

千葉工業大学　社会システム科学部

プロジェクトマネジメント学科

平成25年度　卒業論文

プロジェクトホスティングサービスのためのEVM自動描画システムの開発

プロジェクトマネジメントコース

矢吹研究室

0842062　　工藤亮

千葉工業大学　社会システム科学部

プロジェクトマネジメント学科

平成25年度　卒業論文

プロジェクトホスティングサービスのためのEVM自動描画システムの開発

Development of automatic EVM drawing system for project hosting services

プロジェクトマネジメントコース

矢吹研究室

0842062　工藤亮/Akira KUDO

|  |  |
| --- | --- |
| 指導教員印 | 学科受付印 |
|  |  |

内容

[1.　序論 2](#_Toc378811800)

[1.1　本章の構成 2](#_Toc378811801)

[1.2　研究の背景 2](#_Toc378811802)

[1.3　研究の目的 3](#_Toc378811803)

[1.4　研究の方法 3](#_Toc378811804)

[1.5　プロジェクトマネジメントとの関連 3](#_Toc378811805)

[1.6　本論文の構成 3](#_Toc378811806)

[1.7　参考文献 4](#_Toc378811807)

[2.　ソフトウェア開発プロジェクトについて 6](#_Toc378811808)

[2.1　本章の構成 6](#_Toc378811809)

[2.2　ソフトウェア開発プロジェクトについて 6](#_Toc378811810)

[2.3　ソフトウェア開発の歴史 7](#_Toc378811811)

[2.4　ソフトウェア開発プロジェクトの成功事例 7](#_Toc378811812)

[2.4.1　成功事例 7](#_Toc378811813)

[2.4.2　成功事例の考察 7](#_Toc378811814)

[2.5　ソフトウェア開発プロジェクトの失敗事例 8](#_Toc378811815)

[2.5.1　失敗事例 8](#_Toc378811816)

[2.5.2　失敗原因 8](#_Toc378811817)

[2.6　事例から得られたこと 9](#_Toc378811818)

[2.7　参考文献 10](#_Toc378811819)

[3.　バージョン管理システム 12](#_Toc378811820)

[3.1　本章の構成 12](#_Toc378811821)

[3.2　バージョン管理システムについて 12](#_Toc378811822)

[3.3　バージョン管理システムの種類 12](#_Toc378811823)

[3.3.1　個別バージョン管理システム 13](#_Toc378811824)

[3.3.2　集中型バージョン管理システム 14](#_Toc378811825)

[3.3.3　分散型バージョン管理システム 14](#_Toc378811826)

[3.4　Gitについて 15](#_Toc378811827)

[3.4.1　Gitの歴史 16](#_Toc378811828)

[3.4.2　Gitの特徴 16](#_Toc378811829)

[3.4.3　Gitの利用状況 16](#_Toc378811830)

[3.4.4　GitとGitHubの違い 17](#_Toc378811831)

[3.5　GitHubについて 17](#_Toc378811832)

[3.6　GitHubの利用状況 17](#_Toc378811833)

[3.7　GitHubの基本用語 18](#_Toc378811834)

[3.8　GitHubの機能 19](#_Toc378811835)

[3.9　参考文献 21](#_Toc378811836)

[4.　システム構築について 23](#_Toc378811837)

[4.1　本章の構成 23](#_Toc378811838)

[4.2　APIについて 23](#_Toc378811839)

[4.2.1　APIを公開する・公開されるメリット 23](#_Toc378811840)

[4.3　GitHubのAPIの種類 23](#_Toc378811841)

[4.4　Google Chart APIについて 35](#_Toc378811842)

[4.5　ブックマークレットについて 47](#_Toc378811843)

[4.5.1　ブックマークレットの提案者について 48](#_Toc378811844)

[4.5.2　ブックマークレットの仕組み 48](#_Toc378811845)

[4.5.3　ブックマークレットのプログラム上の特徴 49](#_Toc378811846)

[4.5.4　ブックマークレットの使用方法 49](#_Toc378811847)

[4.6　参考文献 50](#_Toc378811848)

[5.　プロジェクトマネジメントについて 53](#_Toc378811849)

[5.1　本章の構成 53](#_Toc378811850)

[5.2　プロジェクト作業の監視・コントロール・プロセス群について 53](#_Toc378811851)

[5.3　各知識エリアでの識別 57](#_Toc378811852)

[5.4　EVM（Earned Value Management）の概要 60](#_Toc378811853)

[5.5　EVMの関連するプロジェクトマネジメント項目について 61](#_Toc378811854)

[5.6　参考文献 62](#_Toc378811855)

[6.　システムについて 64](#_Toc378811856)

[6.1　本章の構成 64](#_Toc378811857)

[6.2　システムを使用する際の条件 64](#_Toc378811858)

[6.3　EVMを作成するために必要な数値のデータ 64](#_Toc378811859)

[6.4　EVMの計算方法 65](#_Toc378811860)

[6.5　GitHub APIで抽出するデータ 65](#_Toc378811861)

[6.6　GitHub APIで取得したデータの使用目的 65](#_Toc378811862)

[6.7　GitHubからデータ抽出する方法 66](#_Toc378811863)

[6.8　GitHubから複数ページにわたるデータを取得する方法 67](#_Toc378811864)

[6.9　GitHub APIで取得したデータの処理方法 70](#_Toc378811865)

[6.10　IssuesのデータをGitHubから抽出する方法 70](#_Toc378811866)

[6.11　システムの概要 72](#_Toc378811867)

[6.12　システムの機能 72](#_Toc378811868)

[6.13　参考文献 73](#_Toc378811869)

[7.　システム検証について 75](#_Toc378811870)

[7.1　本章の構成 75](#_Toc378811871)

[7.2　検証方法 75](#_Toc378811872)

[7.3　検証用のGitHub内の仮想プロジェクトデータのリポジトリ内容 75](#_Toc378811873)

[8.　結論 84](#_Toc378811874)

[8.1　結果 84](#_Toc378811875)

[8.2　結論 84](#_Toc378811876)

[8.3　今後の課題 84](#_Toc378811877)

第1章

序論

# 1.　序論

## 1.1　本章の構成

第1章では，本論文の序論について述べる．研究背景，目的，方法，プロジェクトネジメントとの関係，本論文の構成について述べる．

## 1.2　研究の背景

ソフトウェア開発はバージョン管理システムを用いて行われるのが一般的である．バージョン管理システムが一般的になった理由は， プロジェクトを運営していく上でいろいろな場面で役立つということである． 開発者どうしのコミュニケーション，リリース管理，バグ管理，コードの安定性の確保，安心して新機能を実験できる環境，各開発者の権限の管理など，あらゆる場面でバージョン管理が利用することができるのである．バージョン管理システムは，これらの内容をひとまとめにして管理する．その中心となる機能が，変更管理である．これは，プロジェクト内のファイルが変更されるたびに，その変更についてのメタデータ（更新日や更新者など）を収集する仕組みである．そして，あとからその変更の内容を再現できるようにするのである．つまり，変更が発生した単位で情報を管理する仕組みといえるのである．

バージョン管理システムはソースコード管理システムなどとも呼ばれ，大規模な開発を行う際には必須と言っても過言ではないのである．また，大規模な開発だけでなく小規模な開発や個人による開発においても，ファイルの変更履歴の記録やバックアップといった用途で活用されており，プロジェクト内のさまざまなファイルの変更履歴を管理するためのテクノロジーや習慣を組み合わせたものである．バージョン管理システムを提供するサービス（以下，プロジェクトホスティングサービス）もSubversionやCVS（Concurrent Versions System），Git，Mercurial，Visual SourceSafeなどと数多くある．

ソフトウェア開発プロジェクトの中心にバージョン管理システムをおくことができる．そのような開発体制においても，一般的なプロジェクトと同様に，PMBOKで提示されているようなWBS（Work Breakdown Structure）やEVM（Earned Value Management）のようなマネジメントの活動がプロジェクトホスティングサービスでサポートされることが望ましいのである．

プロジェクトマネジメントにおいて，プロジェクトの進捗状況や進捗に係わるリスクを把握し，コントロールするためにEVMが用いられる．EVMは，プロジェクトを計画通りに進行させ，進捗状況に対応した判断を行うのに必要であり重要である．GitHub（バージョン管理システム，Gitを提供するサービス．ギットハブ）のような既存のプロジェクトホスティングサービス上では，EVMを見ることはできないである．

そのため，バージョン管理システムを用いるプロジェクトマネジャーは，計画時のコストのデータと計画時の時間のデータを進行中の進捗時のコストのデータと進捗時の時間のデータを取り出して，別々に処理することによってEVMを描かなければならないのである．

## 1.3　研究の目的

GitHub上で自動的にEVMを描画するシステムを開発する．このシステムにより，プロジェクト関係者がプロジェクトの進捗状況を手軽に把握できる．

その結果，このシステムによりプロジェクトマネジャーを含むメンバー全員がプロジェクトの進捗状況を手軽に把握できるようになる．その結果，作業の遅延などへの対応（リスク・マネジメント）やスケジュール管理（タイム・マネジメント）が効率よく行えるようになることも期待できる．

## 1.4　研究の方法

本研究では，バージョン管理サービスのGitHubを調査し，EVMに必要なデータとして，計画時のコストのデータと計画時の時間のデータ，進行中の進捗時のコストのデータ，進捗時の時間のデータをIssueに記述する仕様を決定する．そして，Issueに記述されたデータを抽出するプログラムを作成し，Issueから抽出されたデータをもとにEVMを描画するシステムを構築する．その後，過去のプロジェクトのデータを構築したEVMを描画するシステムに適用し，システムの運用とシステムの検証によるシステムテストを行う．

## 1.5　プロジェクトマネジメントとの関連

　本研究では，バージョン管理サービスのGitHubのデータの中からEVMに必要なデータを抽出し，そのデータをもとにEVMを自動的に作成できるシステムを構築する．このシステムにより，プロジェクトマネジャーを含むプロジェクトメンバー全員がプロジェクトの進捗状況を手軽に把握できるようになる．その結果，作業の遅延などへの対応が可能になり，リスク・マネジメントに貢献することができる．また，スケジュールの管理などへの対応も可能であり，タイム・マネジメントにも貢献することができる．

## 1.6　本論文の構成

　第1章は，序論，第2章から第4章に関しては，各項目の基礎知識，問題点，必要性などをまとめていく．第5章から第6章までは実験の方法，考察，結果を記し結論をまとめていく．

　第1章では，研究の背景，目的，方法，プロジェクトマネジメントとの関連を記述していく．第2章では，ソフトウェア開発について調査する．ソフトウェア開発の歴史，成功事例，失敗事例，失敗事例の原因などを調査する．第3章ではGitHubの基礎知識を述べる．バージョン管理システムについての調査，バージョン管理システムの種類，成功事例などを述べる．第4章では，システムを構築する時に必要な知識について述べる．第5章では，マネジメントとEVMについて述べる．本研究ではコスト・マネジメントとタイム・マネジメントについて述べる．第6章ではシステムについて述べる．システムを使用する際に必要なシステム構造を述べる．第7章では，開発したシステムの検証と検証方法，検証結果について述べる．第8章では検証結果から，結論を記述する．

## 1.7　参考文献

[1] Project Managemant Institute. PMBOK Guide. 第4版, PMI, 2008, p5-6.

[2] 松島造道. CVS-Subversionを使ったバージョン管理. SOURCEFORGE.JP. http://sourceforge.jp/magazine/08/09/09/1038233, 2013-09-26.

[3] Fogel Karl. バージョン管理 第3章 技術的な問題. オープンソースソフトウェアの育て方. http://producingoss.com/ja/vc.html, 2013-09-27.

第2章

ソフトウェア開発プロジェクトについて

# 2.　ソフトウェア開発プロジェクトについて

## 2.1　本章の構成

　本章では，本研究はバージョン管理システムを中心におく開発体制になる前のソフトウェア開発プロジェクトについて調査する．ソフトウェア開発プロジェクトは以前ではどのように行われていたのかを調査し，ソフトウェア開発プロジェクトの歴史を知る．また成功事例，失敗事例，失敗事例の原因などを調査することによって，バージョン管理システムの必要性を明確にする．

## 2.2　ソフトウェア開発プロジェクトについて

　ソフトウェア開発プロジェクトはプロジェクトの開始時から多くの失敗の可能性があると予想される．ソフトウェア開発プロジェクトにはいくつかの特徴的な傾向がある．最近のソフトウェア開発プロジェクトの特徴は以下のようである．

* 従来の開発に比べて期待される開発工期が相対的に短い．
* 要求仕様が固まっていない状態で開発を始めねばならないケースが多い．
* さらに要求の追加・変更が開発中にも継続的に発生にする．
* 利用すべき技術や製品が次々に登場・改訂される中で開発しなければならない．
* アプリケーション固有画面，Web画面，i-モード，そのほか携帯端末などのさまざまなインターフェイスに対応しなければならない．

経済産業省が2005年に公開した仕組みソフトウェア産業実態調査[5]によれば，80%以上

のプロジェクトの開発期間が1年未満であることがグラフから読み取れる．規模が増大しているにもかかわらず，期間は短いままである．結果として，プロジェクトに投入される技術者が増え，最近では数百人規模の開発体制も増加傾向にある．ソフトウェア開発プロジェクトでは開発期間が最近では6ヶ月未満で納品する傾向にある．

このグラフから開発期間が短いことがわかる．そのため，期間内に終わらせるためには，ソフトウェア開発プロジェクトのプロジェクトマネジャーはプロジェクトの進捗管理には細心の注意が必要になる．

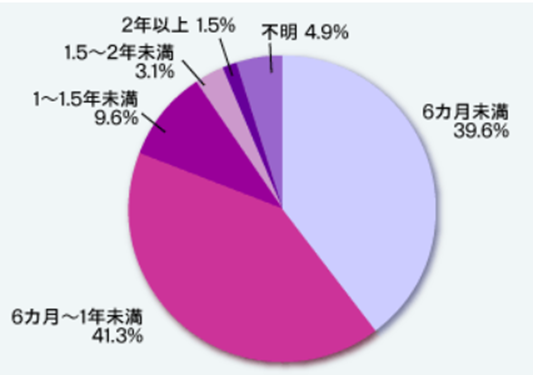


図1-1　ソフトウェア開発プロジェクトの平均的な開発期間[4]

## 2.3　ソフトウェア開発の歴史

　1970年代の後半から1980年代にかけてITの大規模システム構築が急激に発生したが，プロジェクトマネジメント技術や方法論が未熟だったために，勘と度胸で実行せざるを得なかったのである．その結果，多くのプロジェクトに問題が発生したのである．こうした中で数少ない大規模プロジェクトを成功に導いたプロジェクト・リーダーがスーパーSEとして話題になったのである．このためにスーパーSEの育成方法が話題となり，各方面で色々試行したが，スーパーSEの育成は極めて困難なことが分かった．このことを境にスーパーSEを頼る以外の成功手法として「プロジェクトマネジメントの方法論」や「プロジェクト組織の仕組み」などを研究したのである．また，企業経営手法やスポーツチームの成功事例から，人間関係や心理的側面もプロジェクトマネジメントの重要な要素の一部だということも分かったのである．更に，プロジェクト・チームの組織能力向上という視点から，CMM（組織の能力を評価する方法論）が米国で発案され，CMMを組織能力評価のみならず組織能力向上に利用することも始まったのである.[5]

## 2.4　ソフトウェア開発プロジェクトの成功事例

　ここでは，過去に行われたプロジェクトで成功した事例を考察する．事例から，なぜこのソフトウェア開発プロジェクトは成功したのかを考察する．

### 2.4.1　成功事例

　このプロジェクトは日本の企業と中国の企業が協力して行うプロジェクトである．プロジェクト全体のプロジェクトマネジャーは10年以上も開発経験をもつ人物である．内容は要件定義，外部設計は日本の企業が担当し，内部設計，プログラム開発以降は中国，システムテストは日本で行うということになったのである．中国の企業は品質，進捗の管理を独立して行われ，日本の企業へは毎週報告することに決められていたのである．使用についての質疑応答，変更の管理は中国の企業と日本の企業とで決められたやり方でやり取りしていたのである．中国の企業でプロジェクト進行中の問題解決として，日本の企業の技術者にレビューを頼むことになったのである．プロジェクトは当初，決められた計画内のスケジュール中に成果物を納品することができ，品質の面でもバグ数は許量範囲内であり，日本の企業に中国の企業から納品した後のシステムテストは，無事に成功したのである．その後，ユーザーテストを終えて稼働することができたのである．この開発体形は，オフショア開発である．

### 2.4.2　成功事例の考察

　このソフトウェア開発プロジェクトの成功事例から成功した要因は，開発プロジェクトを担当したプロジェクトマネジャーの経験が豊富だという点にある．10年以上も開発の経験があったことで必要作業項目も詳細に指示をすることができる．また，進捗管理に関しても細かい双方の企業での情報のやり取りも毎週やることによりプロジェクトマネジャーが進捗状況を把握することもできる．その結果，中国の企業での進行中の問題を解決するために日本の企業の技術者にレビューを任せることにより，問題解決ができたのである．プロジェクトはスケジュール中に成果物を納品することができ，その後のシステムテストとユーザーテストも完了することができたのである．このソフトウェア開発プロジェクトの開発体形はオフショア開発というプロジェクトの参加メンバーが離れたところにいながら，ネットワークを通じてプロジェクトを進行していく開発体形である．プロジェクトに重要なコミュニケーションをとりにくい環境である中，進捗管理のために毎週の報告をすることでコミュニケーションを可能にし，進捗状況を把握することができたのである．

## 2.5　ソフトウェア開発プロジェクトの失敗事例

　ここでは，過去に行われたソフトウェア開発プロジェクトの事例を考察し，事例の原因からソフトウェア開発プロジェクトにおけるリスク分析を行う．

### 2.5.1　失敗事例

　このソフトウェア開発プロジェクトはデンバー空港の自動手荷物運搬システムを開発するプロジェクトである．1993年の10月29日に，その当時の空港としては非常に巨大で最新の設備を備えたデンバー国際空港がオープンする予定でした．しかし，なぜかオープンの日になっても自動手荷物運搬システムだけはうまく起動しなかったのである．その結果として，空港のオープンが16ヶ月ほど遅延して1995年の2月28日にオープンしたのである．手荷物運搬システムがないのは空港として常識的に考えられないという一般的な考えから空港はオープンの遅延をよぎなくされたのである．その遅れによって，問題の自動手荷物運搬システムを構築していた会社は，デンバー市から空港のオープンの遅延の代償として1日ごとに12,000ドル（1994年の為替レート換算で約120万円）を請求されることになってしまったのである．さらに，起動しなかった自動手荷物運搬システムのバックアップシステムを構築するために，5,000万ドル（同じく1995年で約45億円）を追加請求されたのである．さらに，構築したバックアップシステムが2000年問題に対応していなかったため，さらに巨額のコスト10億円が費やされたのである．

### 2.5.2　失敗原因

　問題の原因が明確にはなってないが，問題定義としてリスク分析や進捗状況の管理が曖昧だと考えられる．また，組み込みテストを行う場合はCPUが熱で暴走させないために，CPUを休み休みテストを行わなければならないのである．そうした遅延やあるいは手戻りがあり進捗計画時に遅延などのリスク計画とそのリスクによる作業の遅延を予想したスケジュールの管理が必要だったのである．さらに，このプロジェクトでは要求が変更されることがたびたびあったのである．このプロジェクトはソフトウェア開発会社だけでなく，デンバー市や航空会社も係わっているプロジェクトであったために，開発チーム以外からも仕様変更などが多数突きつけられてしまったのである．

　つまりこのプロジェクトは，変更要求の多い「外的要因」を抱えていたのにもかかわらず，トラブル耐性の低い体質であったうえに，それらへの対策を行うはずのプロジェクトマネジャーを失うという「内的要因」が重なった結果が原因である．

## 2.6　事例から得られたこと

　成功事例と失敗事例から得られたことは，成功事例では，進捗状況を把握するためにコミュニケーションを必要とするが，開発体形はプロジェクト関係メンバーが離れて活動するというオフショア開発であったが，毎週の報告をすることによりプロジェクトマネジャーが進捗状況を把握することができる．そして，問題が発生した場合でも対処できたということである．失敗事例では，失敗要因となる問題項目が多々ある．まず，内的要因はプロジェクトの進行中にプロジェクトマネジャーが変更になることである．次に，組み込みテスト時にはCPUが熱暴走させないために，CPUを休み休みテストを行うことにより，作業の遅延を招いてしまったのである．外的要因は，プロジェクト規模が大規模であることである．次に，顧客からの要件定義の変更の回数が多く，そのためにシステム開発の不具合も起きやすくなったのである．この場合の原因は，進捗管理を行うプロジェクトマネジャーがプロジェクトの進行中に変更になってしまったことにより，進捗状況に応じて対応できないことである．

## 2.7　参考文献

[1] 伊藤誠一. ソフトウェアの成功と失敗. 初版, 共立出版株式会社, 1997.

