータ博物館 > 実物の所在地情報[2]

▲仮想マシン技術の構成

[3]

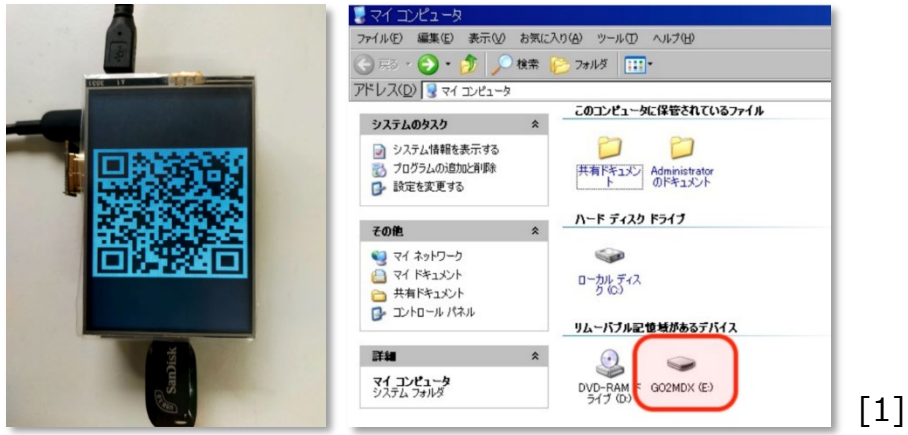
[1] https://livingcomputers.org/
[2] https://museum.ipsj.or.jp/library/shozaichi.html
[3] Joe Mabel. https://commons.wikimedia.org/wiki/File:Living_Computer_Museum_pano_01.jpg

技術的な問題: OS／ドライバ

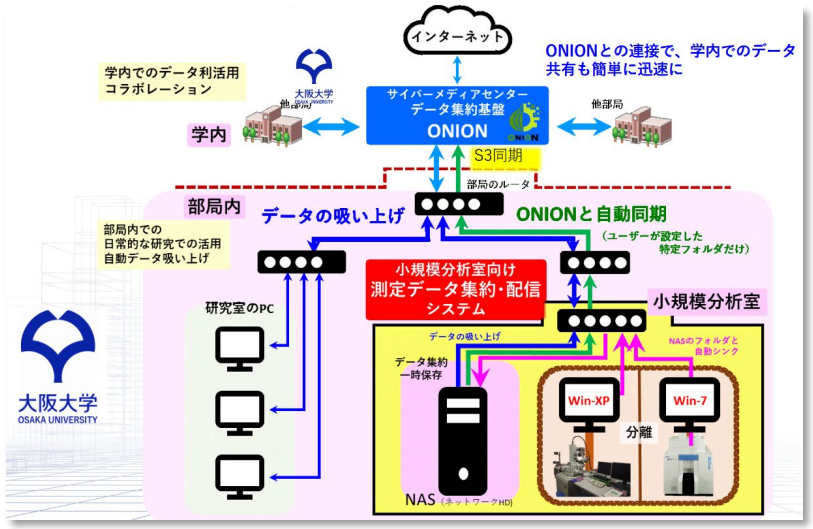
起こりうる問題▶ 非互換変更、バグ修正、サポート終了（新しいハードウェア/OSに対応しない）

解決策▼

- 代替ドライバを開発する
- 仮想マシンを作る
- ハードウェアを保存する／隔離する
  - 古い OS を温存したまま、ネットワークから切り離して使い続ける
    - 解決例1: 東京大学 ARIM-mdx では、オフラインの実験装置からクラウドストレージにデータを送り出すデバイスを開発[1]。実験装置の制御PCにリムーバブルメディアとして接続する
    - 解決例2: 大阪大学コアファシリティでは、小規模分析室向け測定データ集約・配信システムを開発[2]。NASとして分析装置の制御PCとデータ集約基盤を中継する



[1]



[2]

[1] https://arim.mdx.jp/docs/ja/iot.html

[2] 戸所泰人「研究データ管理（RDM）は研究活動をどう促進するか　部局的観点の推進と課題」大学ICT推進協議会 2024年度年次大会 https://rdm.axies.jp/_media/sites/14/2025/01/AXIES2024-RDM-3-%E5%A4%A7%E9%98%AA%E5%A4%A7_%E6%88%B8%E6%89%80.pdf

技術的な問題: 言語処理系／実行環境／ライブラリ

起こりうる問題▶ 消失、非互換変更、依存関係破損、サポート終了（新しいOSに対応しない）

解決策▼

- 仮想マシンを作る
- コンテナ化する
  - 外部の環境変化に対応できない点は仮想マシンと同様
  - コンテナの再ビルドが将来失敗しないとは限らない
- ライブラリのバージョンを厳密に固定する
  - 解決例: パッケージマネージャ「Nix」は、再現可能な実行環境を宣言的に記述するビルドシステムを持ち、依存関係の衝突が原理的に発生しない
    - 提供元のソースコードが消滅した場合、Software Heritage から入手してコンパイルすることも可能

▲コンテナ技術の構成

技術的な問題: ソースコード

起こりうる問題▶ 逸失、入手不能、検索不能、悪意ある変更

解決策▼

- オープンソースにする
  - GitHubなどのリポジトリに公開しておけば、Software Heritage[1]がクロールして永久保存してくれる
- オープンソースにできない場合
  - 社内の話なら、通常の業務データ保存やBCPと同じ話
  - そのソフトウェアに依存する社外の研究があったら？
    研究の再現性に対して責任を負わないの？
- 他者／他社が書いたソースコードが手に入らない／見つからない
  - → 人的・社会的な問題に帰着

[1] https://www.softwareheritage.org/

人的・社会的な問題

- 問題意識／ドキュメント／人材／メンテナンス体制／継続的な予算確保
  - 結局は経営判断であり、投資の優先順位の問題 → 市場主義では解決できない予感
  - 公的資金を投入して実施される研究に対しては、短期的な経済合理性とは異なる長期的な価値基準が必要
- ライセンス（特に、商用ソフトウェアが廃盤になった場合）
  - 商用ソフトウェアを用いた研究が、そのソフトウェアと一蓮托生になってよいのか？
  - ソフトウェア市場の寡占が本質的な問題 → 公正取引委員会の出番ではなかろうか

ソフトウェアは老化する

> *Programs, like people, get old. We can't prevent aging, but we can understand its causes, take steps to limits its effects, temporarily reverse some of the damage it has caused, and prepare for the day when the software is no longer viable. A sign that the Software Engineering profession has matured will be that we lose our preoccupation with the first release and focus on the long term health of our products. Researchers and practitioners must change their perception of the problems of software development. Only then will Software Engineering deserve to be called Engineering. — David Lorge Parnas*

[1]

プログラムは人間と同じように老いていく。老化を防ぐことはできないが、その原因を理解し、その影響を抑えるための手段を講じ、老化が引き起こしたダメージの一部を一時的に回復させ、ソフトウェアが実行不可能になる日に備えることはできる。ソフトウェアエンジニアとして成熟することは、初回リリースへの執着から離れ、製品の長期的な健全性に焦点を当てるようになることだ。研究者と実務家は、ソフトウェア開発の問題に対する認識を変えなければならない。そうして初めて、ソフトウェア工学はエンジニアリングと呼ばれるにふさわしいものになる。

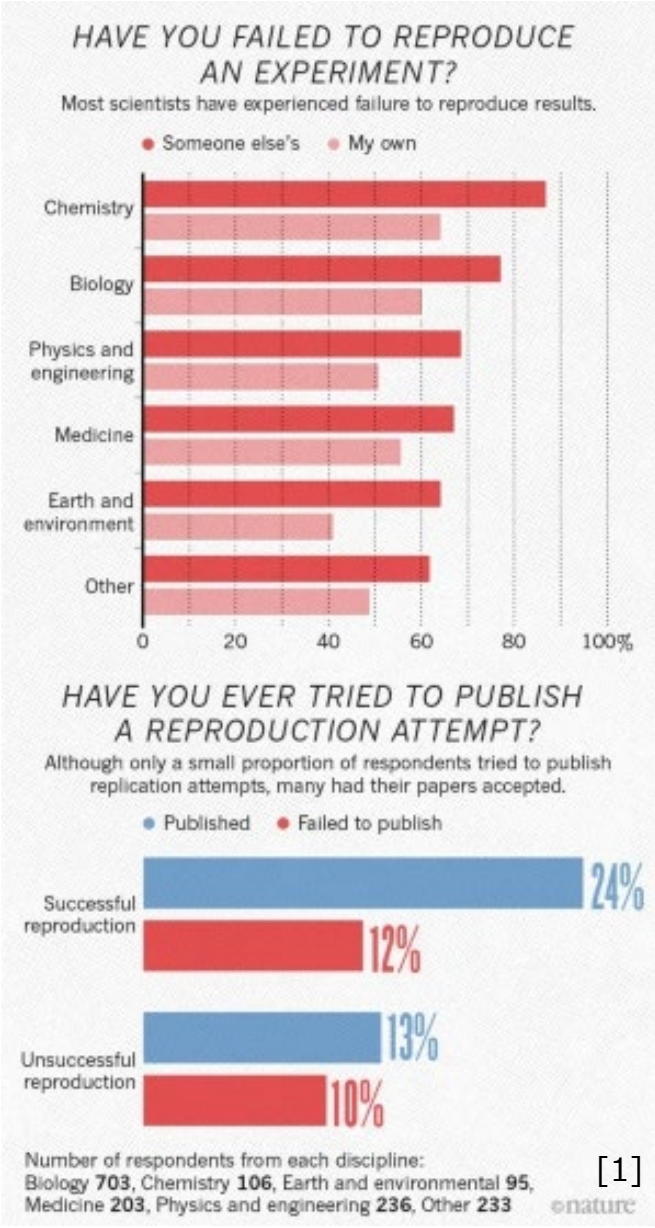
[1] David Lorge Parnas. "Software aging". In Proceedings of the 16th international conference on Software engineering (ICSE '94). IEEE Computer Society Press, pp.279–287. 1994.

データ駆動科学の再現性・再利用性を考える

本節は、藤原一毅「そのグラフ、再現できますか？ ～研究再現性・再利用性を支えるシステムの開発～」信学技報 vol.122, no.204, CPSY2022-18, pp.7-7 (2022) より一部改変して引用したものです

研究再現性の危機

- 2016年、Nature誌が『1500人の科学者が再現性の蓋を開ける』という特集記事を掲載[1]
  - 1576人の研究者を対象とした大規模な調査で、70%以上の研究者が他の科学者の実験を再現できなかった経験があると回答
- 主な要因
  - 研究データの透明性不足: 生データへのアクセスが制限される、実験手順の詳細な記録が不十分、コードや分析手順の共有が不足
  - 統計的手法の誤用: p-値ハッキング（都合の良いデータだけを選択する）、サンプルサイズが小さすぎる、多重検定の問題を適切に処理していない、など
  - 研究プロセスの構造的問題: ネガティブな結果が公表されにくい（出版バイアス）、研究資金獲得のプレッシャー、画期的な発見を重視する学術誌の傾向
- 解決策
  - メソッドと分析の透明性向上: オープンサイエンスの推進、研究データやコードの公開
  - 教育とトレーニング: 統計手法の適切な使用法の教育、研究倫理教育の強化、データ管理・共有のベストプラクティス教育
  - 研究評価システムの改革: 再現性研究への評価・支援、ネガティブな結果の出版促進、研究計画の事前登録制度



