Vol.2020-IOT-48 No.13
2020/3/3

# データ駆動科学研究の再現性を支えるデータ分析環境再構成

横山 重俊[1,3,a)]　浜元 信州[1]　長久 勝[2]　藤原一毅[3]　政谷 好伸[3]　竹房 あつ子[3]　合田 憲人[3]

**概要**：科学技術における第 4 のパラダイムとして「データ駆動科学」とよばれる手法が各研究分野で広がっている．データ駆動科学型研究では，論文発表されたデータ分析の再現性確保に対する要請が特に強く，オープンサイエンスの言葉の元，公共データベース構築によるデータ共有，およびデータ解析ソフトウェアのオープンソース化による研究情報共有が進んでいる．しかし，研究分野の多様性や計算環境再構築技術の未成熟さから，データ分析環境の再現性確保には課題が残っている．本研究では，この課題解決のために，データ駆動型研究分野のデータ分析環境の再現性を推進するデータ分析環境再構成方式の確立を目指す．具体的には，コンテナ技術を用いたオーバーレイクラウド方式の仮想クラウド技術と実行可能な構築手順書構成技術 (LC4RI: Literate Computing for Infrastructure) の組合せで，データ分析環境を再現できる方法を共有し，各所でオンデマンドにデータ分析環境を再現する方式を提案する．

# How to Reproduce Data Intensive Research Environments

Shigetoshi Yokoyama[1,3,a)]　Nobukuni Hamamoto[1]　Masaru Nagaku[2]　Ikki Fujiwara[3]
Yoshinobu Masatani[3]　Atsuko Takefusa[3]　Kento Aida[3]

## 1. はじめに

バイオインフォマティクス分野などのデータ駆動型研究では，論文発表された計算実験の再現性確保に対する要請が強く，DNA 塩基配列の公共データベース構築によるデータ共有，およびデータ解析ソフトウェアのオープンソース化による研究情報共有が進んでいる．しかし，計算環境再構築技術の未成熟さから，実験の再現性確保には課題が残存する [1], [2]．特にオリジナル研究を実施した研究グループと異なる研究グループが異なる計算機環境で再現を試みる場合，データ分析手順の復元と共にその手順が実行される計算機環境の再現の困難さは依然残っている．研究再現を行おうとするタイミングはオリジナル研究から数年後であることも珍しく無いので時間的な隔たりは困難さをさらに強めている．

本研究ではこの課題解決のために，データ駆動型研究分野の計算実験の再現性を推進するデータビリティプラットフォームの普及を目指す．具体的には, ICT 基盤上に構築された計算実験（以後 実験）の再現性をコンテナ技術を用いたオーバーレイクラウド方式の仮想クラウド技術 [3], [4] と実行可能な構築手順書構成技術 (LC4RI: Literate Computing for Infrastructure)[5] の組合せで実現する．

本報告で計算機環境あるいは研究環境という場合，(1) ハードウェア，(2) オペレーティングシステム，(3) 研究分野毎に存在するアプリケーションソフトウェアスタックおよびワークフローエンジン，(4) データ分析ワークフロー群の 4 レイヤ全体を指すものとする．

## 2. 背景

データ駆動型研究分野の研究プロセスを単純化すると，(1) データの蓄積, (2) 実験（データ分析と発見）, (3) 論文化となり，(1) と (3) についてはオープン化が進んでいる．残る (2) についてのオープン化を進める手段を提供することが本研究の対象となる．

データビリティ (Datability) とは，Data と Ability を合

1 群馬大学
Gunma University
2 ライフマティックス株式会社
Lifematics
3 国立情報学研究所
National Institute of Informatics
a) yoko@gunma-u.ac.jp