[2] 株式会社アイティネット. ITプロジェクト失敗の原因と対策. ITプロジェクト管理考.http://www.itnetinc.co.jp/ITPMopinion/projectfailurereason.htm, 2013-09-30.

[3] 国際大学法人東京農工大学. なぜソフトウェア開発にプロジェクト管理が必要か. ソフトウェア開発環境の概要. http://www.tuat.ac.jp/~asiaprog/courses/project/lesson03/, 2013-09-30.

[4] ITmedia Inc. ETSSは人材&スキル不足を解決できるか. MONOist. http://monoist.atmarkit.co.jp/mn/articles/0601/27/news122.html, 2013-09-15.

[5] システム管理者の会. IT業界におけるプロジェクトマネジメントの大まかな歴史. システム管理者の会ポータルサイト. http://www.sysadmingroup.jp/kh/rpt/001/34.html, 2013-10-01.

[6] 勝呂暖生. ソフトウェアプロジェクト管理. 初版, 株式会社翔英, 2012, p16-19.

[7] 北京英普思科技発展有限公司. 成功事例. INPROSYSTEM LED. http://www.inprosystem.com/Templates/cgsl\_jp.html, 2013-10-08.

第3章

バージョン管理システムについて



# 3.　バージョン管理システム

## 3.1　本章の構成

　本章では，最終的な本研究の目的で使用するバージョン管理システムについての基本知識や種類などを述べる．その後，Gitの基本情報やGitHubの現状について述べ，その後，GitとGitHubの違いについて述べた後，専門用語の説明を解説しながら，機能について述べる．

## 3.2　バージョン管理システムについて

　バージョン管理システムは，ファイルの履歴を管理するシステムである．何らかの作業によって生成されたファイルについての複数の履歴を記録し，後から古い履歴の取り出しや，差分の参照ができる仕組みである．このファイルの履歴をリポジトリ，履歴の1つ1つをコミットと呼び，履歴をリポジトリに追加や登録することをコミットすると言うのである．またバージョン管理システムソフトウェアによっては，ファイルの削除や移動の履歴を確認する機能や，特定の利用者がファイルの管理する権限を獲得するロック機能，複数の変更を統合するマージ機能がある．

　バージョン管理システムは，複数人で同一のファイルを編集する必要があるときに，バージョン管理システムの機能が役立つのである．例えば，テキストファイルをファイルサーバに置いて共有していたとする．この状態で編集やコピーを続けていくと，誰が最新版を持っているか，また，誰がいつ，どこを変更したなどの履歴がわからなくなってしまうのである．ここでこのファイルをバージョン管理システムで管理すれば，変更点と変更者，変更日時を管理できるようになるのである．

　またソフトウェア開発においては，バージョン管理システムは特に有効に稼働するのである．変更点を管理できるので，障害がいつ発生したのか，どの時点から問題となっていたのか，いつ修正されたのかなどを簡単に調査できるからである．

## 3.3　バージョン管理システムの種類

　このバージョン管理システム（Version Control System，以下VCS）は，管理の方法から3つの分けることができる．個別バージョン管理システム，プロジェクトメンバーのためのサーバ型の集中型バージョン管理システム，そしてGitに代表される分散型バージョン管理システムである．以下にVCSの種類と代表とするVCSソフトウェアを表3-1に示す．これら3つのVCSについて説明する．

表3-1　バージョン管理システムの種類と概要[2]

|  |  |
| --- | --- |
| VCSの種類 | 概要 |
| 個別VCS | ファイル単位で個別システムのバージョン管理を行う．（SCCS，RCS） |
| 集中型VCS | リポジトリをサーバで一元管理し，コミットなどの操作はプロジェクトメンバーから行う（CVS，Subversion，Perforce） |
| 分散型VCS | リポジトリをプロジェクトメンバーでも管理することができる．このプロジェクトメンバー間でリポジトリの連携ができる．（Git，Bazaar，Mercurial，BitKeeper） |

### 3.3.1　個別バージョン管理システム

　初期のオープンソースソフトウェアとしてのバージョン管理システムといえばRCSが挙げられる．RCSはファイル単位での管理しかできないため，バージョン管理については限定的なことしかできなかったのである．個人での利用であれば問題ないかもしれないが，複数人が利用するプロジェクトでは何らかの運用上の工夫が必要になることが多くなるため，集中型バージョン管理システムであるCVSなどへと発展していくことになったのである．

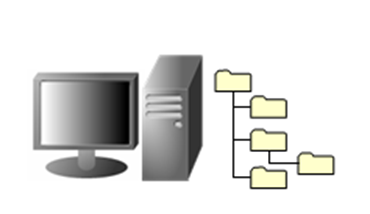


図3-1　個別バージョン管理システム（個人開発）[6]



図3-2　個別バージョン管理システム（少数でのチーム開発）[6]

### 3.3.2　集中型バージョン管理システム

　集中型バージョン管理システムは，複数人で使う場合は通常プロジェクトメンバー・サーバ型の構成をとり，リポジトリはサーバ上に格納されているのである．リポジトリに対する操作はプロジェクトメンバーから行うが，ネットワーク的に接続していなければならないという制約がある．集中型バージョン管理システムとしては，CVSのほかにSubversionや商用ソフトウェアであるPerforceなどがある．これらは現在でも多く利用されているが，次第に分散型バージョン管理システムに移行しつつある．

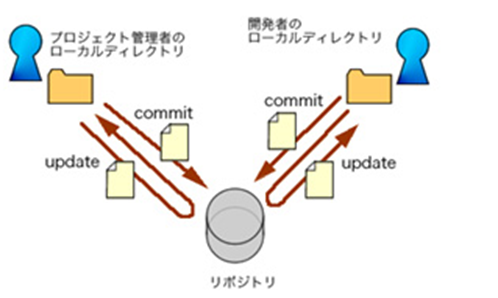


図3-3　集中型バージョン管理[7]

### 3.3.3　分散型バージョン管理システム

　分散型バージョン管理システムでは，リポジトリをサーバ上ではなく，プロジェクトメンバーであるコンピュータ上にも作成するのである．コミットの参照や差分を取得する場合に，手元にあるリポジトリにアクセスするため，集中型にあったネットワーク接続が必要であるという欠点がないのである．また，ローカルファイルシステムにリポジトリがあるので高速に動作することも，利点のひとつである．さらに変更点を送受信する仕組みがあるため，複数のリポジトリ間で連携できるのである．

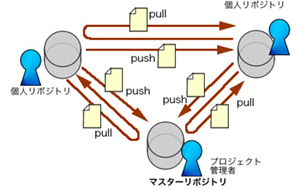


図3-4　分散型バージョン管理システム[7]

以下の表3-2でオープンソースプロジェクトと概要，採用されている分散型バージョン管理システムを示す．

表3-2　オープンソースプロジェクトで採用されている分散型バージョン管理システム[3]

|  |  |  |
| --- | --- | --- |
| プロジェクト | 概要 | VCS |
| Linux | カーネル | Git |
| Android | OS | Git |
| Perl | 言語 | Git |
| Ruby | 言語 | Subversion |
| Ruby on Rails | Webアプリケーションフレームワーク | Git |
| X.Org | グラフィカルウィンドウシステム | Git |
| Samba | Windowsとのファイル共有 | Git |
| Emacs | エディタ | Bazaar |
| Vim | エディタ | Mercurial |
| Mozilla Firefox | ブラウザ | Mercurial |
| Chromium | ブラウザ | Subversion |
| FreeBSD | OS | Subversion |

## 3.4　Gitについて

　Gitは分散型バージョン管理システムであるためサーバを必要としないのである．また各コンピュータ上にリポジトリを持ち，それぞれが互いに連携しあうことができる．さらに，基本的なそれぞれのリポジトリにすべての履歴が保存されるため，差分やログの表示などを高速に行えるのである．

　リポジトリ間連携はネットワーク通信（ssh，http，git，プロトコルなど）やメールを経由して行うのである．ほかにもリポジトリを共有リポジトリとして公開する仕組みや，ユーザー管理と組み合わせる方法があり，集中型バージョン管理システムのような利用形態もとれるのである．

　さらに他のバージョン管理システムとのデータ交換も可能である．既存の他のバージョン管理システムのリポジトリをGitリポジトリへ変更することや，中央リポジトリに他のバージョン管理システムのリポジトリを利用し，手元ではGitを利用する，といった形態をとることもできるのである．

オープンソースのバージョン管理システムとしては，CVSやSubversionが有名で，今でもこれらの集中型システムはよく使われている．しかし，近年になって，Linuxカーネル，X.org，Ruby on Rails，Perlといった有名なプロジェクトがGitに乗りかえて成功裡に使用しているのをみて，Gitを使用し始めようとする分散型バージョン管理システムを使用するプロジェクトは飛躍的に増加してきているのである．

### 3.4.1　Gitの歴史

　Linuxの創始者Linus TorvaldsがGitの開発を始めたのは2005年の春頃のことである．それまで，Linuxカーネル開発のソース管理にはBitKeeperというバージョン管理システムが用いられていたのである．これはBitMover社製の商用のバージョン管理システムである．Bitkeeperは，先進の分散型バージョン管理システムで，カーネルプロジェクトが採用した当時，オープンソースの世界にはこれに匹敵する分散型バージョン管理システムの使用が不可欠であったのである．

　同社はオープンソースプロジェクトに対して無料使用のライセンスを与えていたのですが，どれも機能・性能面で満足できるものではありませんでした．そこでLinuxは，「カーネルプロジェクトから2週間休み」を宣言して現在のGitの原形となるプログラム郡を書き，これを公開したのである．Linus Torvalds氏自身が作成したという背景があり，さらに機能や性能に好評だったためGitはプログラマーに受け入れられる理由の1つである．公開当初は，ハッカーにしか使えないぐらい荒削りのプログラムだったらしいが，多くの開発者の協力を経て，今では世界中のプログラマーに利用されるようになったのである．

### 3.4.2　Gitの特徴

　Gitは，ファイル自身，ファイルの集合としてのツリー，そしてコミット情報という3つの情報を管理しているのである．それぞれの情報はハッシュ値をもとに管理されており，このハッシュ値とはファイルが同一かどうかの判定にも利用されているのである．また，Gitではコミット情報を，SubversionやCVSのような差分管理ではなく，ファイルをそのまま保持しているというのも特徴である．

　さらに時間的な変遷を管理する仕組みreflogや，コミットをメールで送受信する仕組みもある．reflogは時系列でのコミットやブランチ切り替えなど，リポジトリに関する変更を管理する機能である．例えば先週の変更はどのようなものだったのかを後で確認するときなどに効果を発揮するのである．コミットをメールで送受信する仕組みの機能は特にオープンソースプロジェクトで活躍する機能である．コミット情報をパッチとしてメールでやりとりできるため，そのパッチについてプロジェクトのメーリングリストなどで議論することができる．また，ローカルコンピュータ上にリポジトリを持つため，場所や時間，あるいはネットワーク接続の状態に関わらずコミットを扱えるという利点がある．

### 3.4.3　Gitの利用状況

　GitHubの登場で広がるGitの利用率は上昇する傾向である．2008年に登場したGitHubは，Gitリポジトリのホスティングサービスである．ソースコードをもとにソーシャルネットワーキングサービスでもあり，気軽に自分のリポジトリを公開できるのである．近年では多くのオープンソースプロジェクトがGitHubで公開されているのである．GitHubには，他のユーザーのリポジトリの複製を簡単に作成できる「Fork」という機能があり，変更の送信や複製元の変更への追従もできるのである．これらはすべてGitの基本機能であり，実際にどのように動作するかを理解すれば，GitHubを活用しやすいのである．

### 3.4.4　GitとGitHubの違い

　GitではGitリポジトリというデータの貯蔵庫にソースコードなどを入れて利用しているのである．GitHubはGitリポジトリを置く場所をインターネット上に提供しているサービスのことである．つまり，GitHubで公開しているソフトウェアのソースコードは，すべてGitで管理されているのである．

## 3.5　GitHubについて

　GitHubはコードを共有したり，公開したりするためプログラマーのソーシャルネットワーキングサービスとされるのである．GitHubはGitをサポートするためのサービスである．

Gitはプロジェクトの改訂履歴を管理し保存するものである．そして，プログラムのコードに利用されることが多いが，WordのドキュメントでもFinal Cutのプロジェクトでも，どんなタイプのファイルでも管理できるのである．つまり，コンピュータのプログラムに限らず，どんなドキュメントでも，すべてのアップデート履歴を保存し管理できるファイルシステムである．また，GitHubは一般に公開しているコードのリポジトリ集を企業に売り込んだり，就職活動にも役立てている．就職活動においてSEやプログラマーを募集するIT関連業界ではGitHubのリポジトリを履歴書として選考する企業もでてきている．

## 3.6　GitHubの利用状況

　GitHubは，2012年月で約560万リポジトリをホスティングしており，年間稼働率99.69%となっている．参加しているプログラマーは世界中に存在しており，170万人を超えている．そして，現在の2013年は約530万人[8]に達している．

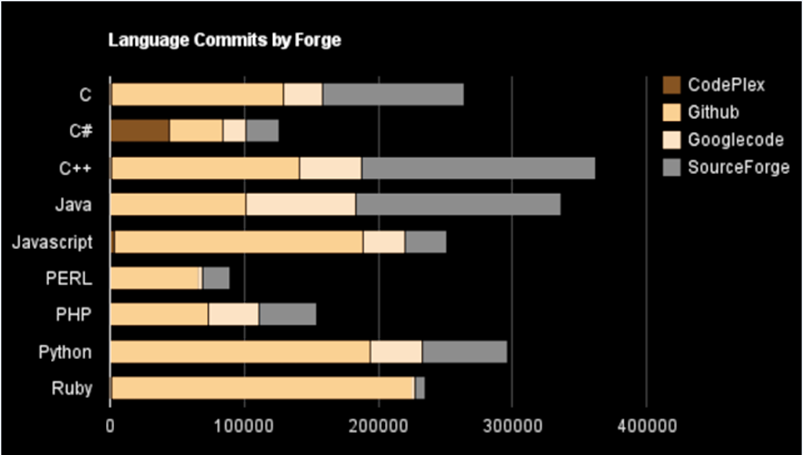


図3-5　GitHub利用状況[7]

## 3.7　GitHubの基本用語

　GitやGitHubについて論述するためには，GitHubの基本用語を知っておく必要がある．ここでは，論述する内容に記載されている基本用語について以下に記述する．

* Gitリポジトリ：

　GitHubで提供するリポジトリである．データベースのようなものである．論文に記述してあるリポジトリはGitリポジトリのことである．リポジトリにはリポジトリの管理ユーザーが公開か非公開かに選択することができる．公開の場合は，誰でも閲覧することができる．だが，非公開に選択すると特定のユーザーのみしか閲覧することはできないのである．

* コミット（commit）：

ファイルの変更履歴情報を閲覧したり，ファイルの変更を保存することである．

* 共有リポジトリ：

共有リポジトリとは，チームメンバーで共有するリポジトリで，ソースコードのメインバージョンが保存されている．

* ローカルリポジトリ：

作業者のコンピュータ上にあるリポジトリである．

* インデックス：

ローカルリポジトリへ反映する変更を一時的に保存しておく場所である．インデックスの内容は，commitによりローカルリポジトリへ反映される．

* アドオン（add）：

ソフトウェアに追加される拡張機能のことである．

* 作業ツリー：

ローカルリポジトリ上にある現在の作業ファイルである．作業ツリーの変更点はaddによるインデックスに追加される．

* ディレクトリ：

フォルダのことである．ファイルを分類・整理するための保管場所である．

* フォロー：フォローができる．
* フォロワー：ユーザーが他のユーザーをフォローすることができる．
* スター：「お気に入り」，「いいね！」のようなもの．つけられたスターはカウントされ，スターのカウントが多いとほかのユーザーからの注目をされていると認識できる．また，そのデータは活発に活動（更新）されているとみなされる．
* リビジョン：

ある期間内までの過去のプロジェクトデータやある程度まとまったプロジェクトデータを記録したものである．

* ウィキ（Wiki）：

その場にページが表示され，ドキュメントやコードがかける場所である．

* Oragnization：

Oragnizationとは普段，個人的にGitHutを使用している人が仕事などでGitHubを仕事用に使用したい場合にアカウントをもう1つ作成するのではなく，同じアカウントで会社用に使用するアカウントとして使用するのがOragnizationである．

## 3.8　GitHubの機能

GitHubはGitをサポートするためにさまざまな機能を備えている．以下には機能について記述する．

* フォーク（forking）：

1つのプロジェクトが複数に分岐していくことである．だれかのリポジトリをほかの人がコピーし改変していくことである．

* ライト（write）：

フォークしたときにコピーしたオリジナルのリポジトリのデータに書き込みをすることである．

* プルリクエスト：

フォークを行い，コピーしたリポジトリのデータにライトしたことをオリジナルのリポジトリのユーザーに通知を送信することである．

* マージ（merge）：

プルリクエストした人がその人のリポジトリに対して行われた変更を自分のリポジトリにも取り入れることである．

* イシュー（Issue）：

Issueとは，1つのタスクを1つのIssueに割り当てて，データの監視や管理を行えるようにするための機能である．1つの機能変更や修正などに対して1つのIssueが割り当てられるため，Issueを見れば，そのタスクの変更や修正に関することがすべてわかるよう管理できるのである．また，イシューにタグやマイルストーンをつけることも可能である．タグ機能は初期の設定の場合では，「バグ」，「重複」，「強化」，「無効」，「質問」が設定されている．タグの種類は増やすことも可能である．また，Issueには，「Open」と「Close」機能があり，Issueを受信した人は受信したIssueを拝見したらOpenをクリックし，拝見し終わったら，Closeをクリックすることで，Issueを発行したユーザーに拝見し終わったことを通知することができる．

* ウォッチ機能：

他の人のデータを見ることができる．他の人の進捗状況やプロジェクト内容を閲覧できる．

* グループ機能：

特定のアカウントユーザーで構成し，特定のユーザー同士でリポジトリを共有したりすることができる．グループ機能を公開の状態にしておけば，グループ以外のユーザーも閲覧はできるが，グループ機能を非公開の状態にするとグループで決められたユーザー以外は閲覧やリポジトリを操作することができないように設定できる．

* 検索機能：

検索機能は「検索ウィンドウ」に検索ワード（ユーザー名やプロジェクト名，コードなど）を入力するとそれに関連した情報を表示することができる．

* ブランチ：

ブランチとは，ある時点のソースコードを分岐させて開発するもので，ソフトウェアのメインの開発をもとに，そこから派生したコードの部分のことである．

* リモートブランチ：

リモートブランチとは，共有リポジトリの中にあるブランチのことである．

## 3.9　参考文献

[1] 岩松信洋. Gitによるバージョン管理. 初版, オーム社, 2011, p1, p4-5.