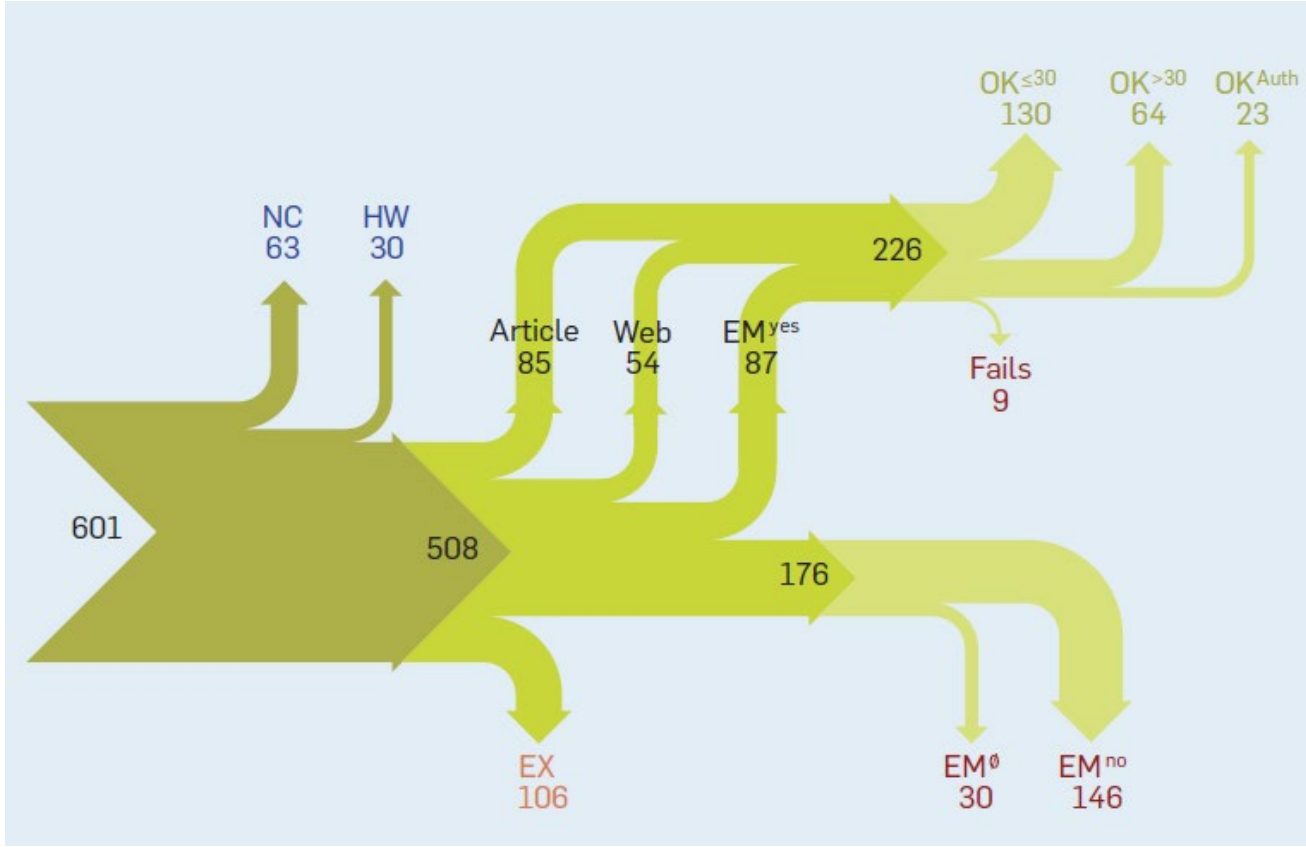
[1]

[1] Monya Baker, "1,500 scientists lift the lid on reproducibility", Nature vol.533, pp.452-454 (2016) https://doi.org/10.1038/533452a

計算再現性も危機

- ACMの論文601件を対象に、そのソースコードの入手可能性とビルドの可否を検証
  - 32.3%が30分以内でビルド成功、48.3%が時間無制限でビルド成功、54.0%が著者ならビルド可能

HW	特殊なハードウェアが必要なため除外
NC	コードによる裏付けがない結果のため除外
EX	著者リストの重複のため除外
BC	結果がコードによって裏付けられている
Article	論文自体でコードが見つかった
Web	Web検索によってコードが見つかった
EM^{yes}	メール要求後、著者がコードを提供
EM^{no}	メール要求に対して、著者がコードを提供できないと回答
$EM^{\emptyset}$	メール要求から2か月以内に著者から返信なし
$OK^{\leq 30}$	コードが利用可能で、システムの構築に30分以内で成功
$OK^{>30}$	コードが利用可能で、システムの構築に30分以上で成功
OK^{Auth}	コードは利用可能だが構築に失敗。著者が妥当な労力でビルド可能と回答
Fails	コードは利用可能だが構築に失敗。著者がコードが壊れている可能性を認めた



[1]

[1] Christian Collberg and Todd A. Proebsting. "Repeatability in computer systems research". Communications of the ACM, vol.59, issue 3, pp.62–69 (2016). https://doi.org/10.1145/2812803

いつ、何のために再現性を求められるか

1. 発表前
   - 共同研究のための再現性

2. 論文投稿時
   - 査読のための再現性

3. 発表後
   - 派生研究のための再現性

4. 不正発覚時
   - 遡及のための再現性

再現性を損なう3つの問題

依存性の問題
（どこで実験するか）

- 実行環境を全く同一にすることが難しい
- ハードウェアの違い、ソフトウェアの違い
- 並列計算の精度、数値計算の安定性

属人性の問題
（誰が実験するか）

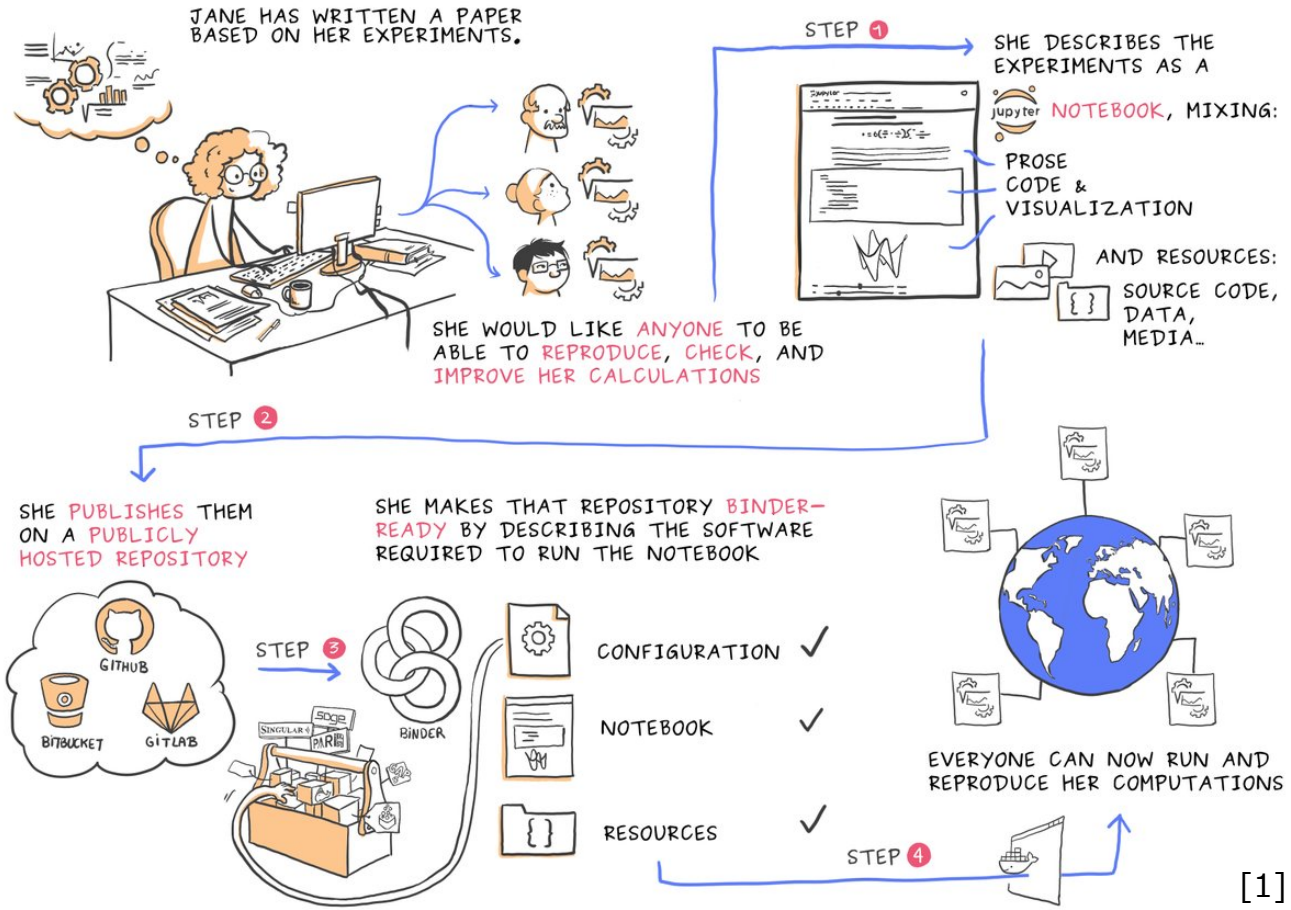
- GUIを用いたファイル管理
- スプレッドシートを用いた前処理
- 手打ちコマンド内のパラメータ指定

永続性の問題（いつ実験するか）

- CD-Rが読み取り不能
- プロプライエタリソフトウェアが入手不能
- 暗黙知を持つ人材が死亡

mybinder.org

- Jupyter Notebookと環境構成情報をまとめてGitHubに保存
  - Jupyter Notebookは、プログラムコード、実行結果、説明文、図表をひとつのドキュメントに統合し、計算内容を理解しながら対話的に実行できるシステム

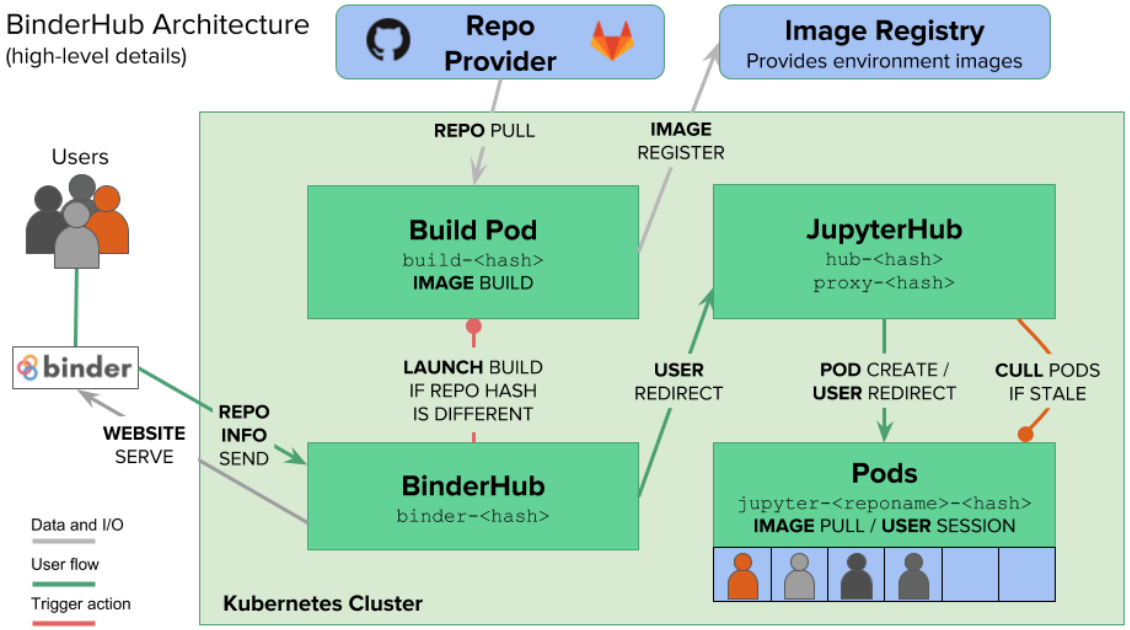


[1]

[1] M. Beg et al. "Using Jupyter for Reproducible Scientific Workflows". Computing in Science & Engineering, vol.23, no.02, pp.36-46 (2021). doi: 10.1109/MCSE.2021.3052101

Binder (BinderHub + repo2docker)

アーキテクチャ



[1]

データモデル

```
name: ipyvolume
channels:
 - conda-forge
dependencies:
 - nodejs
 - pip:
   - traittypes
   - traitlets
   - ipywidgets>=7.4
   - Pillow
   - scipy
   - numpy
   - scikit-image>=0.13
   - requests
   - ipywebrtc>=0.4
   - pythreejs>=1.0
   - matplotlib
   - jupyterlab>=0.34
   - notebook>=5.3
```

