修 士 論 文

Law Enforcing Information Systemに ソフトウェアアカウンタビリティ機能を 追加する機構の研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

秋山 裕俊

2008年3月

修 士 論 文

Law Enforcing Information Systemに ソフトウェアアカウンタビリティ機能を 追加する機構の研究

指導教官　落水浩一郎 教授

審査委員主査　落水浩一郎 教授
審査委員　鈴木正人 准教授
審査委員　片山卓也 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

610002秋山 裕俊

提出年月: 2008年2月

概 要

近年、われわれの生活する社会では、電子化が急速に押し進められており、日常生活の基盤は、電子社会システムによって成り立っている。これらの情報システムのほとんどは、国や地方自治体が定める法律や条例などの各種規則に則って構築される。規則に則って構築される情報システムを利用するにあたって、システムが行った判断や行為について、利害関係者がいくつかのタイプの質問を持つことがある。本研究では、利害関係者が持ちうる質問のうち、「実行結果に関する質問」に対して、システムが利害関係者を満足させる十分な説明を行う、といった機能を既存の情報システムに追加することを目標とする。この説明機能は社会規則および既存システムの履歴情報を利用して実現を行う。

目 次

図 目 次

表 目 次

第1章　序論–研究の背景と目的

1.1　研究の背景と目的

近年、電子政府・電子自治体への取り組みをはじめとして、社会システムの電子化が急速に押し進められている。行政、金融、医療、交通、教育、企業などの活動の基盤部分が電子システム化され、それらがネットワークを介して相互に接続され巨大な電子社会システムが構築されつつある。

われわれの日常生活は、社会基盤としてのこのような電子社会システムの上に成り立っている。したがって、電子社会システムは、これまでの情報システムのように機能を単に提供するだけでは十分ではなく、われわれが安心して生活できることを保障できるように設計・構築され、かつ運用・保守されている必要がある。

本学 21 世紀 COE プログラム「検証進化可能電子社会」[1] では、安心できる電子社会システムが持っているべき要件として、正当性、アカウンタビリティ、セキュリティ、耐故障性、進化性の 5 つからなる安心性要件を提案しており、本研究はこのうちソフトウェアアカウンタビリティ (Software Accountability) および進化容易性 (Ease-of-Evolution) を対象としている。進化容易性では、ソフトウェアの進化性に加えて、進化のための変更コストの低減も特に重視する。

われわれの社会には、電子社会システムを含め属するものすべてが守らなければならない法律や条例および組織が定めている各種規則があり、これらを社会規則と呼ぶ。一般に社会規則は、自然言語で記述された文書であり、条・項もしくは節から構成される。

電子社会システムは社会規則の適用を支援し、社会規則を完全に満たすように構築されていなければならない。さらに、社会規則に従って電子社会システムが正しく構築されていることを保証でき、かつ確認できる必要がある。また、社会は常に変化しており、社会規則もそれに応じて改定される。電子社会システムは、社会規則の改定に応じてシステムを迅速に進化させていかなければならない。そのためには低いコストでシステムを変更できる必要がある。こういった特徴を持つシステムを法令実働化情報システム、Law Enforcing Information System(以下、LEIS と略記する) と呼ぶ。

LEIS におけるソフトウェアアカウンタビリティとは、システムが行った判断や行為に関して、そのシステムの利害関係者が持つ質問に対して納得するよう説明しうることと直観的に定義する。ここで、利害関係者とは、システムの一般利用者、業務担当者、社会規則整備担当者、システム開発者・保守担当者の 4 種類を考える。たとえば、大学の履修規則の場合、学生の履修登録作業および修了要件の達成状況の確認作業を支援する履修管理

システムはLEISであり、学生、事務員、教員、履修管理システム開発者が利害関係者となる。

LEISにソフトウェアアカウンタビリティを適用することで高い信頼性、かつ進化容易性を持つシステムを開発することできる。

LEISはウェブシステムとして実現されることが多く、一般にウェブシステムはユーザインタフェース層、プロセス管理層、データベース層から成る3層モデルに基づいて実現される。文献[9][10]では、3層モデルに基づいたアカウンタビリティを実現するための参照アーキテクチャが提案されている。これを図1.1に示す。

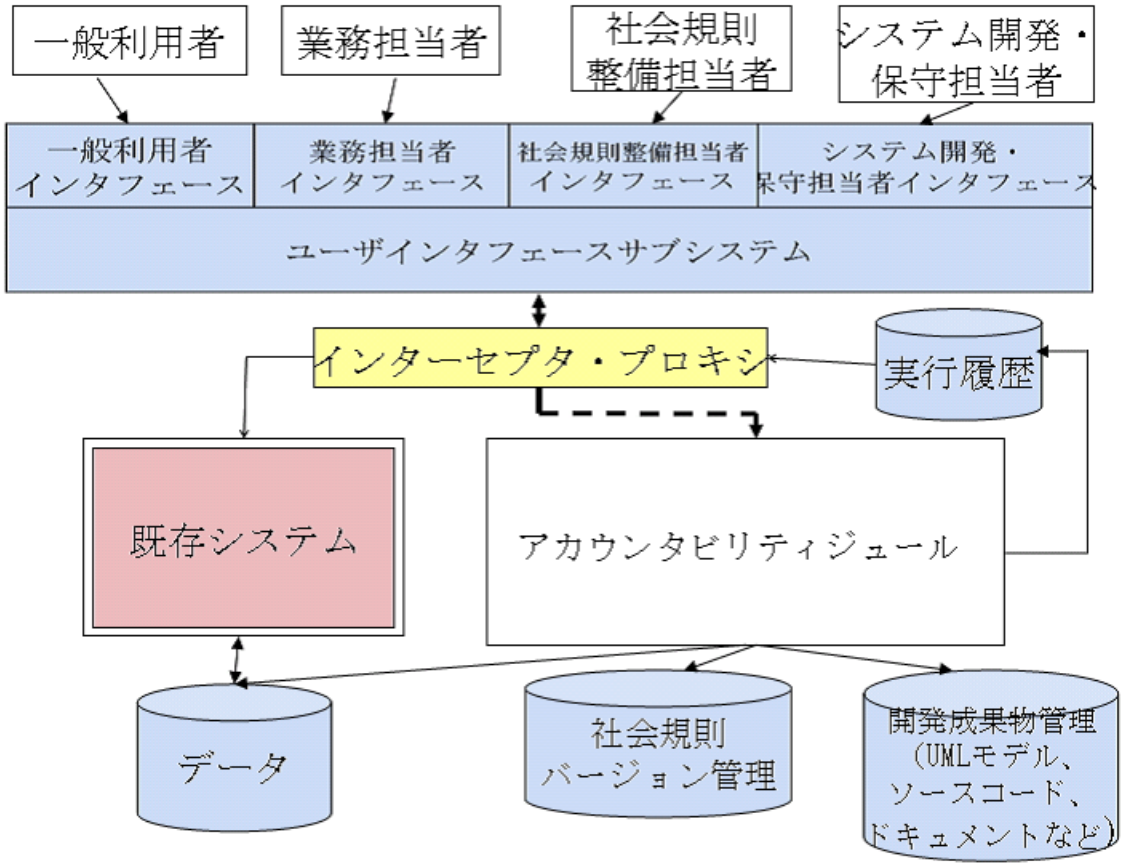

図 1.1: 3層モデルに基づく参照アーキテクチャ

このアーキテクチャの特徴は、既存システムのユーザインタフェース層とプロセス管理層との間にインターセプタプロキシを配置し、既存システムに対しアカウンタビリティモジュールを付加できる構成になっている点である。

LEISがソフトウェアアカウンタビリティの能力を持つとき、それをソフトウェアアカウンタビリティ機能(以下、アカウンタビリティ機能と略記する)と呼ぶ。本研究の目的は、アカウンタビリティ機能を提供するアカウンタビリティモジュールの設計と実現で

ある。アカウンタビリティ機能はシステムの実行履歴 DB および社会規則 DB を用いて実現する。ここで、システムの実行履歴を取得する手法はさまざまなものがあるが、パーフォーマンスのオーバーヘッドが特に問題となる。本研究では、アカウンタビリティモジュールのパーフォーマンスの性能改善も行う。

1.2 関連研究

電子社会システムの安全性や安心性の実現のための研究は現在世界各国で行われている。社会システムにおいて人々の生活に関連の深い法律、条例などの法令に関する研究や電子社会システムを支えるデータベース技術、セキュリティなどの情報技術に関する研究が進められている。

国外では、アメリカ合衆国において NSF(National Science Foundation) が Digital Government Research Project を推進しており、現在では 100 近くの学術研究機関が社会学から情報学にいたるまでの様々な分野で電子社会システムについての研究を進めている。

国内では、名古屋大学の外山グループが法令執務 [2] に関する研究を行っている。法令執務とは、法令文書の起草、改廃など法令文書の作成・管理に関わる作業のことであり、従来は、習慣に基づいた法令文書の書式や法令文の表現に関する知識を持つ専門家が手作業によって行われていた。しかし、法令数は膨大であり、専門家の負担は増大している。そこで、[2] では、条・項など法令の論理的構造を単位として実行される作業を支援するための研究や、法令改正のための旧法令と改正法令から新法令を生成するための統合方式の研究が進められている。

また、要求定義法においてゴール指向要求分析法 [3] に関する研究が行われており、本研究において関連が深い。ゴール指向要求分析とは、システムに対する非機能要求をゴールとして設定し、それを AND–OR 木を利用してサブゴールに階層的に展開していくことにより、具体的な要求を抽出する手法である。葉にあたる部分には通常の機能要求がくる。文献 [7] では、この手法に基づいてアカウンタビリティ木を提案している。