[2] 濱野純. 入門Git. 初版, 斎藤和邦, 株式会社秀和システム, 2010, p1.

[3] 片岡巌. WEB+DB PRESS. 初版, 2012, vol.69, p20.

[4] 国立大学法人東京農工大学. 開発環境の概要とバージョン管理ツール，ドキュメントテーションツールを知る. ソフトウェア開発環境の概要. http://www.tuat.ac.jp/~asiaprog/courses/oss/lesson03/, 2013-09-29.

[5] ITmedia Inc. 分散バージョン管理システムGit入門. ITmediaエンタープライズ. http://www.itmedia.co.jp/enterprise/articles/0902/09/news019.html, 2013-09-29.

[6] GitHub,Inc. 5.4M developers. https://github.com/features/community. 2013-09-29.

[7] fusion. GitHubエピローグ. Yahoo Japan Corporation. http://blogs.yahoo.co.jp/igproj\_fusion/12541553.html, 2013-09-29.

[8] Iwatani. そもそもGitHubとは一体何か. TechCrunch Japan. http://jp.techcrunch.com/2012/07/15/20120714what-exactly-is-github-anyway/, 2013-09-29.

[9] 片岡巌. 開発ツール徹底攻略. 第2版, 2013, p49-53.

[10] 岡本隆. Gitポケットリファレンス. 第2版, 2013, p1, p19-p20.

第4章

システム構築について

# 4.　システム構築について

## 4.1　本章の構成

　この章では，最終的な本研究の目的で開発するシステムを構築する際に使用するGitHubのAPI と目的に必要なデータを抽出する基準としてEVMについて記述をする．

## 4.2　APIについて

　API（Application Programming Interface，アプリケーション・プログラミング・インターフェース）とは，アプリケーションから利用できる，オペレーティングシステムやプログラミング言語で用意されたライブラリなどの機能の入り口となるものである．主に，ファイル制御，ウィンドウ制御，画像処理，文字制御などのための関数として提供されることが多いのである．

　つまり，アプリケーションをプログラムするにあたって，プログラムの手間を省くために，非常に簡潔にプログラムができるように設定されたインタフェースのことである．

### 4.2.1　APIを公開する・公開されるメリット

　Web製作者にとっては，他社の膨大なデータベースや機能を無料で利用できるため，Webサイトの開発のコストを大幅に削除できるので，効率的に制作できる．このため，個人や小資本の会社でも，人気のWebサイトを低コストで効率的に作成されることが可能になったのである．

　また，API提供会社にとっては，自社のみでは考え付かないようなWebサービスや自社のみでは実現できないようなWebサービスを外部の人間が作成してくれるため，API利用サイトから自社のサイトへユーザーが流入してくるのを期待できる．さらに，自社のデータベースや機能を間接的に自社のサイトを利用してくれるユーザーが増加し，それが結果としてデータを集積するスピードが速まり，競争優位に立つことを期待できるのである．

## 4.3　GitHubのAPIの種類

　GitHubのAPIとはGitHubの機能を抽出する時に必要なコードである．GitHubのAPIは機能ごとに設定されている．IssueにはIssueのAPIがあり，マイルストーンには，マイルストーンのAPIがある．以下に機能ごとのAPIを記述する．

* Issue（イシュー）のAPIは以下に記述する．

表4-1　IssueのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /repos/{userName}/{repoName}/issues/{id} | 選択されたIssueを表示する． |
| GET | /repos/{userName}/{repoName}/issues | リポジトリのIssueの一覧を表示する． |
| GET | /issues | ユーザーのIssueの一覧を表示する． |
| POST | /repos/{userName}/{repoName}/issues/{id} | 指定されたリポジトリのIssueを編集する． |
| POST | /repos/{userName}/{repoName}/issues | 指定されたリポジトリにIssueを作成する． |

* Issue Events（イシューイベント）のAPIは以下に記述する．

表4-2　Issue EventsのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /repos/{userName}/{repoName}/issues/events/{id} | リポジトリの指定したIssueの変化を表示する． |
| GET | /repos/{userName}/{repoName}/issues/events | リポジトリのIssueの変化を表示する． |
| GET | /repos/{userName}/{repoName}/issues/{issueId}/events | リポジトリの指定したIssueの変化を表示する． |

* Issue Comments（イシューコメント）のAPIは以下に記述する．

表4-3　 Issue CommentsのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/issues/comments/{id} | リポジトリのIssueのコメントを削除する． |
| GET | /repos/{userName}/{repoName}/issues/comments/{id} | リポジトリのIssueのコメントを表示する． |
| GET | /repos/{userName}/{repoName}/issues/{id}/comments | リポジトリのIssueの返信コメントを表示する． |
| POST | /repos/{userName}/{repoName}/issues/{id}/comments | リポジトリのIssueにコメントを作成する． |
| POST | /repos/{userName}/{repoName}/issues/comments/{id} | リポジトリのIssueにコメントを編集する． |

* Events（イベント）のAPIは以下に記述する．

表4-4　 EventsのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /users/{userName}/events | ユーザーが実行したイベントの一覧を表示する． |
| GET | /networks/{userName}/{repoName}/events | ネットワーク内でユーザーが実行したリポジトリの共有リポジトリの一覧を表示する． |
| GET | /orgs/{org}/events | 組織の共有イベントの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/events | ユーザーが実行したリポジトリのイベントを表示する． |
| GET | /events | 共有イベント名を表示する． |
| GET | /users/{userName}/events/orgs/{org} | 組織でユーザーが実行したイベントの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/issues/events | リポジトリでユーザーが実行したイベントの一覧を表示する． |
| GET | /users/{userName}/received\_events | ユーザーが受信したイベントの一覧を表示する． |
| GET | /users/{userName}/events/public | ユーザーが共有イベント名の一覧を表示する． |
| GET | /users/{userName}/received\_events/public | ユーザーが受信した共有イベントの一覧を表示する． |

* Label（ラベル）のAPIは以下に記述する．

表4-5　 LabelのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/issues/{id}/labels | 指定したリポジトリのIssueから全てのラベルを削除する． |
| DELETE | /repos/{userName}/{repoName}/labels/{id} | 指定したリポジトリから指定ラベルを削除する． |
| DELETE | /repos/{userName}/{repoName}/issues/{id}/labels/{id} | 指定したリポジトリのIssueからラベルを削除する． |
| GET | /repos/{userName}/{repoName}/milestones/{id}/labels | すべてのマイルストーンを表示する． |
| GET | /repos/{userName}/{repoName}/labels | リポジトリ内のラベルがついたものの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/labels/{id} | リポジトリ内の選択されたラベルを表示する． |
| POST | /repos/{userName}/{repoName}/issues/{id}/labels | リポジトリ内のIssueにラベルを追加する． |
| POST | /repos/{userName}/{repoName}/issues/{id}/labels | リポジトリ内のラベルを更新する． |
| POST | /repos/{userName}/{repoName}/labels/{id} | リポジトリ内にラベルを作成する． |
| PUT | /repos/{userName}/{repoName}/issues/{id}/labels | Issueのすべてのラベルを交換する． |

* Gists（ギスト）のAPIは以下に記述する．

表4-6　GistsのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /gists/{id}/star | ギストのスターを解除する． |
| DELETE | /gists/{id} | ギストのスターを削除する． |
| GET | /gists | 認証されたユーザーのギストを一覧表示するか，匿名で呼び出された場合はすべての共有ギストを表示する． |
| GET | /users/{userName}/gists | ユーザー一覧のギストを表示する． |
| GET | /gists/public | 共有ギストの一覧を表示する． |
| GET | /gists/{id}/star | 指定したギストにスターがついているかを表示する． |
| GET | /gists/starred | ユーザーがスターをつけたギストの一覧を表示する． |
| GET | /gists/{id} | 指定したギストの情報を表示する． |
| POST | /users/{userName}/gists | 新しいギストを作成する． |
| POST | /gists/{id}/fork | ギストをフォークする． |
| POST | /gists/{id} | ギストを編集する． |
| PUT | /gists/{id}/star | ギストのスターを更新する． |

* Gists Comments（ギストコメント）のAPIは以下に記述する．

表4-7　Gists CommentsのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /gists/comments/{id} | ギストコメントを削除する． |
| GET | /gists/{gistId}/comments | ギストコメントの一覧を表示する． |
| GET | /gists/comments/{id} | ギストコメントの返信を表示する． |
| POST | /gists/comments/{id} | ギストコメントを編集する |
| POST | /gists/{gistId}/comments | ギストコメントを作成する． |

* GitData（ギットデータ）のAPIは以下に記述する．

表4-8　GitDataのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /repos/{userName}/{repoName}/git/commits/{sha} | 使用ユーザーはGitリポジトリからコミットを表示する． |
| GET | /repos/{userName}/{repoName}/git/trees/{sha} | 使用ユーザーはGitリポジトリからツリー形式を表示する． |
| GET | /repos/{userName}/{repoName}/git/tags/{sha} | 使用ユーザーはGitリポジトリのタグを表示する． |
| GET | /repos/{userName}/{repoName}/git/refs/tags | 使用ユーザーのGitリポジトリのすべての証明書タグを表示する． |
| GET | /repos/{userName}/{repoName}/git/refs | 使用ユーザーのGitリポジトリのすべての証明書を表示する． |
| GET | /repos/{userName}/{repoName}/git/refs/{ref} | 使用ユーザーはGitリポジトリの証明書を表示する． |
| GET | /repos/{userName}/{repoName}/git/blobs/{sha} | 使用ユーザーはGitリポジトリのブロブを表示する． |
| GET | /repos/{userName}/{repoName}/git/trees/{sha}?recursive=1 | 使用ユーザーはGitリポジトリの1つ前のツリーを入手する． |
| POST | /repos/{userName}/{repoName}/git/blobs | 使用ユーザーのGitリポジトリにブロブを作成する． |
| POST | /repos/{userName}/{repoName}/git/trees | 使用ユーザーのGitリポジトリにツリーを作成する． |
| POST | /repos/{userName}/{repoName}/git/refs/{ref} | 使用ユーザーのGitリポジトリ内の証明書を更新する． |
| POST | /repos/{userName}/{repoName}/git/tags | 使用ユーザーのGitリポジトリ内にタグオブジェクトを作成する． |
| POST | /repos/{userName}/{repoName}/git/commits | 使用ユーザーはGitリポジトリ内にコミットを作成する． |
| POST | /repos/{userName}/{repoName}/git/refs/{ref} | 使用ユーザーのGitリポジトリの証明書を作成する． |

* Repo Download（ダウンロード・リポジトリ）のAPIは以下に記述する．

表4-9　Repo DownloadのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/downloads/{id} | リポジトリのダウンロードデータを削除する． |
| GET | /repos/{userName}/{repoName}/ downloads/{id} | リポジトリのダウンロードデータを表示する． |
| GET | /repos/{userName}/{repoName}/downloads | リポジトリのダウンロードデータの一覧を表示する． |
| POST | /repos/{userName}/{repoName}/downloads | リポジトリから新しくダウンロードする． |

* Repo Deploy Key（デプロイ・リポジトリ）のAPIは以下に記述する．

表4-10　Repo Deploy KeyのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/keys/{id} | リポジトリの暗号鍵を削除する． |
| GET | /repos/{userName}/{repoName}/keys/{id} | リポジトリの暗号鍵を表示する． |
| GET | /repos/{userName}/{repoName}/keys | リポジトリの暗号鍵の一覧を表示する． |
| POST | /repos/{userName}/{repoName}/keys | リポジトリの暗号鍵を作成する． |
| POST | /repos/{userName}/{repoName}/keys/{id} | リポジトリの暗号鍵を編集する． |

* Repo Fork（フォーク・リポジトリ）のAPIは以下に記述する．

表4-11　Repo ForkのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /repos/{userName}/{repoName}/forks | フォークデータの一覧を表示する． |
| POST | /repos/{userName}/{repoName}/forks | フォークを作成する． |

* Repo Hook（フック・リポジトリ）のAPIは以下に記述する．

表4-12 Repo HookのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/hooks/{id} | フックを削除する． |
| GET | /repos/{userName}/{repoName}/hooks | フックの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/hooks/{id} | 指定したフックを表示する． |
| POST | /repos/{userName}/{repoName}/hooks | フックを作成する． |
| POST | /repos/{userName}/{repoName}/hooks/{id} | フックを編集する． |
| POST | /repos/{userName}/{repoName}/hooks/{id}/test | フックのテスト（検証）を行う． |

* PullRequest（プルリクエスト）のAPIは以下に記述する．

表4-13　PullRequestのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /repos/{userName}/{repoName}/pulls | リポジトリ内のプルリクエストの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/pulls/{id}/commits | リポジトリ内のコミットへのプルリクエストの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/pulls/{id}/files | リポジトリからファイルへのプルリクエストの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/pulls/{id} | リポジトリから指定したプルリクエストを表示する． |
| GET | /repos/{userName}/{repoName}/pulls/{id}/merge | リポジトリからマージへのプルリクエストを表示する． |
| POST | /repos/{userName}/{repoName}/pulls/{id} | プルリクエストを更新する． |
| POST | /repos/{userName}/{repoName}/pulls | プルリクエストを作成する． |
| PUT | /repos/{userName}/{repoName}/pulls/{id}/merge | マージをプルリクエストする． |

* Pull Request Review Comment（プルリクエスト・レビューコメント）のAPIは以下に記述する．

表4-14　Pull Request Review CommentのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/pulls/comments/{id} | プルリクエストのコメントを削除する． |
| GET | /repos/{userName}/{repoName}/pulls/{id}/comments | プルリクエストへのコメントの一覧をする． |
| GET | /repos/{userName}/{repoName}/pulls/{id} | プルリクエストへの指定されたコメントを表示する． |
| POST | /repos/{userName}/{repoName}/pulls/comments/{id} | コメントを編集する． |
| POST | /repos/{userName}/{repoName}/pulls/{id}/comments | コメントを作成する． |

* Milestone（マイルストーン）のAPIは以下に記述する．

表4-15　MilestoneのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/milestones/{id} | マイルストーンを削除する． |
| GET | /repos/{userName}/{repoName}/milestones | マイルストーンの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/milestones/{id} | 指定したマイルストーンの表示する． |
| POST | /repos/{userName}/{repoName}/milestones | マイルストーンを作成する． |
| POST | /repos/{userName}/{repoName}/milestones/{id} | マイルストーンを更新する． |

* User（ユーザー）のAPIは以下に記述する．

表4-16　UserのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /user | 認証されたユーザーを表示する． |
| GET | /users/{userName} | 指定したユーザーを表示する． |
| POST | /user | 認証されたユーザーを更新する． |

* User Email（ユーザー・メール）のAPIは以下に記述する．

表4-17　User EmailのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /user/emails | 電子メールアドレスを削除する． |
| GET | /user/emails | ユーザーの電子メールアドレスの一覧を表示する． |
| POST | /user/emails | 電子メールアドレスを追加する． |

* User Follower（ユーザー・フォロワー）のAPIは以下に記述する．

表4-18　User FollowerのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /user/following/{userName} | フォローを削除する． |
| GET | /user/following | 認証しているユーザーのフォローしている一覧を表示する． |
| GET | /users/{userName}/followers | フォロワーの一覧を表示する． |
| GET | /user/followers | 認証されたユーザーのフォロワーの一覧を表示する． |
| GET | /users/{userName}/following | フォローしているユーザーの一覧を表示する． |
| GET | /user/following/{userName} | フォローしているかどうかを表示する。 |
| POST | /user/following/{userName} | ユーザーをフォローする。 |

* User Public Key（ユーザー暗証キー）のAPIは以下に記述する．

表4-19　User Public KeyのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /user/keys/{id} | 公開鍵を削除する． |
| GET | /user/keys | 公開鍵の一覧を表示する． |
| GET | /user/keys/{id} | 指定した公開鍵を表示する． |
| POST | /user/keys/{id} | 公開鍵を更新する． |
| POST | /user/keys | 公開鍵を作成する． |

* Repos（リポジトリ）のAPIは以下に記述する．

表4-20　ReposのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /repos/{userName}/{repoName}/branches | ブランチの一覧を表示する． |
| GET | /user/repos | 認証済みユーザーのリポジトリの一覧を表示する． |
| GET | /users/{userName}/repos | 指定したユーザーの共有リポジトリの一覧を表示する． |
| GET | /repos/{userName}/{repoName} | ユーザーはリポジトリを表示する． |
| GET | /orgs/{org}/repos | 指定した組織の共有リポジトリの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/ contributors | 貢献者の一覧を表示する． |
| GET | /repos/{userName}/{repoName}/languages | 言語の一覧を表示する． |
| GET | /repos/{userName}/{repoName}/teams | チームの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/tags | タグの一覧を表示する． |
| POST | /user/repos | 認証されたユーザーのためにリポジトリを作成する． |
| POST | /repos/{userName}/{repoName} | ユーザーのリポジトリを編集する． |
| POST | /orgs/{org}/repos | 組織のリポジトリを作成する． |

* Repo Commit（リポジトリ・コミット）のAPIは以下に記述する．

表4-21　Repo CommitのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/comments/{id} | コミットコメントを削除する． |
| GET | /repos/{userName}/{repoName}/commits | コミットの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/commits/{sha} | 指定したコミットを表示する． |
| GET | /repos/{userName}/{repoName}/comments | コミットへのコメントの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/commits/{sha}/comments | 指定したコミットへのコメントを表示する． |
| GET | /repos/{userName}/{repoName}/comments/{id} | 指定したコメントを表示する． |
| GET | /repos/{userName}/{repoName}/compare/{base}...{head} | 2つのコミットを比較する． |
| POST | /repos/{userName}/{repoName}/comments/{id} | コミットのコメントを更新する． |
| POST | /repos/{userName}/{repoName}/commits/{sha}/comments | コミットのコメントを作成する． |

* Repos Watching（鑑賞用リポジトリ）のAPIは以下に記述する．

表4-22　Repos WatchingのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /user/watched/{userName}/{repoName} | リポジトリのウォッチをやめる． |
| GET | /user/watched | ユーザーによってウォッチされているリポジトリの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/watchers | ユーザーがウォッチしているリポジトリの一覧を表示する． |
| GET | /users/{userName}/watched | ユーザーがウォッチしたリポジトリの一覧を表示する． |
| GET | /user/watched/{userName}/{repoName} | 自分のリポジトリウォッチしているユーザーの一覧を表示する． |
| PUT | /user/watched/{userName}/{repoName} | リポジトリをウォッチする． |

* Repo Collaborator（共有リポジトリ）のAPIは以下に記述する．

表4-23　Repo CollaboratorのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /repos/{userName}/{repoName}/collaborators/{user} | 共同リポジトリを削除する． |
| GET | /repos/{userName}/{repoName}/collaborators | 共同リポジトリの一覧を表示する． |
| GET | /repos/{userName}/{repoName}/collaborators/{user} | 共同リポジトリの協力者を表示する． |
| PUT | /repos/{userName}/{repoName}/ collaborators/{user} | 共同リポジトリのユーザーを追加する． |