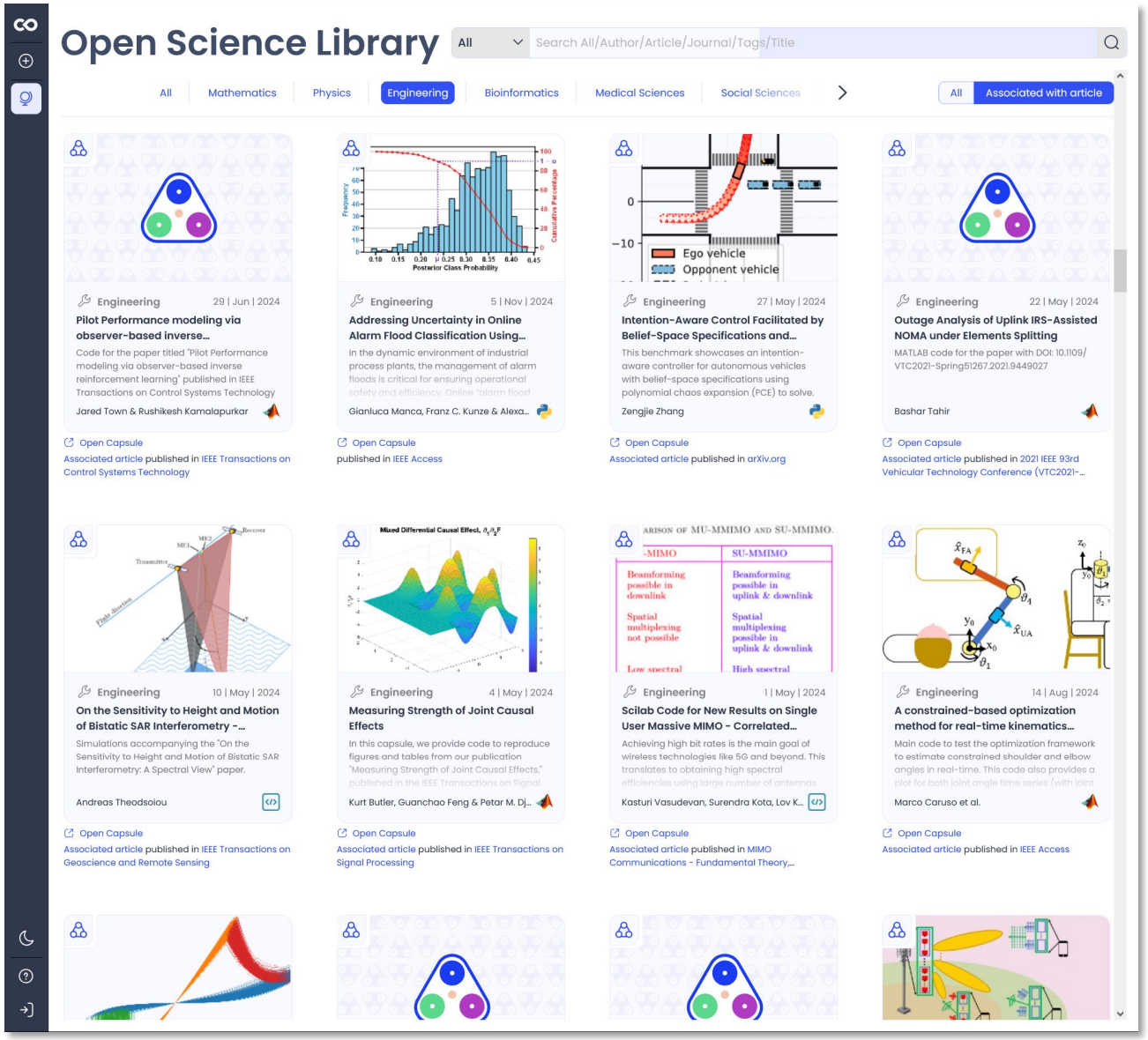
[2]

[1] https://binderhub.readthedocs.io/en/latest/overview.html

[2] https://github.com/binder-examples/requirements

Code Ocean

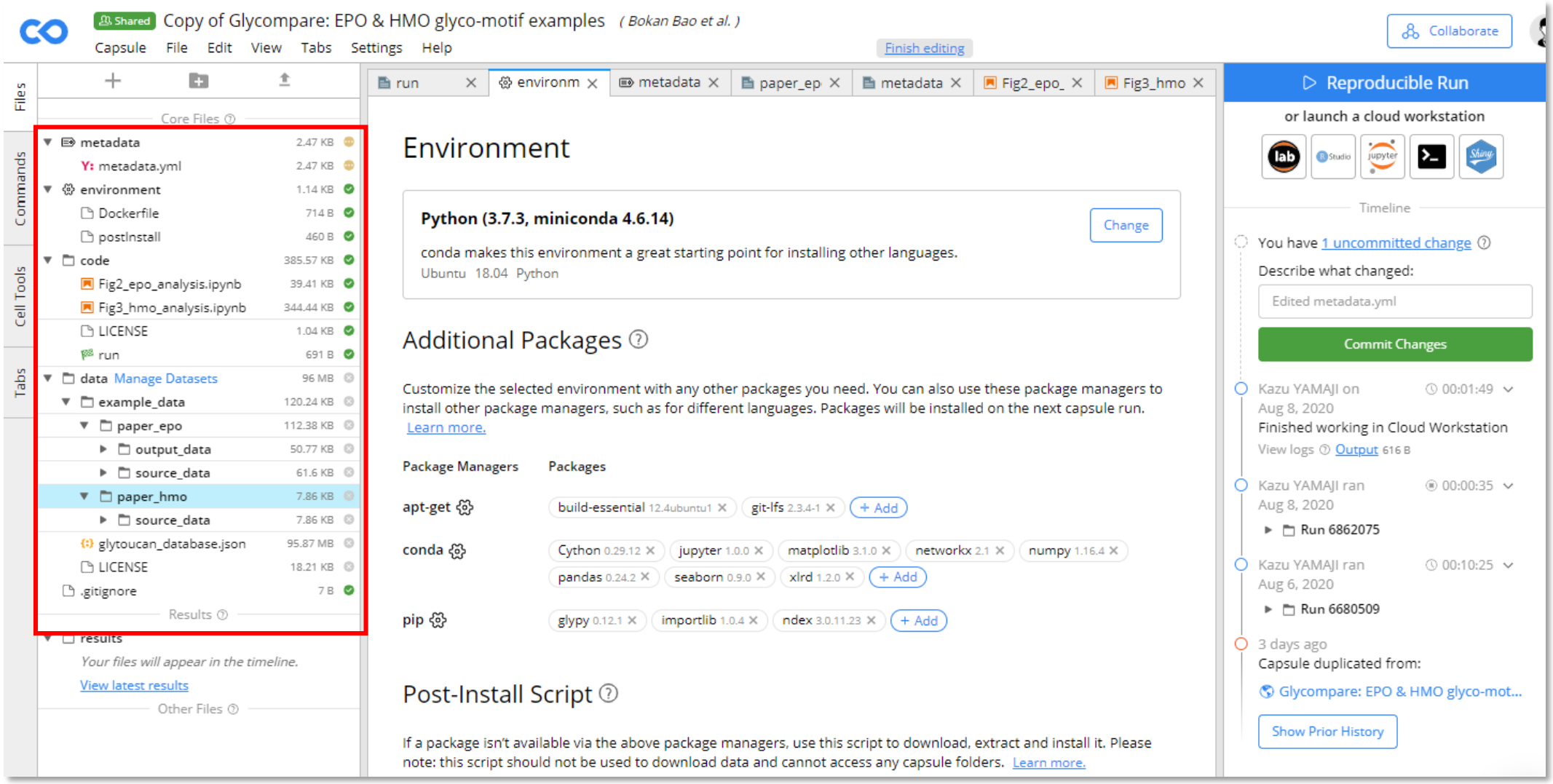
- ソースコード、データ、環境構成情報を含む「計算カプセル」(Compute Capsule) をユーザーが作成
  - 実体は Docker コンテナ
  - ユーザーが追加ライブラリを指定
- Code Ocean 社のクラウド上で実行、保存、公開
  - リポジトリが内部にある
- DOI を自前で付与
- 出版社と契約し査読用に提供
  - 有料、有人サポート



[1]

[1] https://codeocean.com/explore?page=1&filter=withArticle&field=Engineering

CodeOceanのデータモデル

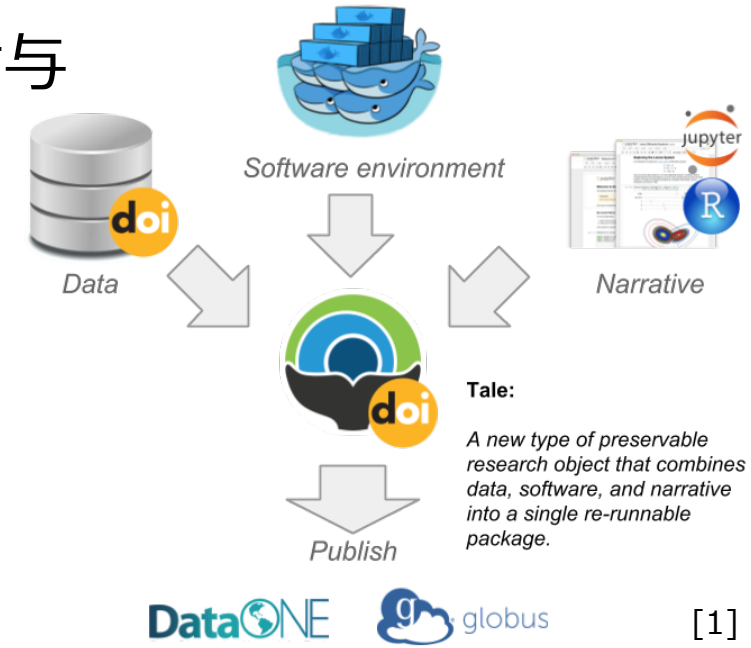


[1]

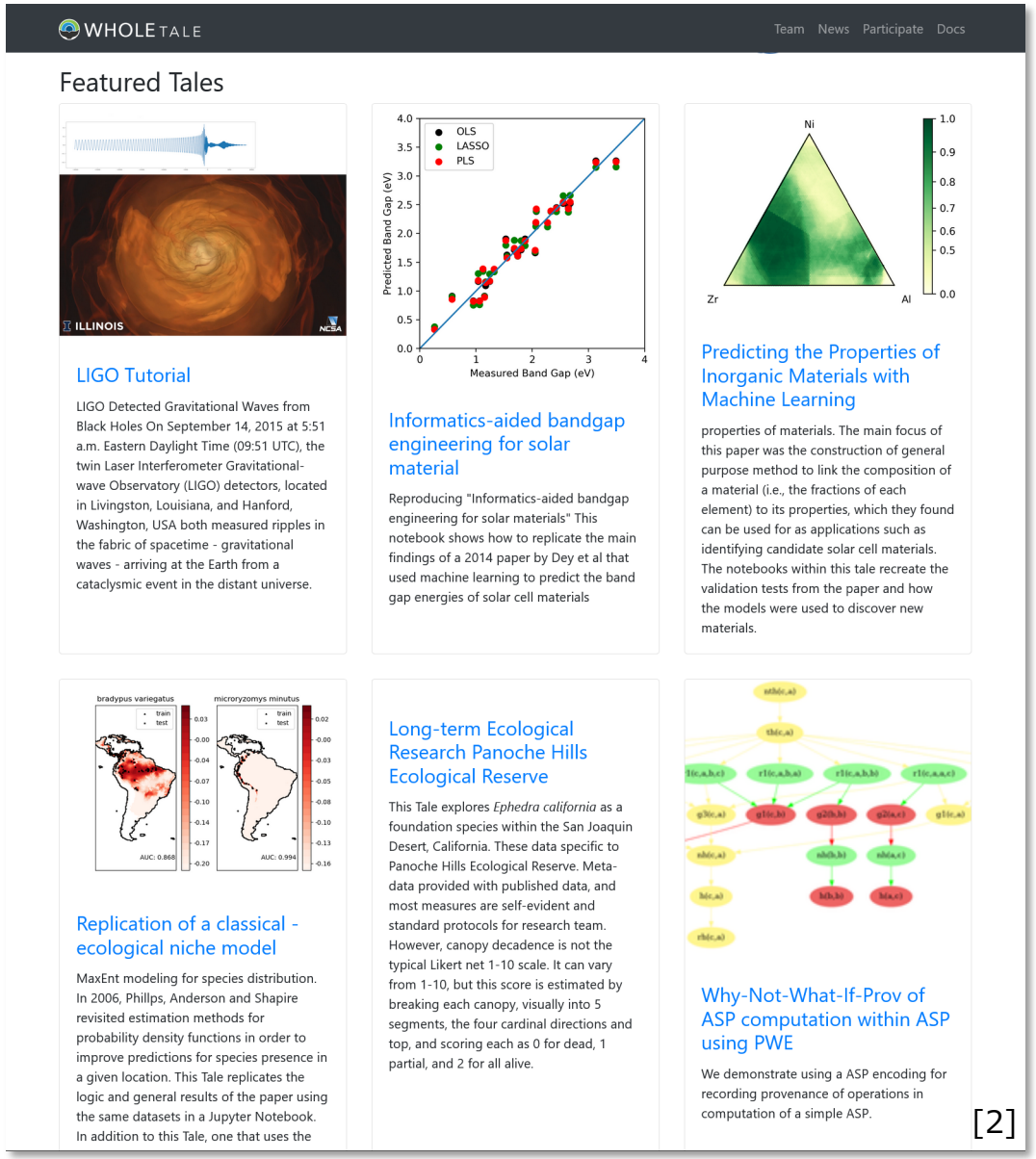
[1] https://codeocean.com/capsule/1787907/tree/v1

