Telelogic Synergy の紹介

リリース 6.6a

本書をご使用になる前に、55 ページの「付録：特記事項」に記載されている情報をお読みください。

本書は、Telelogic Synergy バージョン 6.6a（製品番号 5724V66）および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

目次

はじめに 1

他ツールからの移行 1
表記規則 2
Telelogic Synergy のグラフィカルユーザーインターフェイス 2
Telelogic Synergy のコマンドラインインターフェイス 2
書体と記号 3

Telelogic Synergy を使用する利点 5

構成管理ツールの目的 5
使いやすく、購入後すぐ使える 6
早い立ち上がり 6
新規ユーザーのための生産性の高い作業環境 6
柔軟かつ自動化された作業フロー 7
チームエンジニアリング環境を確保する 7
ワールドワイドな Telelogic Synergy 情報の制御と転送 8
Windows 開発環境とのシームレスな統合 8

Telelogic Synergy の用語 11

Telelogic Synergy データベース 11
タスクとオブジェクト 11
オブジェクトの詳細 14
チェックアウト／チェックイン 14
履歴 15
プロパティ 17
カレントタスク 17
ユーザー、ライフサイクル、状態 17
プロジェクトおよびプロジェクト グルーピング（グループ化） 19
ディレクトリと候補 20
ワークエリア 21
同期 21
オブジェクトの使用、作成、追加、削除、消去 23
更新、ベースライン、タスク、プロセスルール 24
フォルダ 25

ビルド、製品、Makefile 25

Telelogic Synergy の方法論 27

タスクベースの方法論 27
ユーザー 29
プロジェクトと作業フロー 29
リリース 31
プロジェクト目的 32
更新プロパティ 33
Telelogic Synergy のデフォルト作業フロー 35
タスクの利用 35
開発プロセス 36
統合テストサイクル 36
システムテストサイクル 38
システムテストベースラインのリリース 39
次期リリースの準備 39
まとめ 40
並行開発 42
並行同時開発 43
並行プラットフォーム開発 43
並行リリース開発 44
コンポーネントベースの開発 44
コンポーネントの管理 45
コンポーネントの公開 45
コンポーネントの参照 46
プロセスパターン 46

用語解説 47

商標 56

索引 59

1 はじめに

Telelogic® Synergy™ は理解しやすく、機能豊富な構成管理システムです。ソフトウェア開発チームは、Telelogic Synergy を使って、ソフトウェア管理、ドキュメント開発、それらの保守作業を実施できます。Telelogic Synergy は、複雑かつ分散した環境で作業する中規模の開発チームから大規模の開発チームまでをサポートします。

本書の目的は、ユーザーが Telelogic Synergy の用語、概念、方法論の基礎知識を身につける助けとなることです。Telelogic Synergy 製品にはいくつかのインターフェイスがありますが、本書では Telelogic Synergy を主要なインターフェイスとして説明します。

本書では、読者が Windows® や Unix® オペレーティングシステムとそのディレクトリファイル構造の基礎知識を有していることを前提に説明をしてゆきます。

他ツールからの移行

Telelogic Synergy は、特定のファイルの複数バージョンを維持するプロセスを管理するなど、多くの管理を行います。ユーザーは、Telelogic Synergy は初めてでも、これまでに PVCS®（Windows）や RCS や SCCS（UNIX）のようなバージョン管理ツールを使った経験があるかもしれません。このようなツールを使ってもファイルバージョンの管理はできますが、Telelogic Synergy のようなワークフローの管理、製品の再構築性、ルールベースの構成更新まではできません。

Telelogic Synergy は、プロセスという点では他のバージョン管理ツールと大きな違いがありますが、他のツールに精通したユーザーであれば、Telelogic Synergy のいずれかのインターフェイスに移行するのは難しくありません。

表記規則

ここでは、本書で使用する表記規則について説明します。

Telelogic Synergy のグラフィカルユーザーインターフェイス

Telelogic Synergy には、以下に示すグラフィカルユーザーインターフェイスがあります。本書では、Telelogic Synergy 製品（Telelogic Synergy および Telelogic Synergy Classic）について、一般的な名称「Telelogic Synergy 製品」を使用します。特定のインターフェイスについて言及する際には、以下のいずれかの名称を使用します。

- **Telelogic Synergy**

  このインターフェイスは、開発者やビルドマネージャとして作業するユーザー用です。

- **Telelogic Synergy Classic**

  このインターフェイスは管理者のための CM 機能を提供します。

Telelogic Synergy のコマンドラインインターフェイス

本書で示すコマンドラインインターフェイス（CLI: Command Line Interface）の例は、Windows プラットフォームおよび UNIX プラットフォームの両方にあてはまります。