* Organization（組織）のAPIは以下に記述する．

表4-24　OrganizationのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| GET | /user/orgs | 組織に認証されたユーザーの一覧を表示する． |
| GET | /orgs/{org} | 組織を表示する． |
| GET | /users/{userName}/orgs | 組織のユーザーの一覧を表示する． |
| POST | /orgs/{org} | 組織を編集する． |

* Organization Team（組織チーム）のAPIは以下に記述する．

表4-25　Organization TeamのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /teams/{id}/members/{userName} | チームメンバーを削除する． |
| DELETE | /teams/{id}/repos/{userName}/{repoName} | チームからリポジトリを削除する． |
| DELETE | /teams/{id} | チームを削除する． |
| GET | /teams/{id}/members/{userName} | チームメンバーを表示する。 |
| GET | /teams/{id}/members | チームのメンバーの一覧を表示する． |
| GET | /teams/{id} | チーム詳細を表示する． |
| GET | /orgs/{org}/teams | 組織内のチームの一覧を表示する． |
| GET | /teams/{id}/repos/{userName}/{repoName} | チーム内のリポジトリを表示する． |
| GET | /teams/{id}/repos | チームのリポジトリの一覧を表示する． |
| POST | /orgs/{org}/teams | チームを作成する． |
| POST | /teams/{id} | チームを編集する． |
| PUT | /teams/{id}/members/{userName} | チームメンバーを追加する． |
| PUT | /teams/{id}/repos/{userName}/{repoName} | チームリポジトリを追加する． |

* Organization Members（組織メンバー）のAPIは以下に記述する．

表4-26　Organization MembersのAPIと意味

|  |  |  |
| --- | --- | --- |
| APIの種類 | API | 意味 |
| DELETE | /orgs/{org}/public\_members/{userName} | ユーザーのメンバーシップを非公開にする． |
| DELETE | /orgs/{org}/members/{userName} | メンバーを組織から削除する． |
| GET | /orgs/{org}/members/{userName} | メンバーを表示する． |
| GET | /orgs/{org}/members | 組織のメンバーであるすべてのユーザー一覧を表示する． |
| GET | /orgs/{org}/public\_members/{userName} | パブリックメンバーを表示する。 |
| GET | /orgs/{org}/public\_members | パブリックメンバーの一覧を表示する． |
| PUT | /teams/{id}/members/{user} | メンバーを追加する． |
| PUT | /orgs/{org}/public\_members/{userName} | ユーザーの役割を変更する． |

## 4.4　Google Chart APIについて

　ここでは，開発システムに使用するGoogle Chart APIの機能やグラフ作成に必要な項目について説明する．

Google Chart APIとはグーグルが提供しているグラフ作成サービスである．URLにパラメータを指定するだけでグラフをPNG画像として取得できるものである．そのため<img>タグの「src」属性に対してパラメータ付きのURLを指定すれば，ホームページ上にグラフを表示させることができる．以下には，Google Chart APIの使用方法を記述する．

* 使用方法

　Google Chart APIを使用する際には，パラメータの指定をすることでグラフの表示をすることができる．Google Chart APIを使用し，PNG画像を取得するにはGoogleで用意されたURLに必要なパラメータを付けて送信すると生成された画像を取得することができるようになっているのである．書式は以下の通りである．

* https://（指定のURL）?（パラメータ1）&（パラメータ2）&（パラメータn）

まず指定のURLは「http://chart.apis.google.com/chart」である．このURLに対してパラメータを指定し，リクエストを送信する．

　パラメータはURLの直後の後に「?」の後に続けて記述していくのである．パラメータは「パラメータ名=値」の形式となっており，複数のパラメータを指定する場合は区切り文字として「&」を使用し，続けて記述していくのである．

　グラフを表示するにはURLにパラメータを付けてリクエストを送信すると結果としてPNG画像を取得することができる．静止的に作成されたPNG画像と同じように扱うことが可能である．そのため<img>タグの「src」属性にパラメータ付きのURLを指定するだけでグラフを画像として表示することができるのである．

　パラメータの種類は，いくつかあり，グラフを作成に必須なパラメータは以下の表に記述する．

表4-27 必須パラメータ名と種類

|  |  |  |
| --- | --- | --- |
| 種類 | パラメータ名 | 書式 |
| グラフのサイズ | chs | chs=幅の値x高さの値 |
| chds | chds=最小値，最大値 |
| グラフのデータ | chd | chd=s:データを表す文字列  chd=t:データを表す文字列  chd=e:データを表す文字列 |
| グラフの種類 | cht | cht=|c:Y軸のデータを表す文字列  cht=|xy:X軸のデータを表す文字列|  Y軸のデータを表す文字列 |

* グラフのサイズについて

　Google Chart APIを使用してグラフを作成するのに必須な3つのパラメータのうち，ここでは，グラフのサイズについて記述する．

大きさの指定をする際にグラフは，指定したサイズ通りに表示されるように自動的に調整が行われて表示されるのである．サイズは幅および高さをピクセル単位で指定できる．

幅と高さをピクセル単位で指定したあと，幅のピクセル数値と高さのピクセル数値の間にアルファベットの小文字の「x（エックス）」を記述することによってグラフの大きさの指定が完了するのである．

　グラフの種類やデータが同じであっても，指定したグラフの大きさに合わせて自動的にグラフの形状は調整されて表示される．

　グラフを表示する際の注意点として，表示するグラフの大きさには制限がある．グラフの幅と高さに設定可能なピクセルの数値は最大でも1000ピクセルである．また1000ピクセル以内であっても表示するグラフの幅と高さをかけた数値が300000ピクセルを超えると，グラフは表示することができないのである．グラフを作成する際に入力する値が0から100以上の数値を入力すると，折れ線グラフの場合は横線となって表示する．そこで，100以上の数値を入力するときは，入力数値の限界値を指定することで100以上の数値を入力可能にすることができる．パラメータに対して，「最小値」と「最大値」を指定すると，100以上の数値を入力しても横線ではなく，グラフとして表示される．

* グラフのデータについて

ここでは，グラフのデータの指定方法について記述する．データの数値の指定方法が3つあり，簡易形式，数値形式，拡張形式がある．またそれぞれの方法で複数のデータ系列を指定することも可能である．

* 簡易形式でのデータの数値の指定

簡易形式でデータを指定する場合には，数値を直接指定するのではなく文字を使用し，データを指定する．データを表す文字列として使用できる文字は，「AからZ」「aからz」「0から9」である．各文字の対応は以下の表に記述する．

表4-28　簡易形式の文字列対応

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 文字列 | 数値 | 文字列 | 数値 | 文字列 | 数値 |
| A | 0 | a | 26 | 0 | 52 |
| B | 1 | b | 27 | 1 | 53 |
| C | 2 | c | 28 | 2 | 54 |
| ： | ： | ： | ： | ： | ： |
| Z | 25 | z | 51 | 9 | 61 |

簡易形式の場合は文字列対応表からわかるように0から61までの数値しか扱うことができないのである．そのため，URLを短く記述できるメリットがあるのである．この簡易形式の場合は各データの値は1つの文字で表すことができるため，数値と数値の区切り文字を入力する必要がなくなり，数値に対応した文字を連続で記述することができる．

　グラフを複数表示する場合は，1つのグラフごとにデータを表す文字列を作成し，区切り文字としてカンマ（，）を使用して記述する．その際に，グラフの数値の数は同じでなければ，グラフを表示することはできないのである．

故に，グラフを複数表示する場合に1つ目のグラフの数値が4つあり，2つ目のグラフの数値が3つなど数値の数が足りない場合は，アンダースコアを使用することでデータが存在しない場合であっても数値の数を補うことができ，グラフごとに数値の数をそろえることができるのである．ただし，アンダースコアを記述した場合，記述した数値のグラフの部分は，グラフでは表示はされないのである．

* 数値形式でのデータの数値の指定

数値形式でデータを指定する場合には，数値形式の文字列を記述する際には直接指定することができる．指定する数値は整数でも小数点を含む数値であっても指定することは可能である．1つグラフのデータの数値に含まれる複数の数値を記述するには，区切り文字としてカンマ（，）を使用し続けて記述するのである．数値の値として指定できる範囲は0．0から100．0までである．グラフのY軸の最大値は常に100となるのである．100以上の数値を指定した場合は100の数値として扱われるようになるのである．

複数のグラフをデータとして指定することも可能である．複数のグラフのデータを記述する場合は，1つのグラフごとにデータを表す文字列を作成し，区切り文字としてパイプ（|）を使用し続けて記述する．その際に，グラフの数値の数は同じでなければならないのである．それぞれのグラフの数値の数が異なる場合は，描画されるX軸の位置が違ってしまうのである．

そのため，簡易形式と同じくグラフを複数表示する際に1つ目のグラフの数値の数と2つ目移行のグラフの数値の数が異なった場合は，グラフの数値の数を補って表示するための方法がある．数値形式では，グラフの数値が存在しない箇所に（-1）を記述することでグラフの数値の数を補うことができる．（-1）を使用することでグラフの数値が存在しない場合であってもグラフの数値の数をそろえることができる．

ただし，簡易形式と同じで（-1）を記述した場合，記述したグラフの数値の部分は，グラフには，表示されないのである．

* 拡張形式でのデータの数値の指定

　拡張形式でデータを指定する場合には，簡易形式と同じようにアルファベットや数値などの文字を使用して，数値の値を表示することができる．ただし，簡易形式と異なる点は，2つの文字を組み合わせて1つの数値を表すためより大きな数値の値を指定する必要があるところである．

　データを表す文字列として使用できる文字は簡易形式と同じく「AからZ」「aからz」「0から9」に加えて，マイナス「-」，コンマ「．」である．これらの文字を2つ組み合わせて1つの値に対応するのである．各文字の対応表は以下の表の通りである．

表4-29　拡張形式の文字列対応

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 文字列 | 数値 | 文字列 | 数値 | ― | 文字列 | 数値 | 文字列 | 数値 | 文字列 | 数値 |
| AA | 0 | BA | 64 | … | 9A | 3904 | -A | 3968 | ．A | 4032 |
| AB | 1 | BB | 65 | … | 9B | 3905 | -B | 3969 | ．B | 4033 |
| ： | ： | ： | ： | ： | ： | ： | ： | ： | ： | ： |
| AZ | 25 | BZ | 89 | … | 9Z | 3929 | -Z | 3993 | ．Z | 4057 |
| Aa | 26 | Ba | 90 | … | 9a | 3930 | -a | 3994 | ．a | 4058 |
| Ab | 27 | Bb | 91 | … | 9b | 3931 | -b | 3995 | ．b | 4059 |
| ： | ： | ： | ： | ： | ： | ： | ： | ： | ： | ： |
| Az | 51 | Bz | 115 | … | 9z | 3955 | -z | 4019 | ．z | 4083 |
| A0 | 52 | B0 | 116 | … | 90 | 3956 | -0 | 4020 | ．0 | 4084 |
| A1 | 53 | B1 | 117 | … | 91 | 3957 | -1 | 4021 | ．1 | 4085 |
| ： | ： | ： | ： | ： | ： | ： | ： | ： | ： | ： |
| A9 | 61 | B9 | 125 | … | 99 | 3965 | -9 | 4029 | ．0 | 4093 |
| A- | 62 | B- | 126 | … | 9- | 3966 | -- | 4030 | ．- | 4094 |
| A． | 63 | B． | 127 | … | 9． | 3967 | -． | 4031 | ．． | 4095 |

この文字列の対応表のように表現できるデータの数値の値は0から4095までとなっているのである．

　複数のグラフの数値を指定する場合には，1つのグラフのデータごとに数値を表す文字列を作成し，区切り文字としてカンマ（，）を使用して記述することで複数のグラフを表示することができる．

　そして，簡易形式と数値形式と同じく拡張形式でも，複数のグラフを表示する際にはグラフの数値の数は同じ数でなければ正常なグラフは表示されないので，足りないグラフの数値の数を補う必要がある．

そのため，足りない数値の値の場所に，アンダースコア（\_）を使用することにより，データが存在しない場合であってもアンダースコアを入力することによって数値の数を補うことができ，グラフごとに数値の数をそろえることができるのである．

ただし，簡易形式と数値形式と同じくアンダースコアを記述した場合は，記述した数値のグラフの部分は，表示はされないのである．

　以下の表は，各形式でのグラフの数値が単数を指定する場合の書式と複数のグラフの数値を指定する場合の書式である．

表4-30　形式ごとの単数のグラフの数値を指定する場合の書式

|  |  |  |
| --- | --- | --- |
| 形式 | パラメータ名 | 書式 |
| 簡易形式 | chd | chd=s:データを表す文字列 |
| 数値形式 | chd=t:データを表す文字列 |
| 拡張形式 | chd=e:データを表す文字 |

表4-31　形式ごとの複数のグラフの数値を指定する場合の書式

|  |  |
| --- | --- |
| 形式 | 書式 |
| 簡易形式 | chd=s:データを表す文字列，データを表す文字列，データを表す文字列 |
| 数値形式 | chd=t:データを表す文字列|データを表す文字列|データを表す文字列 |
| 拡張形式 | chd=e:データを表す文字列，データを表す文字列，データを表す文字列 |

* グラフの種類

　ここでは，Google Chart APIを使用するのに，必須のパラメータの1つであるグラフの種類の設定について説明する．グラフの種類には折れ線グラフ（Line chart），散布図(Scatter plot)，棒グラフ（Bar chart），ベン図（Vee diagram），円グラフ（Pie chart）の5種類が使用できる．

折れ線グラフを作成する方法は，2種類あり以下の方法がある．

* Y軸のデータのみを指定した場合の作成方法

Y軸のデータのみを指定したときの各Y軸のデータに対するX軸の値は，幅全体に描画するように表示され，グラフはデータの数に合わせて均等に割り当てられるのである．

* X軸とY軸の両方のデータを指定した場合の作成方法

X軸とY軸の両方のデータを指定した場合のデータの記述方法は，同じ系列のX軸のデータを指定し，その後にY軸のデータを指定するのである．X軸の数値とY軸の数値との区切り文字パイプ（|）で区切る．

どちらの方法でも数値が複数あるは，カンマ（，）で分ける．

折れ線グラフの各方法の書式とそのほかのグラフの書式は以下の表に記述する．

表4-32　グラフごとのパラメータ名と書式

|  |  |  |
| --- | --- | --- |
| グラフ名 | パラメータ名 | 書式 |
| 折れ線グラフ | cht | cht=|c  cht=|xy |
| 積み上げ棒グラフ | cht | cht=bhs  cht=bcs |
| 集合棒グラフ | cht | cht=bhg  cht=bvg |
| 円グラフ | cht | cht=p |
| 3Dグラフ | cht | cht=p3 |
| ベン図 | cht | cht=v |
| 散布図 | cht | cht=s |

グラフを作成するにあたりグラフごとに使用変更が可能なオプションは決まっている．

以下の表はグラフの種類によって使用変更が可能なオプションについて記述する．

◎の箇所は変更が可能である．

表4-33　グラフごとの変更可能オプションのパラメータ名

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| オプションの種類 | パラメータ名 | 折れ線グラフ | 散布図 | 棒グラフ | ベン図 | 円グラフ |
| タイトル | chtt | ◎ | ◎ | ◎ | ◎ | ◎ |
| グラフの  凡例 | chd | ◎ | ◎ | ◎ | ◎ |  |
| 色 | chco | ◎ | ◎ | ◎ | ◎ | ◎ |
| チャートと背景の塗りつぶし | chf | ◎ | ◎ | ◎ | ◎ | ◎ |
| 複数の  軸ラベル | chxt | ◎ | ◎ | ◎ |  |  |
| グリッド線 | chg | ◎ | ◎ |  |  |  |
| マーカーを形作る | chm | ◎ | ◎ |  |  |  |
| 水平レンジマーカー | chm | ◎ | ◎ |  |  |  |
| 垂直範囲  マーカー | chm | ◎ | ◎ |  |  |  |
| ライン  スタイル | chls | ◎ |  |  |  |  |
| 領域を  塗りつぶす | chm | ◎ |  |  |  |  |
| 棒の幅との間隔 | chbh |  |  | ◎ |  |  |
| 円グラフのラベル | chl |  |  |  |  | ◎ |

* タイトルと軸ラベルのオプションについて

　ここでは，タイトルと軸ラベルのオプションについて説明を記述する．

* タイトルの入力

タイトルの設定を行うには，パラメータに対して，タイトルとして表示した文字列を指定する．また，タイトル内に空白を入れる場合には「+」を入力する．改行をする場合には，「|」を使用する．

* タイトルの色とフォントサイズ

パラメータに対して，色とフォントサイズをカンマ（，）で区切って指定する．タイトルはグラフの表示エリア内に表示される．タイトルが表示された分だけグラフが表示される部分の縦方向のサイズが小さくなる．

* 凡例

凡例とは折れ線グラフなら折れ線グラフ自体のタイトルのことである．凡例の設定を行うには，パラメータに対して，凡例として表示する系列ごとにラベルを指定できる．複数のラベルを記述する場合はパイプ（|）で区切って続けて指定する．ラベルの色は対応する系列に設定されている色（グラフの線や棒の色）と同じである．

* 軸ラベルの表示

軸ラベルの表示の設定を行うには，パラメータに対して，ラベルを表示する軸を指定する．複数の軸を指定する場合は，カンマ（，）で区切って続けて記述することができる．各軸は複数界記述することが可能である．これは，同じ軸に対して複数の異なるラベルを記述したい場合に使用する．表示する軸は記述した順にインデックスが割りあてられる．これは，1番目に記述した軸のインデックスは0，2番目に記述した軸のインデックスは1である．インデックスは軸のラベルに表示される文字列を指定する場合に利用する．

　指定する値と各軸は以下の通りになっている．

表4-34　軸ラベルの表示指定値と指定軸

|  |  |
| --- | --- |
| 指定する値 | 指定する軸 |
| x | X軸下辺 |
| t | X軸上辺 |
| y | Y軸下辺 |
| r | Y軸上辺 |

* 軸ラベルの表示文字列

軸のラベルの設定を行うには軸ラベルが表示されるように設定されなければならないのである．パラメータに対して，対象の軸のインデックス」を指定した後に「：|」を付け，その後に表示するラベルはパイプ（|）を使用して続けて記述する．複数の軸を対象にする場合はパイプ（|）で区切って続けて記述する．

ラベルはデフォルトでは最小値と最大値，そしてその間の場所に均等に並べられるように配置されるのである．なお表示するように設定したすべての軸にラベルを個別に指定する必要なく，省略した場合はデフォルトで設定されるラベルが表示される．

* 軸ラベルの表示位置

軸ラベルの表示位置の設定を行うには，軸ラベルが表示されるように設定されていなければならないのである．パラメータに対して，「対象の軸インデックス」を指定した後にカンマ（，）を付け，その後，各ラベルの位置はカンマ（，）を使用し，続けて記述する．

複数の軸を対象にする場合はパイプ（|）で区切って続けて記述する．また，軸に表示されるラベルを指定していない状態でラベルの位置を指定すると，指定した位置の値がラベルとして使用される．

* 軸の開始位置の値と終了位置の値

軸の開始位置の値と終了位置の値の設定を行う際に，初期設定では軸の開始位置の値は0，終了位置の値は100で設定されている．パラメータに対して，「対象の軸のインデックス」を指定した後にカンマ（，）を付け，その後に，開始位置の数値と終了位置の数値はカンマ（，）を使用し，続けて記述する．複数の軸を対象にする場合はパイプ（|）で区切って続けて記述する．