Whole Tale

- ソースコード、データ、環境構成情報を含む「テール」(Tale) をユーザーが作成
  - 実体は Docker コンテナ
  - ユーザーが追加ライブラリを指定
- wholetale.org のクラウド上で実行
  - リポジトリは外部依存 (DataOne, Zenodo)
- DOI は外部で付与



[1]

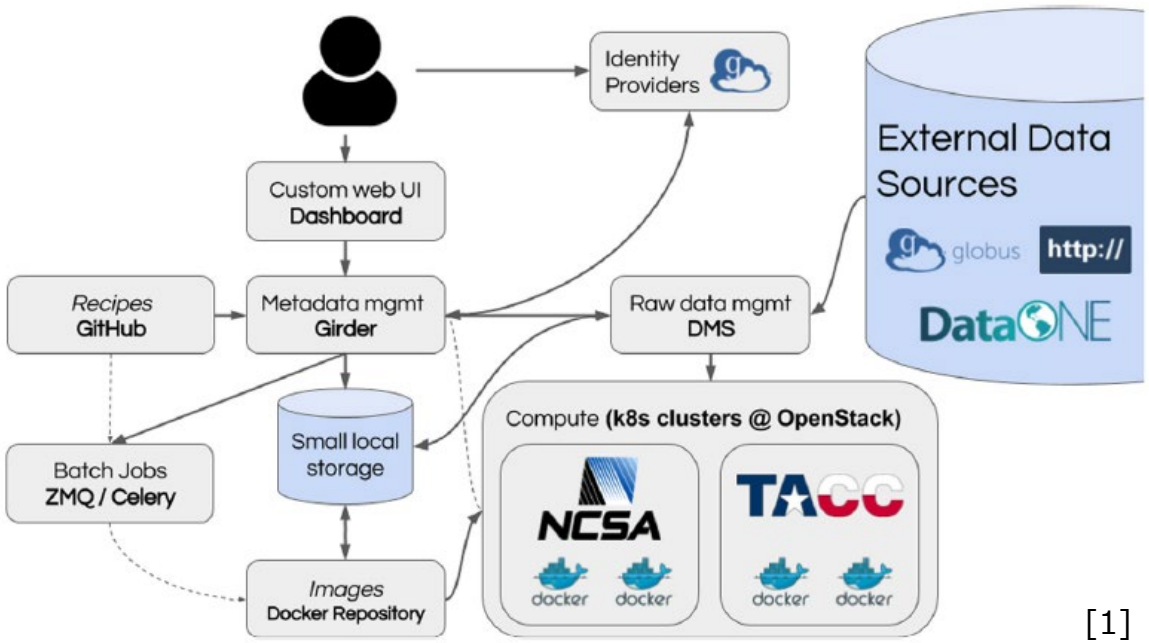


[2]

[1] https://wholetale.org/
[2] https://wholetale.org/#featured

Whole Tale

アーキテクチャ



[1]

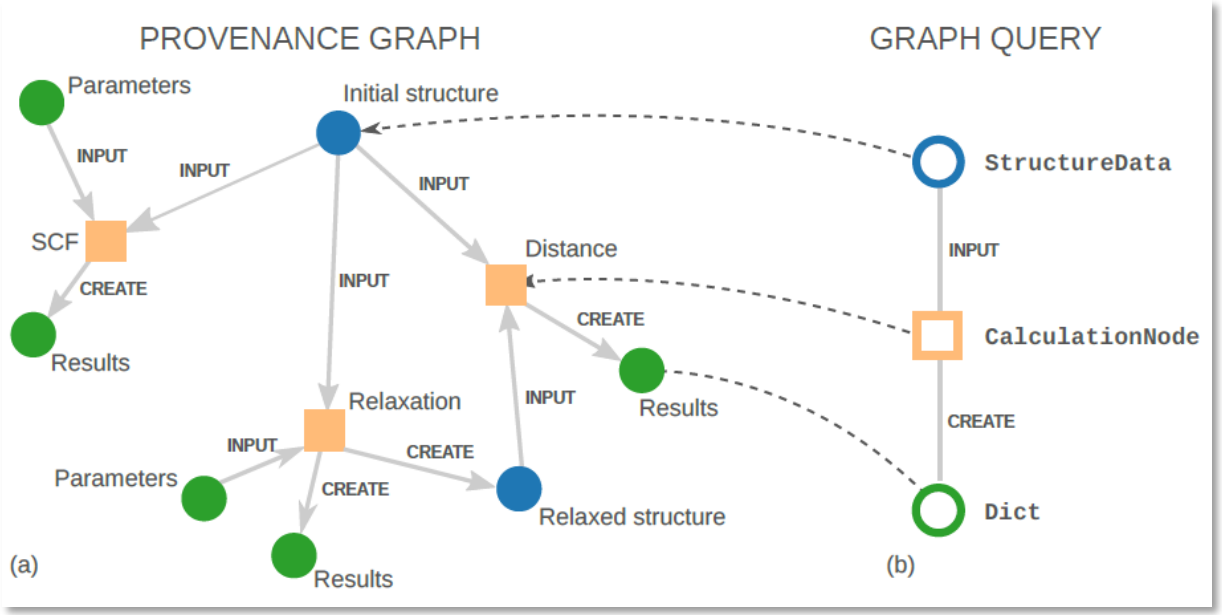
データモデル

```
format: 3
metadata:
  name: Humans and Hydrology Test
  identifier: '8e475f85-d7af-465f-97a1-198b9acdc4fb'
  authors:
    - name: Craig Willis
      orcid: https://orcid.org/0000-0002-6148-7196
  category: science
  description: Test of tale serialization format
  illustration: https://raw.githubusercontent.com/whole-tale/.../demo-graph2.jpg
  entrypoint: wt_quickstart.ipynb
  public: true
data:
  - source: DataONE
    url: http://cn.dataone.org/cn/v2/resolve/urn:uuid:1d23e155-3ef5-47c6-9612-027c80855e8d
  - source: HTTP
    url: http://example.com/data.csv
files:
  - path: notebooks/wt_quickstart.ipynb
    url: https://cn.dataone.org/cn/v2/resolve/urn:uuid:71359f62-b260-4793-a866-418f7fa73aaa
  - path: environment/docker-environment.tar.gz
    url: https://cn.dataone.org/cn/v2/resolve/urn:uuid:71359f62-b260-4793-a866-418f7fa73aaa
environment:
  name: Jupyter Notebook
  url: https://github.com/whole-tale/jupyter-yt
  commit: dc91deafdc959c7edcb8199171b5ac75763323e
  icon: https://raw.githubusercontent.com/whole-tale/rstudio-base/master/RStudio-Ball.png
  archive: environment/docker-environment.tar.gz
  config:
    - command: /init
      environment: CSP_HOSTS=dashboard.dev.wholetale.org,
      port: 8787
      targetMount: /home/rstudio/work
      user: rstudio
```

[1] Adam Brinckman, et al. "Computing environments for reproducibility: Capturing the “Whole Tale””. Future Generation Computer Systems, 94, C (2019), 854–867. https://doi.org/10.1016/j.future.2017.12.029

AiiDA

- 再現可能なワークフローを自動実行するスタンドアロンツール
  - コア、データベース、ワークフローエンジンからなる
- ユーザーは AiiDA API を使って Python プログラムを書く
- AiiDA CLI 経由で実行すると、計算グラフが自動抽出され、来歴として記録される
- 実行はワークフローエンジンで管理され、外部の HPC を利用可
- リポジトリは Materials Cloud Archive



[1]

[1] Huber, S.P., Zoupanos, S., Uhrin, M. *et al.* "AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance". *Sci Data* **7**, 300 (2020). https://doi.org/10.1038/s41597-020-00638-4

アーキテクチャ

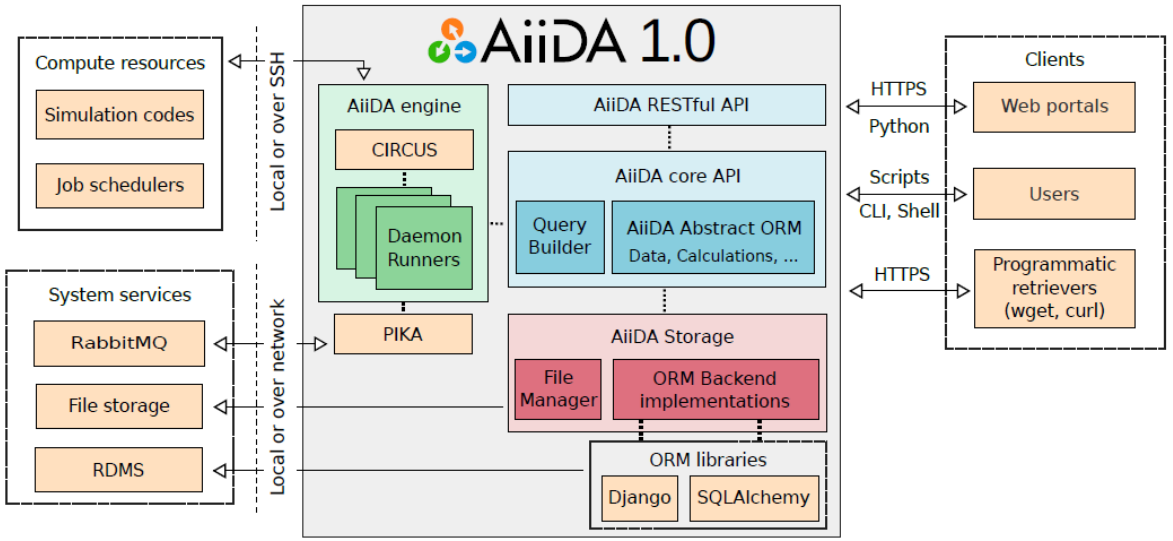
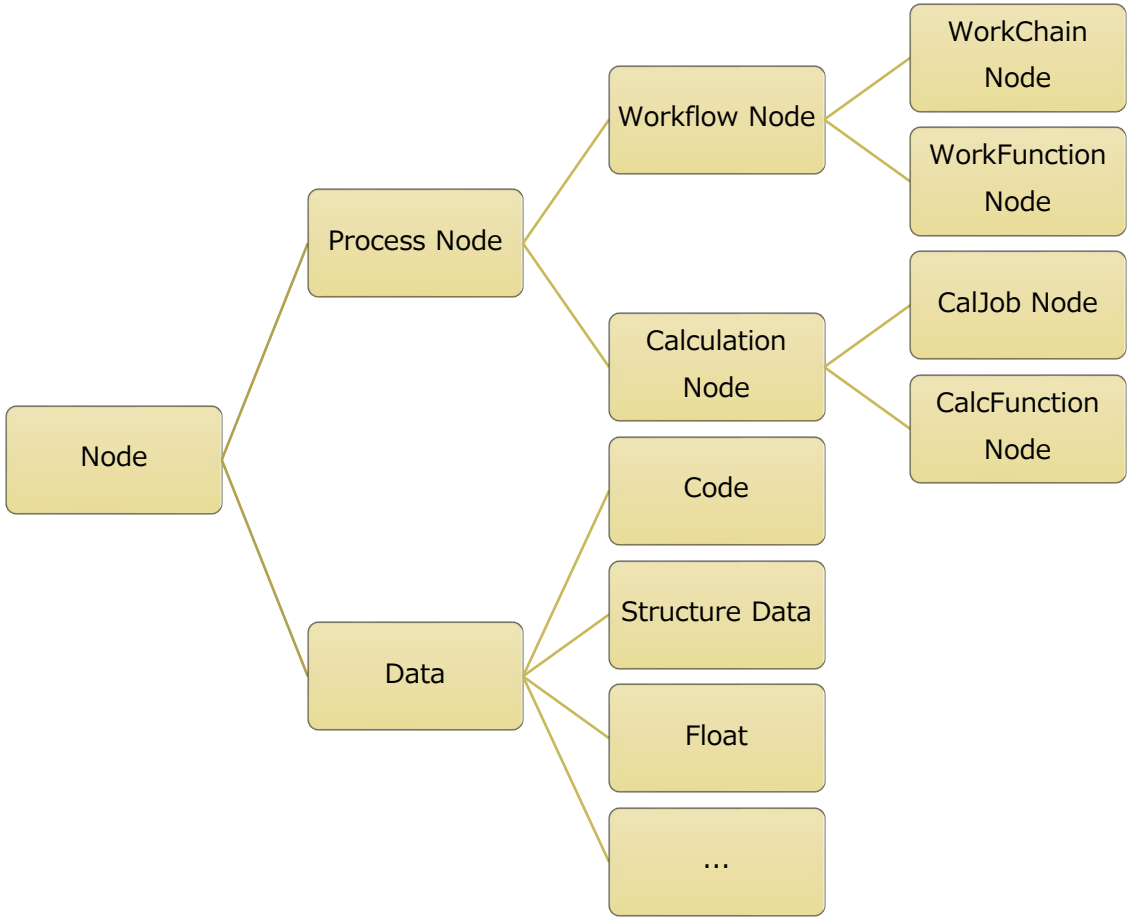