1.3 論文の構成

本論文の構成は以下のとおりである。

- 第 2 章 ソフトウェアアカウンタビリティ実現へのアプローチ
  LEIS の開発プロセスにおいて、ソフトウェアアカウンタビリティ実現の位置づけを述べ、われわれ研究グループの成果であるアカウンタビリティ木と参照アーキテクチャについて説明を行う。

- 第 3 章 ソフトウェアアカウンタビリティ機能の実現方式
  アカウンタビリティ機能を実現するためのモデルの全体像の説明、モデル中に現れる要素の説明を行う。

- 第 4 章 JavaEE におけるソフトウェアアカウンタビリティ機能の設計および実現
  事例研究として開発を行った履修管理システムと、これに追加するかたちで開発を行ったアカウンタビリティモジュールの設計および実現について説明する。

- 第 5 章 評価
  本研究で開発したアカウンタビリティモジュールの機能面および性能面においての評価を行う。

- 第 6 章 結論–まとめと今後の課題
  本研究についてまとめ、今後の課題を述べる。

第2章　ソフトウェアアカウンタビリティ実現へのアプローチ

本章では、われわれの研究グループによる成果を導入しつつ、これを基にソフトウェアアカウンタビリティの実現アプローチについて説明する。

2.1　ソフトウェアアカウンタビリティの定義

アカウンタビリティについて、文献 [4] によると、「一般に説明責任と訳される。具体的には政府・行政などの国民に対する政策成否の説明責任や、経営者の株主に対する財務状況、経営戦略の展開、見直しとその成果などについての説明責任について用いられている」、「行政機関または公務員個人が行った判断や行為に関して、国民が納得するよう説明しうること」とある。

これに基づいて、まず、ソフトウェアアカウンタビリティを以下のように定義する。

> 「LEIS 自体が、行った判断や計算結果に関して、その理由をシステムの利用者に説明できること。言葉をかえれば、LEIS が、LEIS が行った判断や計算結果に対して利害関係者が疑問を持ったとき、利害関係者の質問に答え得ること。LEIS は、利害関係者を満足させる回答をシステムの実行状況を利用して生成できる必要がある。」

ここで利害関係者を、単なるシステム利用者に限定せず、システムの一般利用者、業務担当者、社会規則整備担当者、システム開発・保守担当者の 4 種類に拡張する。いくつかの LEIS とその利害関係者の例を以下に挙げる。

- 大学の履修規則には、大学の教育理念に基づいて、修了のための資格が定義されており、また、資格を得るために必要な様々の条件とその修得法が示されている。学生の履修登録作業および修了要件の達成状況の確認作業を支援する履修管理システムは LEIS であり、学生、事務員、教員、履修管理システム開発者が利害関係者となる。

- 地方自治体には、様々な条例がある。地方自治体システムには、立法担当者、行政担当者、システム開発者、一般市民などの利害関係者が関与する。

- 会社の社内規定には、運営方針に従った様々な決定の基となる規則が定められている。社内システムには、管理者、担当者、従業員などの利害関係者が関与する。

図 2.1 に、地方自治体システムに関与する 4 種類の利害関係者の例と、彼等が持つ典型的な疑問を以下に挙げる。

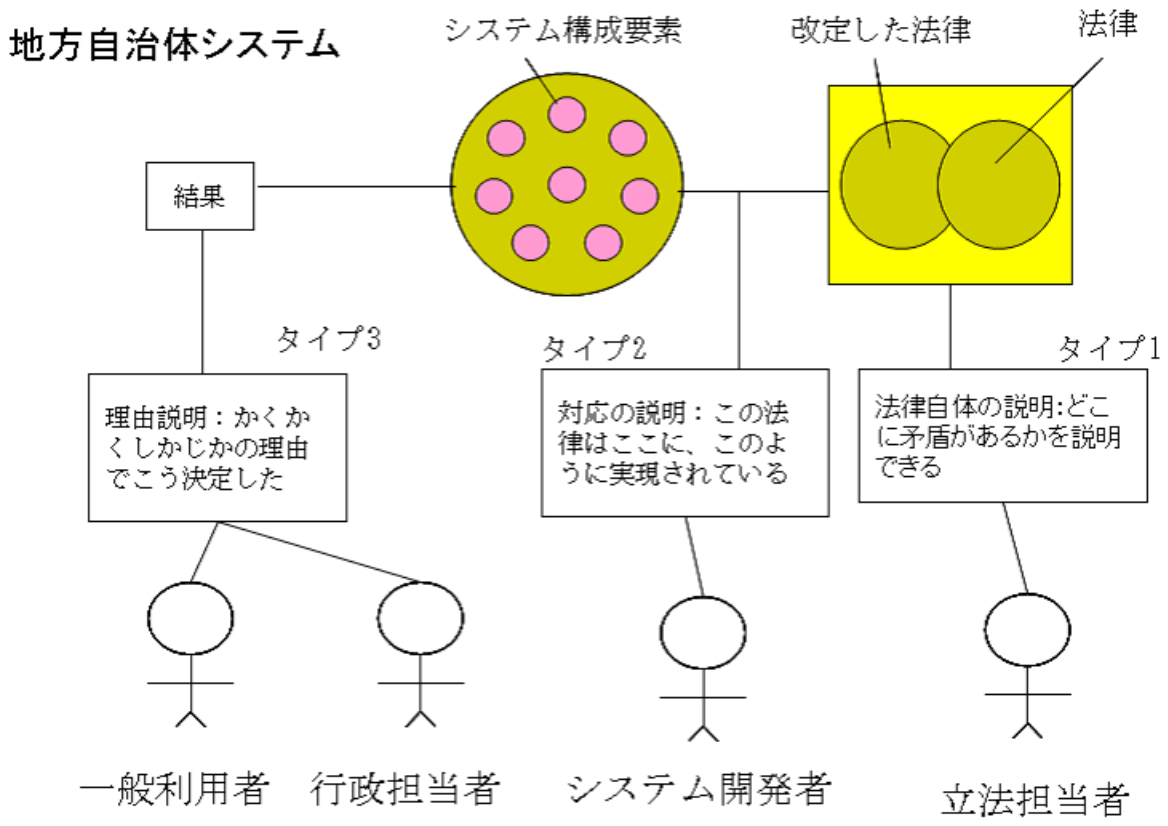

図 2.1: 利害関係者の 3 種類の関心

- 県や市の立法担当者は、新しい法律の制定をはかる場合、当該法律の内容のみならず、従来の法律との整合性にも関心を持つ。「新しい法律と既存の法律の間の関係は？矛盾はないのか？」のどの疑問をもつ。

- システム開発者は、初期のシステム開発時には、開発する情報システムに法律内容を正確に反映させることに関心を持ち、「規則を必要十分に実現したか？」という疑問をもつ。また、法律の改定にともなうシステム保守においては、「法律の改定にあわせて、情報システムを変更したい。法律とシステム構成要素の対応はどのようになっているのだろうか？」などの疑問を持つ。

- 行政担当者やシステムを利用する一般市民は、システムが提供する実行結果に関心を持ち、「情報システムを用いて電子申請や登録を行った。システムが提示した処理結果についての疑問がある。この結果はどのような法律や条令をどのように適用して得られたものだろう？」などの疑問を持つ。

各種の利害関係者が持ちうる疑問を「システムのどの側面に関係するか」という観点から、図 2.1 に示すように 3 つのタイプに分類する。

図 2.1 において

- タイプ 1 は、規則そのものに注目した疑問である。
- タイプ 2 は、規則とシステム構成要素の対応構造に注目した疑問である。
- タイプ 3 は、システムの実行結果に注目した疑問である。

2.2 3 種類のアカウンタビリティ機能と LEIS 開発プロセスとの関係

図 2.2 に、われわれが想定している LEIS の開発プロセスを示す。LEIS 開発プロセスの概要を以下に示す。

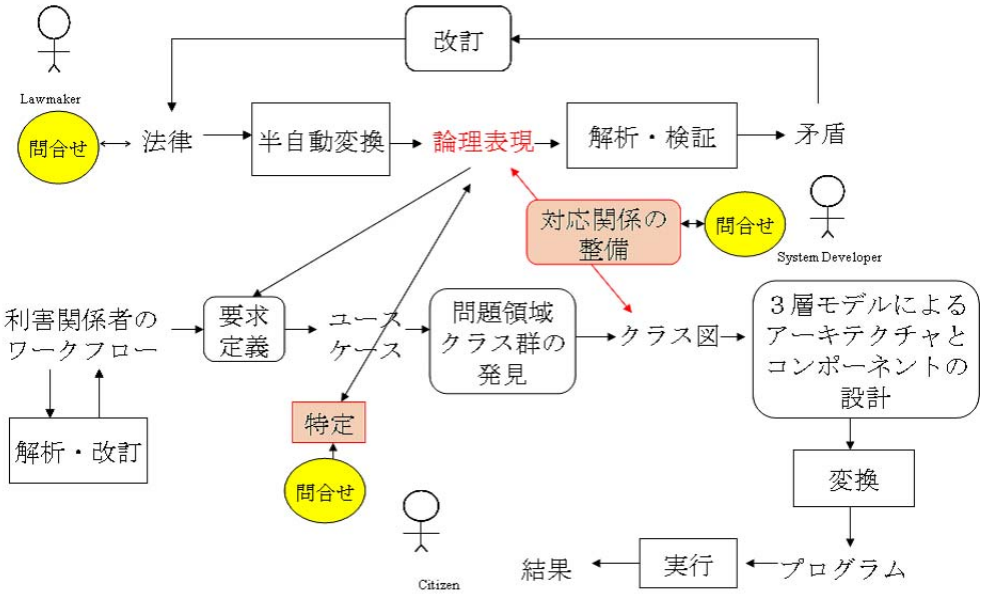

図 2.2: 3 種類のアカウンタビリティ機能と LEIS 開発プロセスとの関係

- 平文で表現された規則 (法律) は、自然言語処理により「論理表現」に自動変換される。
- 論理表現は法推論機構により自動解析され、矛盾が検出される。矛盾解消は人手により行う。ここまでのサイクルを法令デバッギングと呼ぶ
- 法令デバッギングにおける中間表現は LEIS 設計の入力となる。
- コンポーネント設計手法に基づき 3 層モデルアーキテクチャを採用してシステムを自動生成する。