開始位置の数値と終了位置の数値を設定した場合は，ラベルの位置もこの指定した数値の範囲に合わせて数値の指定をする必要がある．また，開始位置が必ずしも終了位置より小さい数値である必要はなく，開始位置の数値が終了位置の数値よりも大きい数値でも問題はないのである．

* 軸ラベルのスタイル

軸ラベルのスタイルの設定を行うには，パラメータに対して，「対象の軸インデックス」を指定した後にカンマ（，）を付け，その後に色，フォントサイズ，水平位置を区切り文字にカンマ（，）を使用し続けて記述する．複数の軸を対象にする場合はパイプ（|）で区切って続けて記述する．

また，フォントサイズと水平方向の位置は省略可能である．水平位置については初期設定の数値はX軸の場合は「中央寄せ」，Y軸の場合は「右寄せ」となっている．設定する場合は以下の表を参照する．

表4-35　水平位置の入力数値と指定場所

|  |  |
| --- | --- |
| 入力数値 | 指定場所 |
| 0 | 中央寄せ |
| 1 | 右寄せ |
| -1 | 左寄せ |

以下の表は，タイトルと軸ラベルのオプションのパラメータ名と書式について記述する．

表4-36　タイトルと軸ラベルのオプションのパラメータ名と書式

|  |  |  |
| --- | --- | --- |
| オプション名 | パラメータ名 | 書式 |
| タイトル | chtt | chtt=タイトル |
| タイトルの色とフォントサイズ | chts | chts=色，フォントサイズ |
| 凡例 | chd| | chd|=ラベル1|ラベル2|ラベルn |
| 軸ラベルの表示 | chxt | chxt=軸1，軸2，軸n |
| 軸ラベルの表示文字列 | chx| | chx|=  インデックス1：|ラベル|ラベル|  インデックスn：|ラベル|ラベル| |
| 軸ラベルの表示位置 | chxp | chxp=  インデックス1，ラベル位置1，  ラベル位置2，ラベル位置n|  インデックスn，ラベル位置1，  ラベル位置2，ラベル位置n |
| 軸の開始位置の値と終了位置の値 | chxr | chxr=  インデックス1，開始位置の数値，終了位置の数値|  インデックスn，開始位置の数値，終了位置の数値 |
| 軸ラベルのスタイル | chxs | chxs=  インデックス1，色，フォントサイズ，水平位置|インデックスn，  色，フォントサイズ，水平位置 |

* 色の指定のオプションについて

ここでは，線や点，背景などグラフで表示される色の設定の方法について説明をする．GoogleChartAPIにおいては色を指定方法は赤青緑をそれぞれ16進数で「00」から「FF」までの値で指定することができる．さらに，透明度のある色を表示することも可能である．

　また，複数の色を指定する場合はカンマ（，）で区切って指定する．

表4-37　色を設定する2種類の書式

|  |  |
| --- | --- |
| 説明 | 書式 |
| 透明度がない色の設定値 | RRGGBB |
| 透明度を表示が可能な色の設定値 | RRGGBBAA |

* データを表す色

　ここでは，グラフで表示されるデータの色の指定方法について説明する．折れ線グラフでは線の色，棒グラフでは棒の色，円グラフではスライスの色などの色の指定である．以下の表はデータを表す色のパラメータ名と書式である．

表4-38　データを表す色のパラメータ名と書式

|  |  |  |
| --- | --- | --- |
| オプション名 | パラメータ名 | 書式 |
| データ（線や棒）を表す色 | chco | chco=色1，色2，色n |

* 背景およびグラフエリア（塗りつぶし）

　グラフはグラフが表示されるグラフエリアと，グラフエリアに加えてタイトルや軸ラベルなどが含まれるグラフの背景がある．ここでは，グラフの背景およびグラフエリアの色の指定方法について説明する．背景やグラフエリアの色の指定方法は以下の3つがある．

* 指定の色で塗りつぶす方法

パラメータに対して，「対象」「s」「色」の順で指定する．対象には背景かグラフエリアかを指定する．背景の場合には「bg」を指定し，グラフエリアの場合には「c」を指定する．背景とグラフエリアのどちらかだけでも指定は可能だが，両方を指定する場合はパイプ（|）で区切り続けて記述する．

* グラデーションで塗りつぶす方法

パラメータに対して，「対象」「|g」「角度」「色1」「オフセット1」「色2」「色2」「オフセット2」の順で指定する．背景とグラフエリアのどちらかだけでも指定することは可能であるが，両方を指定すること場合は，パイプ（|）で区切り続けて記述する．対象には背景かグラフエリアかを指定し，背景の場合には「bg」をグラフエリアの場合には「c」を指定する．グラデーションは指定した2つ以上の色の間で徐々に色が変化していくものである．そこで基点となる位置を決めて色を指定する．位置を表すオフセットには0から1の間で指定し，左端が「0」，右端が「1」となる．そして角度には色が変化していく方向を0から90の間の数値で指定する．角度に90を指定した場合には下から上へ色が変化していきますのでオフセットは「0」が下端，「1」が上端である．

* ストライプで塗りつぶす方法

パラメータに対して，「対象」「|s」「角度」「色1」「幅1」「色2」「幅2」の順で指定する．背景とグラフエリアのどちらかだけでも指定できるが，両方を指定した場合はパイプ（|）で区切り，そのまま続けて記述することが可能である．対象には背景かグラフエリアかを指定し，背景の場合には「bg」をグラフエリアの場合には「c」を指定する．ストライプは指定した2つ以上の色で繰り返し塗りつぶすことが可能である．それぞれの色とどれだけの幅を塗りつぶすのかを「0」から「1」の間でしてする．「1」を指定した場合はすべてのエリアを同じ色で塗りつぶすのである．そして角度にはストライプが繰り返されていく方向を0から90の間の数値で指定する．角度に90を指定した場合には下から上へストライプが繰り返される．

以下の表に，背景およびグラフエリアを塗りつぶす方法ごとのパラメータ名と書式を記述する．

表4-39　背景およびグラフエリアを塗りつぶす方法ごとのパラメータ名と書式

|  |  |  |
| --- | --- | --- |
| オプション名 | パラメータ名 | 書式 |
| 指定の色で塗りつぶす方法 | chf | chf=対象，s，色|対象，s，色 |
| グラデーションで塗りつぶす方法 | chf | chf=  対象，|g，角度，色1，オフセット1，色2，オフセット2，色n，オフセットn |
| ストライプで塗りつぶす方法 | chf | chf=  対象，|s，角度，色1，幅1，色2，幅2，色n，幅n |

* スタイルのオプションについて

　ここでは，マーカーや補助線，グラフの間隔などのグラフの様々なスタイルの設定方法について説明する．

* 折れ線グラフの線の太さと種類

　ここでは，折れ線グラフで表示される線の太さと種類の設定方法について説明する．パラメータに対して，「線の太さ」「描画される線の長さ」「描画されない線の長さ」の3つをカンマ（，）で区切って指定する．複数の系列に対して指定する場合は，パイプ（|）で区切って，続けて記述することができる．描画される線の長さと描画されない線の長さは繰り返し線に適用される．適切な長さを指定することで点線や破線を引く事が可能である．実線を引く場合には描画されない線の長さを「0」に設定すれば，記述することが可能である．

* 補助線

　ここでは，補助線について説明する．補助線は折れ線グラフと散布図のみ表示することが可能である．パラメータに対して，「水平間隔」「垂直間隔」「描画される線の長さ」「描画されない線の長さ」をカンマ（，）で区切って指定することが可能である．間隔の大きさは軸に設定されている値の範囲に合わせて調整する．

* データを表すマーカーの形状

　ここではデータを表すマーカーの形状の設定について説明する．データを表すマーカーの形状は折れ線グラフと散布図のみ設定することが可能である．パラメータに対して，マーカーの種類」「色」「系列」「データのインデックス」「サイズ」をカンマ（，）で区切って指定するのである．同じ系列であってもデータ毎に指定するため，複数のデータに対して設定する場合はパイプ（|）でつなげて続けて記述する．

　データインデックスは何番目のデータに適用するかを指定する．小数点を含む値を指定した場合は，点と点の間の位置にマーカーが表示される．またサイズの単位はピクセルである．なお，マーカーの種類の「h」を選択した場合は，何番目のデータかでなくY軸の原点を「0」，上辺を「1．0」とした場合は水平線を描画する位置を0から1．0の間の値で指定する．マーカーの種類には以下の表のような形状がある．

表4-40　入力データ値とマーカーの種類

|  |  |
| --- | --- |
| データ値 | マーカーの種類 |
| a | 矢印 |
| c | 十字 |
| d | ダイアモンド（ひし形） |
| o | 円 |
| s | 四角 |
| v | X軸上からデータまでの垂直線 |
| V | X軸上から上辺までの垂直線 |
| h | Y軸上から右辺までの水平線 |
| H | 指定したデータのみX軸に水平な線 |
| x | バツ印 |

* 帯領域を表すマーカー

　ここでは，グラフエリアの1部を塗りつぶしたようなマーカーを表示する方法について説明する．この方法は折れ線グラフと散布図のみ使用が可能である．パラメータに対して，「水平または垂直」「色」「開始地点」「終了地点」をカンマ（，）で区切って指定する．複数設定する場合はパイプ（|）でつなげて続けて記述する．水平方向の帯領域を表示するには，「r」を入力し，垂直方向の帯領域を表示するには「R」を指定する．開始地点と終了地点の指定には0．00から1．00までの値を指定する．0．00が下辺または左辺を表し，1．00が上辺または右辺を表すのである．

* 折れ線グラフで線と線の間を塗りつぶす

　ここでは折れ線グラフにおいて複数の系列のデータから線を表示したときに，線と線の間を塗りつぶすことについて説明する．パラメータに対して，「b」「色」「開始系列」「終了系列」をカンマ（，）で区切って指定する．複数設定する場合はパイプ（|）でつなげて続けて記述する．また線が1つであり，X軸と線の間の領域を塗りつぶす場合は以下の書式が使用可能である．

表4-41　X軸と線の間の領域を塗りつぶす場合の書式

|  |  |  |
| --- | --- | --- |
| オプション名 | パラメータ名 | 書式 |
| 折れ線グラフで線と線の間を塗りつぶす（線が1つであり，X軸と線の間の領域を塗りつぶす場合） | chm | chm=B，色，0，0n，未使用 |

以下の表には，各オプション名の色の指定をするパラメータ名と書式について記述する．

表4-42　色を指定するオプション名のパラメータ名と書式

|  |  |  |
| --- | --- | --- |
| オプション名 | パラメータ名 | 書式 |
| 線の太さ | chls | chls=  線の太さ1，描画される線の長さ1，描画されない線の長さ1|  線の太さn，描画される線の長さn，描画されない線の長さn |
| 補助線 | chg | chg=水平間隔，垂直間隔，描画される線の長さ，描画されない線の長さ |
| データを表すマーカーの形状 | chm | chm=  マーカーの種類1，色1，系列1，データのインデックス1，サイズ1|  マーカーの種類n，色n，系列n，データのインデックスn，サイズn |
| 帯領域を表すマーカー | chm | chm=  水平または垂直，色1，未使用，開始地点1，終了地点1|  水平または垂直，色n，未使用，開始地点n，終了地点n |
| 線と線の間を塗りつぶす | chm | chm=  b，色1，開始系列1，終了系列1，未使用|  b，色n，開始系列n，終了系列n，未使用 |

* 日本語の扱い

　ここではタイトルやラベルに表示される文字列に日本語を利用できるかどうかについて説明する．現状では，ひらがなとカタカナのみ表示することが可能である．漢字の表示をすることはできない．GoogleChartAPIでは設定内容はすべてURLにパラメータとして含める必要がある．そのため，タイトルに日本語を含める場合には設定したい文字列をURLエンコードする必要がある．そこで，入力したい日本語の文字列を「Shift-JIS」「EUC-JP」「UTF-8」といったそれぞれの文字コードでタイトル名を入力した場合に，ひらがなとカタカナで表示することが可能である．

## 4.5　ブックマークレットについて

　ブックマークレットとは，通常の「お気に入り」（ブックマーク）にJavaScriptによる特定の動作を仕組んだものである．通常の「お気に入り」の項目を選ぶと，その項目に登録されているWebページに移動する．だがブックマークレットの場合はWebページに移動するのではなく，Java-Scriptによって指定した動作をするのである．

　ブックマークレットには行間が狭くて読みづらいWebページを表示しているとき，行間を広くするブックマークレットを選択すると行間が広くなり，読みやすく変更することができる．また，ブックマークレットを使用する際には表示しているWebページの背景を消したり，マウスカーソルで選択した用語をインターネットの辞書で検索したりといった動作が可能になるのである．

　つまり，ブックマークレットがあればブラウザーの機能を追加したり拡張したりできるのである．

### 4.5.1　ブックマークレットの提案者について

　ブックマークレットを最初に提案したのはbookmarklets.comのスティーブ・キャンガス（Steve　Kangas）氏である．用語の由来は明確になっていないが，InternetExplorerの「お気に入り」と同じ意味の「ブックマーク（bookmark：しおり）」に小さいスクリプト・プログラムを意味する「スクリプトレット（Scriptlet）」が合体したものだろうとされているのである．スティーブ・キャンガス氏が提唱するブックマークレットの特徴に「フリー（Free）」ということがある．つまりブックマークレットを使用する際には，無料となる．

### 4.5.2　ブックマークレットの仕組み

　ここでは，ブックマークレットの特徴や入力方法などについて，説明する．

ブックマークレットの特徴

ブックマークレットはInternetExplorerなどのブラウザーのアドレス欄では単に移動したいアドレス位置（URL）を指定するだけでなく，簡易なJavaScriptプログラムが実行できる，という機能を応用しているのである．

　InternetExplorer6の場合は，セキュリティ機能をオンにしているとこうした動作が阻害されることがあるのである．セキュリティの注意を出しているのは自分で登録したブックマークなので，セキュリティ上の問題はないのである．それでもブックマークの動作が阻害される場合は，ブックマークレットの使用に限ってセキュリティ機能を弱めるか，あるいはどこかのWebサイトを表示した状態でブックマークレットを選択すると良いのである．オフラインなら阻害される機能でも，空白なabout:blankやオンラインでは利用できることが多いのである．

　ブックマークレットは主にWindows用のInternetExplorerを基本としたが，その大半はFirefoxでも利用はできる．Operaでもできるものは多いがバージョン差が多く，また機能的に不安定なこともあり充分に機能が活かせないこともあるのである．ブックマークレットは便宜的に利用されることが多いので，当面の目的が達成できるならプログラムはできるだけ簡単なほうがよいとされているのである．こうしたことからサンプルには簡単だが機能が弱い例と，機能は強いがやや複雑な例を一列に表示しているものである．ブックマークレットで使用した変数は，それが対応するWebページ内のJavaScriptコードの変数に対してグローバルになるため，同一の変数があるとブックマークレットからは予想しえない結果になる．この対処としては，ブックマークレットの動作をすべてfunction()に納めて擬似的にローカルに扱うという手法もある．だが，この状況はそれほど頻繁に発生するわけでもないので，問題が発生した時点で問題を起こした変数を書き換えるほうが得策である．

### 4.5.3　ブックマークレットのプログラム上の特徴

　ブックマークレットはJavaScriptのプログラムからできているが，通常のJavaScriptプログラムとは違った面があるのである．

また，アドレス欄で実行するという制約から一行でまとめる必要がある．また，一行に納められる文字数は500文字程度までと言われているが，実際には300文字を超えるほどのブックマークレットは複雑になりすぎるので作成時には細心の注意が必要である．

　行の冒頭はjavascript:で始めるのである．プログラム上の文の区切りはセミコロン（;）を使用するのである．以下に例として3行で表示されたコードがある．

javascript:

document.body.style.lineHeight=’1.8em’;

void(0);

このような場合のコードは以下のように，1行にする必要がブックマークレットではある．

javascript:document.body.style.lineHeight=’1.8em’;void(0);

さらに，結果は新規ページに表示される．例えば，例えばjavascript:3+4という簡単なアドレス欄でのプログラムでも，結果の7は新規ページに表示されるのである．このようにブックマークレットで指定したJavaScriptの動作に結果があると，それが新規ページに表示されるのである．

新規ページへの移動を避ける手法

　新規ページへの移動を避けたい場合は，以下のような手法がある．

* 全体をvoid()関数で囲む

例：javascript:void(window.status=document.lastModified);

* 処理の最後にvoid(0)またはeval()を追加する

例：javascript:window.status=document.lastModified;void(0);

* 処理全体を関数に納める

例：javascript:(function(c){window.status=document.lastModified})()

### 4.5.4　ブックマークレットの使用方法

ブックマークレットを使用するときは，ブックマークレットを使用したいウェブページをブックマークし，その後，同じウェブページを再度ブックマークするときに，ウェブページのタイトルを機能名に変更する．そして，タイトルを変更したブックマークのプロパティのURLをブックマークレットのURLに変更する．これで，ブックマークレットは完成する．完成後のブックマークレットは，ブックマークしたページを表示したウェブページでブックマークレットのページを表示することで機能する．

## 4.6　参考文献

[1] 田村啓. APIとは. プログラミング実習室. http://www.program-study.com/report/api, 2013-10-06.

[2] APIS.IO. APIS.IO. APIS.IO. http://apis.io/GitHub, 2013-08-30.

[3] GitHub. GitHub Developer. GitHub Developer. http://developer.github.com/v3/issues/, 2013-08-30.

[4] TATSUO IKURA. パラメータの指定方法とグラフの表示. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/ini/index1.html, 2014-01-14.

[5] TATSUO IKURA. サイズの指定. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/size/index1.html, 2014-01-14.

[6] TATSUO IKURA. 最大サイズの制限. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/size/index2.html, 2014-01-14.

[7] 堺ちゃん. Google Ghart APIで複数軸グラフを書く方法. インターネット販促メモ. http://shotets.blog21.fc2.com/blog-entry-29.html, 2014-01-22.

[8] TATSUO IKURA. 簡易形式での指定(xhd=s). AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/data/index1.html, 2014-01-14.

[9] TATSUO IKURA. 数値形式での指定(chd=t). AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/data/index2.html, 2014-01-14.

[10] TATSUO IKURA. 拡張形式での指定(chd=e). AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/data/index3.html, 2014-01-14.

[11] TATSUO IKURA. 折れ線グラフ(Y軸のデータを指定). AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/type/index1.html, 2014-01-20.

[12] TATSUO IKURA. 折れ線グラフ(X軸とY軸のデータをセットで指定). AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/type/index2.html, 2014-01-20.

[13] TATSUO IKURA. タイトル. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index1.html, 2014-01-22.

[14] TATSUO IKURA. タイトルの色とフォントサイズ. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index2.html, 2014-01-22.

[15] TATSUO IKURA. 凡例. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index3.html, 2014-01-14.

[16] TATSUO IKURA. 軸ラベルの表示. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index5.html, 2014-01-22.

[17] TATSUO IKURA. 軸ラベルの表示文字列. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index6.html, 2014-01-22.

[18] TATSUO IKURA. 軸ラベルの表示位置. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index7.html, 2014-01-22.

[19] TATSUO IKURA. 軸ラベルのスタイル. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/label/index9.html, 2014-01-22.

[20] TATSUO IKURA. 色の指定方法. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/color/index1.html, 2014-01-22.

[21] TATSUO IKURA. データを表す色(線の色). AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/color/index2.html, 2014-01-22.

