〈研究資料〉

教育システムの特性とマルチベンダー体制での開発

藤井 稔也

Abstract 昨今、教育システムは複数のサブシステムで構成される大規模なものとなりつつあるため、その開発は小規模チームの手には余りマルチベンダー体制を採らざるを得なくなっている。本論文は大学における教育システムの特性とその開発について述べたものである。参加開発要員の多いプロジェクトにおいては未だシステム工学の効果は限定的であり現場の単純なノウハウやルールの実践や Web ツールの活用が結果につながり、全体の成功の鍵を握る重要なポイントとなり得る。

キーワード：インターネット、教育システム、LMS、マルチベンダー

1. はじめに

インターネット利用を主とする学校においての授業の運営には教育システム開発が不可欠となる。既存の大学においては一度に複数のシステムが導入されるケースは少なく、大抵の場合は既存システムに追加される形で導入されることの方が多い。そのような状況においてはシステムを連携するという考え方は二の次となり、結果として学生や教員は継ぎ接ぎで全く使い勝手の異なるシステムを使わせられるという不便を強いられることとなる。これはシステムの裏方である職員への作業負担も大きい。

一挙にシステムを新規で開発、もしくは刷新という幸運に巡り合わせた場合、どのような方針をとればいいのかを立案することは重要であるが、すべてが新規設計であれば上記のような不具合が自動的に解消される訳でもない。そこには基本設計段階からの仕掛けが必要であり考慮しなければならない事象は組み合わせるシステムや参加するベンダー数に比例して増えることは容易に想像できる。本論文はそのような大規模開発、マルチベンダー体制の実践と反省から記したものである。これは設計運用のひとつの手法であり、すべての教育システム開発に適用できるものではないが、問題点や設計指針が詳らかになることの意義は大きいと考える。なお、著者はソフトウェア開発のプロフェッショナルではあるがソフトウェア工学の研究者ではないことを最初に述べておく。

2. 教育システムとは

近年においては大学を問わず ICT の活用は進んでおり、逆にそれがない事例を探す方が難しい。具体的に教育システムとは何かと問われると統一的な見解があると言えないが、例えば、学籍の管理は教務システムもしくは学事システムと呼ばれるサブシステムで行われるのが通常であり、授業においてもラーニング・マネージメントシステム（LMS）の導入により学生の学習の手助けをしていることが一般的である。

インターネットで教学の全てが閉じる大学においては教務システムや LMS の導入だけでは不十分である。そこには多くのシステムが併存し、学生や教職員など含む全ての関係者のユーザ管理を受け持つシングル・サインオン・システム（SSO）を中心に据える必要がある。また全体システムの入り口となるポータル機能、学籍や学納金など管理する教務システム、LMS や e ポートフォリオ・システム、入試管理システムなどで構成されることになる(図 1)。以下の節ではそれら個々のサブシステムに関して、必要性や役目に応じて述べる。

