Vol.2017-CLE-21 No.20
2017/3/22

Moodle を対象とした xAPI のプロファイルの策定

森本容介[†1]　古川雅子[†2]　山地一禎[†2]

概要：Moodle を対象とした xAPI のプロファイルを策定した．その際，Moodle が持つ学習履歴データの意味を変えずに，できるだけ多くの情報を表現することを要件とした．まず，Moodle の学習履歴とその記録方法を分析し，ステートメント生成の枠組みを決定した．次に，各ステートメントの記述内容を設計した．本稿では，ステートメントの主要な構成要素である actor，verb，object，result，context のそれぞれについて設計を述べる．また，既存のプロファイルとの比較などを行い，本研究における設計を考察した．

キーワード：Experience API，xAPI，Moodle，学習履歴，学習解析

Creation of a Moodle Profile of xAPI

YOSUKE MORIMOTO[†1]　MASAKO FURUKAWA[†2]
KAZUTSUNA YAMAJI[†2]

1. はじめに

1.1 学習解析と xAPI

様々なシステム上に，ICT を活用した学習のログデータが蓄積されている．学習管理システムには，最終的な学習成績や学習成果物だけではなく，テストへの解答内容，電子掲示板への投稿内容，システム上での行動履歴（アクセスログ）などが記録されている．行動主義から社会的構成主義に至る学習観の変化，あるいは学習に用いる端末の多様化や情報技術の進化などに伴い，学習管理システムを用いない学習形態も増えている．ソーシャルメディアを使った活動の記録や，電子書籍のページめくりの記録なども，学習を記録したログデータであるといえる．本研究では，ログとして記録されないものや記録前のものも含め，これらのデータを学習履歴と表現する．学習履歴を収集，分析し，有用な知見の発見や，学習者の将来予測などに役立てる学習解析（learning analytics）が注目され，研究が進んでいる．学習解析の結果に基づき，コースの評価，改善や，学習の個別化，適切な指導介入などを行うことができる[1][2].

多様なシステムの学習履歴を統合して学習解析を行うためには，学習履歴の相互運用性を確保する必要があり，そのような目的に標準規格が有用である．ここでの相互運用性とは，異なるシステム間でのデータの表現形式や意味の統一を指している．標準規格に従うことにより，既存のツールや手法を使用でき，効率的な学習解析が行える．さらに，組織を超えた連携の実現や，データの長寿命化も期待できる[3].

学習履歴に関する標準規格の 1 つとして，Experience API[4][5]（以下，“xAPI”）があげられる．xAPI は，JSON で記述されたステートメント（statement）と呼ばれる構造で学習履歴を表現する．ステートメントは，英語の“I did this.”に相当する構造を持っており，誰が（I / <actor>），何を（this / <object>），どうしたか（did / <verb>）を表現する．さらに，学習の結果を表す<result>，学習文脈を表す<context>を加えて，

> <actor (learner)> <verb> <object>,
> with <result>, in <context>

という形式で学習履歴を表現できる[4]．ステートメントは xAPI における学習履歴の記録単位であり，learning record store（以下，“LRS”）に格納される．

xAPI ではステートメントが従うべき規則を定めているが，ステートメントの内容，つまり学習履歴の粒度や使用語彙は規定していない．それらの設計は利用者（community of practice）に任されている．設計した内容はプロファイル（profile）やレシピ（recipe）と呼ばれる．プロファイルとレシピは異なる概念とされているが[6]，本稿ではまとめてプロファイルと表現する．相互運用性の向上や実践の効率化を考えると，利用者独自のプロファイルを作成するのではなく，広く受け入れられたプロファイルを用いることが望ましい．しかし，そのようなプロファイルの作成には，実践の共有やそれに基づく改良が欠かせない[3][5]．これまでの xAPI を用いた実践は，本来の目的である学習の分析に重点が置かれたものが多い．次章で述べるように，プロファイルについての検討も行われているが，十分とはいえない．

†1 放送大学
The Open University of Japan
†2 国立情報学研究所
National Institute of Informatics