書体と記号

下表はこのドキュメントで使用されている書体と記号の表記規則について説明します。

書体	説明
イタリック	表題や用語に使用されます。ロール (*developer*)、状態 (*working*)、グループ (*ccm_root*) およびユーザー (*laura*) の名前も表します。
太字	選択可能なボタン、アイコンやメニューパスなどのアイテムに使用されます。ダイアログボックス、ダイアログボックスオプション、ツールバー、フォルダ、ベースライン、データベース、リリース、プロパティ、タイプの名前にも使用されます。強調にも使用されます。
`Courier`	コマンド、ファイル名、ディレクトリパスに使用されます。表示どおりに入力するコマンド構文を表します。コンピュータの画面に表示される文字を表します。
`Courier Italic`	ユーザーが指定するコマンド文字列内の値を示します。たとえば、(`drive:\username\commands`)。

サポートへのお問い合わせ

Telelogic 製品のサポートと情報は、Telelogic サポートサイトから IBM Rational Software Support に移行中です。この移行期間中は、サポートの連絡先がお客様によって異なります。

製品サポート

- 2008 年 11 月 1 日より前に Telelogic 製品を取引されたお客様は、Synergy サポート ウェブサイトをアクセスしてください。製品情報の移行後に、IBM Rational Software Support site に自動で転送されます。
- 2008 年 11 月 1 日より前に Telelogic 製品のライセンスをお持ちではなかった新規のお客様は、IBM Rational Software Support site をアクセスしてください。

お客様サポートにお問い合わせいただく前に、問題を説明するために必要な情報をご用意ください。IBM ソフトウェアサポート担当員に問題を説明する際には、担当員が迅速に問題を解決できるように、問題の具体的な内容と必要な背景情報をすべて伝えてください。あらかじめ以下の情報をご用意ください。

- 問題発生時に使用していたソフトウェアとそのバージョン
- 問題に関連したログ、トレース、メッセージなど
- 問題を再現できるかどうか。再現できる場合はその手順
- 回避策があるかどうか。ある場合は、その回避策の内容

その他の情報

Rational ソフトウェア製品、ニュース、イベント、その他の情報については、IBM Rational Software Web site をご覧ください。

2 Telelogic Synergy を使用する利点

構成管理ツールの目的

構成管理システムの目的は、開発チームがソースコードの変更、ドキュメント開発、そして製品管理プロセスを管理する手助けとなることです。以下の話題では、開発チームが直面するいくつかの主要な問題と、複数開発者が複数ファイルに対して行った多くの変更を調整するという課題を軽減、または取り払うことに、構成管理ツールがどのように役立つかということに焦点を置きます。

構成管理ツールは、フラットに展開されたディレクトリ構造からも、深くネスティングされたディレクトリ構造からも、既存のプロジェクトを取り込むことができなければなりません。いったんプロジェクトデータを Telelogic Synergy に移行したら、このツールにより以降の開発作業の基準となるベースラインやプロジェクトバージョンを簡単に作成できるようになります。

理想的には構成管理ツールでは、透過的に、つまり、他の開発者が行った変更によって発生するトラブルから切り離された状態で、開発者が通常通りにソースコードファイルに対する作業に取り組める必要があります。ただし一方で、開発がある程度進んだ段階では開発者が他の開発者の行った変更を自分のプロジェクトに取り込むことができるように、この分離独立性を緩和する必要もあります。構成管理システムは、開発者が独立して作業できるエリアを提供すると同時に、ソースコードに対する変更を共有するチームとして効率的に作業できる環境も提供しなければなりません。

構成管理ツールは、開発チームが各ビルドできわめて安定した実行コードを享有できるようなビルド環境を実現する必要があります。さらに、ビルドの品質レベルの向上に合わせた、作業を継続できるプロセスを提供する必要があります。このようなプロセスを導入すれば、チームが製品リリースに向けて正しく進めるようになり、次のレベルのビルドを破損するようなコードも修正できます。

構成管理ツールは、開発ソフトウェアのさまざまなバージョンを正しく再構築できる機能を備えている必要があります。一般に開発チームは「不具合訂正リリース」と「新機能リリース」の開発作業を並行して行います。そして当然ながら、これらのリリースごとに異なるビルド要件があります。通常は不具合訂正リリース作業の完了後、新機能リリースチームは不具合訂正リリースでの変更を自分たちの作業に取り込みます。これらのことから、構成管理ツールの重要な目標は、何種類ものプロジェクトを同時に立ち上げることができ、ファイルの配置されるディレクトリの場所にかかわらずコードを再利用できる手段を開発者に提供することといえます。並行開発された製品のリリース後、技術サポート要員はカスタマーサポート向けに過去の主要なリリースを完全に再現できる必要があります。そして、並行開発の結果として出荷される製品リリースの数は増えてゆくため、それに応じて、特定バージョンを再構築できる機能の必要性も高まってゆきます。