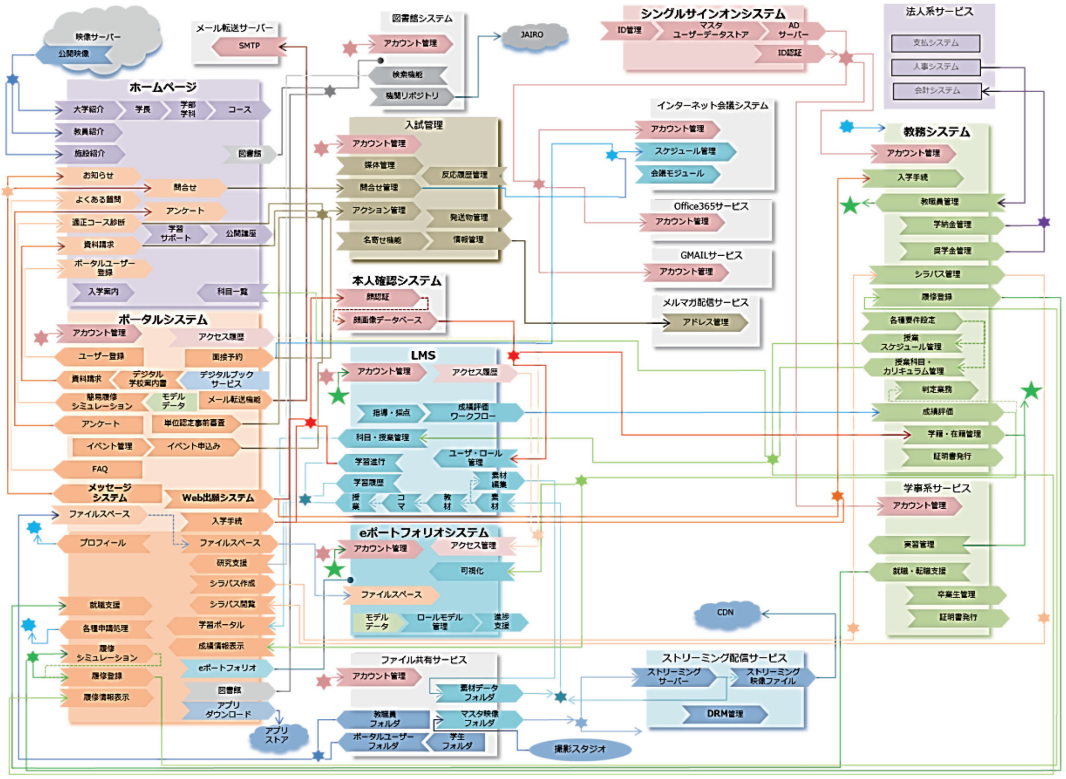

図 1：全体システム構成とサブシステムの相関[1]

2.1. シングル・サインオン・システム

サブシステム毎に利用者にユーザ ID とパスワードを管理させることは、ユーザの利便性を大きく損なうと共にパスワードの流出事故の確率が高まることも意味する。これを一部でも解決するのがシングル・サインオン機構である。シングル・サインオンを実現する方法は幾つかあるが、現実的には、大学システムにおいて学術の多くの場面で利用され外部サービスとの連携が可能なシボレス認証 [1]方式一択となるだろう。シングル・サインオン・システムは、アイデンティティ・プロバイダ（IdP）として働き、LMS をはじめとする各サブシステムはサービス・プロバイダ(SP)として登録することになる。シングル・サインオンはいくつかのクラウドサービスが存在し、それを利用することも可能であるが、採用するかどうかはセキュリティの観点から十分な議論が必要であると思われる。また、多段階・多要素認証やパスワードレス認証も初期の段階で検討すべきであろう。

2.2. ポータルシステム

ポータルとは「入り口」のことであり、大学における仮想的なキャンパスと位置付けることができるシステムである[2]。学生や教職員のみならず、学生候補者や卒業生をもユーザとして扱う必要がある。また、前述のシングル・サインオン・システムのフロントエンドにもなる。LMS は学生数や教員数から必要サーバー数などを算出することが容易である。しかしポータルシステムは ID 発行に制限はなく開かれており最大アクセス数を見積もることが難しいため、大量同時アクセス実績がある CMS[3]と呼ばれるフレームワークの選択が不可避であり、それには柔軟性があり多数の機能モジュールが用意されていることも重要な選択要素である。プロプライエタリなミドルウェアを採用することやスクラッチ開発も可能性はあるが工期やベンダー・ロックインを避ける議論が必要になるであろう。現実的には、そのような CMS として Wordpress [2]や Drupal [3]などが適当と思われるが、開発実績のあるベンダーの選定が最重要となる。また、ポータルシステムはネットからの攻撃に常に曝されるシステムであるため脆弱性への迅速な対応は必須であり運用時の緊急パッチ作業を十分に考慮する必要がある。そのためには、CMS のコア部分に改変を施すことはせずカスタマイズは拡張モジュールを作成するか既存の拡張モジュールの改変のみに留めるなどの工夫は必要である。