ⓒ 2020 Information Processing Society of Japan

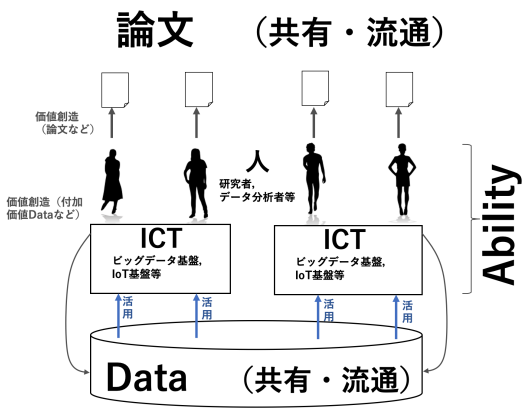


図 1　オープンデータのみの研究スタイル

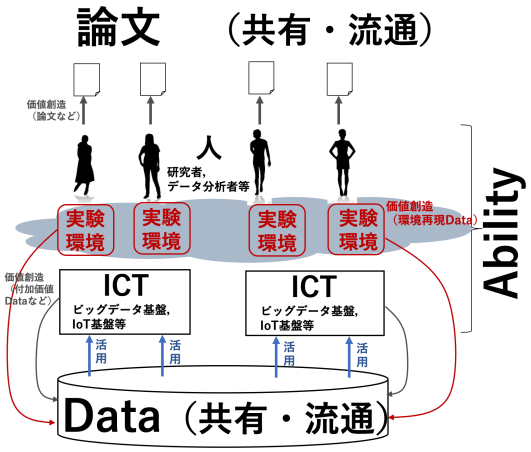


図 2　データビリティプラットフォームのある研究スタイル

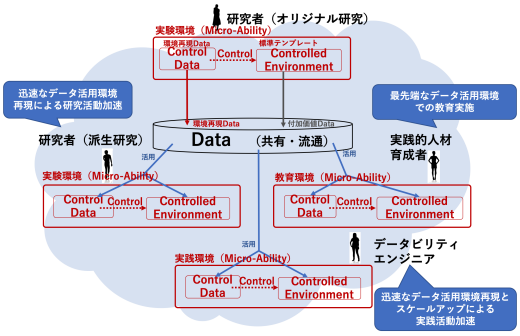


図 3　データビリティプラットフォームによる実験環境流通

わせた造語で「大規模なデータを持続可能 (Sustainability) かつ責任ある形 (Responsibility) で活用する能力」であると定義されており，データ駆動型の研究推進において鍵となる概念である [6].

オープンデータ化によるビッグデータの共有・流通に基づくデータ駆動型研究の推進やデータ駆動型研究人材の育成が，図 1「オープンデータのみの研究スタイル」に示すように進んでいる．論文とともにその際生成される中間データも付加価値 Data としてオープン化される．データビリティプラットフォームはこのオープン化の動きを促進するために不可欠な要素である．

その要素は，共有・流通しているデータを活用する計算実験の再現性向上である．データビリティの Ability 軸で言うと，アカデミッククラウドを代表とする ICT 基盤とそれを活用する人という Ability に加えて，図 2「データビリティプラットフォームのある研究スタイル」に示すように，各実験を Micro-ability として定義し，実験再現のための情報も付加価値 Data に加えて，それを共有・流通させるこのような仕組みをアカデミックコミュニティとして持つことが，データ駆動型の研究推進を成功させる鍵となる．

実験環境に使われるデータ分析基盤は多様であり，それを許容するデータビリティプラットフォームが必要となるため，この点にも留意したデータビリティプラットフォームを用意する必要がある．

そのような柔軟性を持った実験を共有・流通させるための仕組みとして，まず実験を「各分野で標準化された実験環境テンプレート（以後 実験環境テンプレート）」と「各実験を実行するための再現情報（以後 実験ノート）」より構成する．前者の実験環境は Controlled Environment であり，後者の実験ノートはそれを制御する Control Data である．論文と同時に各分野においては標準化された実験環境テンプレートが共有化され，実験ノートを共有・流通することで，実験の再現性が確保される可能性がある．Data の Sustainability や Responsibility はこの実験環境テンプレートや実験ノートについても適用されるべきである．

さらに図 3「データビリティプラットフォームによる実験環境流通」に示すように，付加価値 Data として加えられた実験再現のための情報を，研究フェーズの実験のみならず，実践的人材育成やデータエンジニアがデータ活用を実践する領域へ流通させ，データビリティプラットフォームのさらなる展開も視野に入れたい．

## 3.　関連研究

### 3.1　AiiDA データ分析環境サービス

AiiDA[7] は計算科学のための柔軟でスケーラブルなインフラストラクチャになるようにゼロから設計された洗練されたフレームワークで，各シミュレーションに必要なデータを保存でき，柔軟なプラグインインターフェースにより利用者が任意の数の HPC マシンおよびコードとシームレスに対話できる．シミュレーションの自動化のためワークフローエンジンと共にワークフロー 保存機能も持つ．

### 3.2　Chameleon Cloud データ分析環境再現サービス

NSF Cloud の一つである Chameleon Cloud 上の一つのサービスとして，Jupyter Notebook 環境を活用してこのクラウド上にデータ分析環境再現サービスを提供する研究がなされている [8].

AiiDA は All in One アプローチであり，各研究者がこ

の環境を使うことで，お互いの研究論文の再現性を達成できることを目指している．分野毎の要求や時間を経過することによるインタフェース仕様変更なども全て継続的にプラグイン機構を駆使することで吸収していく姿勢で運用を続けている．このことを前提とすれば，研究者が実施したデータ分析手順を保存するだけで，いつでもその手順に従って他の研究者が再実験を実施することができる環境を実現できる．ただし，広い範囲の研究分野について，この環境を長期に渡って運用・維持して行くことはコスト的な問題を含んでいる．それと同時にソフトウェア維持管理の技術観点からも様々な版のコンフリクトを調整しそれらを同時に実行可能とすることの困難さが今後増大する可能性がある．

一方，Chameleon Cloud 上のデータ分析環境再現サービスはオンデマンドで Chameleon Cloud 上にデータ分析環境再現するというアプローチで，分野毎の要求や時間を経過することによる仕様変更などは再現手順を記述する Jupyter Notebook の中で吸収する方式である．