Telelogic Synergy の利点

Telelogic Synergy は、開発チームが簡単、迅速、安全に作業ができる完全な構成管理環境を提供します。この節では、Telelogic Synergy がソフトウェア開発会社に提供する優れた機能と利点について説明します。

使いやすく、購入後すぐ使える

大半の開発チームは開発システムの停止時間のほとんどない厳しい日程の中で作業をしています。開発者が新しいツールを快適に使いこなして作業できるように、Telelogic Synergy は、以下の機能を標準で提供します。

- 直感的で、使いやすいグラフィカルユーザーインターフェイス（GUI）と製品全体を制御できる CLI
- 変更追跡のためのシンプルなタスクベースのアプローチ
- 多くの標準的なツールや開発環境とのインテグレーション
- テンプレートを活用できる柔軟なプロセスサポート
- カスタマイズ不要で効率よくツールを使用できる
- すぐに使用できる定義済みの権限／オブジェクトライフサイクル、安全性、アクセスルール
- GUI と CLI のヘルプ

早い立ち上がり

開発チームは、インストール後すぐに（通常、即日）Telelogic Synergy を使い始められます。この迅速な立ち上がりを実現するのは、Telelogic Synergy の以下の機能です。

- 自動化された移行ツールを使って、既存プロジェクトのファイルシステムを Telelogic Synergy 制御下に簡単に持ち込むことができます。
- Telelogic Synergy は既存のビルド手順や make 手順と互換性があるため、既存の makefile を使用して製品をビルドできます。

新規ユーザーのための生産性の高い作業環境

Telelogic Synergy の起動後、作業を軌道に乗せるのは難しくありません。新規ユーザーであっても、すぐに Telelogic Synergy を使い始めることができます。構成管理担当者に Telelogic Synergy ユーザーとして登録してもらえれば、プロジェクトをコピーして自分のワークエリアを作成し、以下の作業を実施できます。

- 作業対象のタスクを選択する。
- プロジェクトをコピーし、ファイルを変更する。
- タスクを完了する、つまり、自分が行ったすべての変更をチェックインする。

ここで紹介したステップは、Telelogic Synergy の最も基本的な使用法です。他の数多くの機能を学習すれば、Telelogic Synergy の利点を最大限に活用できるようになります。学習のためには、弊社の提供するトレーニングセッションへの参加をお勧めします。もちろん、基本的な機能を理解するだけでも、すぐに Telelogic Synergy を使用して作業を開始し、開発チームが重要な納期や納品を守れるように支援できます。

柔軟かつ自動化された作業フロー

Telelogic Synergy のタスクベースの方法論は、開発ソフトウェアのビルドと試験を実施するための単純明快な方法を提供します。この方法に従えば、早期に問題を検出でき、求められている品質レベルを確保できます。タスクベースの作業フローでできることは、以下のとおりです。

- 変更の理由を簡単に確認でき、変更を盛り込むために修正されたファイルをすべて特定できる。
- 分離された環境で変更作業を行い、必要に応じて他からの変更を確認できる。
- テスト用ソフトウェアに組み込む変更やリリース用に組み込む変更を細かく制御できる。
- 並行バージョンや変更の盛り込み洩れ等の構成上の矛盾を検知できる。
- 出荷するソフトウェアを保管し、再構築できる。
- ビルド管理操作を自動化できる。
- 同時並行開発を管理できる。
- 別チーム用の異なる作業フローを簡単に設定できる。

チームエンジニアリング環境を確保する

以下の機能によって、開発チームは Telelogic Synergy 自体をさほど意識せずにソフトウェア開発プロジェクトを推進できます。

- データのリポジトリは、市販の信頼できる RDBMS（リレーショナルデータベース管理システム）。
- Telelogic Synergy は、開発者の使用している既存ディレクトリ構造とツールを利用している。
- Telelogic Synergy は、個人用の、独立したワークエリアを開発者に提供。自分の行った変更を他の開発者に公開する前に、このワークエリアでチェックアウトしたファイルに自由にアクセスして、プロトタイプの作成、編集、ビルド、デバックなどを実施できる。
- Telelogic Synergy では、ベースライン構成を正確に作成することで、プロジェクトを再現できる。
- Telelogic Synergy では、タスク関連付けとチェックイン／チェックアウト機能によってファイルやプロジェクトを追跡できる。