1.2 Moodle への xAPI の適用

山田は，MOOC のプラットフォームである OUJ MOOC において，留学生向けの日本語学習科目である Nihongo Starter を提供している[7][8]．OUJ MOOC は，複数のサービス，システムを組み合わせて構成していることが特徴である．受講登録には Facebook，教材には電子書籍と Web ページ，ディスカッションには Facebook と Moodle，小テストの受験やバッジ（Open Badges）の確認には Moodle を用いる．つまり，OUJ MOOC の学習履歴は，複数の場所に異なるデータ形式で保存される．著者らは，本科目の学習解析を行うために，学習履歴の収集を行った[9]．学習履歴の表現形式として xAPI を採用し，ステートメントを集約するために，xAPI に対応したオープンソースソフトウェアの LRS を使用した．

本実践では，Moodle の学習履歴を xAPI のステートメントで表現するためのプロファイルを策定した．このとき，verb の語彙，activity の表現，学習セッションの扱いなどに対する考慮が必要であった．本研究では，Moodle を対象とした xAPI のプロファイルの策定を通して明らかになった課題を整理し，考察する．なお，本稿は，最新の規格や状況を基に，[9]にさらに検討を加え，一部を見直した結果をまとめたものである．そのため，実際に開発したシステムと本稿での設計内容には差異がある．

2. 関連する実践，研究

本章では，xAPI の語彙（verb や activity type），プロファイル，および，Moodle における xAPI の使用に関する実践や研究について述べる．

“The Registry”（The Tin Can API Registry）[10]では，xAPI の語彙やプロファイルを収集し，公開している．本稿執筆時点で，179 の verb，108 の activity type が公開されている．また，Open Badges の取得や，ビデオの再生に関するものなど，12 のプロファイルが公開されている．

xAPI の策定を主導している ADL でも，xAPI の仕様とは別に語彙を公開している[11]．ここでは，30 の verb と 15 の activity type が公開されているが，そのうち 25 の verb と 11 の activity type は，The Registry にも登録されている．また，ADL は，語彙の意味づけや再利用に関する検討も進めている[6]．

Bakharia ら[12]は，xAPI を用いて学習解析を行った経験から，ステートメントの設計について考察を行った．特にソーシャルメディアを使った活動について，ステートメントに含めるべき項目，いいね（like）や共有（share）の表現方法，などについての“ベストプラクティス”を示している．また，xAPI が JSON を採用したことにより，強い型付けが行えないことやプロファイルが機械可読ではないことを指摘し，JSON-LD を使うことを提案している．

Berg ら[13]は，広範な学習に対応したプロファイルを公開するとともに，権威づけられた（authoritative）プロファイルを提供する必要性を指摘している．

xAPI のプロファイルである cmi5[14]は，学習管理システムから起動される教材（AU / assignable unit）に関する規格であり，SCORM に代わる規格として注目を集めている．cmi5 は，学習管理システムやクライアント（Web ブラウザなど）が LRS と通信する手順やステートメントを定めている．一方，SCORM を対象としたプロファイルも，ADL によって策定されている[15].

Moodle のプラグインである logstore_xapi[16]を用いると，Moodle のログを xAPI のステートメントとして LRS に送信できる．このプラグインでは，Moodle 用のプロファイルを作って使用しているといえる．本研究も，Moodle 用のプロファイルを策定することが目的である．logstore_xapi は主要なイベントに対応するステートメントを生成するが，Moodle における学習行動を広くカバーするものではない．

3. プロファイル策定の要件と全体の設計

3.1 要件

本研究では，Nihongo Starter の学習解析に特化せず，できるだけ汎用的に使用できる xAPI のプロファイル策定を目指す. そのため, xAPI の規格から逸脱した設計は行わず，規格で推奨（SHOULD）や非推奨（SHOULD NOT）とされている項目にはできるだけ従うことを前提として，次の 3 点を要件とする．