本報告で述べるアプローチは，オンデマンドでクラウド上にデータ分析環境を再現するという意味では，後者に類似するものである．但し，適用する範囲を Chameleon Cloud に限定しない，分野毎のデータ分析環境に対する雛形を用意するという点などでの相違点がある．

## 4. 問題設定

再現性という言葉が曖昧なため，論文の再現性を議論する際に，まず再現性について定義する必要がある．再現性に相当する英語には Repeatability, Replicability, Reproducibility などが存在し，それらの意味についても様々な定義が存在する [9]．これら三つの言葉について，以下のように整理している例があるため，本報告ではこの定義に沿って説明を進める．

- Repeatability (Same team, Same experimental setup):
  実験は，同じチームが同じ実験手順，同じ実験システム，同じ操作条件，同じ場所での複数の試行において指定された精度で実験結果を取得できる．計算実験の場合，これは研究者が自分の計算を確実に繰り返すことができることを意味する．
- Replicability (Different team, Same experimental setup):
  異なるチームが同じ実験手順，同じ実験システム，同じ動作条件，同じ場所または異なる場所での複数の試行において指定された精度で実験結果を取得できる．計算実験の場合，これは，独立したグループがオリジナル研究者と同じ環境を使用して同じ結果を取得できることを意味する．
- Reproducibility (Different team, Different experimental setup):
  実験は，異なる場所にある異なるチーム，異なる実験システムでの複数の試行において指定された精度で実験結果を取得できる．計算実験の場合，これは，独立したグループが完全に独立して開発した環境を使用して同じ結果を得ることができることを意味する．

本報告で扱う再現性は Reproducibility (Different team, Different experimental setup) のレベルまでを目指す．このレベルが上記三つのレベルの中で最も対象範囲が広く，そのため，達成までの困難性が最も高いからである．実際，直接の関係が無い異なるチームがその違い超えて再現性を達成するということは空間的な隔たりはもちろん，時間的隔たりも存在する．本質的な問題は時間的隔たりが引き起こすと考えている．研究のサイクルが短くなったとはいえ，数年を超えた時間的を隔たりを持つチーム間で再現性を求められることも発生し，その隔たりを超える再現性達成を目標とする．同一研究所で同一チームが再現する場合でも，この時間的隔たりは本質的には異なるチーム間の再現性達成と同様の困難さを持つ．

また，何を再現するのかという観点でも，そのレベルを定めておきたい．

- Methods Reproducibility: 同じ手順を正確に繰り返すことができるように，手順とデータに関する十分詳細な情報が提供されている状態．
- Results Reproducibility: 可能な限り元の研究に近い手順で，独立した研究から同じ結果を得られる状態．
- Inferential Reproducibility: 研究の独立した複製または元の研究の再分析から同じ結論を導き出せる状態．

本報告では Methods Reproducibility を前提にして Results Reproducibility を目標とし，論文に掲載されているデータ分析結果が再現できることを再現性と定義する．なお，Inferential Reproducibility については，研究の意味的なレベルの Reproducibility であるので扱わないこととする．

従ってこのレベルの再現性を実現する目標を達成するために，まず設定しなければならないのは，実験手順と実験環境の時間軸に対する保存耐性である．数年経た後も保存されている情報を元に実験手順とその手順に従って実験が再現できる環境を構築できることを保証するという問題（Robustness）を設定する．

次に，各チームがセキュリティポリシー上の境界を保つことを求められるので、その境界を超えて再現性を達成するためには再現する側で独自に実験環境を構築することが出来なければならない，という問題設定（Anywhere）をする．

また，時間を隔てて継続的に実験環境を保持し続けることは一般的に困難なので，実験環境構築は必要性が生じる毎に行えるという問題設定（On-demand）も行う．

さらに，本報告で再現性の対象とする研究分野に特に最初から制約を設けることはしないので，研究分野毎の実験手順と実験環境の多様性を許容出来なければならないという問題設定（Variety）も必要である．

上記の問題設定の議論をオペレーティングシステム（例えば Linux）の設計と開発の際のそれとして例えて説明してみたい．

オペレーティングシステムはそこで動作していたソフトウェア（バイナリー）が時間を隔てても動作するという Robustness を達成するために ABI(Application Binary Interface)[10] についてとても神経を使って設計・開発を続けて来ている．また，Anywhere, On-demand についてはオペレーティングシステムのディストリビューションサービスがその問題解決を担って来たし，クラウドサービス内のマシンイメージ流通が解決策の洗練化を進めていると言える．

Variety についてはIoT デバイスからスーパーコンピュータに至るホスト上て動作する様々なソフトウェアを対象とする汎用性があることは実証されている．

これまでのオペレーティングシステムでの経験やノウハウを今回のデータ科学研究の再現性を支えるデータ分析環境再構成基盤の設計と開発に持ち込めるか考察する．

持ち込める部分と持ち込めない部分があり，それらを区別して議論したい．そのためにデータ分析環境再構成基盤を４つのレイヤに分ける．最下位のレイヤはハードウェア (Hardware Layer) であり，その上のレイヤはオペレーティングシステム (OS Layer) である．その上には各研究分野特有のソフトウェアスタックが乗るレイヤ（Domain Layer) がある．このレイヤは最上位で実行されるワークフローを実行できるワークフロー エンジンとワークフローエンジンから呼び出される実行エンジン（Kernel）から構成される．最上位のレイヤ (Workflow Layer) ではデータ分析ワークフローが実行される．