- 開発者は、Telelogic Synergy の「更新（*update*）」機能を使用して自分のプロジェクトを更新することによって、必要に応じて、テスト済みのファイルやチェックイン済みのファイルを利用できる。
- Telelogic Synergy では、自動化された同時並行開発サポートとビルトインセキュリティを通じて、同時に発生する変更を管理できる。
- 開発者は、統合フェーズやリリースフェーズのソフトウェアのテストを行っている間でも開発作業を中断せずに続行できる。
- デフォルトで提供されるライフサイクルを使用すると、許可された変更のみがリリースされることが保証される。
- Telelogic Synergy は、データとファイルのアクセス操作を、データベースの各ユーザー向けに設定したセキュリティの下で制御する。

ワールドワイドな Telelogic Synergy 情報の制御と転送

分散型変更管理（Telelogic Synergy Distributed）製品を使用すると、世界中の任意の場所にある任意の数の Telelogic Synergy データベース間で、ソフトウェアの変更を共有できます。Telelogic Synergy Distributed の機能は以下のとおりです。

- 開発者は、使い慣れた共通のユーザーインターフェイスを使用して作業できる。
- 適切な方法論を選択して、データベース間のデータ転送の特性と方向を定義できる。
- ソースオブジェクト、プロジェクト、フォルダ、タスクをどの分散 データベースへも送信できる。送信の際にオブジェクトのグループ化方法に制約はない。
- データベース全体またはデータベースのサブセットを自動または手動で転送できる。
- 転送を起動する前に転送データのリストを最終確認できる。
- 異なる場所にあるデータベースを使って同時並行開発を続行でき、後で Compare および Merge （比較／マージ）機能を使用してコンフリクトを解決できる。

Windows 開発環境とのシームレスな統合

Telelogic Synergy 構成管理ツールは、各社からリリースされている主要な開発環境とのインテグレーションをサポートしてきました。このインテグレーション機能を利用すると、使用している開発環境にソースコントロールや構成管理の機能をもたせることができます。インテグレーションには、簡単に実行できるインストールとセットアッププログラムが含まれます。以下のインテグレーションがサポートされます。

- Eclipse™
- IBM® Rational® Application Developer
- Microsof®t Visual Studio®
- Microsoft Visual C++®
- Microsoft Visual Basic®

利用できるインテグレーションの最新情報については、IBM Rational Software Support site をご覧ください。

3 Telelogic Synergy の用語

本章では、Telelogic Synergy の基本的な概念と用語について説明します。基本概念が系統的に理解できるよう順番に概念を紹介しています。本章は、順番に読み通すことをお勧めします。

手早く参照できるように、本書の巻末には用語集があります。

本書は、すべての Synergy インターフェイスについて説明します。基本的にすべてのインターフェイスに適用される一般的な用語を使用し、ある特定のインターフェイス、たとえば Telelogic Synergy の動作を説明する場合は、そのインターフェイス特有の用語を使用します。

概念と方法論はすべてのインターフェイスについて共通です。たとえば、すべてのインターフェイスについて、タスクベースの方法論がサポートされており、開発者が新しいプロジェクトで作業を開始する際には自分のプロジェクトを更新して最新のメンバーを取り込むことを推奨しています。

Telelogic Synergy データベース

Telelogic Synergy データベースとはデータの保管場所です。ソースファイルとデータファイルを含め、すべての制御データ、そのプロパティ、およびこれらの相互関係が保存されます。Telelogic Synergy データベースは、制御するデータの量、またデータの体系化の方法にしたがって、1 つの場合も複数の場合もあります。