要件 1:
学習履歴をステートメントで表現するとき, Moodle の学習履歴が持つ本来の意味を保持する．

要件 2:
ステートメントを作る時点では，学習履歴の集約（要約）は行わない．

要件 3:
学習履歴に関する情報をできるだけ保持する．

要件 2 はステートメントの粒度を小さくすること，要件 3 はステートメントに学習履歴に関する情報を多く含めることを意味している．

3.2 Moodle における学習と学習履歴

Moodle では, 主にリソースと活動を用いて学習する[17]. “活動”の原語は activity であるが，xAPI にも activity という概念がある. 混乱を避けるため, Moodle の activity は“活動”と表記し，xAPI の activity は原語を用いる．リソースとは，学習者が閲覧する静的な教材である．リソースの例として，PDF や PowerPoint プレゼンテーションなどのファイル，HTML 文書，Web ページへのリンクがあげられる．活動とは，学習者による情報入力を伴う双方向型の教材やツールである．活動の例として，SCORM パッケージや小テストなどの教材，チャットやフォーラム（電子掲示板）などのツールがあげられる．これらのリソースと活動はコ

ースに配置される．学習者は，受講登録したコースに配置されたリソースや活動を選択して，学習する．

Moodle では，多くの機能がプラグインとして実装されている．活動もすべてがプラグインである．活動のインスタンス（コースに設置した個別の活動）のメタデータや，活動を用いて学習した履歴としてのデータは，プラグイン固有のテーブルに記録される．たとえば，小テストの得点は小テストプラグインである mod_quiz，フォーラムへの投稿内容はフォーラムプラグインである mod_forum が持つテーブルに記録される．これらに加えて，Moodle は，以下のような方法で統一的にログを記録する仕組みを有する[18].

Moodle 上で何らかの行動を行うとイベントが発生する．Moodle には，プラグインとしてログストアマネージャ（tool_log）が標準添付されており，tool_log がイベントを検出する．tool_log は，サブプラグイン（tool_log に対するプラグイン）として，ログストアを持つ．イベントを検出した tool_log は，有効化されたログストアにイベントを転送し，ログストアがそれぞれに実装された方法でログを記録する．Moodle 3.2 には，“標準ログ”（logstore_standard），“レガシーログ”（logstore_legacy），“外部データベースログ”（logstore_database）の 3 種類のログストアが標準添付されており，そのうち logstore_standard のみが初期状態で有効化されている．

表 1　Moodle のイベントが持つプロパティの一部
（[9]の表 1 を一部修正して再掲）

プロパティ名	意味
eventname	イベント名
component	イベントを定義している（イベントを発生させる）コンポーネント．“core”（Moodle のコア）や“mod_quiz”（小テスト）など.
action	イベントを発生させた行動．“created”（作成した）や“viewed”（閲覧した）など.
target	action の対象．“blog_comment”（ブログのコメント）や“attempt_summary”（小テストの受験概要）など.
edulevel	教育レベル．“Teaching”，“Participating”，“Other”のいずれか.

Moodle のイベントが持つ情報のうち，主要なものを表 1 に示す．eventname の書式は，“¥*component*¥event¥*target*_*action*”である．たとえば，“¥core¥event¥blog_comment_created”はブログにコメントをした，“¥mod_quiz¥event¥attempt_summary_viewed”は小テストの受験概要を閲覧した，という意味を持つイベントである．logstore_standard は，発生したイベントを，ユーザ ID，コース ID，発生時刻，IP アドレスなどとともに，データベースに記録する．前章で述べた logstore_xapi は，ログストアである．logstore_xapi は，発生したイベントに対応するステートメントを生成し，LRS に送信する．

3.3 ステートメント生成の枠組み

学習者が Moodle 上で行う行動はすべて学習履歴といえる．本研究では，これらの学習履歴を xAPI のステートメントとして表現する方法を設計する．

3.1 で述べた要件 2 のためには，学習者が行動するたびにステートメントを生成すればよい．Moodle 上での行動に伴ってイベントが発生することから，イベントの発生タイミングでステートメントを生成することが自然である．logstore_xapi もイベントを契機にステートメントを発行している．

Moodle で発生するイベントのすべてが学習に関するものではない．たとえば，管理者や教師が行う作業である，コースを分類するためのカテゴリの作成（¥core¥event¥course_category_created）や，コースへの活動の追加（¥core¥event¥course_module_created）などは，学習活動とはいえない．そこで，イベント 1 つ 1 つについて学習活動であるか否かを判断し，学習活動であれば，対応するステートメントを設計する．ここで，イベントの edulevel（表 1 参照）が“Teaching”であれば教授行動，“Participating”であれば学習行動，“Other”であればその他であることから，学習履歴を記録するためには edulevel が“Participating”のイベントを選択すればよいと考えられる．しかし実際は，edulevel が“Participating”であっても学習活動ではないと考えられるイベントや，“Participating”でなくても学習活動であると考えられるイベントが存在する．前者の例として，教師が用語集のエントリを承認したときの“¥mod_glossary¥event¥entry_approved”があげられる．後者の例として，小テスト受験をレビュー（以前に受験した小テストへの解答や正誤を閲覧）したときの“¥mod_quiz¥event¥attempt_reviewed”（edulevel は“Teaching”）や，Moodle にログインしたときの“¥core¥event¥user_loggedin”（edulevel は“Other”）があげられる．