FIG. 1: Schematic overview of the architecture of AiiDA 1.0. [1]

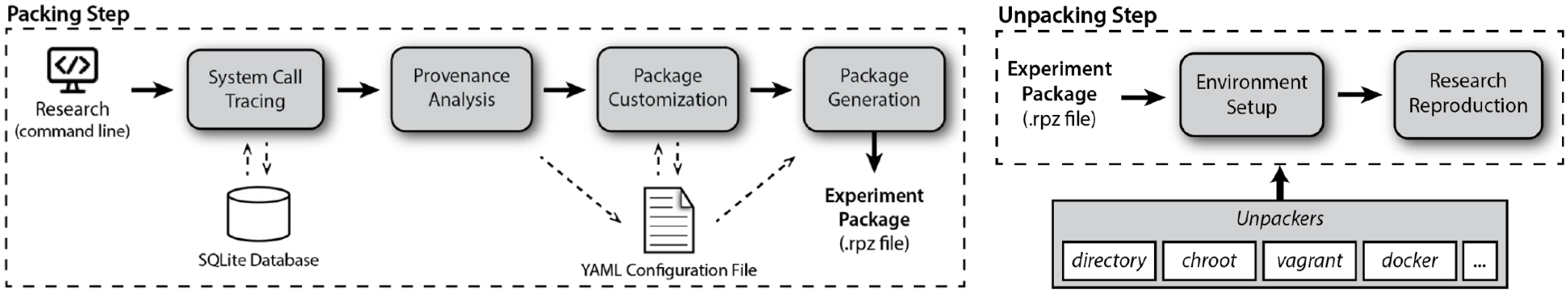
データモデル



[1] Huber, S.P., Zoupanos, S., Uhrin, M. *et al.* "AiiDA 1.0, a scalable computational infrastructure for automated reproducible workflows and data provenance". *Sci Data* **7**, 300 (2020). https://doi.org/10.1038/s41597-020-00638-4

ReproZip

- コマンドラインプログラムの実行を再現可能化するスタンドアロンツール
- 作成者が ReproZip を介してコマンドを実行すると、システムコールがトレースされ、「バンドル」が作成される
  - データファイル、実行可能ファイル（ライブラリ）、環境変数、コマンドラインを含む
- 再現者が ReproUnzip を用いてバンドルを解凍し、仮想環境上で実行する



[1]

```
CREATE TABLE processes(
    id INTEGER NOT NULL PRIMARY KEY,
    run_id INTEGER NOT NULL,
    parent INTEGER,
    timestamp INTEGER NOT NULL,
    is_thread BOOLEAN NOT NULL,
    exitcode INTEGER
    );
```

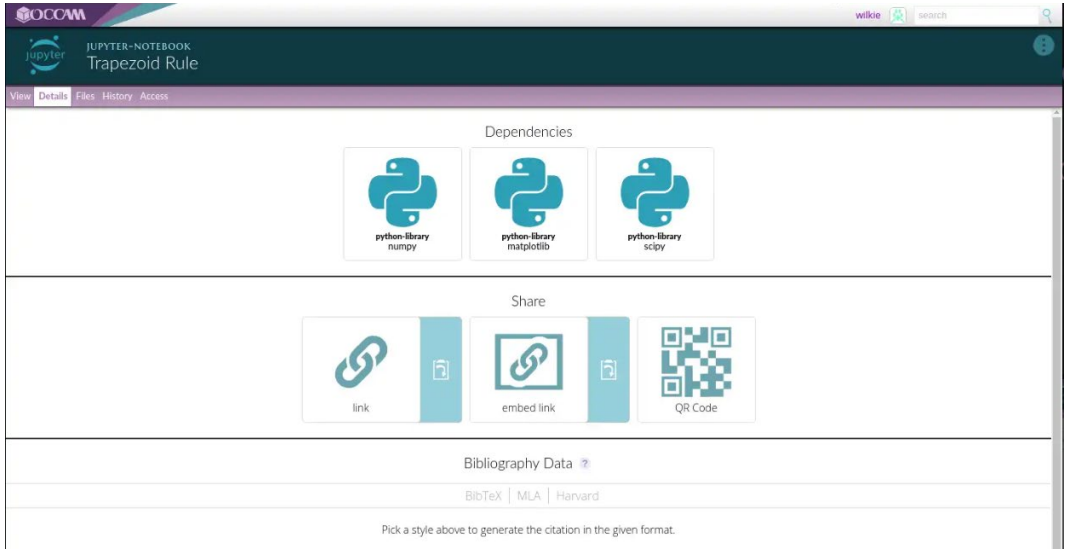
```
CREATE TABLE opened_files(
    id INTEGER NOT NULL PRIMARY KEY,
    run_id INTEGER NOT NULL,
    name TEXT NOT NULL,
    timestamp INTEGER NOT NULL,
    mode INTEGER NOT NULL,
    is_directory BOOLEAN NOT NULL,
    process INTEGER NOT NULL
    );
```

```
CREATE TABLE executed_files(
    id INTEGER NOT NULL PRIMARY KEY,
    name TEXT NOT NULL,
    run_id INTEGER NOT NULL,
    timestamp INTEGER NOT NULL,
    process INTEGER NOT NULL,
    argv TEXT NOT NULL,
    envp TEXT NOT NULL,
    workingdir TEXT NOT NULL
    );
```

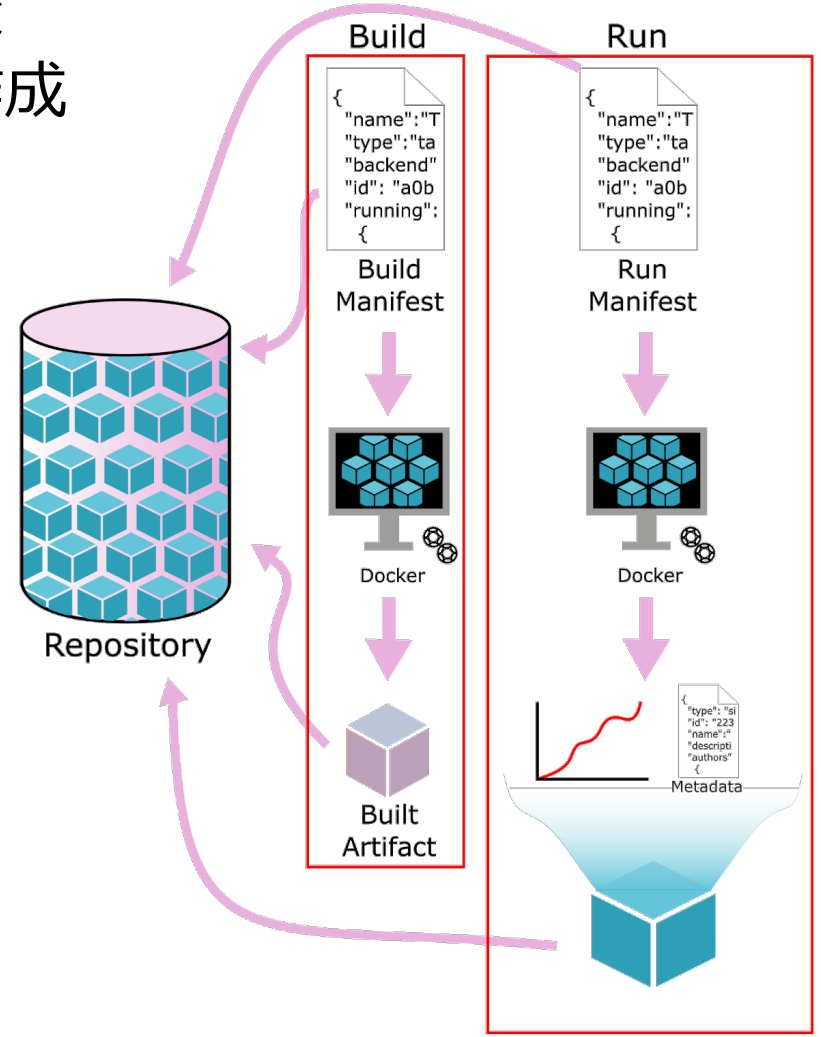
[1] Steeves, V., Rampin, R., & Chirigati, F. (2017). Using ReproZip for Reproducibility and Library Services. *IASSIST Quarterly*, *42*(1), 14. https://doi.org/10.29173/iq18

Occam

- ソースコード、ビルド環境、実行環境、その他必要なファイルを指定して「オブジェクト」をユーザーが作成
- オブジェクトは実行時にビルドされ、Occam 内の VM 上で実行される
  - 外部からダウンロードされたファイルは内部にミラーされる
- ソフトウェアの長期的な保存と再利用をサポート



[1]