上記の処理の流れにおいて、ソフトウェアアカウンタビリティに関与する情報は、図 2.2 の以下の箇所に保持される。

- 論理表現
  タイプ 1 のアカウンタビリティ機能を実現する際の、すなわち、法令デバッギングの一次情報となる。
- 対応関係
  論理表現とクラス図の対応関係を保持する。タイプ 2 のアカウンタビリティ機能を実現する際の一次情報となる。
- 論理表現、対応関係、システムの実行履歴
  タイプ 3 のアカウンタビリティ機能を実現する際の一次情報となる。

上記、論理表現と対応規則を利用して、ソフトウェアアカウンタビリティ機能を以下のように実現する予定である。

- タイプ 1 のアカウンタビリティ
  法令デバッギングの過程に機能を盛り込む。
- タイプ 2 のアカウンタビリティ
  対応関係を保持するデータベースと検索システムを開発する。
- タイプ 3 のアカウンタビリティ
  システムの実行履歴を保持する履歴データベースを設計し、問い合わせの内容と実行履歴を利用して、対応関係のサブセットを特定する。これにより論理表現のサブセットが特定できる。

本研究では、タイプ 3 のアカウンタビリティ機能を提供するアカウンタビリティモジュールの設計と実現を行った。これ以降は、タイプ 3 のアカウンタビリティ機能の実現方式に議論を限定する。

2.3 アカウンタビリティ木

自然言語処理により論理表現に変換された社会規則を利用してアカウンタビリティ機能を実現するためには、変換された社会規則を構造化する必要がある。

文献 [5] において、ゴール指向要求分析法 (GORA) を採用することにより、「規則群」と「規則を作る人が意図し、理解し、表現した世界」とを階層的に関連づけて構築する手法が提案されている。すべての規則は、その規則をつくる人が意図した目標を達成するために存在し、その目標はさらに上位の目標を達するために存在するという構造を作ることができる。これがゴール木である。ゴール木の構成方針を図 2.3 に示す。

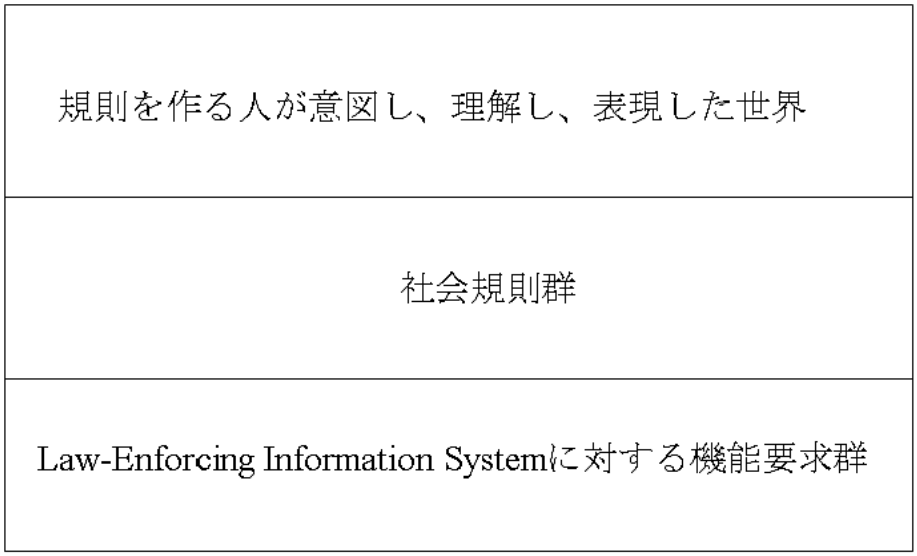

図 2.3: ゴール木の基本方針

ゴール木は、社会規則の改定の制御を行うため、および利害関係者からの規則制定のいきさつに関する質問に答えるために有用である。また、社会規則はたがいに関連しあっているので社会規則自体を構造化する必要がある。構造化することで、関連規則の検索が容易になる。

エックホフの法理論 [6] によれば、規則間には種々の関連が存在する。この関係を利用することにより社会規則自体の構造化と型付けを行うことができる。

ゴール木の各ノードは、図 2.3 における第 1 層のどのような考え方で規則が作成されたかなど、利害関係者によって理解されたセマンティクスを表現し、木の葉に第 2 層の社会規則の条文を対応させ、かつ、法理論に基づき木の葉を型付けした木をアカウンタビリティ木と呼ぶ。なお、木の葉間には、クロスリンクとして条文間の関連を設定する。図 2.4 にアカウンタビリティ木を示す。

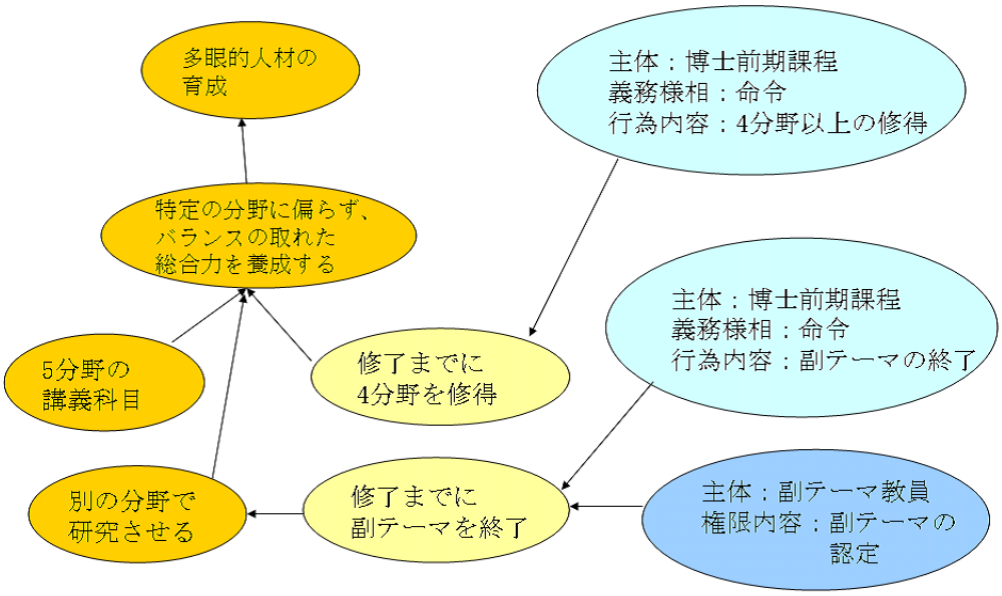

図 2.4: アカウンタビリティ木の一例

杉森 [7] は GROA によるゴール木の定義とエックホフの法理論によりアカウンタビリティ木を関係データベースとして実現した。本学の履修規則に対するアカウンタビリティ木を定義し、それに基づき図 2.5 に示すスキーマを定義した。

以下に、各テーブルについて説明する。

- 業務目標テーブル
  アカウンタビリティ木におけるゴールとサブゴールを保持する。たとえば、図 2.5 における「多眼的人材の育成」、「特定の分野に偏らず、バランスの取れた総合力を養成する」、「5 分野の講義科目」、「別の分野で研究させる」の各ノードとその間の関係を保持する。

- 規則要求テーブル
  アカウンタビリティ木の葉の情報を保持し、組織に所属する者の行為、またはその組み合わせによって、なるべき状態を示す。たとえば、図 2.5 における「修了までに 4 分野を修得」、「修了までに副テーマを終了」の情報を保持する。

- 義務規範テーブル
  履修規則のうち、義務規範に属するものを保持する。どのカテゴリの人間に、どのような行為が指令されているか、といった内容である。主体、様相、行為の属性を持つ。たとえば、図 2.5 における「主体： 、業務様相： 、行為内容： 」のかたちのものが義務規範テーブルの情報である。

- 性質決定規範テーブル

履修規則のうち、性質決定規範に属するものを保持する。あるカテゴリに入る人物、状態などの現象がどのようなものであるか、といった内容である。カテゴリ名、カテゴリ内容の属性を持つ。たとえば、図 2.5 における「副テーマ教員」は、性質決定規範テーブルの「カテゴリ名：副テーマ教員、カテゴリ内容：副テーマ開始時に、各学生に研究科によって定められた教員」と定義される。

- 権限規範テーブル
  履修規則のうち、権限規範に属するものを保持する。どのカテゴリの人物に、どのようなことに影響をあたえる権限があるか、といった内容である。主体、権限対象の属性を持つ。たとえば、図 2.5 における「主体：、権限内容：」のかたちのものが権限規範テーブルである。

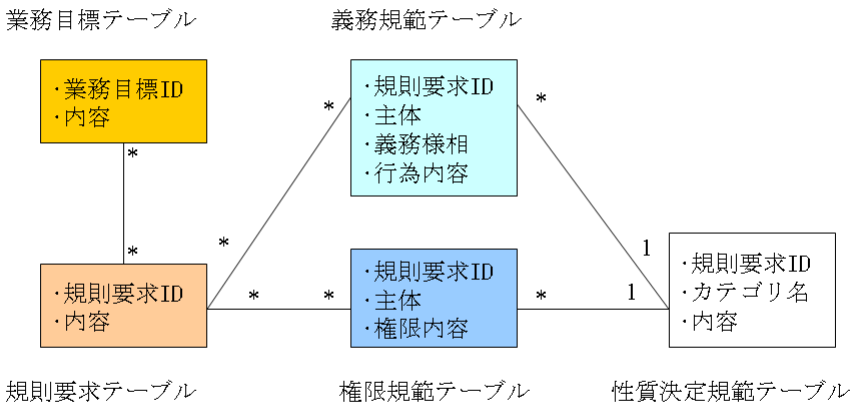

図 2.5: エックホフの法理論に基づいたデータベーススキーマ

2.4 参照アーキテクチャ

われわれはアカウンタビリティモジュールを既存のシステムに低コストで装着するための参照アーキテクチャを提案した。参照アーキテクチャ設計にあたっての要件は以下の通りである。