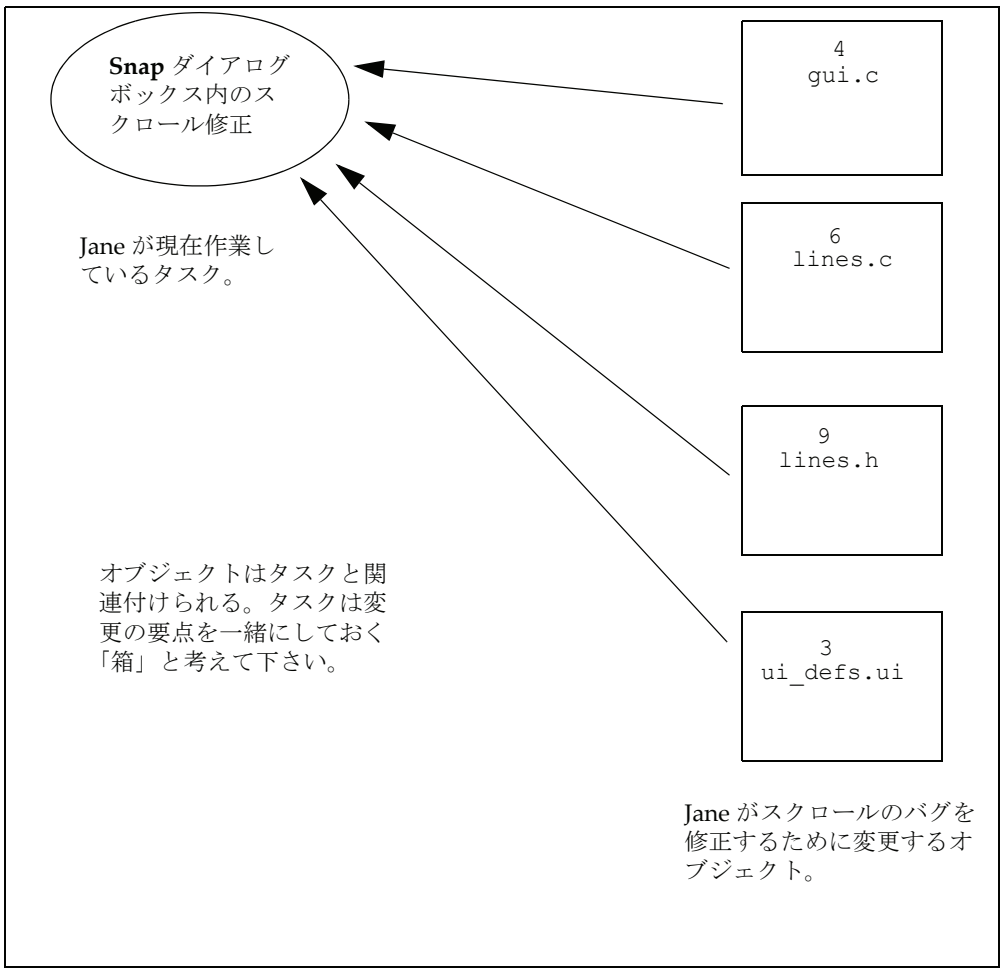
タスクとオブジェクト

タスク（*task*）とはソフトウェアアプリケーションに必要な一まとまりの論理的変更です。タスクは、要求されている変更のために必要なソフトウェアへの修正をグループ化します。また、変更内容の説明と変更作業を完了する責任者の名前も保持します。

オブジェクト（*object*）とは、ファイルやディレクトリといったデータの集まりです。たとえば、ソースファイル、makefile、テスト結果、ディレクトリ、ドキュメントなどです。変更の追跡用として、オブジェクトの各改訂版をオブジェクトバージョンといいます。各オブジェクトバージョンには、より詳細に定義するための一連のプロパティ（例、名前（**name**）、所有者（**owner**）、**作成時刻（create time**）があります。

たとえば、あるユーザーがアプリケーションの GUI について、バグを報告してきたとします。GUI グループのリーダーは、バグの修正作業を Jane という名前の開発者に割り当てます。彼女には、"**Fix scrolling in Snap dialog**"（「**Snap** ダイアログボックス内のスクロールの修正」）と呼ぶタスクが割り当てられます。彼女が Telelogic Synergy を使用して、**Snap** ダイアログ内のスクロールの問題を修正するためのオブジェクトをチェックアウトする時、常にオブジェクトがタスクと関連付けられます。

これらの概念については、下図を参照してください。



タスクとオブジェクトには関連があります。タスクでグループ化されるオブジェクトを、「タスクと関連付けされたオブジェクト」と呼びます 。特定の問題を修正するためのすべてのオブジェクトは、タスク名で記述されるある論理グループの中にあります。上図の右側にあるオブジェクトバージョンは、**Snap ダイアログボックス内のスクロール修正**というタスクと関連付けられます。なぜならば、これらのオブジェクトバージョンにはこのタスクを完了する（問題を修正する）ために必要なコード変更が含まれているからです。各オブジェクトの上部にある数字がオブジェクトのバージョンを示します。

タスクの名前（この場合、"**Fix scrolling in Snap dialog**"（「**Snap** ダイアログボックス内のスクロールの修正」））を概要（*Synopsis*）と言います。また、新規タスクを作成する場合、Telelogic Synergy はそのタスクに番号を割り当てます。番号と概要の他に、タスクに

は変更についての情報が盛り込まれます。たとえば、担当者の名前などです。タスクがある開発者に割り当てられると、自動的に開発者の名前が担当者名として設定されます。さらに、タスクを作成すると、以下のプロパティが設定されます。

- リリース（*release*）

  ソフトウェアアプリケーションのバージョンを示すラベルです。**エディタ /2.0** や **Telelogic Synergy/6.6** といった開発ソフトウェアアプリケーションにとって重要なリリース名から構成される値です。開発しているソフトウェアに固有のリリース値をビルドマネージャが設定します。