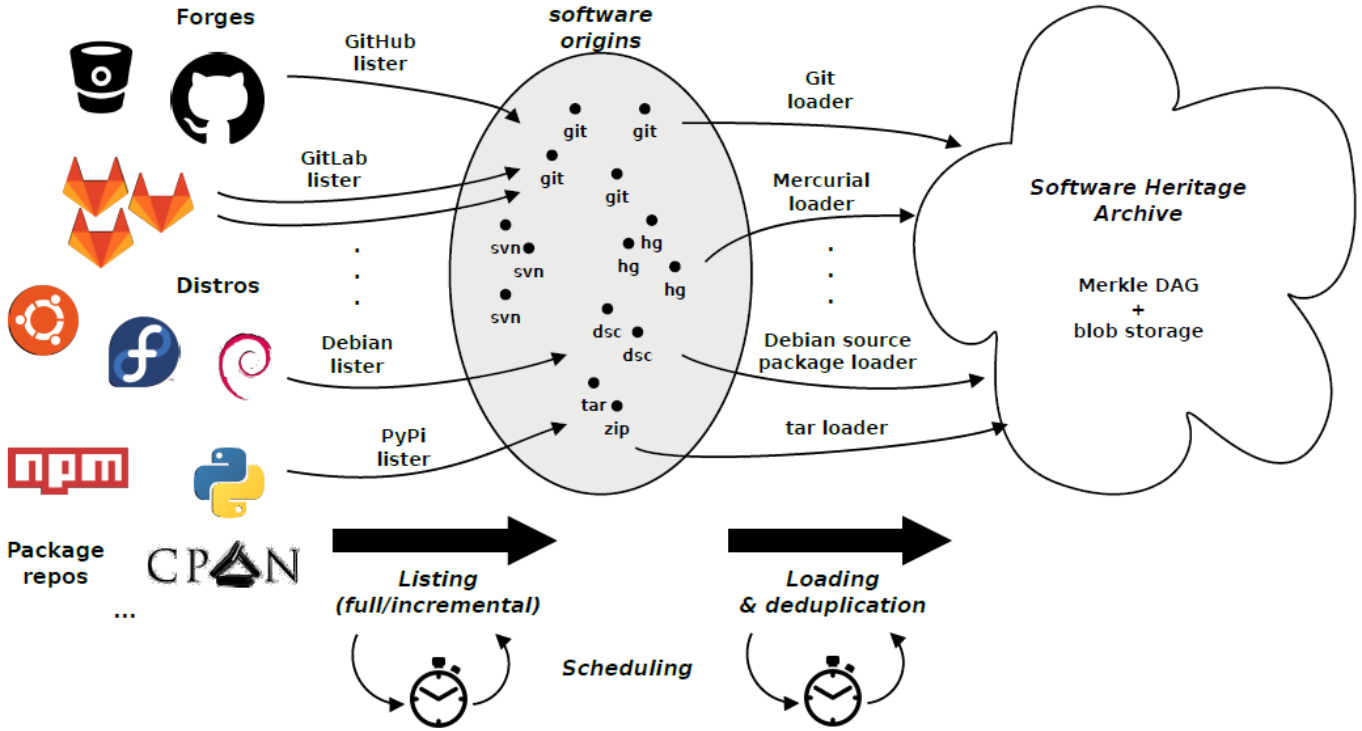
[2]

[1] https://occam.cs.pitt.edu/

[2] Luís Oliveira, et al. "Supporting Long-term Reproducible Software Execution". First International Workshop on Practical Reproducible Evaluation of Computer Systems (2018). https://doi.org/10.1145/3214239.3214245

Software Heritage

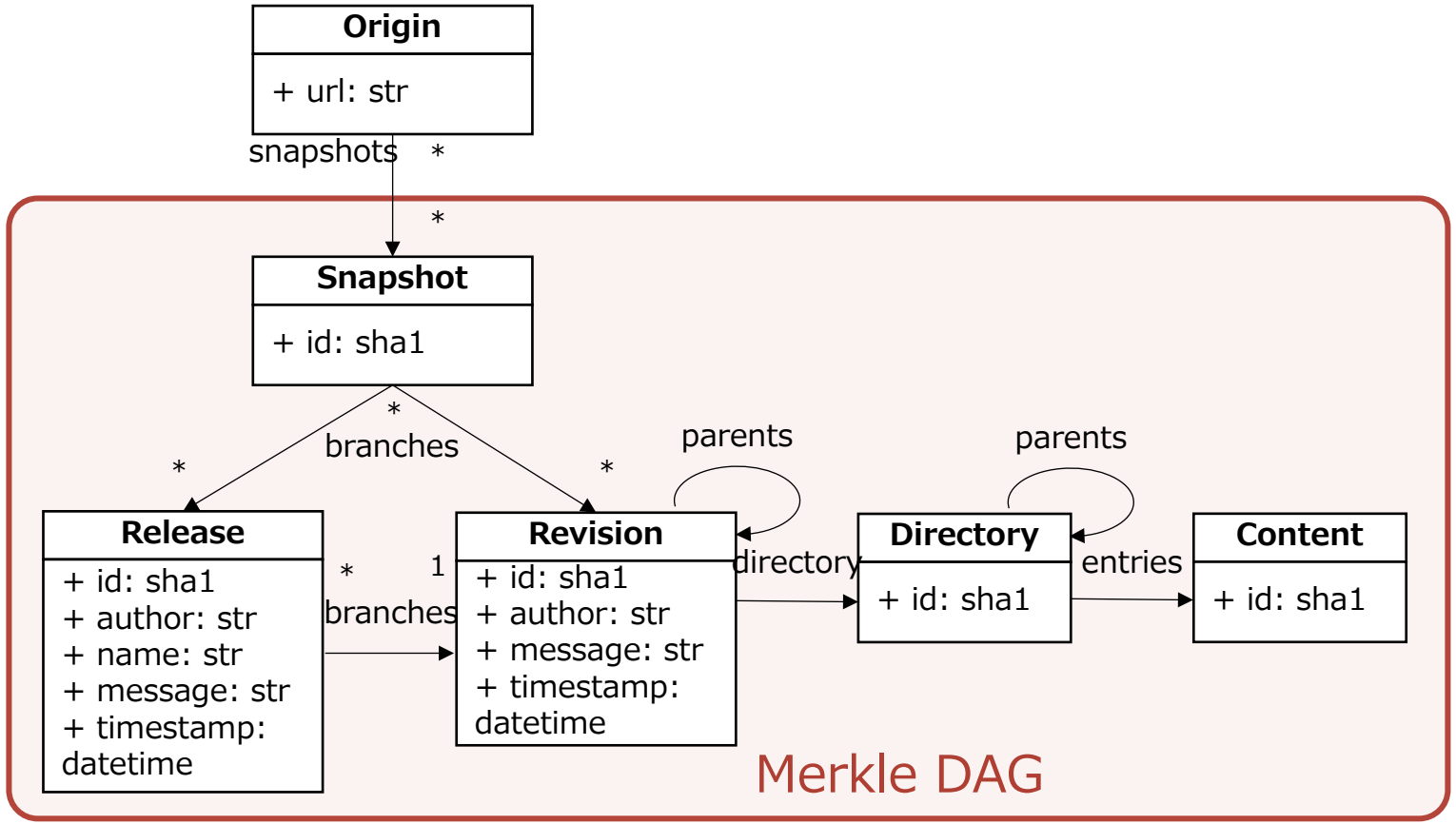
- すべてのオープンなソフトウェアを将来に向けて収集、保存、共有するアーカイブ
- リポジトリをクロールしてすべてのオブジェクトを取得
  - ファイル、ディレクトリ、リビジョン、リリース
- Software Heritage ID をオブジェクトに付与し、ファイルを葉とする DAG 構造で保存する



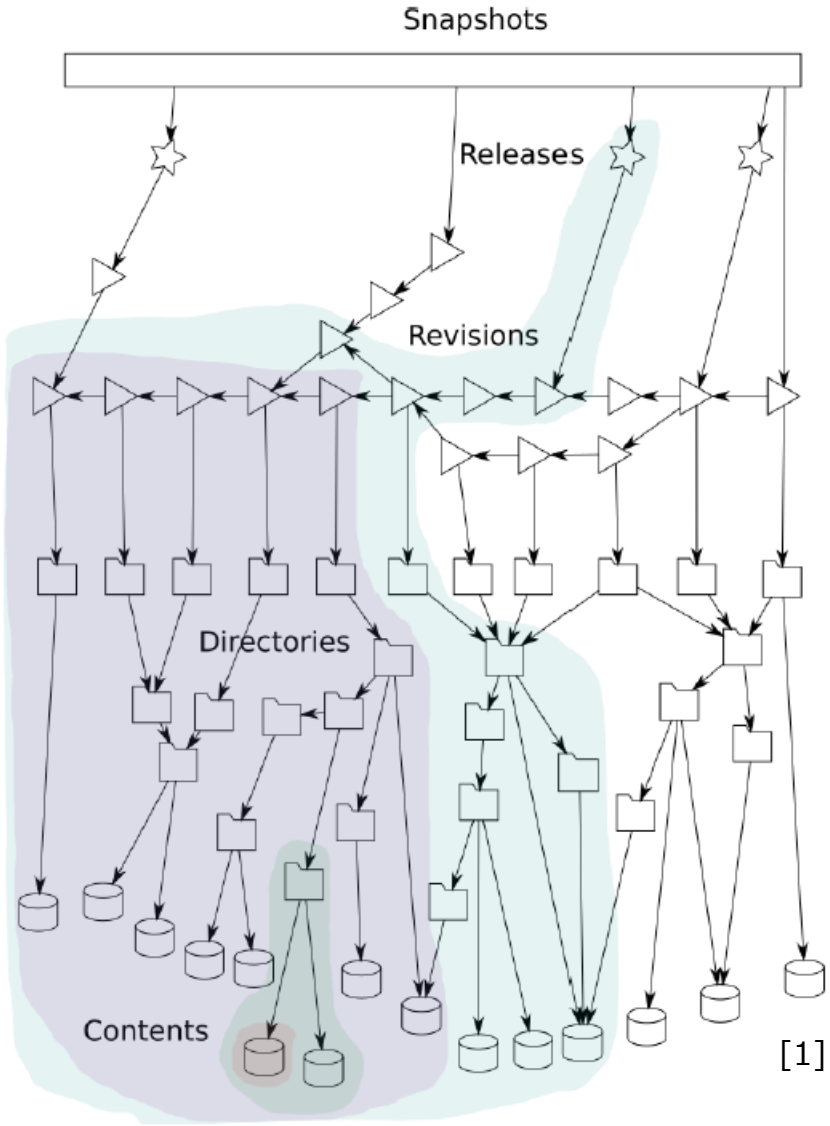
[1]

[1] R. D. Cosmo. "Software Heritage: Why and How We Collect, Preserve and Share All the Software Source Code". IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Society, pp.2-2 (2018)

Software Heritage のデータモデル



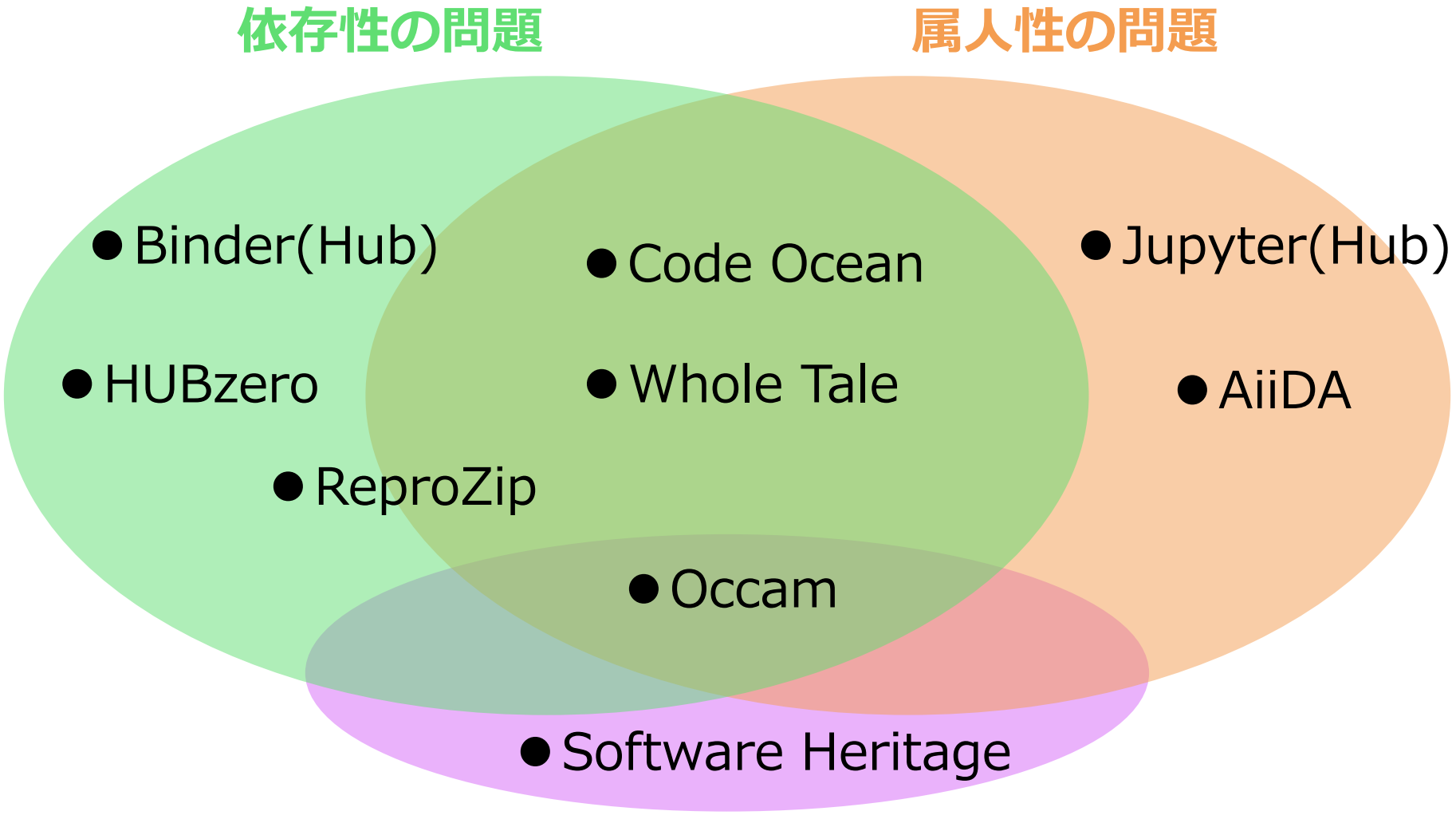
[1]