Moodle 3.2.1 の標準配布物では 433 のイベントが定義されており，edulevel が“Participating”のイベントは 145 である．そのうち，3 つのイベントは学習活動とはいえない．残りの 288 のイベントは，学習活動か否かの判断が難しいものや，リリースされて間もない Moodle の新機能に関連したものも多く，本研究でも検討が不十分である．さしあたり，edulevel が“Participating”のイベントから 3 つを除外し，“Participating”以外のイベントから 4 つを加えた，計 146 のイベントを対象に検討することとした．

これらのイベントに対して，生成するステートメントを設計する．つまり，actor，verb，object，result，context の表現方法を決める．このとき，要件 3 を満たすために，プラグイン固有のテーブルなども参照する．なお，1 つのイベントに対して 1 つのステートメントを生成することが多いが，そうではないものもある．

4. ステートメントの設計

本章では，本研究におけるステートメントの設計について述べる．ページ数の制約により，細かい部分や記述例を示すことができないため，設計の要点のみを示す．また，適宜，他の実装例などを参照しながら考察を行う．

4.1 actor

actor の表現方法は，すべてのステートメントで同一である．xAPI では，学習者の識別子として，mailto スキームの IRI，その SHA-1 ハッシュ，OpenID（URI），何らかのシステムのアカウント，の 4 つのうちの 1 つを用いる．本設計では，Moodle のアカウントを用いることにした．システムのアカウントを用いる場合は，システムのホームページとシステム内での識別子の組で actor を表現する．そこで，システムのホームページとして Moodle のフロントページの URL，システム内での識別子としてユーザ ID を用いる．

ここで，actor の表現は，プロファイルの設計において重要ではない．Moodle のユーザは必ずメールアドレスを持っているため，mailto スキームの IRI を使うこともできる．システムをまたぐ学習解析を行う場合は，その方が都合がよいと考えられる．個人情報に関する方針などに応じて適宜選択すればよい．

なお，異なる actor が同一人物であるかの判定は，xAPI のスコープ外である．CLA toolkit[19]は，複数のシステム（SNS やブログ）上での活動履歴を xAPI で収集する．CLA toolkit では利用者に各システムのユーザ ID を入力させており，ステートメントを生成する時点で，同一人物のステートメントには同じ actor（mailto スキームの IRI）を用いることができる．logstore_xapi は，本設計と同様，Moodle のアカウントを用いている．

4.2 verb

verb は，同種の行動を結びつけ，異種の行動を区別する役割を持つ．そのためには，同じ意味を持つ行動には同じ verb を用いることが必要である．たとえば，CLA toolkit では，Facebook のシェアと Twitter のリツイートに同じ verb（http://activitystrea.ms/schema/1.0/share）を割り当てるなど，システムをまたいだ verb の統一を試みている[12][19].

xAPI では 1 つの例外（発行したステートメントを取り消すために使用するもの）を除いて verb の定義は行っていない．独自の verb を定義することもできるが，プロファイルの策定に当たっては，広く採用されている既存の verb を用いることが推奨されている．しかし，表 2 や表 3 のように，同種の verb が重複して公開されている例も見られる．

表 2　The Registry に重複登録されている verb の例

行動	verb（IRI）
完了した	http://activitystrea.ms/schema/1.0/complete http://adlnet.gov/expapi/verbs/completed
レビューした	https://brindlewaye.com/xAPITerms/verbs/reviewed/ http://id.tincanapi.com/verb/reviewed