もし，オペーレーティングシステムの設計・開発と同様にこの上位の２レイヤの設計・開発が「優しい終身の独裁者」の存在する単一のコミュニティで一枚岩的に行えれば，これら４レイヤについてオペレーティングシステムでの経験やノウハウを持ち込むことで我々が持つデータ科学研究の再現性に対する問題は解決するであろう．

しかし，現実にはそのようにこれらのレイヤは出来て来ていない．特に Domain Layer については，このレイヤを構成するソフトウェアスタックには様々なソフトウェアコンポーネントがあり，それらは異なるコミュニティが独立に設計・運用を続けている．ここに我々が取り組まなければならない問題の本質が存在する．

## 5. 解決策

議論を進める前提として研究再現に必要なデータやソースコードなどは時間を経過しても保存され利用可能な状態が保たれているものとする．この前提で前章で設定した問題に対し，そこで述べた４つのレイヤ，特に Domain Layer についてどのようにして解決して行くのか以下に説明する．

ソースコードが提供されているのだから，例えば Makefile などを利用して Domain Layer レイヤのソフトウェアを再現できるはずであるけれど，例えばビルド手順の想定する環境と異なる環境で再現する場合には手動で修正を要することがある．また，ライブラリに依存する場合，再現時に差異が生じないようにバージョン情報などを明示する必要がある．さらに再現時にコンパイルなどの変換処理が必要となるため，使う言語処理系に依存することで元の結果との差分を生じる可能性を持つ．

この問題を解決する方策として，クラウド利用が普及することと合わせて次の二つの方策が用いられることが増えている [11]．

- マシンイメージの保存・流通
  構築し利用したソフトウェアをそのまま仮想マシンイメージとして固めて流通・利用する方法である．
- 構築手順の保存・流通
  コンテナ技術 Docker の Dockerfile のように構築手順を記述し，それを流通させたり，Ansible, Puppet, Chef などの設定自動化ツールを用いて細かなソフトウェア設定を行ったりする方法である．

第４章で設定した４つの問題（Robustness，Anywhere, On-demand, Variety）を解決するという観点からそれぞれの方策を見てみると，Anywhere, On-demand, Variety についてはどちら方式もこれらの問題を解決可能である考えられる．マシンイメージの場合，そのイメージのフォーマットのクラウドプロバイダー間での揺らぎから Anywhere についてはマシンイメージのフォーマット変換に依存するものの大きな問題ではない．

一方，時間軸で考えた Robustness でみると，全てをマシンイメージの中に固めて流通させる前者の方が強いと考えられる．すなわち後者の場合，時間経過によって構築手順で指示されている実施項目のために必要な資源が既に確保できない事態が起こりえて，この部分についてはその資源管理が分散しているため，将来の確保が保証できないという本来的な問題を持っている．

前者についても厳密に言うと過去のマシンイメージが将来確実に実行可能であるかどうかは各クラウドプロバイダの各時点での運用ポリシーに依存するので管理不能ではある．しかしながらこれら二つの方策を比較する限り，時間経過によって構築手順で指示されている実施項目のために必要な資源が既に確保できない事態など，時間経過による変化からの影響を受けない分，マシンイメージの保存・流通によりデータ分析環境の再構成を行う方向に進める方が有効な方式である．

ただ，現実にはデータ分析環境の再構成についての議論はここで終わらない．以上の単純な議論はデータ分析環境が単一ノードに収容できる比較的簡単な場合に限って有効であり，我々が扱う対象はマルチノードから構成される計

算機クラスタ上でないと実行できないデータ分析の再現までを対象とするため，さらなる考察が必要となる．

マシンイメージの保存・流通の方策の延長で複数ノードを前提に考えると，通常はいわゆる Infrastructure as Code での解決を目指すこととなる．Ansible, Puppet, Chef などの設定自動化ツールを用いてデータ分析環境構築を”自動化”するというアプローチであり，構築に必要なマシンイメージ群とこのマシンイメージ群を束ねて必要な計算機クラスタを再構築するための設定自動化ツールが理解する構築手順情報を保存・流通させることである．

この方策は空間方向の分散には有効であると考えられる．設定自動化ツールが構築手順情報をもとに問題なく自動実行できるのであれば再現者はクリック一つで必要なデータ分析環境がオンデマンドにどこでも構築・利用できることとなる．残念ながらこの方策は我々の本質的な問題である時間軸方向の Robustness に強くない．実際，Infrastructure as Code に沿ってシステムを運用管理する際にはコードの低可読性，人が読む手順書との整合性維持が困難，未知要因によるエラーの発生などの問題に対処することが求められることを体験している．どうしても自動化コードの属人化が発生してしまっているのが現状である．