- プラットフォーム（*platform*）

  この論理的変更の対象であるハードウェアプラットフォームを指定します。**HP-UX** や **WIN2K** といった開発ソフトウェアアプリケーションにとって重要なプラットフォーム名から構成される値です。開発しているソフトウェアに固有のプラットフォーム値をビルドマネージャが設定します。この値は便宜的に使用されるだけなので指定は任意です。

- サブシステム（*subsystem*）

  ソフトウェアサブシステムを指定します。たとえば、クライアント／サーバーソフトウェアアプリケーションを開発する場合は、サブシステムとして考えられるのは、**クライアント**、**サーバー**、**コミュニケーションなどでしょう**。会計ソフトウェアアプリケーションを開発する場合は、サブシステムは **AR**、**AP**、**GL などである可能性もあります**。開発しているソフトウェアに固有のサブシステム値を CM アドミニストレータが設定します。この値は便宜的に使用されるだけなので指定は任意です。

  タスクとソースオブジェクトは、Telelogic Synergy データベースの中に存在します。タスクにはバージョンはなく、あるライフサイクル（19 ページで説明）に従います。タスクは他のタスクを含むことはできません。

オブジェクトの詳細

Telelogic Synergy データベースで管理されるすべてのオブジェクトは、**名前**、**バージョン**、**タイプ**、**インスタンス**などのプロパティによって一意に識別されます。

デフォルトで、「4 部名称」（オブジェクトスペックまたは完全名称とも呼ぶ）は以下のように記述されます。

name-version:type:instance（名前 - バージョン：タイプ：インスタンス）

４部名称の例には、**main.c-3:csrc:2**、**draw.c-beta:csrc:7** などがあります。

オブジェクトの名前は、制限文字以外の任意の文字の組み合せです（使用できない文字のリストについては、Telelogic Synergy CLI Help を参照）。タイプは、デフォルトのタイプ（例、**csrc**、**ascii** 等）から選択するか、自分で作成できます。名前、バージョン、タイプは指定可能ですが、インスタンスは Telelogic Synergy が算出して決定します。

インスタンスは、同じ名前とタイプを持つ複数のオブジェクトを区別するために使用します。しかし、バージョンとは意味が異なります。たとえば、プロジェクトに 20 個の異なる makefile があるとします。それぞれに `makefile` という名前が付けられ、それぞれが異なるディレクトリにあり、またそれぞれにたくさんのバージョンがあります。**makefile-4** を使用したい場合、そのオブジェクトのクエリは、**makefile-4** と呼ぶ 6 つのオブジェクトを返すこともあります。この場合、`instance` プロパティがどの makefile オブジェクトを使用するかを区別します。インスタンスの値は通常数値ですが、たとえば Telelogic Synergy Distributed を使用するデータベースなどでは、英数字の場合もあります。

複数のディレクトリで固有の値をもつオブジェクトバージョンを使用できます。この場合は、パス名を使って特定のオブジェクトバージョンを参照できます。4 部名称の固有の ID は、ファイルの場所が変更されても不変です。

チェックアウト／チェックイン

既存のオブジェクトを修正するには、そのオブジェクトの変更可能なバージョンを作成する必要があります。この修正可能なオブジェクトバージョンは、チェックアウト（*check out*）操作で作成できます。この操作でオブジェクトの既存のバージョンから新規バージョンが作成されます。新規バージョンには、既存のバージョンからすべてのプロパティがコピーされます。

チェックアウト操作は、PVCS **get -l** コマンド（Windows）、あるいは RCS **co -l** や SCCS **get -e** コマンド（UNIX）の操作に似ています。しかし、Telelogic Synergy では、オブジェクトをチェックアウトする必要があるのは、それを修正する予定がある場合のみです。単に内容を見たい場合には、オブジェクトをチェックアウトする必要はありません。また、どんな種類のオブジェクトでもチェックアウトできます（ファイル、ディレクトリ、シンボリックリンク、実行形式ファイルなど）。

チェックイン（*check in*）操作ではオブジェクトバージョンを書き込み不可にして保管します。いったんチェックインをすると、他のユーザーがオブジェクトを利用できるように

なります。オブジェクトをチェックインすると、誰が修正／使用できるかを定義する、状態（*state*）プロパティが変更されます。チェックイン操作は、PVCS **put -u** コマンド（Windows）、あるいは RCS **ci -u** や SCCS **delget** コマンド（UNIX）の操作に似ています。

Telelogic Synergy では同じファイルを複数回チェックインできます。たとえば、テストの準備ができた時にファイルをチェックインし、また、リリースの準備ができた時にも別の状態でチェックインできます。