- アカウンタビリティモジュールを既存の情報システムに低コストで結合できる。
- 既存システムを最小の変更コストで再利用されうる。
- 大部分の LEIS は、3 層モデルに基づくウェブベースシステムであることを考慮する。
- さらに EJB3.0 に対応できる。
- アカウンタビリティモジュールを 3 層モデルに結合する拡張構造が必要である。

LEIS はウェブシステムとして構築されることが多く。一般にウェブシステムはユーザインタフェース層、プロセス管理層、データベース管理層の 3 層モデルに基づいて構築されている。

参照アーキテクチャの特徴は、既存システムのユーザーインターフェース層とプロセス管理層との間にインターセプタプロキシを配置し、既存システムに対しアカウンタビリティモジュールを追加できる点である。また、アカウンタビリティ機能はシステムの実行履歴 DB と社会規則 DB を利用して実現する。

3 層モデルに基づいたアカウンタビリティを実現するための参照アーキテクチャを図 2.6 に示す。

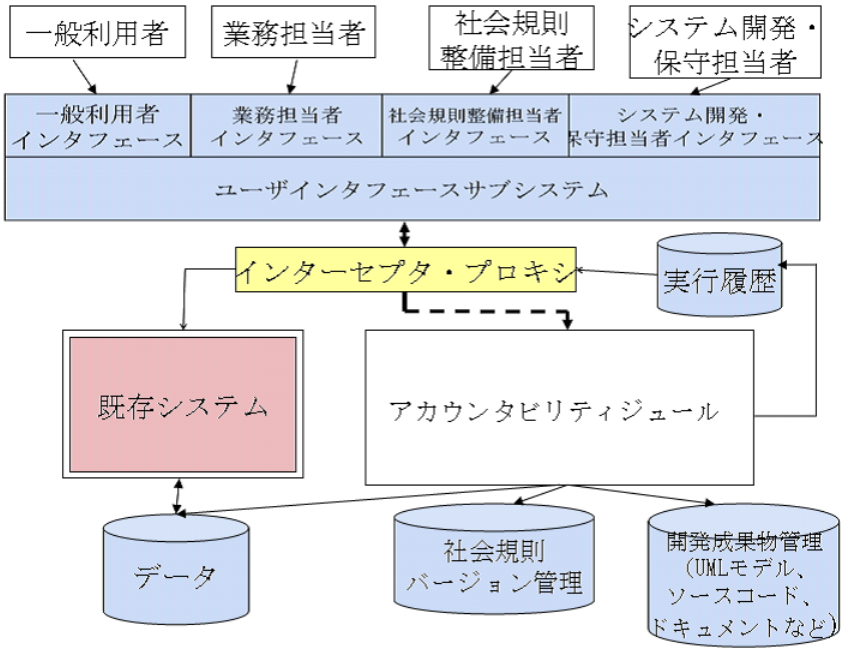

図 2.6: アカウンタビリティを実現する参照アーキテクチャ

第3章　ソフトウェアアカウンタビリティ機能の実現方式

本章では、アカウンタビリティモジュールの定義を行い、アカウンタビリティ機能を実現するための説明モデルの全体像の説明およびモデル中に現れるクラス定義書と履歴情報の説明を行う。なお、クラス定義書と履歴情報に関しては概要のみを述べ、詳細の設計に関しては次章で述べる。

3.1　ソフトウェアアカウンタビリティモジュールの定義

タイプ3のアカウンタビリティ機能を実現するために、ソフトウェアアカウンタビリティモジュールを開発した。これは、アカウンタビリティ木とアカウンタビリティ機能をもつソフトウェアモジュールのことである。

アカウンタビリティモジュールの役割は、利用者にアカウンタビリティ機能を提供することである。利用者がシステムの実行結果に対して疑問や質問をもった場合、利用者はアカウンタビリティモジュールに実装されたアカウンタビリティ機能を利用することでその疑問や質問を解決できる。利用者からの質問があった場合、この質問を受け付ける方式はさまざまなものがあるが、本研究では、利用者にいくつかの質問候補を提示し、その中から選択させる方式をとる。この質問候補をリストにしたものを質問リストと呼ぶ。

質問候補としては、実行結果自体をリストとして、その中のどの実行結果に対して質問があるかを利用者に選択させる方法がある。しかし、この方法では、実行結果の情報量によって見やすさが変化するため問題である。

そこで、本研究では、「登録ボタンをチェックする」などのシステムに対する利用者の行為をとし、どの行為により得られた実行結果に対して、質問があるかを利用者に選択させる方法を採用した。

また、利用者への回答の説明は、社会規則DBと実行履歴DBを利用し、質問に関連する社会規則および利用者の状況の2つの情報を利用者に提示するかたちで行う。ここで、利用者の状況とは、

> 「システムに規則を適用して開発を行った場合に、システム内部で用いられる
> 利用者の状態を示す情報」

である。たとえば、本学の理由規則のうち、副テーマ提出要件では、「基幹・専門・先端

講義科目２科目以上の修得を行うこと」とある。この規則をシステムに適用した場合、利用者の状況となるものは、基幹・専門・先端講義科目の修得科目数である。

3.2　ソフトウェアアカウンタビリティ機能の実現法

前節で述べたアカウンタビリティモジュールの特徴をふまえた上で、本研究では、アカウンタビリティ機能を実現するための図 3.1 の説明モデルを提案した。

ここでは、この実現方式について説明を行う。

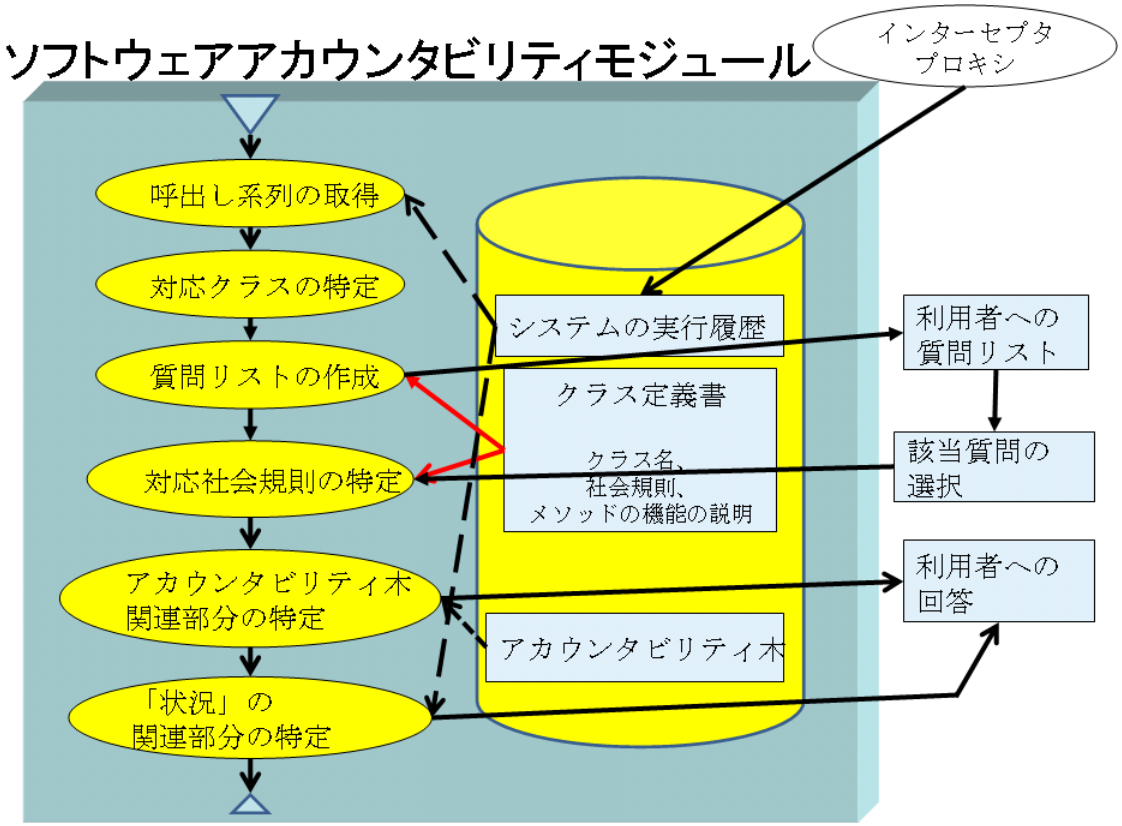

図 3.1: タイプ３のアカウンタビリティ機能の実現方式

アカウンタビリティ機能は以下の手順で実現される。

- インターセプタプロキシがシステムの実行履歴情報を記録する。
- 実行履歴のうち、呼出し系列の情報はクラス定義書を用いて質問リストに変換する。
- 質問リストは複数の質問からなり、利用者は質問リストより１つの質問を選ぶことができる。

- クラス定義書を用いて、説明に必要となる社会規則を特定する。
- 論理表現はアカウンタビリティ木の葉として表現されている。
- 実行履歴のうち、引数の値等は、質問者の状況を表すデータとして説明に利用する。