表 3　The Registry と ADL Vocabulary で verb が異なる例

行動	The Registry	ADL Vocabulary
ログインした	https://brindlewaye.com/xAPITerms/verbs/loggedin/	https://w3id.org/xapi/adl/verbs/logged-in
基準を満たした	http://activitystrea.ms/schema/1.0/satisfy	https://w3id.org/xapi/adl/verbs/satisfied

このような verb を用いた場合，別途マッピングを定義する必要があったり，解析結果やステートメントの寿命に影響を与えたりすることが考えられる．様々な実践や研究が集約され，語彙が整備されることが望まれる．

本設計では，各ステートメントの verb を，The Registry で公開されているものの中から選択した．重複登録されている verb は，説明を読み，より意味的に近いと考えられる方を選択した．その結果，Wiki ページの履歴からのリストア（対応するイベントは ¥mod_wiki¥event¥page_version_restored）や，ブックの印刷（¥booktool_print¥event¥book_printed）など，7 種類のステートメントに対して適切な verb が見つからなかった．ここで，Moodle のイベントの action（表 1 参照）が，verb に相当すると考えられる．3.1 で述べた要件 1 を満たすため，適切な verb が見つからなかったステートメントは，対応するイベントの action を使うこととした．verb の識別子は IRI であるため，イベントの action から IRI へ変換する規則も定めた．

4.3 object

xAPI では，“I did this.”の this に相当する object で，学習の対象を表現する．object には，学習活動（activity），人（agent），別のステートメント（statement reference）などを記述できる．本研究では，object が activity であるステートメントと，statement reference であるステートメントを設計した．

まず，object が activity である場合について述べる．activity には IRI である識別子が必須で，メタデータを含めることができる．ステートメント自体にメタデータを含めてもよいし，IRI が指す先でメタデータを提供してもよい．xAPI では，IRI に対するメタデータは不変であることが想定さ

れており，activity を更新した場合は IRI を変更すべきとされている．ステートメントにメタデータを含めるよりは，IRI に対応するメタデータを，別のサービスから提供することがシステムの設計としては理想的である．しかし，そのようなサービスの構築，運用はコストがかかり，xAPI の簡便性が損なわれる．

Web 上で学習を行ったのであれば，学習場所の識別子として URL を用いることが自然である．本設計でも，activity の識別子として Moodle の URL を用いる．たとえば，“ログインした”であればフロントページの URL，“コースを閲覧した”であればコースページの URL，“問題に解答した”であれば問題のプレビューページの URL とした．これは，規格上の理想的な設計ではないが，現実的な方法である．logstore_xapi でも Moodle の URL を用いているほか，既存のプロファイルも，Web 上の activity の識別子には，そのサイトやページの URL を用いている例が多い（たとえば[13]）．

3.1 で述べた要件 3 のため，ステートメントに activity のメタデータを可能な限り含める．名前（name）と詳細（description）は Moodle のデータベースに格納されている（対応する列がある）ことが多い．activity の種類（type）も記述できるが，語彙は定められていない．verb 同様，The Registry を参照し，対応する activity type がある場合は，指定する．その他，規格上記述できるすべてのメタデータを含める．小テストに使われる問題であれば，問題文，解答形式，正答なども記述できる．解答形式については，表 4 のように，Moodle の問題タイプと xAPI に記述できる形式（interactionType）とのマッピングを定義した．

表 4　Moodle の問題タイプと xAPI の interactionType
（[9]の表 4 を再掲）

Moodle の問題タイプ	xAPI の interactionType
description	other
essay	long-fill-in
match	matching
multianswer	choice
multichoice	choice
numerical	numeric
shortanswer	fill-in
truefalse	true-false

次に，object が statement reference であるステートメントを発行するフォーラムの例を述べる．Moodle のフォーラムは，スレッドごとに，最初の投稿を起点として，投稿に対して返信を投稿する形式で議論を行う．以降，スレッドを“ディスカッション”，ディスカッションへの投稿を“ポスト”，そのうち 1 件目を“第一ポスト”，2 件目以降を“返信ポスト”と表現する．ポストの編集（投稿内容の修正）と削除も考えると，ディスカッションごとに，図 1 のような木構造を想定することができる．