開発者のジェーンの例を用います。彼女は、スクロールのバグを修正するために、ファイルを修正することになっています。彼女は、自分に割り当てられた仕事（タスク）を完了するために修正する必要のある **gui.c** や **lines.c** などのファイルを、チェックアウトします。ファイルを修正してバグを直すと、彼女は、次の製品ビルドでこれらの修正したファイルが取り込まれるように、タスクを完了します。

チェックアウトとチェックインのプロセスは、開発サイクルにおける変更管理の重要な一部です。開発者があるオブジェクト（たとえば、ファイルなど）を修正目的でチェックアウトすると、その開発者はそのファイルの個人作業用版の所有者となります。Telelogic Synergy では、デフォルトで、別の開発者がこのファイルの自分の作業用のバージョンをチェックアウトできるようになっています。この手法は並行開発（*parallel development*）と呼ばれ、同じファイルの異なるバージョンは、パラレルバージョン（*parallel versions*）と呼ばれます。したがって、ファイルのあるバージョンがすでに使用されているという理由だけでは、作業プロセスの遅延は起こりません。

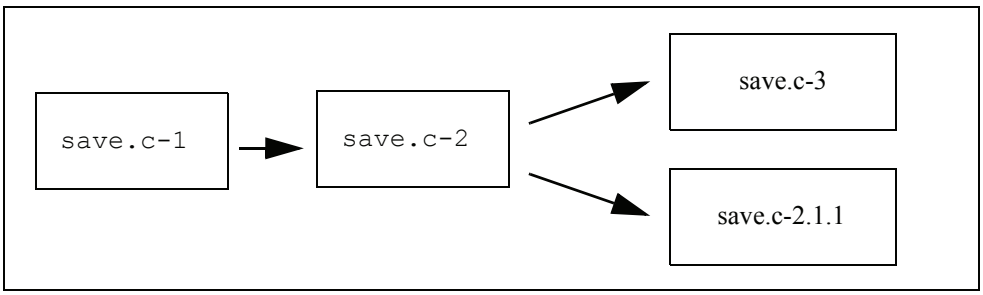
パラレルバージョンは、ある段階に来たところでマージする必要があります。Telelogic Synergyのマージ機能を使うと、あるファイルの2つのパラレルバージョンの情報をマージできます。2 つのオブジェクトバージョンをマージすると、第 3 のバージョンが作成されます。Telelogic Synergy は、直近の共通祖先を使用して、新バージョンに盛り込むべき変更を提示します。ファイル内にコンフリクトがない場合は、新バージョンはすぐに使用できます。コンフリクトが存在する場合は、コンフリクトを起こしているコードからどれを新バージョンで使用するのかを選択する必要があります。

履歴

オブジェクトの履歴（*history*）には、オブジェクトのすべての既存バージョンと、それらのバージョン間の関係が表示されます。ここでいう履歴とは、現行オブジェクトバージョン以前に作成されたすべてのオブジェクトバージョン（先行バージョン）と、現行オブジェクトバージョン以後に作成されたすべてのオブジェクトバージョン（後継バージョン）を指しています。

save.c ファイルの履歴を下図に示します。矢印は、どのバージョンがどこからチェックアウトされたかを示します。たとえば、バージョン 2 はバージョン 1 からチェックアウトされたものです。バージョン 1 はバージョン 2 の先行バージョンであり、バージョン 3 と

2.1.1 はバージョン 2 の後継バージョンです。バージョン 3 と 2.1.1 はパラレルバージョンです。



プロパティ

オブジェクトのプロパティ（*properties*）は、あるオブジェクトと他のオブジェクトを区別するために使用します。オブジェクトの基本プロパティは、4 部名称（名前（`name`）、タイプ（`type`）、インスタンス（`instance`）、バージョン（`version`）で識別できる項目であり、さらに、所有者（`owner`）、状態（`status`）、プラットフォーム（`platform`）、リリース（`release`）などがあります。プロパティは Telelogic Synergy の［プロパティ］ダイアログボックスに表示されます。

プラットフォーム属性およびリリース属性は、ビルドやテスト用にソフトウェアバージョンを集める際に重要になります。Telelogic Synergy は、プラットフォームとリリースの値がビルドやテストを行いたい構成と一致するバージョンを集めます。

カレントタスク

カレントタスク（*current task*）は、現在作業を行っているタスクのことです。あるタスクをカレントタスクとして指定するということは、オブジェクトのチェックアウト時には必ずそのオブジェクトがカレントタスクと関連付けられることを意味します。あるタスクとして定められた変更作業がすべて終わったら、そのタスクを完了 *(complete)* します。タスクを完了すると、そのタスクに関連付けられたすべてのオブジェクトバージョンがチェックインされます。