2.3. メッセージシステム

マルチベンダー体制で複数のシステムを構築する場合、学生や教員間のコミュニケーションの機能は各サブシステムで実装するのではなく統一する必要がある。恐らくポータルシステムにメッセージシステムを構築し、他のシステムにはその API を提供することが適当である。メッセージ機能には、ユーザ間のコミュニケーションの他、お知らせのようなブロードキャストメッセージ機能も含まれる。注意すべき点は、メッセージが多量になると利用者に高機能な機能を提供する必要がでてくることにある[4]。この場合、ポータルの機能モジュールとして実現するには工数が嵩みスケーラビリティの欠如が表面化してくる。そのような要件がある場合は、ポータルシステムはお知らせ機能のみ実装し、メッセージ機能は GMAIL に代表されるクラウドサービスを選定するなど外部メッセージシステムを導入するべきであろう[5]。

2.4. Web 出願システム

入学の手続きを担うものが Web 出願システムである。証明書等の送付をオンライン化することは現時点でもまだ難しいが、必要情報の入力や送金処理は Web 上で可能であり受験者の利便や職員の手間の省力化のため導入されることが多くポータルシステムの一機能として実現することが多い。振り込みやクレジットカード決済、コンビニ決済などを扱う決済機能や学納金管理機能、入試管理機能と連携する他に、処理の進捗度合を受験者や職員に知らせる機能や複雑なステップ制御が必要である。また他のサブシステムとのデータ連携が必須なシステムであり、ほとんどスクラッチ開発とならざるを得ず比較的開発の難しいサブシステムであるといえる。

2.5. ホームページ

ホームページの Web サーバーは教務システムや LMS を収容するデータセンター内に構築することが通常であるが、外的要因による突発的なアクセス数変動があると学生の学習活動に影響を与える可能性がある。また、このサブシステムが同じシステムに同居することは、万一、データセンターに障害が発生した際の告知手段を大学は一切失うことにもなる[6]。そのため、ホームページの Web サーバーは物理的に異なる場所や GCP 等のクラウドサービスを利用することが望ましい。この場合、ポータルシステムとの連携が難しくなるという問題が生じる。例えば、ポータルシステムに実装したお知らせ機能 API をホームページが利用する場合、Web ページのアクセス数によってポータルシステムが高負荷になる可能性やポータルシステム障害時にレスポンスが返ってこないためにホームページが表示されないなどの不具合である。この問題に対処するためにはホームページ側にリバースプロキシを置くなどの工夫が必要であろう。また、クラウドサービスに構築すると自動的にコンテンツの CDN 機能が効くというような利点も生まれるのでお勧めでもある。

2.6. LMS

ラーニングマネージメントシステム(LMS)には、大学の授業運営に必要な機能や要件を満たしたオープンソースの Moodle [4]、Sakai [5]、Canvas [6]といったシステムを選定することが多い。ただし、メディア授業を中心に据える大学においてはビデオ教材の DRM 機能や本人確認機能などカスタマイズで実現する必要がある。それが可能な開発ベンダーは数が限られており、例えば Moodle モジュールの開発実績のある十分な規模のベンダーは日本に数社もいない。そのため開発の時期と規模によっては契約すら困難である可能性がある。代案としてはシンプルな構造の LMS をスクラッチ開発し必須な機能のみを追加するという方法がある。他方、カスタマイズ対応可能なプロプライエタリな LMS も存在するのでそれらの採用も一つの手であるが、多くは企業内教育用に開発されたものであり、その工数はスクラッチ開発以上の工数になる可能性もある。大抵のベンダーはカスタマイズを文言の変更程度で済むと考えている可能性など、意識のすり合わせに苦労することも考慮すべきである。この判断は非常に難しい。

スクラッチ開発を選択する場合に必要なことは Moodle 等の構造や大学教育の仕組みを熟知した技術者の存在である。モジュール機構や多数のモジュールの存在が必要以上に複雑そうに見せているが、Moodle の根幹は教材や素材を管理するデータベース、カリキュラム科目から生成する授業を管理する機能[7]、成績管理などであり、そういった機能を学生や教員、職員といったロールに対して適切なユーザインターフェイス（UI）を表示する Web サービスでしかない。ベンダーにそれを伝える能力があればスクラッチ開発は実際のところ非現実ではない[8]。

LMS の仕様を検討する際に採用の取捨選択で悩む機能がいくつかあるが、ここでは外部教材機能[9]と学習履歴データベースを取り上げたい。外部教材には IMS Global [7]が定めた LTI がある。Moodle などオープンソースの LMS には標準的に用意されていることが多いが、スクラッチ開発やプロプライエタリ LMS の場合には、この機構を利用するために IMS