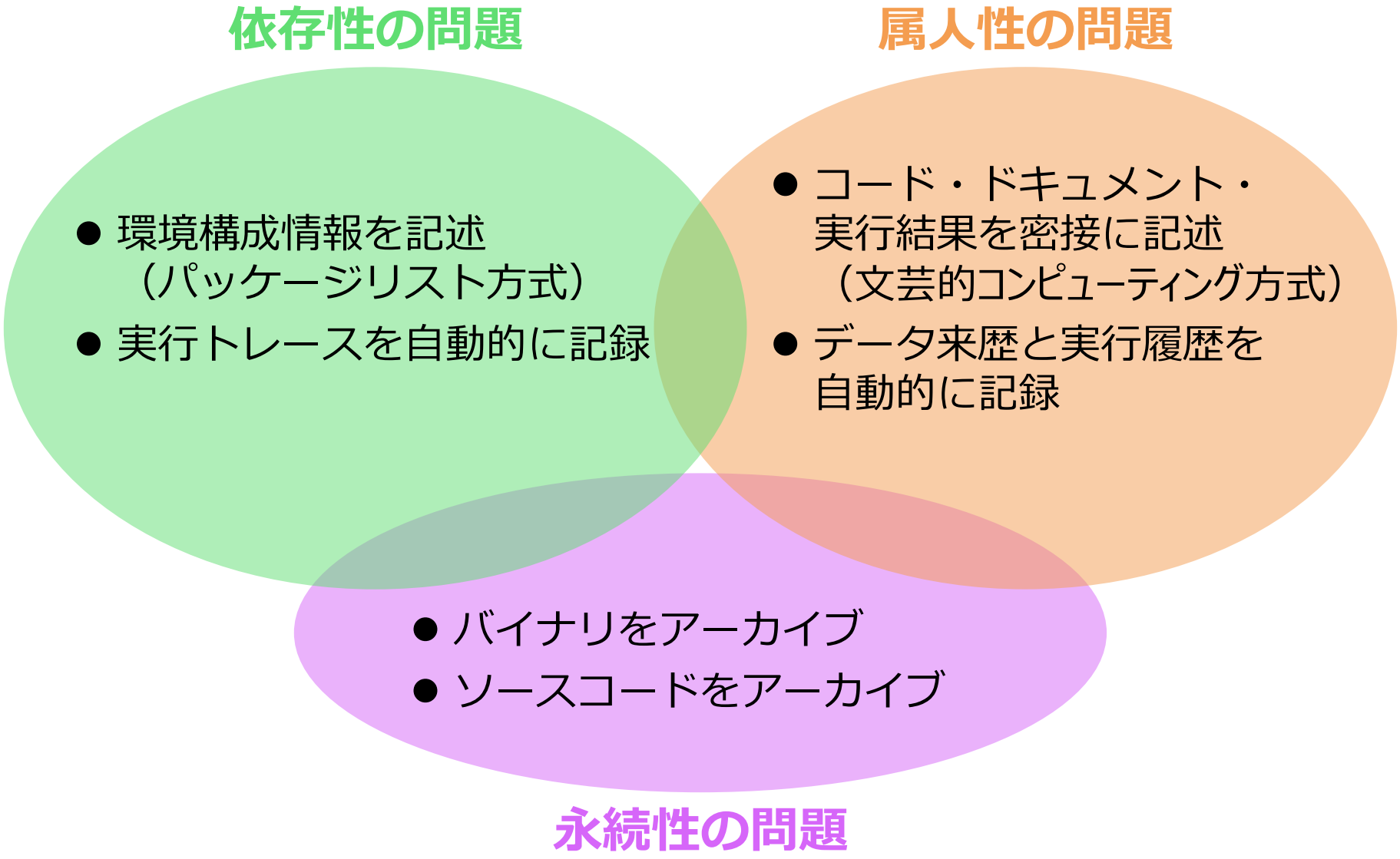
[1]

[1] R. D. Cosmo, M. Gruenpeter and S. Zacchiroli. "Referencing Source Code Artifacts: A Separate Concern in Software Citation". *Computing in Science & Engineering*, vol. 22, no. 2, pp. 33-43 (2020). doi: 10.1109/MCSE.2019.2963148.

各システムが解決する課題



再現性を損なう3つの問題へのアプローチ



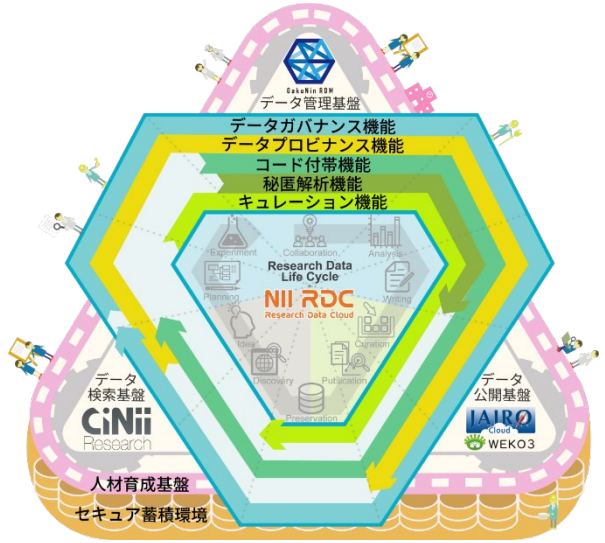
解決されていない問題

- 知的財産権／プライバシーの問題
  - 非公開のデータや機微情報／個人情報を扱う研究の再現性
  - データ所有者の同意が取れなかったら？
- プロプライエタリソフトウェアなどに依存する実験の永続性
  - ソフトウェア開発者が倒産 → 合法的に入手できない
- 人的サポートの問題
  - 研究者や研究支援者のトレーニング
  - データマネジメントプランとの連携

わたしたちができること

NII RDC (Research Data Cloud) の開発

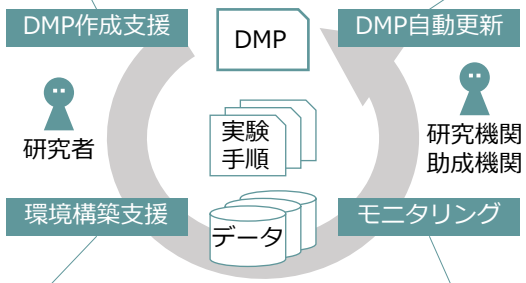
- 大学・研究機関向けのデータ管理サービスとして2021年度から提供
- 管理・公開・検索の基本3機能に加え、高度化7機能を開発中