3.3 クラス定義書

本研究で提案したアカウンタビリティ機能の実現方式では、クラス定義書が重要な役割を持つ。ここでは、クラス定義書の概要のみを述べ、設計の詳細については次章で述べる。

実行履歴情報から社会規則の特定を行うためには、条・節からなる社会規則の条文が、システム内部で実行されている、どの処理に対応しているかを明確にする必要がある。しかし、規則が実装されたシステムを考えた場合、1つの条文と、それを実装するクラスとの間には明確な対応関係がなく、1つの条文はさまざまなクラスのさまざまな部分に分散して実装され対応している。また、どの社会規則の条文がソースコードのどの部分と対応しているかなどの情報は、システムの仕様書に記録されない場合もある。

本研究では、条文と、条文が実装される処理との対応を示した対応表を作成し、それを利用することで、条文と処理との対応関係を明確にした。この対応表をクラス定義書と呼ぶ。また、クラス定義書の情報は既存システムを設計した開発者が書くものである。

さらに、クラス定義書は以下に述べるような特徴を持つ。

前章で、質問リストに関して、利用者の「登録ボタンをチェックする」などのシステムに対する行為をリストにすると述べた。この利用者のシステムに対する行為は機能呼び出しなどの1つの処理になる。クラス定義書は、この機能呼び出しなどの処理を利用者に理解できる言葉に置き換える役割を持つ。ここで、1つの置き換えられた情報は質問リストの1つの項目になる。

つまり、クラス定義書は以下の役割を持つ。

- 社会規則の条文と、その条文が実装される処理との対応をつける。
- 処理を利用者に理解できる言葉に置き換える。

クラス定義書を利用することで、利用者が質問リストから選択した情報から対応規則を一意に決定することが可能である。

3.4 履歴情報

アカウンタビリティ機能を実現するためにはシステムの実行履歴情報から必要となる履歴情報を抽出しなければならない。ここでは、どのような履歴情報が必要であるかを議論する。履歴情報の具体的な取得方法については次章で行う。

われわれがシステムの実行結果に対して疑問、質問をもつ場合、どのような規則が適用され、そのような実行結果になったかである。本研究では、これらの利用者からの質問に対する回答を、関連のある社会規則および利用者の状況を利用者に提示するかたちで行う。

以下に、システムが利用者に対して、説明を行うために必要な情報を示す。

- 利用者が実行結果に対して質問を持ったとき、その実行結果がシステムのどの処理によるものか、という情報。
- その処理は、どのような社会規則に則っているか、という情報。
- また、処理中にシステム内部では、利用者の何の状況が使用されているか、という情報。

このうち、「その処理は、どのような規則に則っているか」という情報は、クラス定義書に記述される情報である。

これらの点をふまえると、必要となる履歴情報は以下のようになる。

1. 実行結果がどの処理によるものか、という情報。
2. 処理中に使用される利用者の状況の情報。

第4章　JavaEEにおけるソフトウェアアカウンタビリティ機能の設計と実現

本研究では、まず、LEIS の事例研究として履修管理システムをユースケース駆動オブジェクト指向開発方法論に従って開発した。これは、JavaEE プラットフォームを用いたクライアント・サーバ方式のウェブアプリケーションである。次に、このシステムに追加するかたちで、JavaEE プラットフォームを対象とした実装アーキテクチャに基づいてソフトウェアアカウンタビリティモジュールの開発を行った。本章では、参照アーキテクチャと実装アーキテクチャとの対応を示し、JavaEE の中でも特に JBossSeam ウェブアプリケーションフレームワークを用いた場合の、それぞれのシステムの設計および実現について述べる。

4.1　JavaEEプラットフォームを対象とした実装アーキテクチャ

図に示した参照アーキテクチャおいて、アカウンタビリティ機能はシステムの実行履歴 DB と社会規則 DB を利用して実現する。ここで、実行履歴の取得方法には大きく 2 種類考えられ、それぞれ取得できる実行履歴内容が異なるため実現できるアカウンタビリティ機能も異なる。

1 つ目の方法は、ユーザインタフェース層とプロセス管理層との間に配置したインターセプタプロキシからのみ実行履歴情報を取得するものである。インターセプタプロキシは、これらの層の間で行われる機能呼び出しやデータの受け渡しのすべてを実行履歴として取得することができ、DB に格納できる。この方法の場合、既存システムのプロセス間理層は一切変更を加えずにアカウンタビリティ機能を追加できるが、既存システムのプロセス管理層の内部で行われている処理の実行履歴情報を取得することができない。このため、使用できる実行履歴は層の間のインタフェース、つまりプロセス管理層の機能呼び出し名と引数およびその返り値のみであり、実現できるアカウンタビリティ機能は限られたものとなる。

2 つ目の方法は、インターセプタプロキシで取得できる実行履歴に加えて、既存システ

ムのプロセス管理層の内部で行われる処理の実行履歴も取得するものである。例えば、既存システムが Java で実装されている場合、以下のような取得方法がある。

- 通常デバッガにより利用される JPDA が提供する API を利用すると、既存システムが動作している Java 仮想マシンから処理の詳細な実行履歴を取得できる。
- AOP 技術を適用すると既存のソースコードを変更することなくシステムの内で行われている処理の実行履歴を取得可能である。

前者の JPDA が提供する API を利用する場合、システムの動作に関する詳細な実行履歴情報を取得でき、高度なアカウンタビリティ機能を実現できる利点がある。しかし、この方法では、Java の実行環境から取得を行い、ウェブアプリケーションフレームワークを含む、すべての処理の実行履歴を取得するため、パフォーマンスが低いという難点がある。

これに対し、後者の AOP 技術を利用する場合、実行履歴を出力するための処理を既存システムのソースコードに差し込むかたちで実装を行うため、取得できる実行履歴情報は限られたものになるが、パフォーマンスは高い。

本研究では、システムのパフォーマンスを考慮し、後者の方法でアカウンタビリティ機能の実現を行う。そのための JavaEE プラットフォームを対象とした実装アーキテクチャを導入した。図にアカウンタビリティ機能を実現する JavaEE プラットフォームを対象とした実装アーキテクチャを示す。

図に示した参照アーキテクチャは 3 層モデルに基づいており、実装アーキテクチャも 3 層モデルに基づいているため対応づけることができる。

JavaEE アーキテクチャは instrumentation と javassist の 2 つの技術を使用して実現される。instrumentation は JDK1.5 に導入された java.lang.instrumentation パッケージが提供する API である。instrumentation の特徴を次にあげる。

- instrumenation パッケージは動的にバイトコードを操作するための枠組みを提供する。
- アプリケーションプログラムが実行される前にエージェントと呼ばれるプログラムを実行でき、このエージェントプログラムにより JVM 上で実行されるプログラムを観測できる。

instrumentation の技術を用いることで VM 上にロードされたクラス名などの情報を取得可能である。さらにアカウンタビリティ機能を実現するために本研究ではバイトコードを操作するための API である javassist を使用した。

instrumentation および javassist が提供する機能を併用して用いることで、実行しているアプリケーションプログラムのクラス名、メソッド名、メソッドの引数名、引数の値といった情報を取得可能である。

4.2　履修管理システムの開発

履修管理システムは、ユースケース駆動オブジェクト指向方法論 COMET[] に基づいて開発を行った。本システムは本学が定める履修規則に則っている。

ここでは、中間成果物を示しながら、履修管理システムの概要およびシステム構造を述べる。

4.2.1　ユースケースモデル

開発にあたって、システムのメインの利用者である本学情報科学研究科の学生全員を対象としたアンケート調査により要求獲得を行った。アンケートの回答を集計し、履修管理システムへの機能要求を整理した。図 4.1 にこれを基に定義したユースケース図の一部を示す。

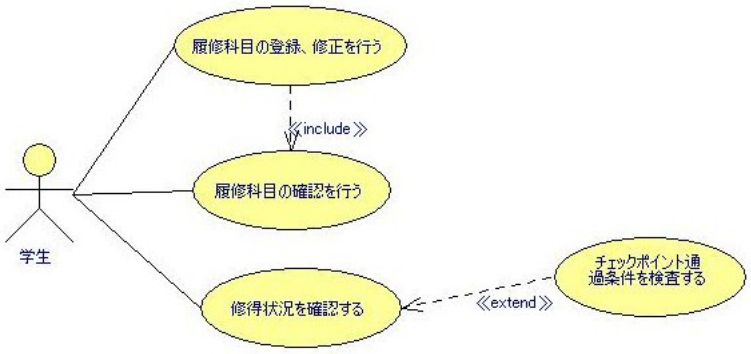

図 4.1: 履修管理システムのユースケース図

学生は、履修管理システムを利用することで履修登録を行うことできる。以下に図 4.1 における各ユースケースの説明を示す。

- 「履修科目の登録、修正を行う」ユースケース
  履修科目の登録および修正を行うことができる。
- 「履修科目の確認を行う」ユースケース
  履修登録を行った科目を確認することができる。
- 「修得状況を確認する」ユースケース
  修得済みの科目名、取得単位数、評価点の一覧を確認することができる。
- 「チェックポイント通過条件をチェックする」ユースケース
  「修得状況を確認する」ユースケースの中で学生が要求した場合に実行される。本学が定める履修規則には、修士課程を修了するために 4 つのチェックポイント (「副

テーマ開始要件」、「研究計画書提出要件」、「就職推薦要件」、「修了要件」) があり、それぞれの要件を満たしていなければチェックポイントを通過できない。このユースケースでは、現在の修得状況における各チェックポイントの通過可否を確認できる。