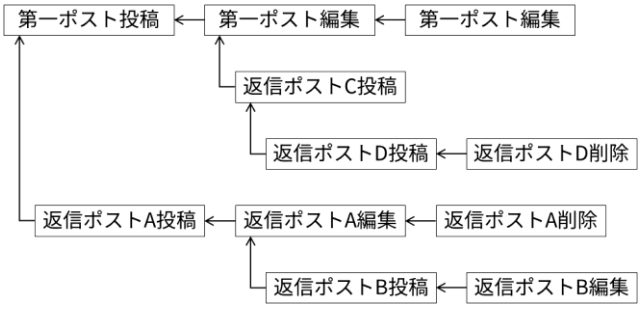

図 1　ポストの木構造の例
（[9]の図 3 を一部修正して再掲）

この例では，第一ポストに対して返信ポスト A が投稿され，その後第一ポストが編集されている．編集後の第一ポストに対して返信ポスト C が投稿され，その後さらに第一ポストが編集されている．返信ポスト A は一度編集された後，削除されている．このような木構造を表現するため，ポストの投稿, 編集, 削除のステートメントの object には，自身の親となるステートメントへの参照を記述する．つまり，図 1 において，“返信ポスト A 投稿”は“第一ポスト投稿”に対する行動であり，“返信ポスト A 削除”は“返信ポスト A 編集”に対する行動であると考える．

Moodle では, ディスカッションの追加時に第一ポストの投稿も行われる．このとき, “ディスカッション追加”と“第一ポスト投稿”の 2 つのステートメントを発行する．前者の object は activity であり，識別子はディスカッションの閲覧ページの URL とする．後者の object は statement reference であり, “ディスカッション追加”のステートメントを参照することとする．

Moodle でポストを投稿すると, 投稿後に自動的にディスカッションの閲覧ページに遷移する．このとき，logstore_xapi では，投稿自体のステートメントは発行されず，その後の閲覧のステートメントのみ発行される．本設計では，両者を発行する．

なお，フラグメント識別子まで考えると，ディスカッションのそれぞれのポストを URL で記述できる．フォーラムのステートメントにおいて object に statement reference を使う代替案として，object を activity にして，識別子をその URL とすることが考えられる．その場合，4.5 で述べるように，自分の親の情報を context に記述すればよい．本設計で採用した方法の方が xAPI の規格の意図には合致していると考えられるが，代替案の方がデータを扱いやすい可能性がある．

4.4 result

result には，学習結果を記述できる．得点，合否，学習時

間などの記述方法が定められているため，データを取得できる場合は，規格に従って記述する．小テストなど学習結果が意味を持つステートメントでは, result の記述が重要である．

ここでは，フォーラムへの投稿内容を result に記述する方法を紹介する．ポストには，件名とメッセージが必須であり，ファイルを添付することもできる．メッセージは，response プロパティに記述することが適切である．添付ファイルは，result の値としてではなく，result と同階層の attachments に記述するが，ここでは省略する．件名を記述する適切なプロパティは定義されていない．そこで，拡張可能なプロパティである extensions を用いる．前節で述べたポスト固有の URL も extensions プロパティに記述し，図 2 のように表現する．

```
"result": {
  "response": "<p>私もそう思います。</p>",
     … メッセージ
  "extensions" : {
    "subject": "Re: resultについて", … 件名
    "moreInfo": "https://www.example.com/
      path/to/moodle/mod/forum/discuss.php
      ?d=XX#pYY" … 自ポストのURL
  }
}
```

図 2　フォーラムへの投稿内容を表現する result の例

4.5 context

context には，学習文脈を記述できる．result 同様，データを取得できる場合は，規格に従って記述する．本設計において，ほとんどのステートメントに記述するデータは，親アクティビティ，学習プラットフォーム，ユーザエージェントである．親アクティビティは，contextActivities プロパティに記述する．たとえば，Moodle では，“問題”を組み合わせて小テストを構成する．問題は，小テストとは独立に管理される．小テストに問題を取り込んだとき，問題の親はその小テストであり，その親はコースページ，その親はサイト全体である．問題への解答結果を表現するステートメントでは，親アクティビティとしてコースモジュール（コースページに配置された小テスト）を記述する．学習プラットフォームとは, Moodle である．object が activity であるステートメントは，platform プロパティに学習プラットフォームを記述する．Web 上での学習であれば，学習に用いた端末(ユーザエージェント)も重要なデータである．ユーザエージェントは，extensions プロパティに記述する．logstore_xapi は，コースの設定情報や Moodle の内部情報も含めて，多くのデータを context に記述するが，ユーザエージェントは記述しない．