① データガバナンス機能

データの種類、取得元、蓄積場所、保護レベル、資金源などを含むDMPを半自動的に作成

研究進展に伴いDMPを自動的に更新



ルールに則った解析環境・蓄積環境を自動的に構築

研究データの実態を把握し、所属機関の経営層や研究費助成機関へフィードバック

② データプロビナンス機能

データの出所・来歴を自動的に追跡・記録

公開データの再利用状況を把握し、利活用を促進

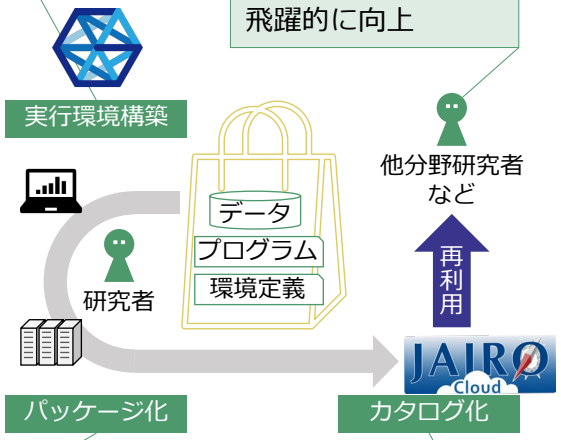
研究成果の真正性を遡及的に検証可能。エンバーゴ期間も管理

利用実績に基づきデータ生産者を学術的に評価

③ データ解析機能

全国の計算資源を活用してデータ解析環境を簡便に構築・共有

研究成果の再現性が飛躍的に向上



ソフトウェアとその実行環境をパッケージとして公開

パッケージを論文と紐づけて検索可能化

GakuNin RDM データ解析機能

❶ 個人用のデータ解析環境をNIIクラウドに※1自動構築。
データ管理基盤「GakuNin RDM」に保存されたデータをすぐに解析へ

❷ 多様なデータ源や計算資源と連携※2

❸ 解析結果をプロジェクトメンバーと共有

❹ プロジェクトを「計算再現パッケージ」として公開

❺ 他の研究者が発見・再利用。発展的な研究へ

※1 機関所有計算機や商用クラウド等も利用可
※2 開発中の機能も含む

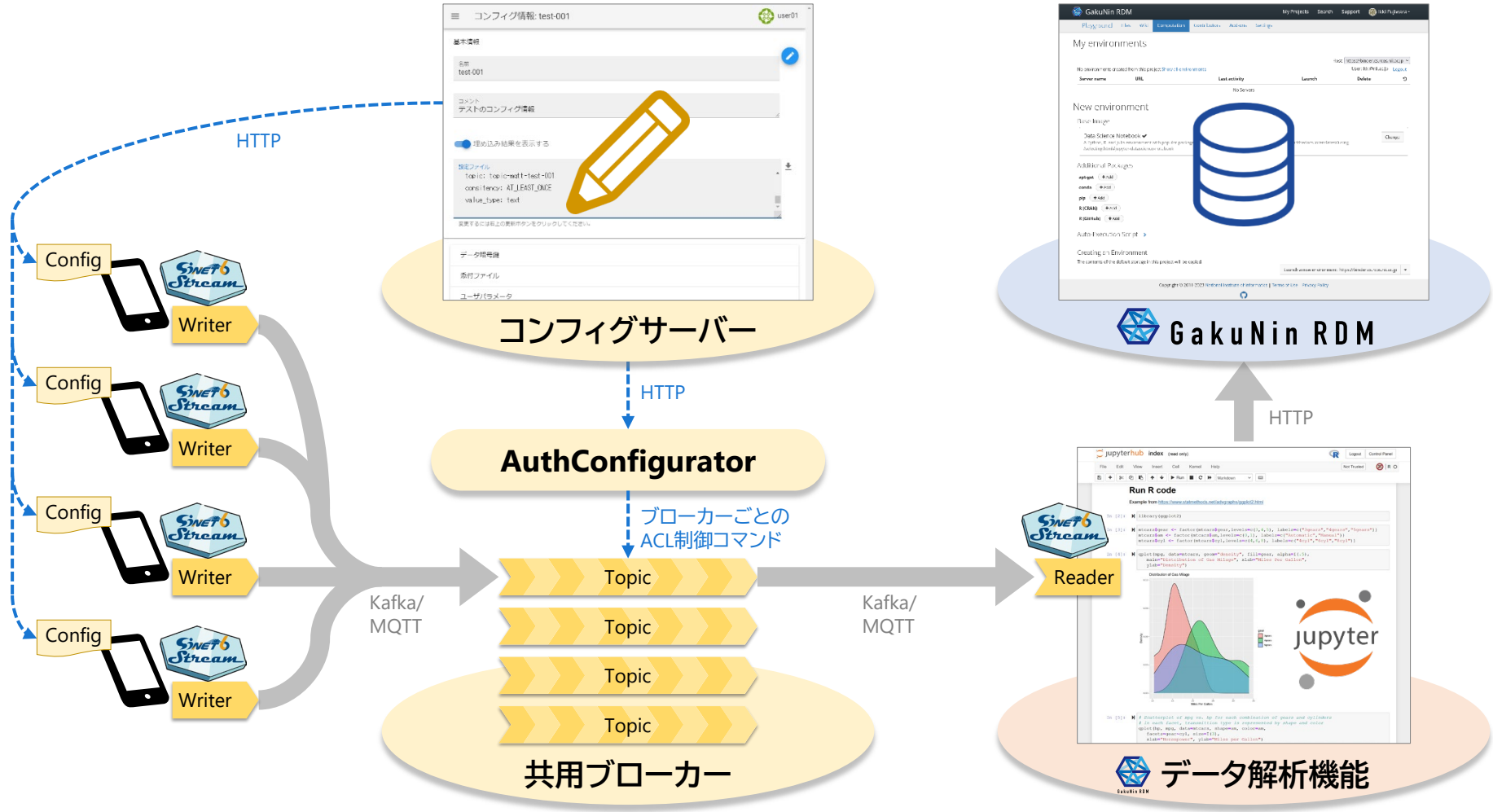
**再現可能な
データ駆動科学の
普及を目指す**

計算再現パッケージ機能

- GakuNin RDM のプロジェクトをパッケージ化して公開基盤へエクスポート
  - データ、プログラム、解析環境定義を含む。メタデータは RO-Crate 形式
- 後続研究者が自分のプロジェクトとしてインポート
  - 先行研究者のデータ解析過程を再現し、発展的な研究をすぐに開始

SINETStream お試しサービス

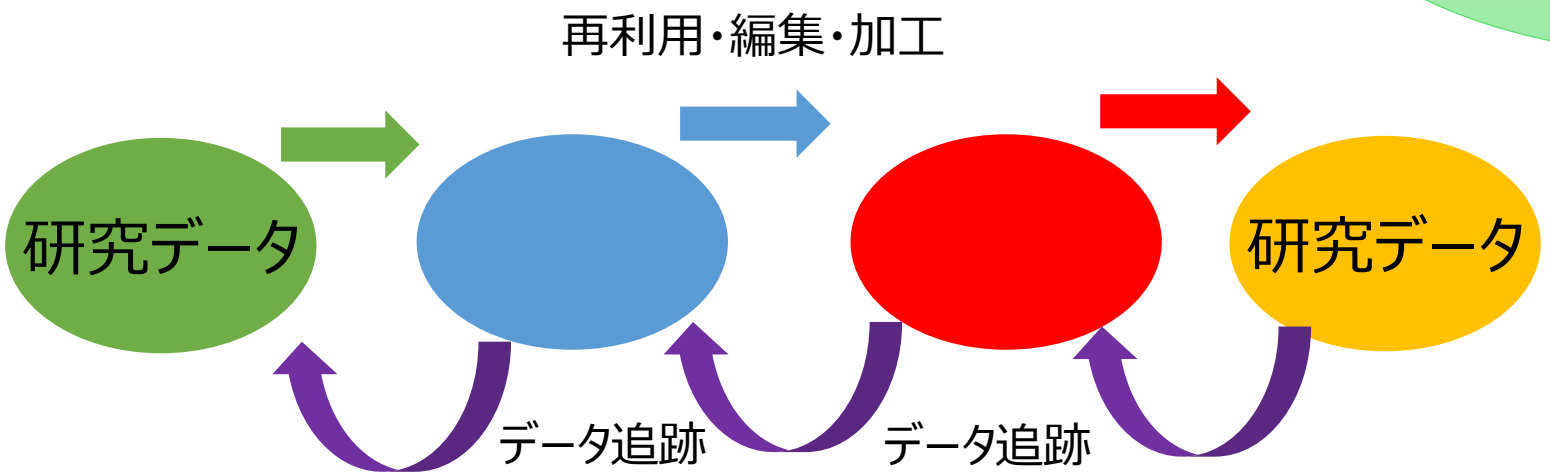
- データ収集基盤 *SINETStream* を用いて IoT 機器などからストリーミングデータを収集し、リアルタイムに分析。GakuNin RDM で保存し、共有・利活用へ



データプロビナンス機能

- 研究データの変遷に関する情報を保存し、見える化する機能
  - データ原本、実験手順、操作履歴、解析結果に関するメタデータの記録、ならびに情報抽出
  - 論文等の研究業績に関する研究データの編集履歴の記録、情報抽出

研究データの来歴を収集・保存し、研究データの信頼性・再利用性向上に役立てる



データプロビナンス機能: 活用事例

データの証跡に関連する情報を一括抽出

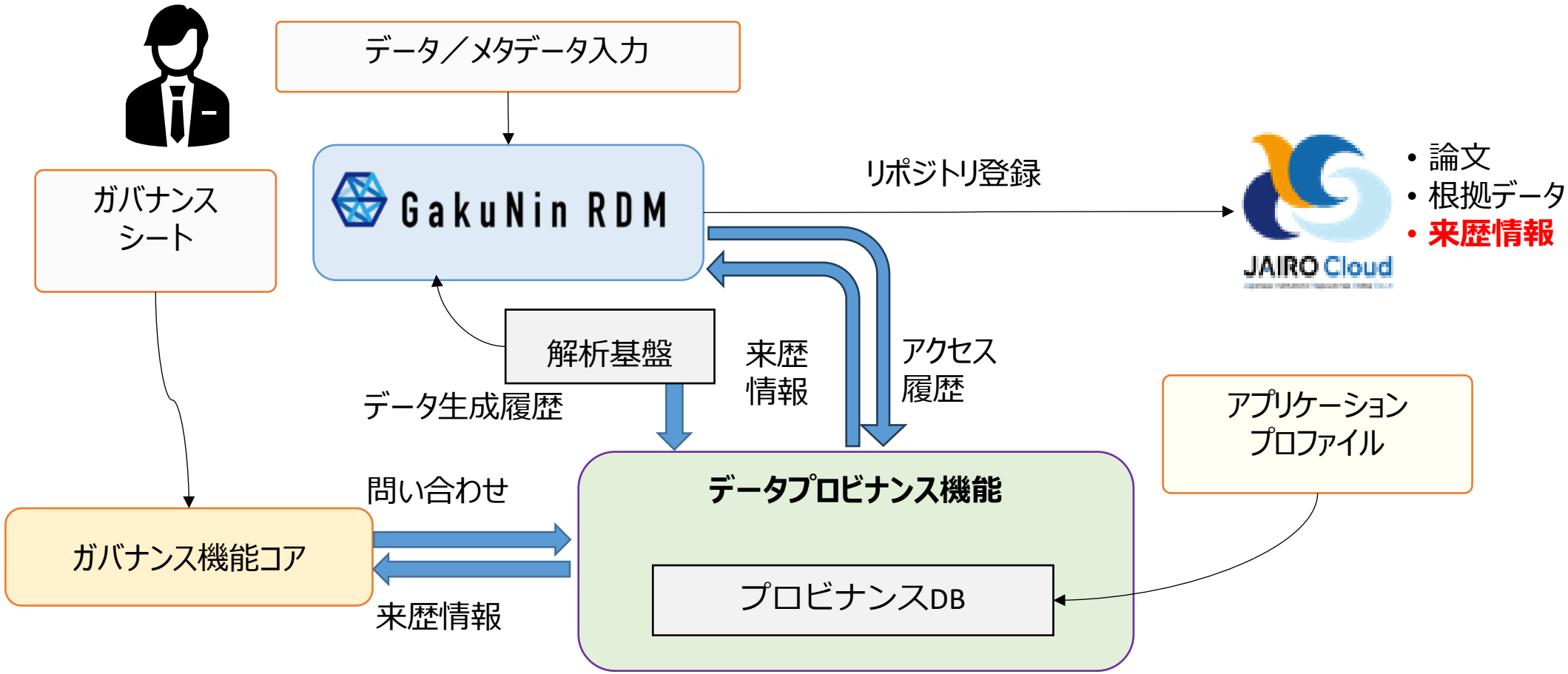
- 論文に記述されるデータの証跡に関係する情報をデータベース化
- 図・表の根拠となる情報とその結びつきを記述しJSON化して出力

論文・プレゼン資料の版管理

- 論文バージョンと用途との紐付け
- プレゼン素材の著作権・ライセンス情報の保持
- 差分箇所の抽出・マーキング・一覧

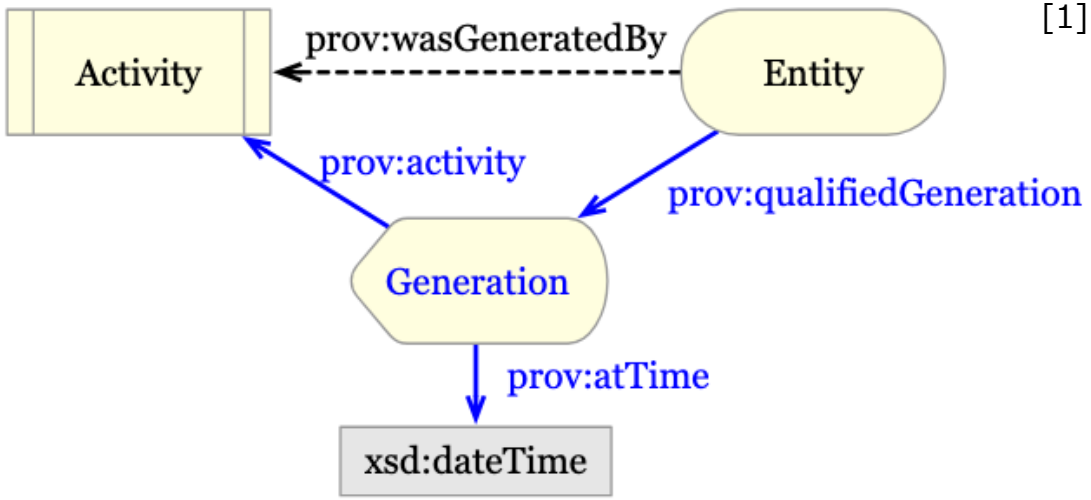
データプロビナンス機能: システム構成

- 研究データの来歴情報をNII RDC各基盤から収集しDBに保存
- 論文内の図・表に対する根拠となる来歴情報を抽出



プロビナンスのデータモデル

- W3C の PROV-JSON 規格で記述
- 入力:
  - データとデータのつながり
  - 生成方法
  - データ自身の情報（メタデータ）
- 例:
  - オープンデータセット *S* を使い
  - プログラム *P* による
  - 計算機実験 *Activity* を用いて
  - 得られた結果が *T* である
  - 計算機環境は *Entity* である
  - 実施者は *Agent* である



[1]

```
{
   "wasGeneratedBy": {
      "_:wBG1": {
         "prov:entity": "e1",
         "prov:activity": "a1",
         "prov:time": "2001-10-26T21:32:52",
         "ex:port": "p1"
      },
   },
   ...
}
```

[1] Heinrichs, B. and Politze, M. "Asynchronous Data Provenance for Research Data in a Distributed System". The 23rd International Conference on Enterprise Information Systems. vol.2, pp.361-367 (2021) DOI:10.5220/0010478003610367

まとめ

おわりに

- データ駆動科学の再現性・再利用性をどこまで追求できるか？
  - 物理レイヤから政治レイヤまで多岐にわたる問題
  - 民間企業の話であれば、経済合理性に帰結させてもよいが…
  - 公的資金を投入して実施される研究に対しては、経済合理性とは異なる価値基準が求められる
  - 人材育成や世論醸成を含む長期的な取り組みが必要

銀の弾などない

- 今日の~~講座~~放談会の趣旨
  - 社内／所内で取り組みを始めるきっかけにしていただければ嬉しい
  - NIIで一緒に取り組んでくれる人が見つかればもっと嬉しい