Global へのライセンス料が発生することなど注意事項がある。学習履歴データベースはラーニングレコードストア（LRS）とも呼ばれるものでありxAPI (旧 Tin Can)や Caliper などの採用により実現できる。学習ログ活用は個々の学生に最適な学習方法の提示ができる可能性があり検討に値する機能であろうが、その活用は途上といえる。

2.7. DRM および視聴制御機能

Web でビデオ教材を視聴することは標準のブラウザ機能でも可能であるが、著作権処理機能や視聴制御機能であるDRM機能を有したストリーミング再生機能が必須である。LMSに Moodle を採用した場合、標準モジュールにもサードパーティ開発のモジュールにもその機能は存在しない確率が高く、新規に開発するなどの工数が必要である [8]。また、インターネット経由のビデオ視聴は帯域を必要とするため安定した学習環境提供ためには、規模にもよるが CDN の利用が望ましい。なお、メディア授業において学生が正しくビデオを視聴していることを大学側が確認するため早送り機能の抑制ができるといったビデオプレイヤーも必要である[10]。

2.8. 本人確認システム

インターネットを利用したメディア授業においては、学生本人が受講したかどうかの証拠を得る機構が必須となる。成りすましを防止する機能であるが、その検出には生体認証技術を利用することになる。追加機器の購入が不要であり現時点で最も有力な方法を考えるとノート PC やスマートフォンに標準装備されるカメラを利用した顔認証技術の利用となろう。その利用には入学時等に確認済みの顔写真取得が大前提となる。W3C で標準化が進む WebRTC 技術[11]を利用し撮影した顔画像をサーバー上で照合する仕組みが最有力となる。現在、利用している顔認証技術[12]は 90%近い認識率に達しているとはいえリアルタイム認証を行うと条件によっては過負荷になり学習に進むことのできない学生が発生する可能性がある。学生の利便の面からビデオ視聴や単位認定試験には顔画像を複数枚取得しサーバーに蓄積するのみにし、後にバッチ処理で照合する方式の採用も考慮するべきであろう。この場合、顔認証サーバーの構成を最小限に抑えられるというメリットがあるが、照合不可件数分の職員による目視確認が必要になるというデメリットもある。

2.9. e ポートフォリオ・システム

e ポートフォリオ・システムは、授業の振り返りや学習日誌であると定義するもの、教員のメンタリング手段であると定義するもの、学習に限らない活動成果の記録と提示手段と定義するものと実に様々なものがある [9]。一例として挙げれば、オープンソースの e ポートフォリオである Mahara [10]は Moodle と連携させ授業での日々の学習日誌をつけさせることで学習者に客観的に見つめ直させるというメタ認知に基づいた学修支援システムであるが、LMS に Moodle を採用するのであれば有力な候補となろう。しかし、運用には日々の学習を学生が日誌に書かせるインセンティブをどのように学生に与えるかなど工夫は必要であり、まだ研究途上の仕組みではないかと思われる。基本的に e ポートフォリオは、

LMS で扱う授業内に留まらない入学から卒業までのスパンの学修支援機能と考えると、学生自身の履修計画の立案や履修登録、履修に関する教員による相談、単位習得した科目の成績等の閲覧機能などを最小限、提供するべきであろうかと思われる。

2.10. 教務システム

教務システムは学籍やカリキュラム、シラバスなどを管理する総合的なシステムである。カリキュラムや科目情報、学生情報など LMS にデータ連携するなどの機能を担うサブシステムである。学校運営のノウハウの詰まったシステムであるとも云えるため。実績を持つベンダーのパッケージを採用するのが無難である。しかし、教務システムは事実上、構造的にシンプルなデータベースの複合的な集合体でしかなく、大学業務に精通した技術者が居る条件であればスクラッチ開発も十分可能であり、実現すれば柔軟で先進的なシステムとなる可能性もある。