[22] 大島. Google Chart APIでマーカーを表示する. ilovex. http://www.ilovex.co.jp/blog/system/cat818/google-chart-api.html, 2014-01-22.

[23] TATSUO IKURA. 折れ線グラフの線の太さと種類. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/style/index2.html, 2014-01-22.

[24] TATSUO IKURA. 補助線. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/style/index3.html, 2014-01-22.

[25] TATSUO IKURA. データを表すマーカーの形状. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/style/index4.html, 2014-01-22.

[26] TATSUO IKURA. URLエンコードのテスト. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.ajaxtower.jp/googlechart/japan/index1.html, 2014-01-22.

[27] 佐藤信正. ブラウザー・スクリプティング. 第初版, ルナテック, 2005, p14-16, p20-24.

第5章

プロジェクトマネジメントについて

# 5.　プロジェクトマネジメントについて

## 5.1　本章の構成

　この章では，本研究と関係のあるマネジメントの定義について記述する．本研究の目的に最も関係のあるプロジェクトマネジメント・プロセス群と知識エリアの分類の表を参照にプロジェクトマネジメント・プロセス群の監視・コントロール・プロセス群について記述する．

## 5.2　プロジェクト作業の監視・コントロール・プロセス群について

　プロジェクト作業の監視・コントロール・プロセス群の目的は，プロジェクトが計画通りに進んでいるかを確認するものである．計画通りでない場合は，その対処の内容と対処してよいかの伺い書を作成することである．プロジェクトマネジャーは，作業者から報告される実績報告を元に，計画通り（プロジェクトマネジメント計画通り）にプロジェクトが進んでいるか確認するのである．計画通りにプロジェクトが進行していない場合は，処置に必要なことをまとめて提言を行うのである．

　PMBOKによれば，プロジェクト作業の監視・コントロールは，プロジェクトマネジメント計画書に定義されたパフォーマンス目標を達成するため，進捗を追跡し，レビューし，統制するプロセスと定義されているのである．このプロセス郡が役立つ主な点は，プロジェクトマネジメント計画書との差異を識別するために，プロジェクトのパフォーマンスを定期的かつ一貫して監視し，測定することである．

プロジェクト作業において監視とコントロールは以下のように定義する．

1. 監視

　プロジェクトを通して実行されるプロジェクトマネジメントの一側面である．監視にはパフォーマンス情報の収集，測定，配布，およびプロセスの改善を推進するために行う測定結果とその傾向の評価が含まれるのである．この継続的な監視を行うことによって，プロジェクトマネジメント・チームはプロジェクトの健全度を洞察し，特に注意が必要な箇所を識別にできるのである．

1. コントロール

是正処置や予防措置を行うことおよび再計画を立てること，さらに実施した処置によりパフォーマンスの課題が解決されたかどうかを判断するために行動計画をフォローアップすることが含まれるのである．

プロジェクト作業の監視・コントロール・プロセスで行うことは，以下に記述する．

* プロジェクトマネジメント計画書とプロジェクトの実施との比較する．
* 是正処置や予防処置が必要かどうかを決定するためのパフォーマンスの評価，および必要に応じた処置の提言する．
* 新たなリスクの特定および既存のプロジェクト・リスク分析，追跡，監視．そのためには，リスクを特定し，状況を報告し，適切なリスク対応計画を実行することを確実する．
* プロジェクトを通して，プロジェクトの成果物と関連文書について，正確でタイムリーな情報データベースの維持する．
* 状況報告，進捗測定，予測等の基礎となる情報の提供する．
* 承認済み変更が発生した際に，それらの実施状況の監視する．

この継続的な監視により，プロジェクト・チームは，プロジェクトの状況を把握し，引き続き注意が必要な領域を特定する．監視・コントロール・プロセス郡は，あるプロセス郡内で実施される作業の監視・コントロールだけでなく，プロジェクト全体の作業の監視とコントロールも行う．複数のフェーズをもつプロジェクトにおいては，監視・コントロール・プロセス郡は，プロジェクトマネジメント計画書に沿ってプロジェクトに対し是正処置や予防処置を講じるように，プロジェクト・フェーズ間の調整をする．このレビューの結果，プロジェクトマネジメント計画書の更新が提言され，承認される場合がある．例えば，あるアクティビティの終了日を守れないことが明らかな場合は，現在の要員計画の調整，残業時間の増加，予算とスケジュール目標間のトレードオフなどが必要になる．

このプロセスのインプットは，「プロジェクトマネジメント計画書」，「実績報告書」，「組織体の環境要因」，「組織のプロセス資産」である．ツールと技法は，「専門家の判断」である．アウトプットは「変更要求」，「プロジェクトマネジメント計画書更新版」，「プロジェクト文書更新版」であり，以下で各項目を記述する．

1. インプット

* プロジェクトマネジメント計画書

　プロジェクトマネジメント計画書には，計画プロセスで作成したすべての補助のマネジメント計画書およびベースラインを統合し，集約する．プロジェクトマネジメント計画書には以下のものが含まれるが，これらに限定されないのである

* プロジェクトに対応して選択したライフサイクル，および各フェーズに適用するプロセスである．
* プロジェクトマネジメント・チームがテーラリングした結果，得られた次のような項目である．
* プロジェクトマネジメント・チームが選択したプロジェクトマネジメント・プロセスである．
* 選定した各プロセスの実行のレベルである．
* プロセスを実行するために使用するツールと技法の記述する．
* 当該プロジェクトをマネジメントするために選定したプロセスを適用する方法である．これは選定したプロセス間における依存関係と相互作用，およびもっとも重要なインプットとアウトプットも含まれる．
* プロジェクト目標を達成するための作業の実行方法である．
* 変更を監視し，コントロールする方法を文書化した変更マネジメント計画書である．
* コンフィギュレーション・マネジメントの実施方法を文書化しコンフィギュレーション・マネジメント計画書である．
* パフォーマンス測定ベースラインの一貫性を維持する方法である．
* ステークホルダー間のコミュニケーションのためのニーズと技法である．
* 未解決の課題や未決定事項の処理を促進するために，内容，範囲，時期などをレビューする重要なマネジメント・レビューである．

プロジェクトメネジメント計画書の記述レベルは概要・詳細いずれの場合もある．また，

いくつかの補助計画書で構成することもできる．それぞれの補助計画書は，当該プロジェ

クトが必要とする程度まで詳しく記述する．いったんプロジェクトマネジメント計画書が

ベースラインとして確立されたあとは，変更要求が提起され，統合変更管理を通して承認

された場合に限りプロジェクトマネジメント計画書を変更することができる．

プロジェクト・ベースラインには以下のものがあるが，これらに限定されるものではない．

* スケジュール・ベースライン
* コスト・パフォーマンス・ベースライン
* スコープ・ベースライン

補助計画書には以下のものがあるが，これらに限定されるものではないのである．

* スコープ・マネジメント計画書
* 要求事項マネジメント計画書
* スケジュール・マネジメント計画書
* コスト・マネジメント計画書
* 品質マネジメント計画書
* プロセス改善計画書
* 人的資源計画書
* コミュニケーション・マネジメント計画書
* リスク・マネジメント計画書
* 調達マネジメント計画書

　スコープ・ベースライン，スケジュール・ベースライン，およびコスト・ベースライン

がまとまってパフォーマンス測定ベースラインとなり，統合したパフォーマンスを測定す

るための統括的プロジェクト・ベースラインとして使用する．パフォーマンス測定ベース

ラインは，アーンド・バリューの測定に使用される．

* 実績報告書

　プロジェクト・チームは，アクティビティ，業績，マイルストーン，特定した課題，問題などについて詳しく記述した報告書を作成する．重要な情報を報告するために用いる実績報告書には以下のものがあるが，これらに限定されるものではない．

* 現状の状況
* 期間中の重要な実績
* 予定されたアクティビティ
* 予測
* 課題
* 組織体の環境要因

　プロジェクト作業の監視・コントロール・プロセスに影響を与える組織体の環境要因には以下のものがあるが，これらに限定されるものではない．

* 国家標準や業界標準など
* 企業の作業認可システム
* ステークホルダーのリスク許容度
* プロジェクトマネジメント情報システム（例：スケジュール・ソフトのような自動化ルール一式，コンフィギュレーション・マネジメント・システム，情報収集・配布システム，他のオンライン自動化システムとWebインターフェース）
* 組織のプロセス資産

　プロジェクト作業の監視・コントロールのプロセスに影響を与える組織のプロセス資産には以下のものがあるが，これらに限定されるものではない．

* 組織のコミュニケーション要求事項
* 財務管理手順（例：作業時間報告，勘定コード，必要な支出や支払いのレビュー，標準契約条件）
* 課題と欠陥のマネジメントの手順
* リスク・コントロールの手順．これには，リスク区分，発生確率の定義と影響度，発生確率・影響度マトリックスなどが含まれる．
* プロセス測定データベース．これは，プロセスやプロダクトの測定データを利用可能にするために使用される．
* 教訓データベース

1. ツールと技法

* 専門家の判断

　プロジェクトマネジメント・チームは，プロジェクト作業の監視・コントロール・プロセスから得られる情報を評価するために，専門家の判断を利用する．プロジェクトマネジャーは，プロジェクト・チームと協力し，プロジェクトに対する期待とプロジェクトのパフォーマンスを一致されるように，必要な処置を決定する．

1. アウトプット

* 変更要求

　計画値と実績値を比較した結果，変更要求が発行される．それにより，プロジェクトや成果物のスコープは拡大され，調整され，あるいは縮小される．変更により，プロジェクトマネジメント計画書，プロジェクト文書，またはプロダクトの要素成果物が影響を受ける．変更には，以下の種別のものがあるが，これらには限定されるものではない．

* 是正処置

　予期される将来のパフォーマンスをプロジェクトマネジメント計画書に沿ったものとするために，プロジェクト作業に出される文書による指示．

* 予防処置

　プロジェクト・リスクによって好ましくない結果が発生する確率を低減するために取られる行動の文書による指示．

* 欠陥修正

　プロジェクト構成要素の欠陥の公式な文書による特定．それには欠陥の修正あるいは構成要素の全面的な取り換えのいずれかの勧告を含める．

* プロジェクトマネジメント計画書更新版

　プロジェクトマネジメント計画書を構成するもので，このプロセスにおいて更新されるものには以下のものがあるが，これらに限定されるものではない．

* スケジュール・マネジメント計画書
* コスト・マネジメント計画書
* 品質マネジメント計画書
* スコープ・ベースライン
* スケジュール・ベースライン
* コスト・パフォーマンス・ベースライン
* プロジェクト文書更新版

更新されるプロジェクト文書には以下のものがあるが，これらに限定されるものではないのである．

* 予測
* 実績報告書
* 課題ログ

## 5.3　各知識エリアでの識別

PMBOKのプロジェクトマネジメント・プロセスでは，プロジェクト作業の監視・コントロール・プロセス郡は，9つの各知識エリアでは「プロジェクト作業の監視・コントロール」，「総合変更管理」，「スコープ検証」，「スコープ・コントロール」，「スケジュール・コントロール」，「コスト・コントロール」，「品質管理」，「実績報告」，「リスクの監視・コントロール」，「調達管理」に判別される．以下にその内容を記述する．

1. プロジェクト作業の監視・コントロール

　プロジェクト作業の監視・コントロールは，知識エリアでは，プロジェクト統合マネジメントに属しており，プロジェクトマネジメント計画書で定義されたパフォーマンス目標を達成するため，進捗を追跡し，レビューし，統制するプロセスである．監視には，状況報告，進捗測定，予測が含まれる．実績報告書には，スコープ，スケジュール，コスト，資源，品質，リスクなどのプロジェクトのパフォーマンスに関する情報が提示される．これらの情報は，他のプロセスのインプットとなる．

1. 総合変更管理

　統合変更管理は，知識エリアではプロジェクト統合マネジメントに属しており，すべての変更要求をレビューし，変更を承認し，さらに要素成果物，組織のプロセス資産，プロジェクト文書，プロジェクトマネジメント計画書などへの変更をマネジメントするプロセスである．統合変更管理プロセスは，プロジェクトの開始から完了に至るまで実行される．プロジェクトマネジメント計画書，プロジェクト・スコープ記述書，およびそのほかの要素成果物は注意深く，継続的に変更をマネジメントすることにより維持しなければならないのである．それは変更の却下や承認によって行われ，確実に，承認済みの変更だけが改訂ベースラインに組み込まれるようにする．

1. スコープ検証

　スコープ検証は，知識エリアではプロジェクト・スコープ・マネジメントに属しており，完成したプロジェクトの要素成果物を公式に受け入れるプロセスである．スコープ検証には，顧客やスポンサーとともに要素成果物をレビューし，それが満足のいくような形で完成したことを確認し，顧客やスポンサーから要素成果物の公式な受け入れを得ることが含まれる．スコープ検証と品質管理との違いは，スコープ検証が要素成果物の受け入れを主要な関心事としているのに対して，品質管理は要素成果物に規定されている品質要求事項を満たしているかどうかを主要な関心事としている点にある．品質管理は一般にスコープ検証に先行して行うが，これら2つのプロセスを並行して行うこともある．

1. スコープ・コントロール

　スコープ・コントロールは，知識エリアではプロジェクト・スコープ・マネジメントに属しており，プロジェクト・スコープと成果物スコープの状況を監視し，スコープ・ベースラインに対する変更をマネジメントするプロセスである．プロジェクト・スコープをコントロールすることにより，すべての変更要求および提案された是正処置や予防処置が，確実に統合変更管理プロセスを通して処理する．実際に変更が生じたときは，プロジェクトのスコープ・コントロールはスコープの変更をマネジメントするために適用され，そしてほかのコントロール・プロセスとの間の統合が行われる．

1. スケジュール・コントロール

　スケジュール・コントロールは，知識エリアではプロジェクト・タイム・マネジメントに属しており，プロジェクトの進捗を更新するためにプロジェクトの状況を監視し，スケジュール・ベースラインに対する変更をマネジメントするプロセスである．

1. コスト・コントロール

　コスト・コントロールは，プロジェクトの予算を更新するためにプロジェクトの状況を監視し，コスト・ベースラインの変更をマネジメントするプロセスである．予算を更新するには，現在までに支出した実コストを記録する必要がある．予算の増額の認可は，総合変更管理プロセスを通してのみ承認される．支出に対して達成した作業の価値を考慮に入れずに資金の支出だけを監視しても，プロジェクトにとってなんらの意味を持たないのである．それは単にプロジェクト・チームの支出が認可された資金の範囲内に留まっていることを示すにすぎないのである．そのようなことから，コスト・コントロールのほとんどの作業は，プロジェクト資金の支出と，その支出によって達成された実際の作業との関係を分析することにあてられる．コスト・コントロールを効果的に行うには，認可されたコスト・パフォーマンス・ベースラインとそのベースラインの変更をマネジメントすることが重要である．

1. 品質管理

　品質管理は，パフォーマンスを査定し，必要な変更を提案するために，品質活動の実行結果を監視し，記録するプロセスである．品質管理のプロセスは，プロジェクトの期間を通して実行される．品質標準には，プロジェクトのプロセスと成果物の目標を含まれる．プロジェクトの結果は，要素成果物，およびコストやスケジュールのパフォーマンスなどのプロジェクトマネジメントの結果からなる．品質管理のプロセスは，品質管理部またはそれと同様の名称をもつ組織内の部門によって実行されることが多いのである．品質管理活動では，プロセスやプロダクトの品質不良の原因を特定し，それらを除去する処置を推奨し，実行する．

1. 実績報告

　実績報告は，状況報告，進捗測定，予測などのパフォーマンス情報を収集し，配布するプロセスである．プロジェクトの進捗とパフォーマンス，さらにプロジェクトの結果の予測の理解と伝達を行うために，実績報告プロセスではベースラインと実績データとの比較を定期的に収集し，分析する．

1. リスクの監視・コントロール

　リスクの監視・コントロールは，プロジェクト全期間にわたり，リスク対応計画を実行し，特定したリスクを追跡し，残存リスクを監視し，新たなリスクを特定し，リスクのプロセスの有効性を評価するなどのプロセスである．リスクの監視・コントロールでは，代替戦略の選択，コンティンジェンシー計画や代替計画の実行，是正処置の実施，プロジェクトマネジメント計画書の修正などを行うのである．リスク対応担当者は，計画の有効性，予想外の影響，リスクに適切に対処するために必要な修正などについて，プロジェクトマネジャーに定期的に報告する．また，リスクの監視・コントロール・プロセスでは，将来のプロジェクトの役に立つように，教訓データベースやリスク・マネジメントのテンプレートなどの組織のプロセス資産も更新する．

1. 調達管理

　調達管理は，調達先との関係をマネジメントし，契約のパフォーマンスを監視し，必要に応じて変更と是正を行うプロセスである．購入者と納入者の双方は，同様の目的で調達契約を管理する．各当事者は，双方が契約上の義務を果たすこと，およびそれぞれの法的な権利が保護されることを確実に行われなければならないのである．調達管理プロセスは，納入者のパフォーマンスが契約上の要求事項を満たすこと，および購入者は契約条件に従って行動していることを確実にするものである．契約関係は法的な性質を有することから，プロジェクトマネジメント・チームは，調達を管理するときに取る行動の法的な意味を明確に知っていることが非常に重要である．複数の納入者が関係する大規模なプロジェクトにおいては，さまざまな納入者間のインターフェースをマネジメントすることが調達管理の重要な側面である．

## 5.4　EVM（Earned Value Management）の概要

　PMBOK Guide[1]によれば，アーンド・バリュー・マネジメント（EVM）には，さまざまな形式があるが，それはパフォーマンス測定の方法として広く使われている．EVMはプロジェクトのスコープ，コスト，スケジュールの測定結果を統合するもので，プロジェクトマネジメント・チームがプロジェクトのパフォーマンスや進捗状況を評価し，測定することに役立つ．EVMは1つの統合ベースラインを設定することが欠かせないプロジェクトマネジメント技法であり，プロジェクト期間中は，その統合ベースラインに対してパフォーマンスを測定する．EVMの原理は業種を問わず，すべてのプロジェクトに適用することができる．EVMでは各ワーク・パッケージと各コントロール・アカウントについて，以下の3つの基本的な側面を表現し監視する．

1. プランド・バリュー（Planned Value）

　プランド・バリュー（PV）は，アクティビティまたはワーク・ブレークダウン・ストラクチャーの構成要素について，その遂行すべき作業に割り当てられた認可済みの予算のことである．PVは，認可された詳細な作業と，その認可された作業に対する予算であり，プロジェクトのライフサイクル全体にわたりフェーズに振り分けられたものである．PVのすべてを称してパフォーマンス測定ベースライン（PMB）と呼ぶことがある．プロジェクトに対するプランド・バリューの合計値は，完成時総予算（BAC）とも呼ばれる．

1. アーンド・バリュー（Earned Value）

　アーンド・バリュー（EV）は実施した作業の価値のことであり，アクティビティまたはWBS要素の当該作業に割り当てられた承認済み予算に基づいて表示したものである．それは，認可を受け，完了した作業に当該完了作業の認可済み予算を付したものである．測定されるEVはPVからなるベースライン（PMB）に関連付けられている必要がある．WBSについて測定されたEVの値は，認可済みPV予算より大きくなることはない．EVの表示は，プロジェクトの完了割合を表するために使用されることが多い．進行中の作業を測定するには，各WBS要素に対する進捗測定基準を確立する必要がある．プロジェクトマネジャーは，現在の状況を把握するためには増分にする値でEVを監視し，長期のパフォーマンス傾向を把握するためには累計値でEVを監視する．

1. 実コスト（Actual Cost）

　実コスト（AC）は，アクティビティまたはWBS要素の作業を完了するために実際に発生し，記帳されたコストの合計である．それは，EVを測定した作業を完了するために発生したコストの合計である．このACは，PVとして予算化されたものやEVとして測定されたものと同じ内容のもので比較しなければならない（例：直接労働時間のみ，直接費のみ，間接費を含めたすべての費用）．ACには上限がなく，EVを達成するために支出さらた費用はすべて測定する．

実コストと承認されたベースラインとの差異も同じく監視する．

## 5.5　EVMの関連するプロジェクトマネジメント項目について

プロジェクトマネジメントの知識エリアでは，コミュニケーション・マネジメントの領域における1つツールとして位置付けられるが，ほかにも，タイム・マネジメント，コスト・マネジメント，調達マネジメントとの関係性がある．また，EVMは，スコープ，スケジュール，コストなどプロジェクトの計画をベースラインとして設定し，そのベースラインを基礎としてプロジェクトの状況を監視し，監視の内容から分析と評価し，コントロールすることから，プロジェクトマネジメントの基本原則を取り扱う総合的な関係を持ち合わせている．

## 5.6　参考文献

[1] Project Managemant Institute. PMBOK Guide. 第4版, PMI, 2008, p39, p59-64, p81-82, p89, p93, p116, p123-125, p160, p179-182, p206, p266, p308, p335.