4.2.2 画面遷移

本システムはウェブアプリケーションであるため、画面遷移と画面レイアウトを設計した。その結果を図 4.2 に示す。

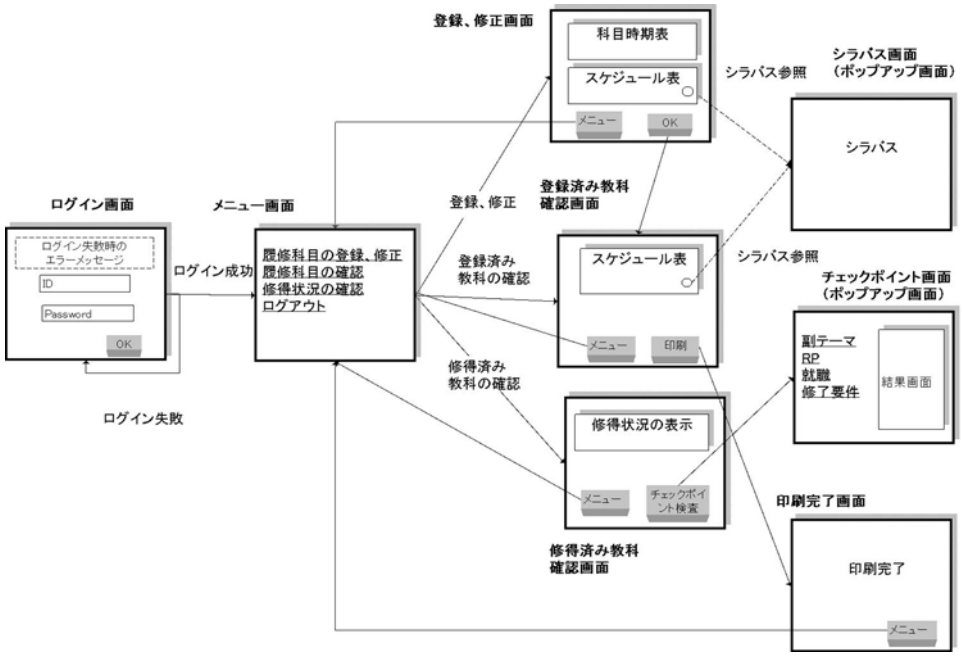

図 4.2: 画面遷移図

メニュー画面のメニュー項目が各ユースケースに対応する。ログイン後すべての画面で「メニュー」ボタンがあり、いつでもメニュー画面に戻ることができる。

4.2.3 プラットフォームおよびウェブアプリケーションフレームワーク

現在、複数のウェブアプリケーションフレームワークが存在し、利用可能である。本開発では、JavaEE プラットフォームを選択し、フレームワークとして JBossSeam[8] を採用した。JBossSeam は、プレゼンテーション層の技術である。JSF(JavaServerFaces) と分散コンポーネント技術 (Enterprise JavaBeans)3.0 を統合して使用可能であり、ステートフルコンポーネントを扱うことができ、宣言的な方法でアプリケーションの状態管理を行うことが可能であるといった特徴のあるフレームワークである。

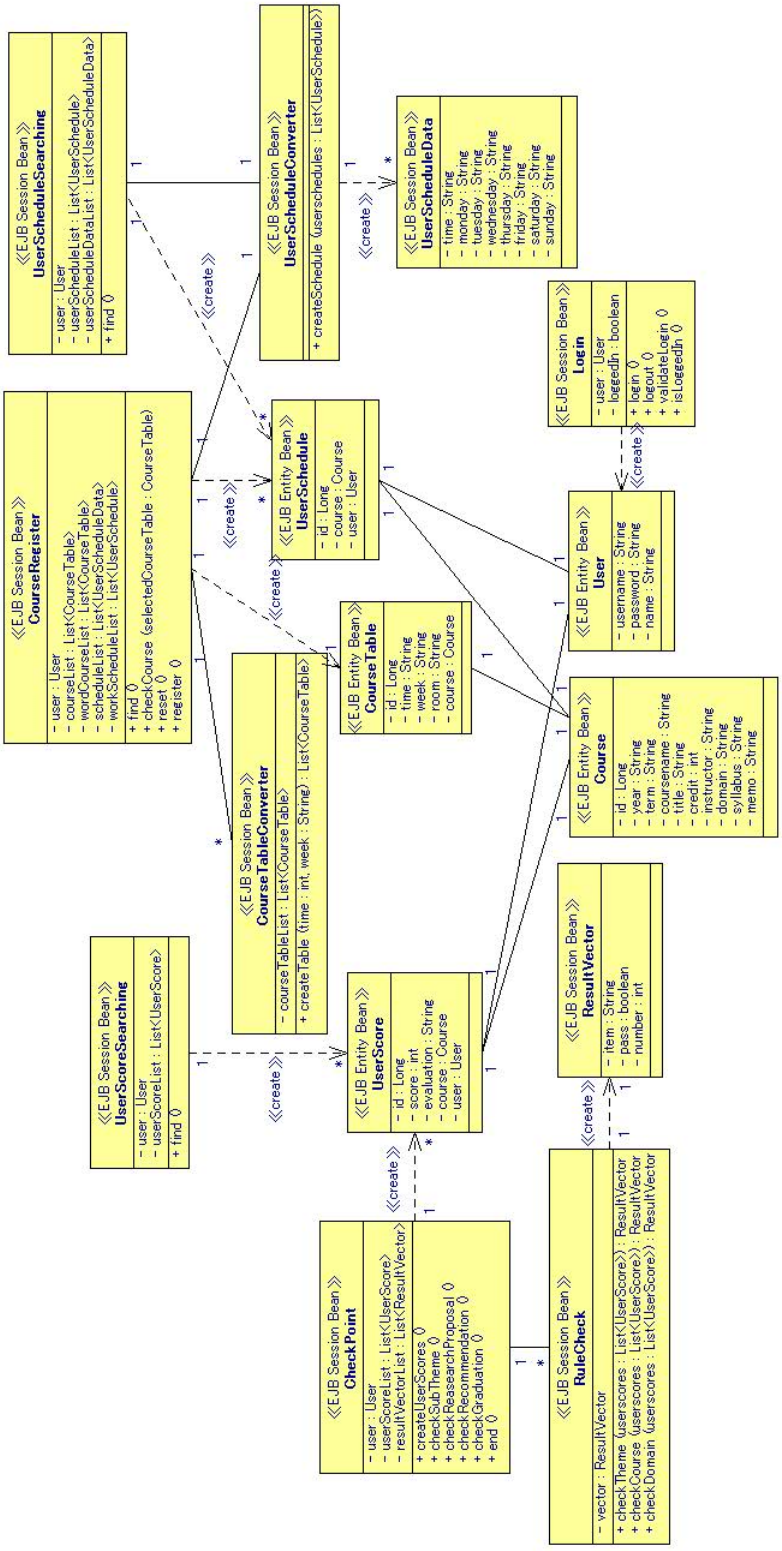

図 4.3: 履修管理システムのクラス図

4.2.4 クラス図

4.3 図にクラス図を示す。「EJB Entity Bean」ステレオタイプのついたクラスは O/R (Object/Relational) マッピングされたクラスであり、プロパティはリレーショナルデータベースに永続化される。「EJB Session Bean」ステレオタイプのついたクラスはビジネスロジックを実装したクラスであり、画面遷移などのイベントに応じて呼び出されるメソッドを持つ。

4.3 アカウンタビリティモジュール

アカウンタビリティ機能を提供するアカウンタビリティモジュールは、研究事例として開発した履修管理システムに追加するかたちで EJB コンポーネントで実現を 行う。

図 4.4 にアカウンタビリティモジュールのクラス図を示す。

アカウンタビリティ機能の呼び出しとその結果である説明の表示を行う必要があるため、履修管理システムのユーザインタフェースの一部に変更を加える。アカウンタビリティ機能のユーザインタフェースはどのようなものが良いかさまざまな角度から検討する必要があるが、本研究では、システムの実行結果を表示する画面においてアカウンタビリティボタンを追加し、利用者がそのボタンをクリックするとアカウンタビリティ機能が呼び出される方式を用いる。

具体的には、図にしめした画面遷移図の「チェックポイント通過条件を検査する」画面内の検査結果表示部にアカウンタビリティ機能を呼び出す「Accoutability」ボタンを追加する。学生がこのボタンをクリックするとウィンドウがポップアップし、質問リストが表示される。学生は質問リストから 1 つの質問を選択することができる。1 つの質問に対して学生が選択を行うと、その質問に対する結果として「履修規則」および、利用者の状況となる「学生の履修状況」が表示される。

実装としては、学生が「Accoutability」ボタンをクリックすると、質問リストを作成するロジックを実装したセッション Bean のメソッドが呼ばれ、履歴情報 DB から読み込まれたデータをクラス定義書により変換を行う。次に、それをもと に質問リストを作成し、ユーザインタフェース層に返す。質問リストは利用者がシステムに対して行った行為の内、最新の 5 項目を表示する。利用者はその内の 1 つの質問を選択することができる。学生が利用者が質問を選択すると、回答を作成するロジックを実装したセッション Bean のメソッドが呼ばれる。選択された質問のデータをもと に、履歴情報 DB と社会規則 DB から該当するデータを読み込み、ユーザインターフェース層に返す。