3.　基盤システム

前章で述べた教育システムは、ラックに組んだサーバーハードウェア上にインストールすることが必要である。これを基盤システムと呼ぶことが多い。大学組織においては、これらをキャンパス内の建物に設置することが多数であったが、新規の設置やリプレイス案件においては、災害対策されたデータセンターに置くことが一般的になってきている。これは、大学構内では点検での停電が不可避であり、その際のマシンの停止と再開の手順は職員の業務負担になる上に機械の寿命を縮めることなどの理由からである。また、クラウドサービスを全面的に採用することも一つの解ではある。

3.1.　データセンターとクラウドとの混成

保守や運用のトータルコストを考えると GCP や AWS のようなクラウドのマネージドサービスを全面的に採用することは一つの選択肢である。しかし前述のシングル・サインオン・システムでの ID 管理や教務情報の安全な管理での懸念から一足飛びに移行は難しい面がある。また、採用したパッケージソフトウェアによっては多くの階層のミドルウェアを必要とする可能性があるためデータセンター収容させ仮想サーバー構成とクラウドサービスを組み合わせることが現実解といえるだろう[13]。

本学の基盤システムの構成は実績のある機器やコンピュータハードウェアの選定に留めている。仮想技術はノンストップでの機器保守のためのハイパーバイザであり実績もある VMware ESXi の採用のみであり、ファイヤーウォールやバランサのコストダウンに繋がるネットワーク仮想化技術などには手をつけていない。IPv6 技術も実用段階であるが採用は見送った。要するにベンダーがその技術の扱いに習熟しているかという点によってはまだ採用に慎重である必要があり機が熟すのは少し先の将来であろうかと考えた結果である。今後の課題となるが、機器の故障や能力増強の保守コスト削減のため思い切ったクラウド移行やネットワーク仮想化、IPv6 への移行などは断続的に進めていくべきと考えている。

3.2. データバックアップ

まず、データセンターは災害に強い立地にあることもありデータは基本的に安全といえる。また、前述の仮想サーバー基盤によりハードウェアの故障があったとしてもノンストップで修理や保守が可能になっている。データ実体は多重化されたストレージアレイ上にあり万全といえるが定期的な仮想サーバーの実行イメージのバックアップも取っている。万が一ではあるが、データセンターが壊滅したとしても立地の異なる施設に教務情報のバックアップを定期で実行している[14]。

4. マルチベンダーによる開発体制

昨今、大規模なシステム開発においてマルチベンダー体制による開発は不可避になっている。進捗管理を含めてシステムベンダーに委託することは一般的であるが、異なる会社の業務管理を束ね、恙無く遂行できるスキルを持つプロジェクト管理者を探すことは至難の業でもある。また、マルチベンダー体制を採った場合にそれぞれのベンダーの質が一定でないことに起因する問題は悩ましく同質に育てることも不可能に近い。結論として全てを解決する「銀の弾丸」[15] はないということにはなるが、寄る辺となる方法論は持っておきたいところである。

4.1. ソフトウェア工学

ソフトウェア開発を学問的に扱うのがソフトウェア工学である。誕生して半世紀以上たつ分野であるが、実践者にとって必要不可欠なものには達していない印象がある。これはその分野の研究者に於いても同様で工学として確立したものではないという自嘲的な記述[11]もみられるように。大規模なソフトウェア開発において方法論は未だ体系化されておらず現場では試行錯誤が続けられている状況のように見える。

一つの試みとして定量的アプローチによる科学的マネジメントの普及を目的とし複数のベンダーからデータを収集し公開 [12]されていることは注目に値するが、ソフトウェア開発の何かに結び付く成果には達していないようである。また、プロジェクトの失敗の多くの原因は要求・要件定義のステージにあるというコンセンサスは多くの研究者や実践者にあり、それが要求工学 [13]という分野を生み出し活発な研究が行われているが、マルチベンダーによる開発管理に役立つようには見えない。

近年、ウォーターフォールモデルに対するアンチテーゼとして提唱されたソフトウェア開発手法にアジャイルがある。迅速かつ適応的であり反復を基本としエクストリーム・プログラミング [14]に代表される幾つかの手法がある。しかし参加する技術者全てにその知識と高い意欲が要求されるため大規模システム開発になればなるほどスキルを有する技術者を集めることへの困難さが増す。