[2] こうちゃん. プロジェクト作業の監視コントロール. PMへの道. http://sayaji.seesaa.net/article/96501685.html, 2013-09-15.

[3] 情報処理振興事業協会. EVM活用型プロジェクトマネジメント導入ガイドライン. 情報技術・市場評価基盤構築事業, 2002, p2-4.

[4] 木村良一. EVMの基礎. http://www.mss.co.jp/technology/report/pdf/17-06.pdf#search='EVM%E5%88%86%E6%9E% 90%E3%81%AE%E5%9F%BA%E7%A4%8E', 2013-10-01.

第6章

システムについて

# 6.　システムについて

## 6.1　本章の構成

　この章では，本研究と目的となるシステムについて記述する．本研究の目的であるEVM自動描画システム開発を使用する際の条件，システムに必要なデータの入力や入力されたデータの抽出方法，EVMに必要なデータのシステムによる計算方法などについて記述する．

## 6.2　システムを使用する際の条件

　本研究で開発するシステムは，GitHub上にEVMをを描画するというシステムだがシステムを使用する際には，決められた使用上の条件事項がシステムを使用する際に，必要になる．ここでは，システムの使用上の条件事項を記述する．

* 時間単価はプロジェクトメンバー全員同額とする．
* EVMに必要な計算は，時間単位で計算する．
* Issuesは一人に割り振ることにする．
* 時間単価はメンバーに閲覧されてもよいものとする．
* Issuesのcommentは1つのIssuesに1日1件しか記述できないものとする．
* EVMの集計は，日にちごとにPV，AC，EVごとに集計し，EVMを作成する．
* リポジトリ1つに対して，1つのグラフを作成する．

## 6.3　EVMを作成するために必要な数値のデータ

　システムによってEVMを描画する際に必要となるEVMを作成する数値のデータは，計画予定単価（PV），実際にかかった時間単価（AC），実績価値（EV）の数値のデータが必要となる．それぞれの数値のデータをIssuesに入力しておく必要がある．Issuesに入力する際には，データをJSON形式で入力する必要がある．そのため，各項目は英語表示で入力する．英語表記することによって，GitHub APIで数値のデータを抽出する際に，文字のバグなどの問題を事前に防ぐことが可能である．以下がGitHubのIssues，commentに入力する項目と各項目の英語表記である．英語表記が必要な項目は，カッコ内に記述する．

1. Issuesに入力する項目

* タスク名
* タスク期限日（end）
* 計画予定時間（time）
* 時給金額（rates per hour）
* 材料費（material costs）

IssuesのJSON形式での記入例：



1. commentに入力する項目

* 直接労働時間数（direct hours）
* 進捗測定基準度（measure work in progress）
* 使用材料費（material costs）

commentのJSON形式での記入例：



## 6.4　EVMの計算方法

Issues，commentの内容からEVMを作成するために必要な数値のデータの計算方法は以下に記述する．

* PVの計算方法

材料費（material costs）＋計画予定時間（time）×時給（rates per hour）＝計画予定単価

* ACの計算方法

使用材料費（material costs）＋直接労働時間（direct hours）×時給（rates per hour）＝

実際にかかった時間単価

* EVの計算方法

全体の計画予定単価（最終累計PV）×進捗測定基準（measure work in progress）＝

実績価値

※全体の計画予定単価（最終累計PV）とはプロジェクト終了時の計画予定単価，つまりPVの最終合計数値のことである．

## 6.5　GitHub APIで抽出するデータ

開発するシステムによりグラフを描画するために，必要なデータをGitHub APIによって抽出する．抽出するデータと各データの利用方法は以下に記述する．

GitHubからGitHub APIによって抽出するデータの項目

* Issuesのclose時の時刻
* Issuesのcomment時の時刻
* Issuesの内容
* commentの内容

## 6.6　GitHub APIで取得したデータの使用目的

* 「Issuesのclose時の時刻」の使用目的

Issuesのclose時の時刻によって，いつ作業をしたかを判断する基準として使用する．

判断基準：

12/25の23：30にcloseした場合，12/25に作業したという事が判明する．

12/27の01：30にcloseした場合，12/26に作業したという事が判明する．

* 「Issuesのcomment時の時刻」の使用目的

Issuesのclose時の時刻と同じ目的であり，Issuesのcommentした時の時刻によって，いつ作業をしたかを判断する基準として使用する．

* 「Issuesの内容」の使用目的

Issuesの内容には，「タスク名」「タスク期限日」「計画予定時間（time）」「時給（rates per hour）」「材料費（material costs）」が入力されている．この入力されたデータは，PVとEVを計算する際に使用するので必要である．

* 「commentの内容」の使用目的

commentの内容には，「直接労働時間（direct hours）」「進捗測定基準（measure work in progress）」「使用材料費（material costs）」が入力されている．この入力されたデータは，ACとEVを計算する際に使用するので必要である．

## 6.7　GitHubからデータ抽出する方法

GitHub APIを使用し，データを抽出する方法については以下に記述する．

1. GitHub APIを使用する際の事前の準備に，curlをインストールする．

sudo apt-get install curl

1. ログイン情報入力を省略する方法

以下のプログラムは，ログイン情報を何度も入力するのは手間がかかるので，画面に表示をさせると手順がかかるので，ユーザー名とパスワードをファイルgithub.passwdに書いておく必要がある．（ユーザー名とパスワードの間は半角のコロン（:））．

echo 'ユーザ名:パスワード' > github.passwd

chmod 600 github.passwd

1. 以下のプログラムは，正しく書けたことを確認するコードである．

cat github.passwd

1. 以下のプログラムは，ログイン情報をファイルに書き込んだら，以下のように利用する．$（コマンド）の部分は，コマンドの実行結果で置き換えられることに注意する必要がある．

curl -s -u $(cat github.passwd) "https://api.github.com/repos/Diogenesthecynic/FullScreenMario/issues"

1. APIの呼び出し結果の処理

　以上の結果を処理するためには，プログラムを書かなければならないが，内容を取り出す程度の簡単な処理なら以下のようなプログラムでは，パイプで実現できる．

curl -s -u $(cat github.passwd) "https://api.github.com/repos/Diogenesthecynic/FullScreenMario/issues" | grep title

表6-1　コマンドの説明

|  |  |
| --- | --- |
| コマンド | 説明 |
| cuel | HTTPやFTPでのダウンロードやアップロードするための様々な機能を持ったコマンドである． |
| echo | 引数に指定された文字列や変数の内容を表示する．文字列の場合は，表示したい文字列を「"」で囲むのである．変数の場合は，変数名の前に「$」をつけるのである． |
| >（リダイレクト） | 入出力の方向を変えることを意味する．具体的には処理結果の出力先を画面からファイルへの書き込みや切り替えを行えたり，入力元をキーボードからファイルからの読み出し，または，切り替えたりすることである．リダイレクトを用いると，コマンドの処理結果をファイルに書き込んだり，処理結果をファイルに書き込んだり，処理結果のリスト・ファイルを作成したりすることができる． |
| cat | ファイルの内容を表示する．複数のファイルを連結して出力することもできる． |
| |（パイプ） | コマンドの実行結果を，次のコマンドの入力として扱えるようにするシェルの機能である．パイプを利用するには，「|」という記号を使用する．すると，左のコマンドの結果を右のコマンドに入力として渡すことができる．パイプを用いることで，出力結果をいったんファイルに落とすことなく，複数個のコマンドを1回の操作で実行できるようになる． |
| grep | 文字列を検索する．ファイル中の文字列を検索する．検索パターンとして正規表現を使用できる．また，指定した検索パターンにマッチする文字列を表示する． |

## 6.8　GitHubから複数ページにわたるデータを取得する方法

1. このプログラムはPythonで記述する．ここでは，以下の2つの部分にわけることにする．

* データを取得しファイルに保存するだけの単純なプログラムapi.py
* ファイルからデータを読み出して処理するプログラムjp

2つの部分に分けることにより，何度もGitHubにアクセスすることを避けることができ，シェルスクリプトなどを活用できるからである．

1. GitHubから複数ページにわたるデータを取得する方法の準備

　以下のプログラムは，PythonとHTTPアクセスのためにrequestsをインストールする．（requestsにはレスポンスヘッダのlink属性を読み取る機能が用意されている．）

sudo apt-get install python python-setuptools

sudo easy\_install requests

1. api.pyの利用

「GitHub APIを使う練習」の作業の通り，github.passwdを作成する．パスワードに「:」を使っていると使用できないのである．

1. api.pyのプログラムは以下に記述する．

#!/usr/bin/python

# coding: UTF-8

import sys, json, requests

tmp = open('github.passwd').readline().rstrip('\n').split(':');

username = tmp[0]

password = tmp[1]

#print >> sys.stderr, username,password

url = sys.argv[1]

count = 0

while (url is not None):

#print url

r = requests.get(url, auth=(username, password))

print >> sys.stderr, r.headers['status'],

for item in r.json():

count = count + 1

print json.dumps(item)

if (r.links.has\_key('next')):

url = r.links['next']['url']

else:

url = None

print >> sys.stderr, count

api.pyに使用されているコマンドは以下に記述する．

表6-2　api.pyに使用されるコマンド

|  |  |
| --- | --- |
| コマンド | 説明 |
| python | さまざまな分野のアプリケーションで使用されている，極めてパワフルな動的プログラミング言語である． |
| coding | プログラミング言語を使ってソフトウェアの設計図にあたるソースコードを作成することである．「プログラミング」とほぼ同義だが，仕様書やフローチャートなど抽象的な設計文書の内容を，プログラミング言語を使って具体的なコードに変換していく，という意味合いが強いのである． |
| UTF-8 | UNICODEで定義される文字集合で記述された文字列を、バイナリデータに変換する方法の1つである．UTF-8では、1文字を1～6バイトの可変長のバイナリデータに変換する． |
| import sys | 「sys」モジュール以下のすべての変数や関数が「sys.」を先頭につける形で利用できるようになる． |
| requests | データを取得するコマンドである．取得するデータの指定方法は，コマンドの前後に記述する． |
| tmp | ファイルの種類の1つである． |
| readline() | 1行ごとに読み込み，その処理を繰り返すのである． |
| rstrip() | 文字列の末尾部分を除去したコピーを返すのである．引数rstrip(chars)は除去される文字集合は指定する文字列である． |
| split() | splitを単語の境界として文字列を単語に分割し，分割された単語からなるリストを返している． |
| print | 出力先を一時的にファイルへ変更する．「>>」の後に出力オブジェクトを渡し，カンマ（，）で区切ってから出力を行う文字列を渡すのである． |
| sys.stderr | インタープリタの標準入力・標準出力・標準エラー出力に対応するファイルオブジェクトである．stderrは必ずしも組み込みのファイルオブジェクトである必要はなく，write()メソッドを持つオブジェクトであれば使用することができる．stderrを別のオブジェクトに置き換えても，os.popen()，os.system()，osのexecなどから起動されたプロセスが使用する標準I/Oストリームは変更されない． |
| sys.argv | Pythonスクリプトに渡されたコマンドライン引数のリストである．argv[0]はスクリプトの名前となるが，フルパス名かどうかは，OSによって異なる．コマンドライン引数に-cをつけてPythonを起動した場合，argv[0]は文字列’-c’となる．スクリプト名なしでPythonを起動した場合，argv[0]は空文字列になる．argv[1]にコマンドライン引数で与える． |
| count | 文字列count中に部分文字列countが出現する回数を返している． |
| r.headers | ヘッダーを読み込むコマンドである． |
| None | 関数にデフォルト値が渡されないときのように，値がないことを表すために容易られる． |
| links | リンクするというコマンドである． |
| json.dumps | オブジェクトをJson形式のデータをサポートするファイル的オブジェクトへのストリームとして直列化する． |
| r.json | Json言語でデータを読み込むという意味である． |
| item | キーと値のリストを取得する． |
| requests.get | 指定されたデータを取得するコマンドである． |
| auth  （authentication） | 認証といる意味である．正当性を検証する作業である． |
| r（read） | 読み込むというコマンドである． |

1. 以下のプログラムはユーザー名Diogenesthecynic，リポジトリ名FullScreenMarioのopenissuesのデータをファイルopenissues.txtに結果を保存している．ユーザー名とリポジトリ名は，取得するデータの対象によって，変更する必要がある．APIでデータを取得するのには時間がかかるため，このように1度ファイルに保存しておくのが，よいのである．

このプログラムで使用されている?per\_page=100はなくても，このコマンドは稼働するが，このコマンドは，1度に取得するデータを最大にしておき，APIを利用する回数を抑える．

python api.py "https://api.github.com/repos/Diogenesthecynic/FullScreenMario/issues?per\_page=100" > openissues.txt

以下のプログラムはプログラムは，closedしたIsseusを取得する．

python api.py "https://api.github.com/repos/Diogenesthecynic/FullScreenMario/issues?per\_page=100&state=closed" > closedissues.txt

以下のプログラムはcommitのデータをapi.pyを利用し，抽出するプログラムである．

python api.py "https://api.github.com/repos/jquery/jquery/commits?per\_page=100" > commits.txt

## 6.9　GitHub APIで取得したデータの処理方法

　「GitHubから複数ページにわたるデータを取得する方法」で取得したデータは、データが1行に1件JSON形式で書かれている。シェル上でJSON形式のデータを扱いたいときは、jqを使うのが便利。

1. GitHub APIで取得したデータの処理方法の準備

　作業ディレクトリにjqのバイナリをコピーする．

1. jqを実行できるようにする．

chmod +x jq

1. jqの利用

「GitHubから複数ページにわたるデータを取得する方法」の作業でできたファイルの，openissues.txt，closedissues.txt，commits.txt，events.txtを例に使用する．

1. データを確認の方法

less openissues.txtでは結果が見づらいので，jqを使用し，./jq '.' openissues.txt | lessとすると，整形して表示される．さらに，./jq '.' -C openissues.txt | less –Rとすれば，カラーで表示することがこのプログラムでできる．

## 6.10　IssuesのデータをGitHubから抽出する方法

　Issuesのタイトルだけを一覧表示するには，./jq '.title' openissues.txtのコードを使用する．結果を別のファイルに保存したければ，>を使用することで，別のファイルに結果を保存できる．

Issusesの作成日時だけを一覧表示するには，./jq '.created\_at' openissues.txtのコードを使用する．

タイトルは重複する可能性があるから，./jq '.id,.title,.created\_at' openissues.txtのように，IDをつけておくとよいと考えられる．以下のプログラムによる結果は，行数を3で割ったあまりが0のときは改行をし，それ以外（つまり1か2）のときは，カンマを付けて整形すればよいのである．

./jq '.id,.created\_at,.title' openissues.txt | awk '{ printf("%s", $0); if (NR % 3 == 0) printf("\n"); else printf(","); }'

プログラムが正常に稼働し，プログラム結果を取得することができたら，> openissues.csvなどを付けてファイルに保存し，ExcelやUbuntuのLibreOffice Calcで読み込むのである．closedissues.txtに変更すると，Issuesのオープン日時とクローズ日時，タイトルの一覧を作成するには，以下のプログラムを使用する．

./jq '.id,.created\_at,.closed\_at,.title' closedissues.txt | awk '{printf("%s", $0); if (NR % 4 == 0) printf("\n"); else printf(","); }'

このプログラムはclosedissuesについてのデータをAPIで取得するものである．このプログラムの結果をCalcで読み込んでD1に=(C1-B1)\*86400などと入力してD列全体にコピーすれば，Issuesがオープンしてからクローズするまでの時間が求められる．

以下にGitHub APIで取得したデータの処理方法で使用したコマンドを記述する．

表6-3　 GitHub APIで取得したデータの処理方法で使用したコマンド

|  |  |
| --- | --- |
| コマンド | 説明 |
| chmod | ファイルやディレクトリのアクセス権限を変更する．所有者，グループ，その他のユーザーに対して，それぞれ読み出し，書き込み，実行権限を与えられる．ここで実行権限をファイルに指定するとコマンドやスクリプトなどとして使用できる．パーミッションは「記号」または「数値」で指定する．変更を行えるのは，そのファイルの所有者だけである． |
| less | テキストファイルの内容を閲覧する．ファイルの内容をページ単位で自由に閲覧できる． |
| awk | テキスト処理の伝統的なツールである． |
| ?per\_page=100 | 1度に取得するデータを最大にする． |

## 6.11　システムの概要

　ここでは，システムがどのように使用されるかを説明する．まず，プロジェクトマネジャーは，Issuesを使用し，メンバーに作業の指示をする．この時のプロジェクトマネジャーが指示するIssuesの内容は，「タスク名」「タスク期限日」「計画予定時間数」「時給金額」「材料費」を記述する．そして，メンバーがIssuesによってタスクの指示を受け，作業を終了し進捗報告をする際のcommentの内容には，「直接労働時間数」「進捗測定基準度」「使用材料費」を記述する．このIssuesによる指示と報告から，EVMを作成するのに必要なPV，AC，EVを計算するために必要な数値をGitHub APIで抽出し，ブックマークレットを利用して，GitHubのページ上にEVMを描画することができる．

　このシステムによって，GitHub上でのソフトウェア開発において，標準的なプロジェクトマネジメント手法を導入しやすくなることが期待される．

## 6.12　システムの機能

　プロジェクトマネジャーの指示の内容とメンバーの進捗報告の内容から，システムがEVMを作成するのに必要なデータを抽出する．そのデータを決められた計算方法で計算し，その数値からEVMを作成し，ブックマークレットを利用し，GitHubのページ上にグラフを挿入する．

## 6.13　参考文献

[1] Linuxコマンド一覧. UNIX入門. http://www.k4.dion.ne.jp/~mms/unix/linux\_com/echo.html, 2013-12-05.

[2] 日経BP. オープンソース/Linux. IT pro by日経コンピュータ. http://itpro.nikkeibp.co.jp/article/COLUMN/20060228/231012/, 2013-12-05.

[3] 日経BP. Linuxコマンド集. IT pro by日経コンピュータ. http://itpro.nikkeibp.co.jp/article/COLUMN/20060227/230728/, 2013-12-05.

[4] Linuxコマンド一覧. UNIX入門. http://www.k4.dion.ne.jp/~mms/unix/linux\_com/less.html, 2013-12-05.