時間軸方向の隔たりへの耐性を作ることの難しさが大きな壁となっている．この難しさの元凶はマシンイメージ群と設定自動化ツールを制御する Infrastructure as Code の Code の組み合わせで全てをフリーズすることが出来ないことである．単一ノードの場合は全て必要な情報はイメージの中に閉じ込めることが出来て，外界との関係性を切断することができることがデータ分析再現に向いていた．

マシンイメージ群と設定自動化ツールを制御する Infrastructure as Code の Code の組み合わせでは, Infrastructure as Code の Code の中に外部との接点を持つ部分が存在し，その部分の時間軸側の変化への追随が必要なので，単純にマシンイメージ群と設定自動化ツールを制御する Infrastructure as Code の Code の組み合わせでフリーズできたと思うのはあくまでも幻影である．実際に環境変化のためにマシンイメージ群と設定自動化ツールを制御する Infrastructure as Code の Code の組み合わせによる再構成を実施した際に障害に遭遇するとその障害対応は難しく，対処もアドホックになることが稀ではない．以上の議論を表 1「方式比較」にまとめた．

そこで我々は，コンテナ技術を用いたオーバーレイクラウド方式の仮想クラウド技術と実行可能な構築手順書構成技術 (LC4RI: Literate Computing for Infrastructure) の組合せで，データ分析環境を再現できる方法を共有し，各所でオンデマンドにータ分析環境を再現する方式を提案する．

**表 1　方式比較**

| | 単一ノード | | | マルチノード | | |
|---|---|---|---|---|---|---|
| | プラグイン | マシンイメージ | 実行可能構築手順書 | プラグイン | マシンイメージ | 実行可能構築手順書 |
| Robustness | X | ○ | ○（人の介入前提） | X | △ | ○（人の介入前提） |
| Anywhere | X | ○ | ○ | X | ○ | ○ |
| On-demand | △ | ○ | ○ | △ | ○ | ○ |
| Variety | △ | ○ | ○ | △ | ○ | ○ |

### 5.1　学認クラウドオンデマンド構築サービス

学認クラウドオンデマンド構築サービス [3], [4] は，SINET5 と学術・商用クラウドの資源を活用したアプリケー ションの構築・運用を支援する. 学認クラウドオンデマンド構築サービスでは，まず基盤ソフト ウェアにより，SINET5 の L2 VPN を介した利用機関と 1 つ以上のクラウドとのネットワーク設定支援を行い，仮想クラウドの構築準備をする. 次に，基盤ソフトウェアで用意した環境上での仮想クラウド構築を支援する．また，アプリケーションテンプレートによりアプリケーション環境構築・運用の手順書を提供し，クラウド活用を支援する．

学認クラウドオンデマンド構築サービスアプリケーションテンプレートは，研究・教育目的で広く利用されているアプリケーションの構築・運用手 順をテンプレート化したものを Jupyter Notebook 形式で提供する. これにより，クラウドでのアプリケーション構築に不慣れな利用者に対してクラウド の利活用を支援する. 現時点では，以下の研究環境に関連するアプリケーションテンプレートを含め各種アプリケーションテンプレートの開発を国立情報学研究所が中心に進めており，引き続き利用者コミュニティと協力しながら開発対象のアプリケーションテンプレートを増やしていく予定である．

- HPC テンプレート: OpenHPC で提供されている科学技術計算のための計算ライブラリ群およびキューイングシステムからなるクラスタ環境を構築と，構築したクラスタでの基本的な性能評価が行える．
- ゲノム解析テンプレート：Galaxy ワークフローツールによるゲノムデータ解析環境をクラウド上 に構築するものである．
- CoursewareHub テンプレート：Jupyter Notebook で開発した教材を実行する JupyterHub ベースの CoursewareHub [12] と呼ばれる教育実践環境をクラウド上に構築するものである．

### 5.2　Literate Computing for Infrastructure

Literate Computing for Infrastructure(LC4RI)[5] で使う意味での「Literate Computing」は，「IPython」の開発者である Fernando Perez が，2013 年以降，ブログ [13] や論文 [14] で言及しているものを指す．

これは，ドキュメントとコードを同じテキストで一元管理する「Literate Programming」[15] の考え方を，IPython のブラウザベースの Notebook スタイルインタフェースである「Jupyter Notebook」[16] でコンピュータの操作に適用しようというものである．元々は，オープンサイエンスの文脈から提唱されている．

この「Literate Computing」をシステムの構築や運用に用いる実践として，Notebook をそのまま実行可能な手順書として使える「for Reproducible Infrastructure」とは，構築手順書として正しく動く Notebook があれば，簡単に繰り返し何度でも同じシステムを再構築できるということである．また，ドキュメントとコードだけではなく，動作結果も一体として管理・比較参照可能とすることで，システム固有の事情や，システムの改変の経緯などの情報を留め，運用者間で個々の環境や作業における関心を共有可能とすることで，人が介在する柔軟な自動化を実現するということである．