4.3.1 クラス定義書

クラス定義書には、通常の開発者用の情報に加えて以下の情報を書く。

- クラス名

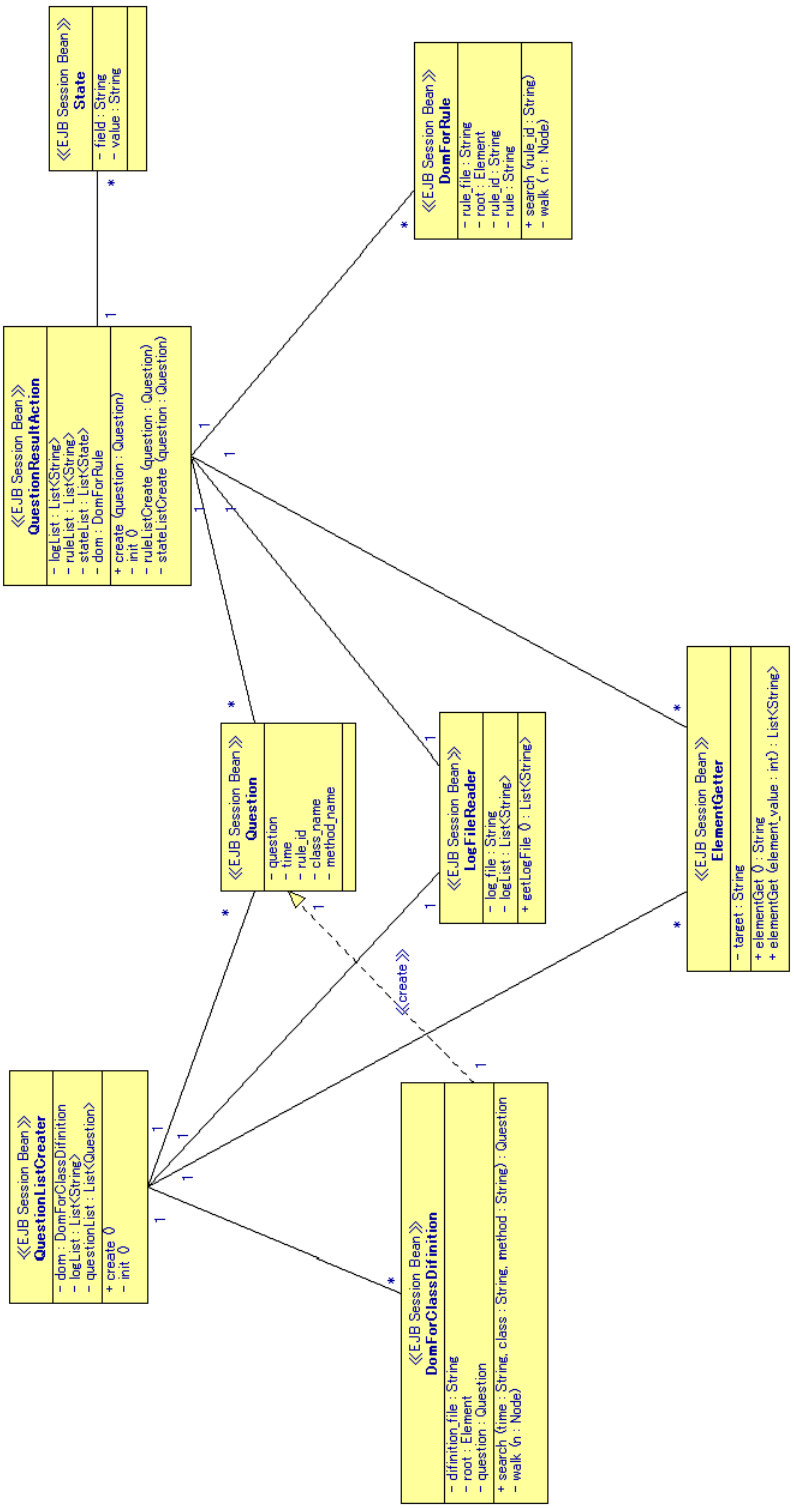

図 4.4: アカウンタビリティモジュールのクラス図

- メソッド名
  そのクラスが持つ社会規則に関連のある、すべてのメソッドに対して書く。

  - メソッドの機能の説明
    システムの利用者が社会規則をどのように利用しているかという観点から、メソッドの機能を利用者に理解できる言葉で表現する。
  - そのメソッドが関連しているすべての社会規則

4.3.2 履歴情報

履歴情報として必要である情報は以下である。

- 利用者からの要求によりシステムが行った処理の情報
- 利用者の状況を表す情報

次に、それぞれの履歴情報の収集を行う際に、対象となる処理を述べる。
「利用者からの要求によりシステムが行った処理の情報」に関しては、ユーザインタフェース層とプロセス管理層との間で機能呼び出しを行う処理を収集の対象とする。「利用者の状況を表す情報」に関しては、「利用者からの要求によりシステムが行った処理」の間に履修管理システムの DB に対して読み書きを行った処理を収集の対象とする。
フォーマットは以下のようになる。

- 利用者からの要求によりシステムが行った処理の情報
  ユーザインタフェース層とプロセス管理層との間で行われる機能呼び出しを持つすべてのセッション Bean に対して

  - タイムスタンプ
  - 処理の始まり、もしくは処理の終り
  - クラス名
  - メソッド名

- 利用者の状況を表す情報
  すべてのエンティティBean に対して

  - クラス名
  - メソッド名
  - メソッドの引数および戻り値

4.4 動作例

実行画面のスナップショットを示し、それについて説明する。

まず、学生がアカウンタビリティボタンをクリックすると質問リストが表示される。図 4.5 に質問リストを表示したスナップショットを示す。

図 4.5: 質問リストを表示したスナップショット

質問リストの 1 つの質問をクリックすると、その質問に対する回答として、履修規則と学生の履修状況が表示される。図 4.6 に質問に対する回答を表示したスナップショットを示す。

図 4.6: 質問に対する回答を表示したスナップショット

第5章　評価

本研究では、アカウンタビリティモジュールに対して機能面と性能面の2つの側面から評価を行った。文献 [12] において、JBossSeam がサーポートするインターセプタ機構を利用してアカウンタビリティ機能の実現を行っている。

評価の比較対象として、この実現方式によりアカウンタビリティ機能の実現を行った履修管理システムをとりあげて評価を行った。まず、JBoss が提供するインターセプタ機構について述べる。

JBoss Seam はインターセプタ機構をサポートしており、これを利用して実行履歴を修得する。具体的には、まず、EJB 配備記述ファイル ejb-jar.xml ファイルにインターセプタを動作させる対象コンポーネントとして、ユーザインタフェース層から呼び出されるコンポーネントすべてに設定する。インターセプタクラスを定義し、前処理や後処理において呼び出し先のコンポーネント名、メソッド名、引数の値、メソッドの戻り値を実行履歴 DB に書き出す処理を行う。

5.1　アカウンタビリティモジュールの機能評価

説明モデルで実現されたアカウンタビリティ機能とインターセプタ機構で実現されたアカウンタビリティ機能を比較し、本方式の機能面での評価を行う。

説明モデルおよびアカウンタビリティ機構で実現したアカウンタビリティモジュールは、それぞれ実現法式が異なる。だが、「システムの利用者を満足させる回答をシステムの実行状況を利用して生成する」という方針は同様であるため、ここでは、2つの方式の特徴を比較し評価を行う。

以下に、それぞれの方式で実現されたアカウンタビリティモジュールが利用者に提示する回答を示す。まず、この結果に対して考察する。

アカウンタビリティ機能の実行結果として説明モデルは、履修規則と学生の履修状況の2つの項目を利用者に提示することができる。これらの情報は、表示するのみで履修状況と規則の適応を示すことができない。また、履修状況として提示される情報のうち、重複した情報などの不必要な情報が含まれる。さらに、これらの情報は開発者よりの情報である。

これに対し、インターセプタ機構は、履修規則と規則の制定目的、学生の履修状況、説明文の4項目を提示できる。説明文に関しては履修規則と規則の制定目的、履修状況の3つの情報を組み合わせて学生にとってわかりやすい説明文を生成するしており、規則との

図 5.1: 説明モデルの回答のスナップショット

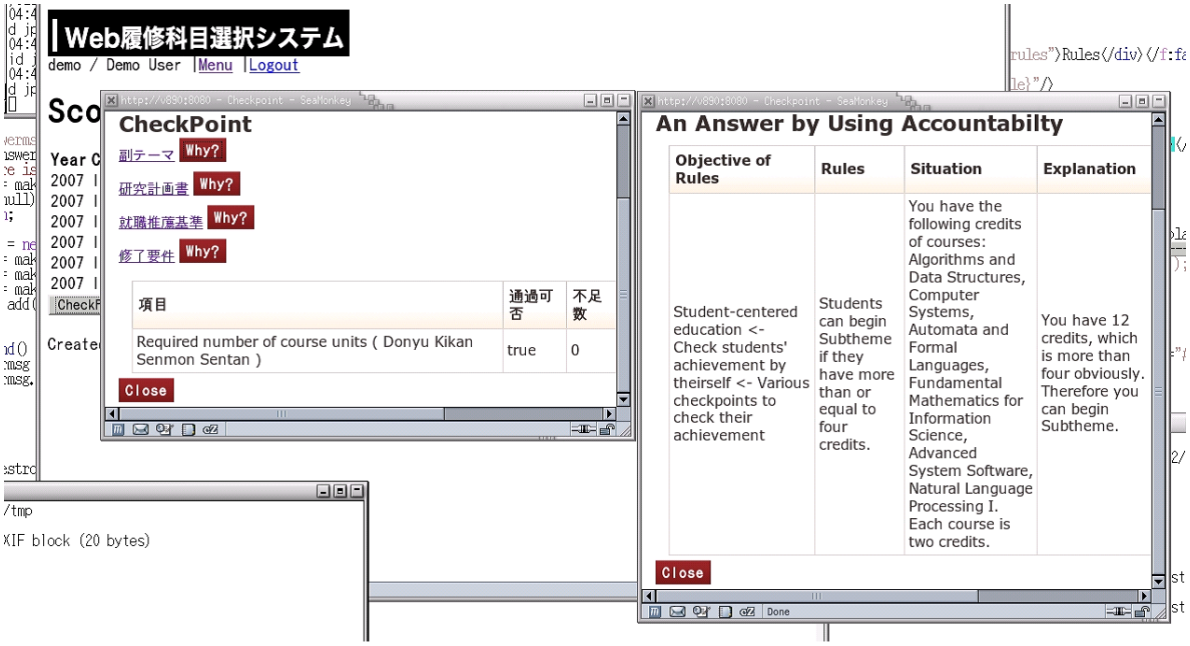

図 5.2: インターセプタ機構の回答のスナップショット

適応を示し説明することができる。
結果として、本方式では以下のことが言える。

- 利用者の状況に含まれる重複などの不必要な情報をはぶくためのフィルタリングが必要である。
- 利用者に情報を提示するのみで規則との適応を示すことができない。これは、説明モデル中に規則との対応を示すための機構が組み込まれていないからである。
- 利用者の状況を、より見やすくする必要がある。