ここでは，context を用いて，小テストにおける受験回を表現する方法を紹介する．本設計において，小テストを受験したときに発行されるステートメントの例を，表 5 に示す．この例では，1 問の問題が含まれるページが 2 ページ（計 2 問）から構成される小テストを受験している．小テストは複数回受験することができるため，各ステートメントがどの受験回のものであるかを区別する必要がある．本設計では，図 3 のように，context の extensions プロパティに受験回を記述することとした．このステートメントは，表 5 の 6.または 7.に相当する．これまでに説明した設計の具体例として，ステートメントの全体を示している．

表 5　小テストのステートメント例

1. 受験を開始
2. ページを閲覧（1 ページ目）
3. ページを閲覧（2 ページ目）
4. 受験概要を閲覧
5. 受験を送信
6. 1 問目の結果
7. 2 問目の結果
8. 受験をレビュー

```
{
  "id": "7a8baf96-fa5d-11e6-ba57-e0db550d
    04aa",
  "actor": {
    "account": {
      "homePage": "https://www.example.com
        /path/to/moodle/",
      "name": "21" … MoodleのユーザID
    }
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/
      answered",
    "display": {
      "en-US": "answered"
    }
  },
  "object": {
    "id": "https://www.example.com/path/to
      /moodle/question/preview.php?id=XX&c
      ourseid=YY", … 問題のプレビューページ
    "objectType": "Activity",
    "definition": {
      "name": {
        "ja-JP": "足し算" … 問題名
```

```
      },
      "description": {
        "ja-JP": "1+1は?" … 問題文
      },
      "type": "http://adlnet.gov/expapi/ac
        tivities/cmi.interaction",
      "interactionType": "numeric",
        … 問題タイプ（数値問題）
      "correctResponsesPattern": [
        "2[:]2" … 正答
      ]
    }
  },
  "result": {
    "score": {
      "scaled": 0,
      "raw": 0,
      "min": 0,
      "max": 50
    },
    "success": false,
    "completion": true,
    "response": "3" … 解答
  },
  "context": {
    "contextActivities": {
      "parent": [{
        "id": "https://www.example.com/pat
          h/to/moodle/mod/quiz/view.php?id=
          ZZ" … 親となる小テスト
      }]
    },
    "platform": "Moodle 3.2.1",
    "extensions": {
      "attempt": 3, … 受験回
      "userAgent": "Mozilla/5.0 (Windows N
        T 10.0; Win64; x64; rv:51.0) Gecko/
        20100101 Firefox/51.0"
          … ユーザエージェント
    }
  },
  "timestamp": "20170224T102110,000+0900"
}
```


図 3　問題の解答結果を表現するステートメントの例

受験回は，一般的にはセッションと考えることができる．xAPI はセッションの概念を持っていないが，教材によってはセッションの区別が重要である．logstore_xapi は，本設計と同様の手法で受験回を区別している．cmi5 では，AU を起動するときに，学習管理システムがセッション ID を発行する．それを context の extensions プロパティに含めることにより，AU のセッションを区別している．ADL による SCORM のプロファイル[15]においても，同様の手法である．SCO を初期化（initialize）する際にアテンプト ID を発行し，それを context の extensions プロパティに含める．xAPI の仕様書には，registration（An instance of an Actor experiencing a particular Activity.[5]）をアテンプトやセッションと考えることもできると記述されているが，この手法を適用したプロファイルの例は見つからなかった．

5. まとめと今後の課題

xAPI は，学習履歴を表現するステートメントの構造を定めているが，ステートメントの粒度や記述内容は定めていない．それらは，利用者がプロファイルとして設計する必要がある．本研究では，Moodle を対象とした xAPI のプロファイルを策定した．このとき，Moodle が持つ学習履歴データの意味を変えずに，多くの情報を正確に表現できることを意図した．本研究では，既存のプロファイルも参照し，理想的な設計を目指した．