４つの問題（Robustness，Anywhere, On-demand, Variety）と我々が提案するコンテナ技術を用いたオーバーレイクラウド方式の仮想クラウド技術と実行可能な構築手順書構成技術 (LC4RI: Literate Computing for Infrastructure) の組合せの関係は以下のようになっている．

Anywhere, On-demand については，コンテナ技術を用いたオーバーレイクラウド方式の仮想クラウド技術を実装し，サービス提供している学認クラウドオンデマンド構築サービスの持っている特徴を活用することで達成できる．Robustness については，上に説明したように時間軸の変化への耐性が強い方式を構築自動化や完全フリーズなどによる達成を諦めることがコアアイデアであり，それを諦めることを出発点として人の介在を積極的に支援する方式を採用するアプローチをとる．

この基本的な考え方に加えて，実行可能構築手順書の継続的インテグレーションという枠組みを提供することによる人の介在への支援も行っている．具体的な取り組みとしては例えば国立情報学研究所のトップエスイープロジェクト内のソフトウェア開発実践演習プロジェクト MicroOpsCI [17] がある．構築手順の継続試験自動化を行い，障害が発生した場合は人が逐次対応するというアプローチである．

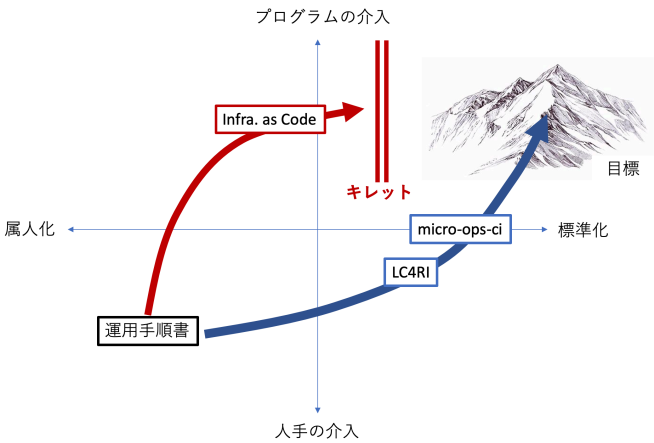


図 4　LC4RI+MicroOpsCI というアプローチ

この人の介在を前提とする方式を採用する理由は図４に示すように完全な自動化を目指すアプローチでは超えられない部分を LC4RI により迂回する新たなアプローチに可能性を見るためである．

実際，マシンイメージ群と設定自動化ツールを制御する Infrastructure as Code の Code の組み合わせで構築を実施し，時間変化により利用しているソフトウェアのディフォルト設定が変化することなどで，正常に構築が実施できなくなった場合，障害への対処に必要なエラーログの収集などは Infrastructure as Code の Code を読み解き，それから紐付いた点在する情報を元に分析，障害発見を行い，その対応を Infrastructure as Code の Code 自体の変更やコンフィグレーション情報の変更として管理して行く必要があり，さらに属人化が進む傾向にある．

このアプローチと比較して，実行可能構築手順書の継続的インテグレーションで障害を発見した場合，実行可能構築手順書により仕様・コード・実行結果を一元管理していることで，障害場所の発見容易化，保守性の向上，属人性増加の回避を実現できる．

具体的には Jupyter Notebook でと実行可能な構築手順書を構成することで属人性の削減や関連する人々の間のコミュニケーションの精度を向上させ，この実行可能な構築手順書の中でクラウド資源の確保も含めた基盤構築に関わる部分を直接リアルなクラウドプロバイダの API などを呼び出すのではなく，学認クラウドオンデマンド構築サービスが提供する仮想クラウド提供のための API などを呼び出すことにより，これら二つの技術を組み合わせたソリューションとする．

さらに，アプリケーション分野毎に典型的な構築手順書をアプリケーションテンプレートと呼ばれる形式で開発・共有することを行うことで最後の問題 Variety への対応を行う．

## 6.　研究再現実験例

本報告で述べたデータ分析環境再現方式の利用イメージを伝えるために具体的な研究再現実験例を示す．

今回は，教育実践研究分野をデータ駆動型研究の一分野とし，その分野の一つの研究を対象とした実験再現について報告する．

再現対象であるオリジナル教育実践実験は，群馬大学で実施された情報教育とした [18]．実験ノートとしては，この教育実践で用いられた教材としての Jupyter Notebook と講師が講義時に使用した実験実施用の Jupyter Notebook とする．このオリジナル教育実践研究は，国立情報学研究所のクラウド内に構築された CoursewareHub を利用して実施された．