4.2. データ連携

データ連携はシステム間のデータのやり取りのことである。例えば、サブシステムで必要

な形式を好きに決めさせるなど、データ連携の取り決めがトップダウンで決定されていないとプロジェクトが立ち往生し手戻りが多数発生しがちとなる。ポイントは幾つかあると考えられる。標準規格をできる限り採用する[16]こと、最新のテクノロジーを追うのではなく平易で枯れた技術[17]を採用すること、そして初期に仕様書を纏め全てのベンダーの技術者に提示することなどであろうか。

また、データ連携で忘れてはならないのは利用できるデータ形式、命名則やデータの寿命、転送プロトコルなどを厳密に定義しておくことであり[18]、勝手なルールをサブシステム内であっても使わないという申し合わせである。教育システムにおいてはサブシステム間で共通に使用する学籍番号や教職員番号、受験番号、科目ナンバーなど全ての管理番号にチェックデジットを含ませた 9 桁程度の固定桁番号に統一するなど施したが、このような単純化も重要であろう[19]。その上で、異種間連携のためのミドルウェアである EAI や RPA を導入することが出来れば、問題が起こったとしても二重三重の保険となり安心感が増す。

4.3. 進捗管理システム

従来のプロジェクト管理では、進捗は会議を催して意識合わせをすることが一般的であるが、ベンダー数や参加人員が多くなるとこの方法では合意形成が急激に困難になる。プロジェクト管理責任者が WBS を作成し管理することも進捗が進むにつれて計画との乖離が生じれば簡単に破綻する。そういった場合にプロジェクト管理責任者が問題を隠蔽に走れば間違いなくデスマーチが発生するであろう。プロジェクトの進捗管理には簡易的には Excel シートを利用することや Microsoft Project のような PC のアプリケーションを使うことで済ますことも多いが、多人数の係わる進捗管理はプロジェクト参加者への「見える化」(Observability)が必須であり、その用途には Redmine に類する Web ベースのプロジェクト管理システムが最適である。Redmine はタスクの階層化など厳密に利用すれば PMBOK を網羅しガントチャート化して進捗を一覧することも可能である。そのように厳密に管理することも一つの方法であるが、単純に仕様や問題、バグ発生に対して一つのチケットを起票し解決していく単純なスタイルでも目的は十分に果たすことが可能であり、より好ましい場合もある[20]。

5. おわりに

本論文は、2018 年 3 月にシステム開発が完了し開学に結びついた知見を書き記したものである。全体の開発を通して、初動時の体制づくりと明確なルール作りがマルチベンダー開発体制では最も重要であることを個人的には再認識した。大規模開発ではアジャイル開発は限定的にしか機能しないことや、最新のテクノロジーの安易な採用が足枷にしかならない事例も目の当たりにした。見方によっては、それはソフトウェア工学発展の余地といえるのかもしれない。