5.2 アカウンタビリティモジュールを追加した履修管理システムの性能評価

履修管理システムの実行履歴を収集する際、オーバーヘッドが問題となる。本研究では、実行履歴を収集する機構を javassist と instrumentation の 2 つの技術を組み合わせて実装した。ここでは、性能測定を行うことにより運用システムとして十分であるかを評価した。
文献 [11] の指針に従って、履修管理システムに対して性能評価を行った。

1. 性能評価の目的と性能評価システムの定義

   アカウンタビリティモジュールを追加した履修管理システムが利用者の要求に対して、十分な速度で動作するかを確認することが性能評価の目的である。

   性能評価の対象となるシステムに関して、アカウンタビリティモジュールを追加した履修管理システムを対象とする。履修管理システムはクライアント・サーバ方式のウェブアプリケーションである。履歴情報の収集はサーバ側で行われる処理であり、クライアント側のブラウザの処理速度、ネットワークのプロトコルの通信速度には影響されないので、サーバ側で動作しているアプリケーションのみを対象とする。

2. 対象システムのサービス
   ここでは、履修管理システムが学生に対して提供するサービスについて考察する。

   履修管理システムは学生に対して、履修科目の登録・修正、履修科目の確認などのサービスを提供している。つまり、図 4.1 のユースケース図のユースケース「履修科目の登録・修正を行う」やユースケース「履修科目の登録を行う」の「凩」レベルで書かれたユースケースがサービスに相当すると考えられる

3. パフォーマンスメトリックスの選択

   今回の性能評価では、システムエラーは特に考慮しない。システムに対するエラーを考慮しない場合、パフォーマンスメトリックスは以下である。

- レスポンスタイム
- スループット
- ユーティリゼーション

4. システムパラメータとワークロードパラメータ

   性能評価は以下の性能を持つシステム上で行った。

   OS： Windows

   CPU： 1.5GHz

   メモリ： 0.99Hz

   ワークロードパラメータの選択は、履修管理システムが行う処理の内、アカウンタビリティモジュールによって実行履歴の収集が行われている処理に着目する。以下の項目がワークロードパラメータとして挙げられる。

   - 図 4.1 で示した、各ユースケースで行われる処理の実行回数
   - データベースのアクセス回数

5. ファクターの選択

   ロギング手法の種類として、本研究で提案した javassist と instrumentation の技術を利用する手法と JbossSeam が提供するインターセプタ機構を利用する手法がある。

   インターセプタは、JBoss フレームワークによって呼び出され実行されるが、呼び出しにかかる実行時間を計測することは実装上難しいので、この評価では、インターセプタによって前処理、後処理がおこなわれる位置に、実行履歴を出力するためのプログラムを書く方法をとった。

6. 評価の仕方

   性能評価は、まず、ユースケースの種類と実行回数を決める。つぎに、これらのユースケースを順に実行するテストケースを実装し、実行時間の測定を行い、そこで得られた結果を分析して評価を行う。

   テストケースの実装に関して、上記で述べた 2 種類のロギング手法に加え、実行履歴を収集しない場合の 3 種類の方法 でテストケースを実装する。結果として得られる以下の 2 つを比較することで評価を行う。

   - 実行履歴を収集しない場合の実行時間とインターセプタ機構を利用した場合の実行時間との差分
   - インターセプタ機構を利用した場合の実行時間と説明モデルを利用した場合の実行時間との差分

テストケースを考えるにあたって、履修管理システムを利用する学生が集中してシステムにアクセスを行った場合を考える。本学の学生、250 人が以下の行為を逐次的に行った場合をテストケースとし、実行時間を測定した。

ユースケース「チェックポイント通過条件を検査する」の実行： 3 回

ユースケース「履修科目の確認を行う」の実行： 1 回

ユースケース「履修状況の確認を行う」の実行： 2 回

7. 得られたデータの分析

表 5.1 に測定したデータを示す。単位はミリ秒である。

表 5.1: 計測されたデータ

実行履歴を収集しない	インターセプタ機構を利用	説明モデルを利用
8,125	8,188	88,250
6,203	9,234	89,985
9,187	11,109	89,891
8,485	5,469	88,063
5,797	11,937	89,406
10,344	10,328	87,031
8,437	5,250	88,437
8,750	11,203	88,328
8,672	10,047	88,953
9,438	5,687	87,344

測定したデータに散らばりがあるので 10 回測定を繰り返し、それぞれのデータに対して平均を出した。

実行履歴を収集しない場合： 8,343.8 ミリ秒

インターセプタ機構を利用して収集を行った場合： 8,845.2 ミリ秒

説明モデルを利用して収集を行った場合： 88,568.8 ミリ秒

実行履歴を収集しない場合の実行時間を基準値 100 %とみたとき、対する 2 つのロギング手法で収集した場合の実行時間の割合を以下に示す。

実行履歴を収集しない場合： 100 %

インターセプタ機構を利用して収集を行った場合： 106.0 %

説明モデルを利用して収集を行った場合： 1061.5 %

8. 考察

   実行履歴を収集しない場合とインターセプタ機構を利用して収集を行った場合との実行時間は、ほぼ同様の割合である。また、説明モデルを利用して収集を行った場合の実行時間は、インターセプタ機構を利用して収集を行った場合の実行時間の約 10 倍である。

   この負荷の原因は、利用者の状況の情報を収集するために、データベースにアクセスする、すべてのエンティティBean クラスの実行履歴を取得しているためである。

   本研究で提案した実現アーキテクチャを用いて、アカウンタビリティ機能を実現した場合、性能評価の結果として、以下のようなことが言える。

   負荷の高い原因は、エンティティBean が収集する履歴情報の量に影響しており、システムに対するデータベースのテーブル数によりシステムの負荷が高くなると言える。

   だが、履修管理システムは、利用者とシステム間でやり取りを行うインタラクティブなシステムである。また、クライアント・サーバ方式のシステムであり、サーバマシンよりもクライアントマシンの方が性能が低く処理時間も遅くなる場合が多い。さらに、クライアントとサーバ間でネットワーク遅延が発生する可能性も考えられる。

   これらを考慮すると、10 倍の処理速度でありパフォーマンスの面では十分とは言えないが、本実現法で開発した運用可能なシステムであると考えられる。

第6章　結論−まとめと今後の課題

本研究では、利害関係者の LEIS に対する質問の内、「実行結果に関する質問」を対象としたソフトウェアアカウンタビリティ機能の実現方式を提案した。まず、事例研究として履修管理システムを開発し、実現方式に従って、履修管理システムに追加するかたちでアカウンタビリティモジュールの設計、実装を行った。

本研究の提案した方式では実行履歴の取得する場合に、性能の負荷が特に問題であったため、性能評価を行い本方式の有効性を評価した。実行履歴の収集には、Javassist と instrumentation という 2 つの技術をもちいて行った。

性能評価の結果、十分であるとは言えないが本提案方式で実現したアカウンタビリティモジュールが運用可能であることを示した。

今後の課題として以下があげられる。

- 履修管理システム以外のシステムへの適用
  本研究では LEIS として履修管理システムをとりあげ、このシステムに追加するかたちでアカウンタビリティモジュールを実現した。アカウンタビリティモジュールは、さまざまな LEIS に対して、モジュールを追加するだけで運用可能でなければならない。そのために、アカウンタビリティモジュールを他のシステムへ適用し、本提案方式の有効性を確認する必要がある。

参考文献

[1] 片山卓也. 検証進化可能電子社会–情報科学による安心な電子社会の実現–. 情報処理, vol. 46,No. 5, pp.515–521, 2005.

[2] 外山グループ. 法令執務支援. http://www.kl.i.is.nagoya-u.ac.jp/research/.

[3] 山本修一郎. 要求を可視化するための要求定義・要求仕様書の作り方. ソフトリサーチセンター, 2006.

[4] 自由国民社 (編). 現代用語の基礎知識. 自由国民社, 2006 年度版, 2006.

[5] 落水浩一郎. ソフトウェアアカウンタビリティに関する基礎考察. 信学技報 SS2006–33,pp.49–54, 電子情報通信学会ソフトウェアサイエンス研究会,2006.

[6] T. エックホフ,N.K. ズンドビー. 法システム：法理論へのアプローチ. ミネルヴァ書房, 1997. 都築廣巳 [ほか] 訳.

[7] 杉森隼人, 落水浩一郎. ソフトウェアアカウンタビリティ実現のための GORA と法理論の利用に関する報告. Technical Report IS–RR–2007–005, 北陸先端科学技術大学院大学,2007.

[8] JBoss.org. JBoss Seam. http://labs.jboss.com/jbossseam/.

[9] 早坂良, 堀雅和, 藤枝和宏, 落水浩一郎. アカウンタビリティおよび進化容易性を持つソフトウェアアーキテクチャと—v!—3 層モデルとの対応. 情処研報 2005–SE–150,pp1–8. 情報処理学会ソフトウェア工学研究会,2005.

[10] 早坂良, 落水浩一郎. 履修管理システムにおけるオントロジを用いたアカウンタビリティ設計手法. 情処研報 2006–SE–151,pp.73–80. 情報処理学会ソフトウェア工学研究会,2006.

[11] Raj,Jain. THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS. John,Wiley Sons,Inc.

[12] 早坂良, 秋山裕俊, 杉森隼人, 北山真太郎, 鈴木正人, 落水浩一郎 履修管理システムにおけるソフトウェアアカウンタビリティ機能の実現法 信学技報 SS2007–14, 電子情報通信学会,2007.