この CoursewareHub を教育実践実験環境として，商用クラウドやプライベートクラウド上に構築するために，学

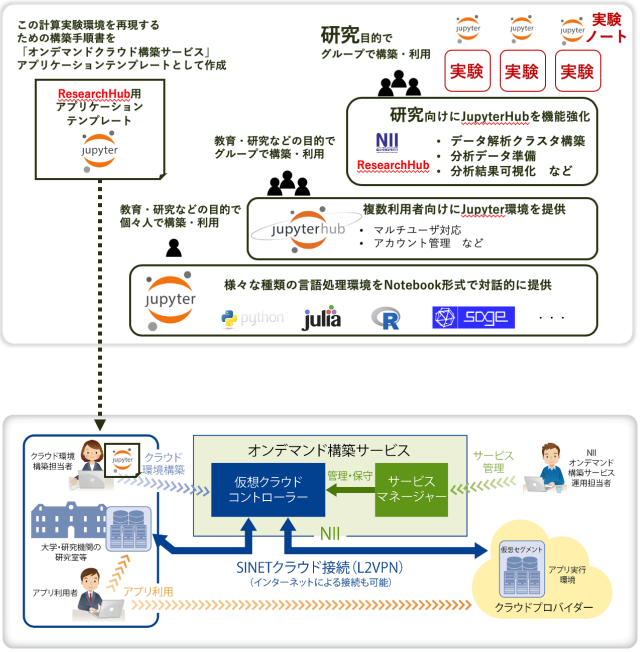


図 5　教育実践実験の再現構成

認クラウドオンデマンド構築サービスのアプリケーションテンプレートとして CoursewareHub 用のものを開発した．このアプリケーションテンプレート（以後 CoursewareHub テンプレート）を使って，各実験環境の要求条件（構築先クラウド/ノード数/マシンサイズ/ホスト名/サーバ証明書類 など）に合わせた Jupyter Notebook 実行環境をクラウド上に動的に構築することで，実験ノートを含んだ実験環境である Micro-ability を実現する．

本方式においては，実験環境構築段階で，Jupyter Notebook を LC4RI に沿って学認クラウドオンデマンド構築サービスを利用して実験環境構築を行うために活用するのと同時に，実験実施時に実験手順を記述する実験ノートのためにも利用する．これら二つの異なるレベルで Jupyter Notebook を利用することに注意していただきたい．

以上説明した実現方式で，図 5 の教育実践実験の再現例に示すように，筆者らの論文 [18] について，上記方針に沿って商用クラウド上に再現することが可能であることを確認できた．実験実施手順が記述された Jupyter Notebook（実験ノート）を github から構築した実験環境内に clone することも Jupyter Notebook で記述し実行することで，実験ノートを実験環境に持ち込んだ．教育実践実験全体の再現例を図 6（次頁）に示す．

## 7.　今後の取り組み

今後，CoursewareHub テンプレートおよび今回利用した実験を実施する手順を記述した Jupyter Notebook を github などで開示する予定であり，これらの情報と国立情報学研究所の提供する学認クラウドオンデマンド構築サービスを利用し，今回報告した実験などの再現性を体験できるような環境を整えて，本報告で述べた再構成方式の有効性の実証を進めて行きたい．

## 参考文献

[1] Feitelson, D. G.: From repeatability to reproducibility and corroboration, *ACM SIGOPS Operating Systems Review vol. 49, no.1*, pp. 3–10 (2015).

[2] Gruning, B. and et al.: Practical Computational Reproducibility in the Life Sciences, *Cell Systems volume 6, issue 6*, pp. 631–635 (2018).

[3] 竹房あつ子，佐賀一繁，丹生智也，横山重俊，合田憲人：学認クラウドオンデマンド構築サービスの推進，AXIES 2018 年度年次大会 (2018).

[4] 国立情報学研究所：学認クラウドオンデマンド構築サービス，`https://cloud.gakunin.jp/ocs` (accessed on 01-07-2020).

[5] 長久　勝，政谷好伸，谷沢智史，中川晋吾，合田憲人：Notebook を介した作業ノウハウの継承・移転を分析するための基盤，インターネットと運用技術研究会, 情報処理学会，pp. 1–6 (2019).

[6] Isak, C.: Datability, `https://techacute.com/datability/` (accessed on 01-07-2020).

[7] AiiDA: AiiDA, `http://www.aiida.net/` (accessed on 01-07-2020).

[8] Anderson, J. and Keahey, K.: A Case for Integrating Experimental Containers with Notebooks, *11th IEEE International Conference on Cloud Computing (CloudCom 2020)* (2019).

[9] Plesser, H. E.: TerminoReproducibility vs. Replicability: A Brief History of a Confusedlogy, *Front Neuroinform* (2018).

[10] 中井悦司：「Docker イメージ」のポータビリティと Linux カーネルの ABI，`https://www.school.ctc-g.co.jp/columns/nakai/nakai43.html` (accessed on 01-07-2020).

[11] 桑田喜隆，　石坂徹，横山重俊，合田憲人：研究用の計算機環境の保存と再利用に関する考察，人工知能学会　知識流通ネットワーク研究会第 18 回 (2016).

[12] 国立情報学研究所：CoursewareHub, `https://github.com/NII-cloud-operation` (accessed on 01-07-2020).