注

[1] システム設計時に作成したものを加筆修正したものである。本論文で記述するサブシステムは太字で表した。線はデータの流れを表しているが厳密なものではない。
[2] 東京通信大学では、その意味でサブドメインに campus を使用している。なお、LMS は仮想的な教室という意味で Class room の room を取って使用している
[3] Contents Management System の略
[4] 百件前後のメッセージの管理はフラットで問題は生じないが、千の桁を超えると複数のフォルダ管理や検索機能などが必要になるなどの機能の提供である
[5] 連携にはユーザ ID の他、平文でのパスワード転送要求するクラウドサービスもあるため、セキュリティの観点から選定は慎重に行うべきである
[6] Twitter や Facebook 等の外部 SNS に公式アカウントを作成することも一つの対処法であり代替手段は複数用意すべきであろう。また、メールシステムもデータセンター収容ではなくクラウドサービスに用意することで障害時に堅牢となる
[7] カリキュラムの概念である「科目」と実際に行われる「授業」の関係性や違いを殆どの教職員は認識していないことが多いが、LMS の設計においては非常に重要な概念である
[8] 東京通信大学の LMS は記述言語に Python を利用した、ほぼスクラッチ開発のシステムであるが、大きな支障なく稼働している
[9] 外部教材利用の仕組みで本稿に取り上げないかったものに SCORM がある。これはレンダリング・エンジンを LMS に組み込む必要があり工数が掛かる。教材資産を SCORM で既に保持している場合でなければ実装する価値は薄いように思われる
[10] 必須ではないが、中断した位置を覚える機能や二度目以降の視聴には早送りや倍速機能が現れるなどが学生の利便のため実装されるとよいと思われる。また視聴開始時間や終了時間のログが記録されることからブラウザのアドオンなどによる不正も検知可能である
[11] PC でカメラ機能を利用するには以前は Adobe Flash を採用することが一般的であったが、現在では、PC およびスマートフォンで利用可能になりつつある WebRTC が最適である
[12] 顔認識技術に採用した NeoFace SDK は NEC の提供するパッケージであったが販売停止している。現在はクラウドサービスである NeoFace Cloud が API を提供する同等の製品となるようである
[13] これは開発開始の時点での判断を表明したものであるが、本論文の執筆時においてもまだ正しいと思われる
[14] 教務データが存在すれば、全てのシステムやデータを失ったとしても、大学の復興が可能である
[15] フレデリック・ブルックスが彼の著書「人月の神話」で使用した表現であるが、この著書は正にこうしたプロジェクト管理を述べた本であり大いに参考になる
[16] 個別に用意せずとも優れたドキュメントが揃っており、それを実現するオープンソースのライブラリソフトウェアも豊富に存在するという利点がある
[17] 例えば、Web サービスのデータ規約としてより新しい規格である GraphQL ではなく REST のような旧い技術を採用するというようなことである。任天堂でゲームボーイを開発した横井軍平は「枯れた技術の水平思考」という哲学と方法論を実践しているが、それに通ずることではないかと考える
[18] 共通に利用する語彙の共通化や、データの桁数、文字列型で扱うか数値化したコードマスタとして扱うかは重要なポイントである
[19] 人間が扱う番号はチェックデジットを設けるべきであり、ショルダーハックを防ぐ意味で 7 桁以上が必要だが、無暗に長いと本人が覚えられない
[20] 参考情報であるが、東京通信大学の教育システム開発において使用した Redmine では運用開始から開学時点の 1 年間で 1 万件弱のチケットが起票されたが、管理的に破綻することなく運用されている。

参考文献

[1] “Sibboleth,” [オンライン]. Available: https://www.internet2.edu/products-services/trust-identity/shibboleth/. [アクセス日: 29 10 2019].

[2] “Wordpress,” [オンライン]. Available: https://wordpress.org/. [アクセス日: 29 10 2019].

[3] “Drupal,” [オンライン]. Available: https://www.drupal.org/. [アクセス日: 29 10 2019].

[4] “Moodle,” [オンライン]. Available: https://moodle.org/. [アクセス日: 29 10 2019].

[5] “Sakai LMS,” [オンライン]. Available: https://www.sakailms.org/. [アクセス日: 29 10 2019].

[6] “Canvas,” [オンライン]. Available: https://www.instructure.com/canvas/. [アクセス日: 29 10 2019].

[7] “IMS Global,” [オンライン]. Available: https://www.imsglobal.org/. [アクセス日: 29 10 2019].

[8] 藤井稔也、篠原信夫, “複数にまたがるシステムで学習行動履歴を統合するログ管理システム,” 情報システム学会, 2012.

[9] 日本教育工学会, 教育分野における e ポートフォリオ, ミネルヴァ書房, 2017.

[10] “Mahara,” [オンライン]. Available: https://mahara.org/. [アクセス日: 29 10 2019].

[11] 丸山勝久, “日本のソフトウェア工学の今と未来,” *情報処理,* 第 巻 49, 第 7, pp. 793-798, 2008.

[12] 情報処理推進機構　社会基盤センター, ソフトウェア開発 データ白書 2018-2019, 情報処理推進機構, 2019.

[13] 情報サービス産業協会(JISA), 要求工学知識体系 第 1 版, 近代科学社, 2011.

[14] ケント ベック, エクストリーム・プログラミング, オーム社, 2015.

藤井 稔也（ふじい としや）　東京通信大学 情報マネジメント学部 准教授