1.2 で述べた通り，実際に開発したシステムと本研究における設計内容には差異がある．開発したシステムでは，実践上の都合により，logstore_standard が書き出すログと Web サーバソフトウェア（Apache）のアクセスログを用いて，事後にステートメントを生成する．しかし，xAPI の考え方からも，実装上の問題からも，学習活動が行われた時点でステートメントを発行することが必要である．後者の問題の例として，フォーラムへの投稿が修正，削除されると，元の投稿内容は Moodle のデータベースには残らない．設計通りに実装するためには，リアルタイムにステートメントを発行する必要がある．また，logstore_standard ではユーザエージェントを記録していないため，開発したシステムでは Apache のアクセスログと照合する必要があった．リアルタイムにステートメントを発行すれば，Apache のアクセスログを参照する必要はない．

本研究では，Moodle のイベントをベースに設計を行った．しかし，すべての状況や，活動の設定による動作の違いまで検討できたわけではないため，引き続き検討を行いたい．また，Moodle のログ解析だけに用いるのであれば，xAPI を使う必要はない．異種システムにまたがる学習解析における，本設計の妥当性についても検討したい．

謝辞　本研究の一部は，JSPS 科研費 15K12424 の助成を受けた．

参考文献

[1] New Media Consortium. Learning Analytics. NMC Horizon Report:

2014 Higher Education Edition, 2014, p.38-39.
[2] New Media Consortium. Learning Analytics and Adaptive Learning. NMC Horizon Report: 2016 Higher Education Edition, 2016, p.38-39.
[3] Cooper, A.. “Learning Analytics and Interoperability - The Big Picture in Brief”. http://laceproject.eu/publications/briefing-01.pdf, (参照 2017-02-24).
[4] ADL. “Experience API Specification v1.0.1”. https://github.com/adlnet/xAPI-Spec/blob/1.0.1/xAPI.md, (参照 2017-02-24).
[5] ADL. “Experience API Specification v1.0.3”. https://github.com/adlnet/xAPI-Spec/blob/1.0.3/xAPI.md, (参照 2017-02-24).
[6] ADL. “Companion Specification for xAPI Vocabularies”. https://github.com/adlnet/companion-specification-for-xapi-vocabularies, (参照 2017-02-24).
[7] 山田恒夫. MOOC とは何か ポスト MOOC を見据えた次世代プラットフォームの課題. 情報管理, 2014, vol.57, no.6, p.367-375.
[8] 山田恒夫. 放送大学 MOOC「NIHONGO Starter（にほんご にゅうもん)」の開発. 大学教育と情報, 2014, 2014 年度, no.3, p.14-15.
[9] 森本容介, 古川雅子, 山地一禎. Moodle 上での活動を Experience API で表現する手法の検討. 教育システム情報学会研究報告, 2016, vol.30, no.5, p.99-106.
[10] “The Registry”. https://registry.tincanapi.com/, (参照 2017-02-24).
[11] “ADL Vocabulary”. http://xapi.vocab.pub/datasets/adl/, (参照 2017-02-24).
[12] Bakharia, A. et al.. Recipe for Success - Lessons Learnt from Using xAPI within the Connected Learning Analytics Toolkit. Proc. Sixth International Conference on Learning Analytics & Knowledge, 2016, p.378-382.
[13] Berg, A. et al.. The Dutch xAPI Experience. Proc. Sixth International Conference on Learning Analytics & Knowledge, 2016, p.544-545.
[14] ADL. “cmi5 Specification Profile for xAPI”. https://github.com/AICC/CMI-5_Spec_Current/blob/quartz/cmi5_spec.md, (参照 2017-02-24).
[15] ADL. “Experience API SCORM Profile”. https://github.com/adlnet/xAPI-SCORM-Profile, (参照 2017-02-24).
[16] “Moodle plugins directory: Logstore xAPI”. https://moodle.org/plugins/logstore_xapi, (参照 2017-02-24).
[17] 赤倉貴子, 柏原昭博. e ラーニング/e テスティング. ミネルヴァ書房, 2016, 212p.
[18] “Moodle developer documentation”. https://docs.moodle.org/dev/Main_Page, (参照 2017-02-24).
[19] Kitto, K. et al.. Learning Analytics beyond the LMS : the Connected Learning Analytics Toolkit. Proc. Fifth International Conference on Learning Analytics & Knowledge, 2015, p.11-15.