[5] TATSUO IKURA. 複合文とブロック. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.pythonweb.jp/tutorial/if/index2.html, 2014-01-22.

[6] ats. シンプルで覚えやすい仕様，パワフルなPythonのWebアクセスモジュールrequest. trivial technologies. http://coreblog.org/ats/python-http-module-request/, 2014-01-22.

[7] 株式会社インセプト Incept Inc. 認証. IT用語辞典e-Words. http://e-words.jp/w/E8AA8DE8A8BC.html, 2014-01-22.

[9] TATSUO IKURA. キーと値のリストを取得（keysメソッド，valuesメソッド，itemメソッド）. AjaxTower IT技術の習得を目指す方向けの学習サイト. http://www.pythonweb.jp/tutorial/dictionary/index8.html, 2014-01-22.

[10] Copyright. JSONエンコーダおよびデコーダ. Documentation. http://docs.python.jp/2.6/library/json.html, 2014-01-22.

第7章

システム検証について

# 7.　システム検証について

## 7.1　本章の構成

　この章では，本研究と目的となるシステムで目的のデータや数値を抽出ができるか，または，EVMを表示できるのかなどを検証する．

## 7.2　検証方法

　ここでは，システムの検証の方法について記述する．システムの検証をするためGitHub内に検証用のリポジトリを作成する．リポジトリ内に仮想プロジェクトデータとしてIssuesを発行する．入力されたデータを元に以下の検証方法で，システムの検証をする．

1. GitHub APIの検証

GitHub APIを利用して，「Issuesのclose時の時刻」「Issuesのcomment時の時刻」「Issuesの内容」「commentの内容」を抽出できるか検証する．

1. EVM作成の検証

抽出したデータから，EVMを作成できるかを検証する．

1. GitHub上にEVM表示の検証

作成されたEVMをGitHub上に表示できるかを検証する．

## 7.3　検証用のGitHub内の仮想プロジェクトデータのリポジトリ内容

システムの検証をするためGitHub内に検証用のリポジトリを作成する．作成したリポジトリでは，プロジェクトマネジャーがメンバーにIssuesで指示を行い，その指示に従って作業をし，進捗報告をする内容である．以下に仮想プロジェクトデータに関する詳細を以下に記述する．

* 仮想プロジェクトデータのリポジトリの詳細
* リポジトリ名：costover
* 作成されたIssues内容
* Issues1：

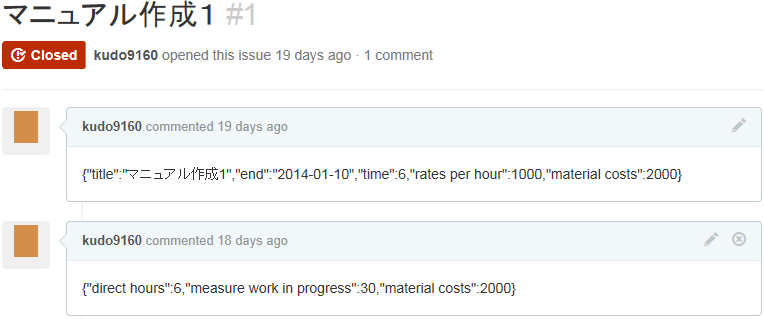


図7-1　テストIssues1

* Issues2：

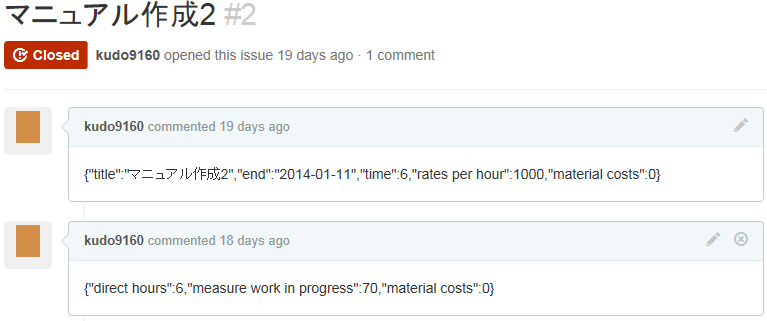


図7-2　テストIssues2

* Issues3：

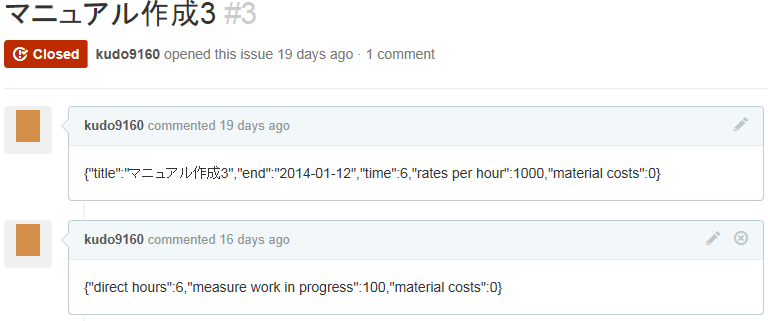


図7-3　テストIssues3

各Issuesからわかることと詳細について以下に記述する．

表7-1　仮想プロジェクトデータのリポジトリ詳細

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Issue名 | 期限 | 計画  予定時間 | closeした  時間 | 直接  労働時間 | 進捗  測定基準 | 材料費 |
| マニュアル作成1 | 2014年  1月10日 | 6時間 | 21：12 | 6時間12分 | 30％ | 2000円 |
| マニュアル作成2 | 2014年  1月11日 | 6時間 | 17：18 | 6時間18分 | 70％ | 0円 |
| マニュアル作成3 | 2014年  1月12日 | 6時間 | 23：12 | 6時間12分 | 100％ | 0円 |

指定された計算方法による計算後の数値

以下は，仮想プロジェクトデータのリポジトリ詳細から指定された計算方法によって計算した各Issuesの累計PV，累計AC，累計EVの数値である．

表7-2　仮想プロジェクトデータの累計数値

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Issue名 | 期限 | 累計PV | 累計AC | 累計EV |
| マニュアル  作成1 | 2014年  1月10日 | 8000円 | 8200円 | 6000円 |
| マニュアル  作成2 | 2014年  1月11日 | 14000円 | 14500円 | 14000円 |
| マニュアル  作成3 | 2014年  1月12日 | 20000円 | 20700円 | 20000円 |

完成イメージEVM

　以下は，計算された数値からシステムで作成されるであろうEVMの完成イメージ図である．

図7-4　仮想プロジェクトデータのEVMのイメージ図

1. GitHub APIの検証

GitHub APIを利用して，「Issuesのclose時の時刻」「Issuesのcomment時の時刻」「Issuesの内容」「commentの内容」を抽出できるか検証する．

GitHub APIで使用されるプログラムは以下に記述する．

Issuesの内容を抽出するプログラム

./jq '.id,.created\_at,.closed\_at,.title,.body' closedissues.txt | awk '{printf("%s", $0); if (NR % 5 == 0) printf("\n"); else printf(","); }' > closedissues.csv

commentの内容を取得するプログラム

./jq '.id,.created\_at,.body' closedcostoverissuescomments.txt | awk '{printf("%s", $0); if (NR % 3 == 0) printf("\n"); else printf(","); }' > closedcostoverissuescomments.csv

使用したプログラムから抽出した内容

　使用したプログラムから抽出した内容は以下に記述する．

Issuesの内容

表7-3　ID・open時の時刻・closed時の時刻

|  |  |  |
| --- | --- | --- |
| 25326122 | 2014-01-09T15:17:13Z | 2014-01-12T14:12:03Z |
| 25326011 | 2014-01-09T15:15:46Z | 2014-01-11T08:18:06Z |
| 25320128 | 2014-01-09T13:49:42Z | 2014-01-10T12:12:44Z |

表7-4　Issuesタイトル・Issues内容

|  |  |  |
| --- | --- | --- |
| マニュアル作成3 | {\title\":\"マニュアル作成3\" | \"end\":\"2014-01-12\" |
| マニュアル作成2 | {\title\":\"マニュアル作成2\" | \"end\":\"2014-01-11\" |
| マニュアル作成１ | {\title\":\"マニュアル作成1\" | \"end\":\"2014-01-10\" |

表7-5　Issues内容

|  |  |  |
| --- | --- | --- |
| \"time\":6 | \"rates per hour\":1000 | \"material costs\":0}\r\n" |
| \"time\":6 | \"rates per hour\":1000 | \"material costs\":0}\r\n" |
| \"time\":6 | \"rates per hour\":1000 | \"material costs\":2000}" |

commentの内容

表7-6　ID・comment時の時刻・comment内容

|  |  |  |
| --- | --- | --- |
| 32022731 | 2014-01-10T12:12:44Z | {\direct hours\":6 |
| 32090580 | 2014-01-11T08:18:06Z | {\direct hours\":6 |
| 32123323 | 2014-01-12T14:12:03Z | {\direct hours\":6 |

表7-7　comment内容

|  |  |
| --- | --- |
| \"measure work in progress\":30 | \"material costs\":2000}" |
| \"measure work in progress\":70 | \"material costs\":0}" |
| \"measure work in progress\":100 | \"material costs\":0}" |

GitHub APIの検証結果

　GitHub APIを使用したプログラムから抽出した内容は，仮想プロジェクトデータのリポジトリ詳細と照らし合わせると，同じであるためGitHub APIの検証結果は，正確なデータを取得することができたと考えられる．

1. EVM作成の検証

抽出したデータから，EVMを作成できるかを検証する．抽出したデータからEVMを作成するには，Google Chart APIを使用する．Google Chart APIにより，GitHub APIで抽出したデータからEVMを作成するためのURLは以下に記述する．

http://chart.apis.google.com/chart?chs=640x240&chd=t:-1,8000,14000,20000,-1|-1,8200,14500,20700,-1|-1,6000,14000,20000,-1&cht=lc&chco=0000ff,ff0000,00ff00&chxt=x,y&chxl=0:|0|20140110|20140111|20140112||1:|0|5000|10000|15000|20000|25000|2:&chg=5000,-1,1,5&chm=d,0000ff,0,-1,10|s,ff0000,1,-1,10|x,00ff00,2,-1,10&chdl=PV|AC|EV&chds=0,25000

Google Chart APIにより，抽出したデータからEVMを作成するためのURLから作成されたグラフは，以下に記述する．

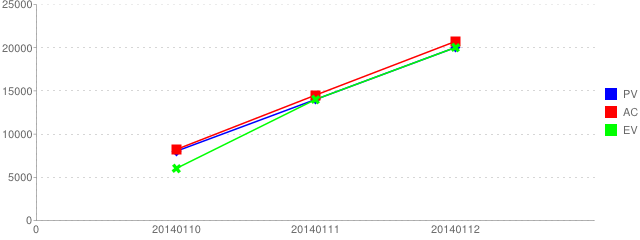


図7-5　Google Chart APIによって作成したEVM

EVM作成の検証結果

GitHub APIを使用したプログラムから抽出した内容からGoogle Chart APIを使用し，抽出したデータから作成したEVMと仮想プロジェクトデータから予想したEVMの完成イメージ図と照らし合わせると，同じであるためEVM作成の検証結果は，正確なデータを取得することができたと考えられる．

1. GitHub上にEVM表示の検証

作成されたEVMをGitHub上に表示できるかを検証する．EVM作成の検証結果から作成されたGoogle Chart APIによって作成したEVMをGitHub上で表示するために，以下のコードを使用する．

* //img要素の生成

var myEvm = document.createElement("img");

* //img要素のsrc属性の設定

myEvm.setAttribute("src", "http://chart.apis.google.com/chart?chs=640x240&chd=t:-1,8000,14000,20000,-1|-1,8200,14500,20700,-1|-1,6000,14000,20000,-1&cht=lc&chco=0000ff,ff0000,00ff00&chxt=x,y&chxl=0:|0|20140110|20140111|20140112||1:|0|5000|10000|15000|20000|25000|2:&chg=5000,-1,1,5&chm=d,0000ff,0,-1,10|s,ff0000,1,-1,10|x,00ff00,2,-1,10&chdl=PV|AC|EV&chds=0,25000");

* //img要素のstyle属性の設定（必須ではない）

myEvm.setAttribute("style", "border:solid gray 1px; padding:10px;");

* //EVMの挿入場所

var myContainer = document.getElementById("js-repo-pjax-container");

* //EVMの挿入

myContainer.parentNode.insertBefore(myEvm, myContainer);

コードのsrc属性の設定の内容がGitHub APIで抽出した内容の数値である．このコードをブックマークレットで使用するURLに構築すると以下のようになる．

javascript:(function() {

var myEvm = document.createElement("img");

myEvm.setAttribute("src", "http://chart.apis.google.com/chart?chs=640x240&chd=t:-1,8000,14000,20000,-1|-1,8200,14500,20700,-1|-1,6000,14000,20000,-1&cht=lc&chco=0000ff,ff0000,00ff00&chxt=x,y&chxl=0:|0|20140110|20140111|20140112||1:|0|5000|10000|15000|20000|25000|2:&chg=5000,-1,1,5&chm=d,0000ff,0,-1,10|s,ff0000,1,-1,10|x,00ff00,2,-1,10&chdl=PV|AC|EV&chds=0,25000");

myEvm.setAttribute("style", "border:solid gray 1px; padding:10px;");

var myContainer = document.getElementById("js-repo-pjax-container");

myContainer.parentNode.insertBefore(myEvm, myContainer);

})();

このURLを使用したブックマークレットを使用し，GitHub上にEVMを挿入すると以下のように表示される．

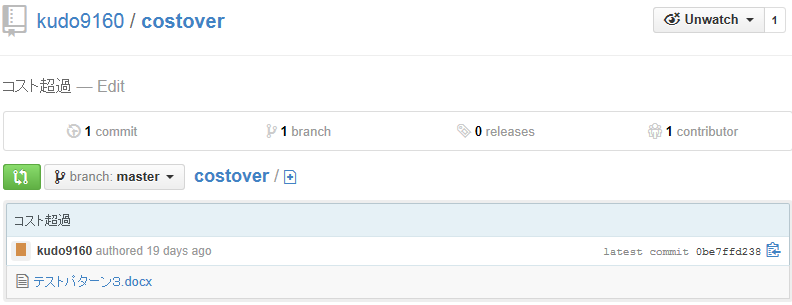


図7-6　EVM挿入前の仮想プロジェクトデータの検証用のGitHubのリポジトリ画面

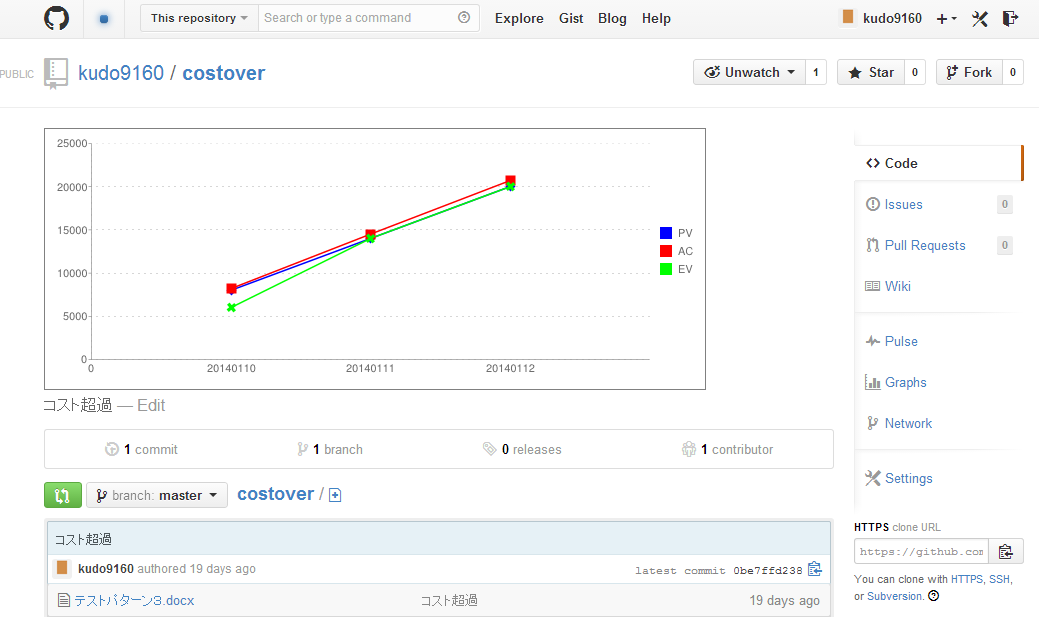


図7-7　EVM挿入後の仮想プロジェクトデータの検証用のGitHubのリポジトリ画面

GitHub上にEVM表示の検証結果

Google Chart APIによって作成したEVMと仮想プロジェクトデータから予想したEVMの完成イメージ図と照らし合わせると，同じであり，GtiHubのリポジトリ画面上にEVMを表示することが可能であると認識できる．この事から，GitHub上にEVM表示の検証結果は，正確なデータを表示することができたと考えられる．

第8章

結論

# 8.　結論

## 8.1　結果

本研究の結果として，開発したシステムによってGitHub上でEVMを描いた様子が図である．このように，EVMのために必要な情報をIssuesに記述しておくことによって，図1のようなGitHub上にEVMを描画することができる．

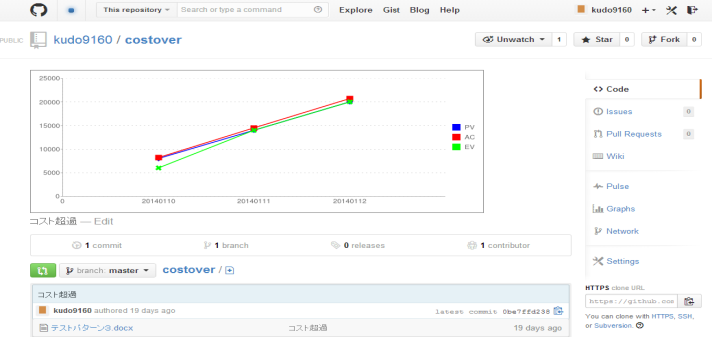


図8-1　GitHubのページ上に描かれたEVM

## 8.2　結論

ソフトウェア開発においてよく利用されているサービスGitHub上で，プロジェクトマネジメントのための重要なツールであるEVMを利用可能にするシステムを開発した．本システムは，GitHubでタスクを管理するのに用いられるIssuesというしくみを利用している．Issuesでのタスクの記述方法を統一するだけで利用可能であることが，本システムの大きな利点であろう．本システムによって，GitHub上でのソフトウェア開発において，標準的なプロジェクトマネジメント手法を導入しやすくなることが期待される．

## 8.3　今後の課題

本研究で開発したシステムによるEVMの作成には，細かい条件上システムによるEVMを作成が可能である．このシステムでタスクのみのEVMを作成する場合は，タスクごとにリポジトリを作成する必要がある．またメンバーごとにEVMを作成するには，システムに新たな仕様を加える必要があり，システムの改善が必要となる．さらに，このシステムでEVMを作成することができるのは，1つのリポジトリをにつき，1つのEVMしか作成することができないため，複数のリポジトリの内容からEVMを作成するには，システムに新たな仕様が必要である．また，本研究で開発したシステムは，EVMに必要な数値を取得するための計算方法もこのシステム用に決めた計算方法のみとなっている．EVMでは，進捗測定基準度の計算方法には，工数を使用した方法もあるが，本研究のシステムは，工数を使用した方法には対応していないため，EVMを作成することができないのである．本研究で開発したシステムは，EVMを作成する多数の方法に現段階では対応することが困難と考えられ，本研究の方法以外に対応するにはシステムの改善必要と考えられえる．