[13] Perez, F.: ”Literate computing” and computational reproducibility: IPython in the age of data-driven journalism, `http://blog.fperez.org/2013/04/literate-computingand-computational.html` (accessed on 01-07-2020).

[14] Perez, F. and et al.: Python: components for interactive and parallel computing across disciplines, *AGU Fall Meeting Abstracts* (2013).

[15] Knuth, D.: Literate programming, *The Computer Journal volume 27, issue 2*, pp. 97–111 (1984).

[16] Jupyter: Jupyter Notebook, `http://jupyter.org/` (accessed on 01-07-2020).

[17] 国立情報学研究所 トップエスイーソフトウェア実践実習：MicroOpsCI –ソフトウェア開発実践演習ポスター 30P1.pdf, `https://www.topse.jp/images/` (accessed on 01-07-2020).

[18] 横山重俊，浜元信州，政谷好伸，合田憲人：Jupyter Notebook を活用した情報教育実践，情報処理学会 情報教育シンポジウム SSS2019，pp. 2–9 (2019).

ⓒ 2020 Information Processing Society of Japan

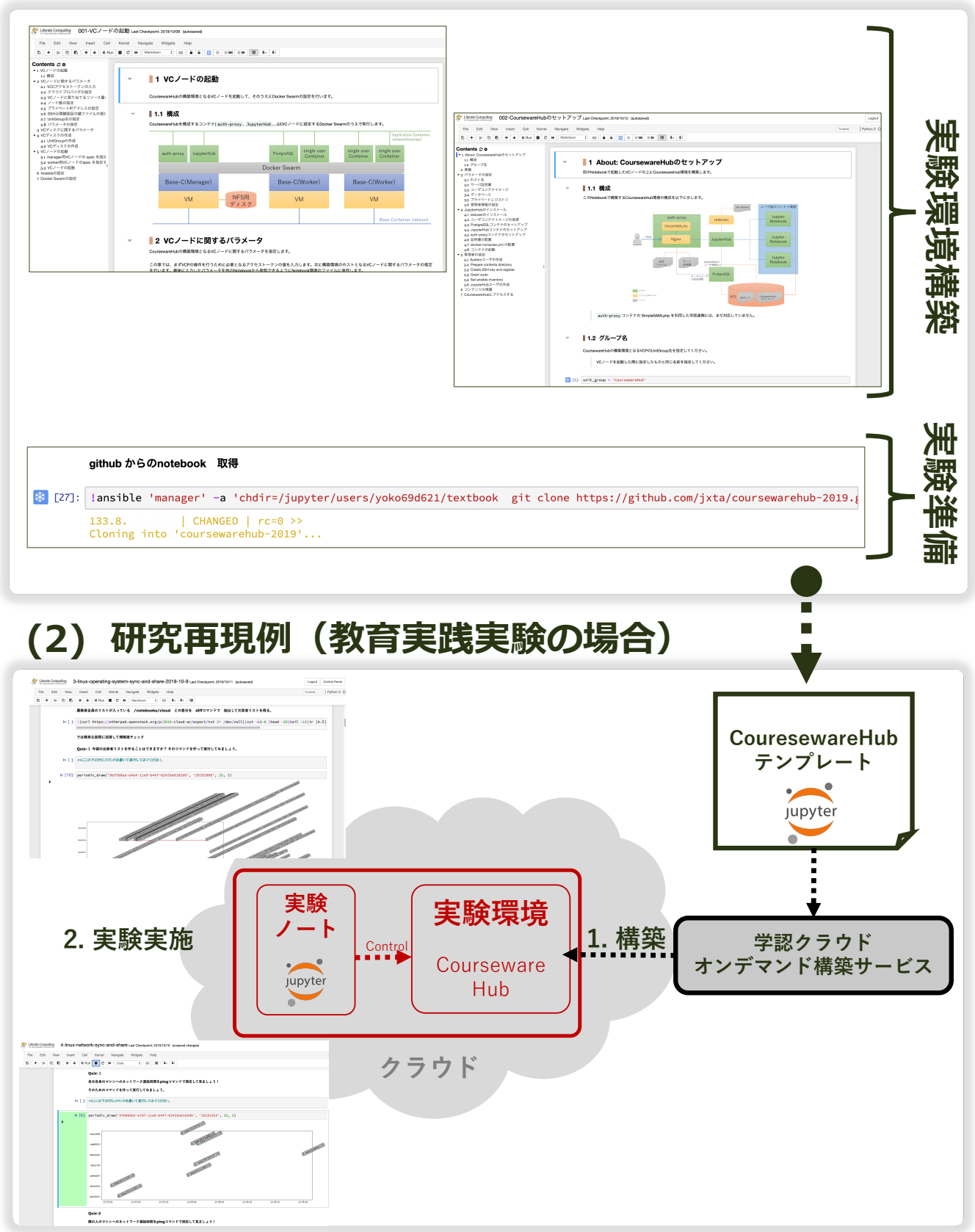


図 6　教育実践実験の再現例