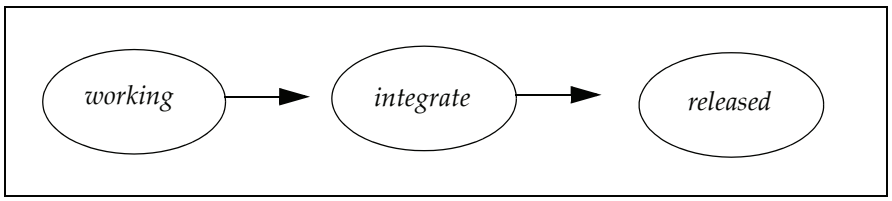
タスクを使用すれば、Telelogic Synergy を利用した開発ライフサイクル全体を通して、特定の意味を持つ一まとまりの変更内容を一つの「ユニット」として管理できます。タスクの情報を使用することで、あるビルドに盛り込みたい機能やテストで確認したい機能などを指定して特定のソフトウェアバージョンを集めることができます。また、ソフトウェアの変更点を表現するものとしてタスクを利用すると、あるタスクに関連付けられたオブジェクトバージョンはすべて一緒に使用すべきであること、そして、プロジェクトにおいて、あるタスクに関連付けられたオブジェクトバージョンの一部は使うが他は使わないというやり方は機能しないことが明らかになります。Telelogic Synergy は、これらの情報に基づいてソフトウェア開発ライフサイクルの早期の段階で構成上の問題を検出してくれます。

ユーザー、ライフサイクル、状態

Telelogic Synergy セッションでは、ユーザーは開発者（*developer*）またはビルドマネージャ（*build manager*）として作業します。あるユーザーが開発者かつビルドマネージャとして作業するように設定されている場合、そのユーザーは特に制約なく Telelogic Synergy の適切な操作を実行できます。通常は、開発者はソフトウェアの開発と試験のための操作を行います。ビルドマネージャは、ソフトウェアをインテグレートして開発者がアクセスできる開発エリアを構成、ビルドするとともに、テスト用エリアの構成、ビルド、リリース用ソフトウェアの準備を行います。

すべてのオブジェクトは、1 つのライフサイクル（*lifecycle*）に従います。ライフサイクルとは、オブジェクトの取り得る状態と、オブジェクトが現在の状態に基づいて遷移できる状態を指しています。また、オブジェクトの状態（*state*）は、ライフサイクルにおけるオブジェクトの段階と実行可能なアクションを定義しています。

デフォルトでは、Telelogic Synergy のタスクベースの方法論でオブジェクトが使用する 3 つの状態は、*working*（作業中）、*integrate*（統合）、*released*（リリース済み）です。下図は、デフォルトのオブジェクトの状態のライフサイクルにおける順序を示しています。



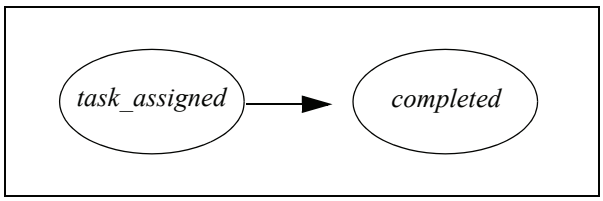
各状態の説明は以下のとおりです。

- *working*（作業中）状態は、すべての新規オブジェクトバージョンが取り得る状態です。新規作成された時や他のバージョンからチェックアウトされた時に、この状態になります。*working* 状態のオブジェクトは所有者による修正が可能です。
- *integrate*（統合）状態は、ビルド管理統合テストで使用されるオブジェクトが取る状態です。*integrate* 状態のオブジェクトは修正できません。
- *released*（リリース済み）状態は、リリースされたオブジェクトやマイルストーンに到達したオブジェクトが取り得る状態です。*released* 状態のオブジェクトは、修正できません。

別の観点でのライフサイクルを採用したい場合もあります。Telelogic Synergy には、*rejected*、*shared*、*visible* といった状態がオプションとして含まれています。

タスクに関連付けられたオブジェクトのライフサイクルは、そのタスクの状態に密接に関係します。オブジェクトをプロセスを通して動かすのは、タスクのライフサイクルです。たとえば、オブジェクトのチェックインはいつでもできますが、そのオブジェクトが統合テストで取り上げられるのは、そのオブジェクトと関連付けられたタスクが完了してからです。

デフォルトでは、タスクは *registered*、*task_assigned*、*completed* の状態を取ることができます。下図 はデフォルトのタスクの状態のライフサイクルでの順序を示します。



task_assigned（タスク割り当て済み）状態は、ある開発者に割り当てられたタスクに使用されます。タスクが *task_assigned* 状態にある場合、修正は可能であり、担当者（タスクを割り当てられたユーザー）が使用できます。デフォルトでは、ビルドマネージャがタスクを割り当てます。開発者は、自分自身にしかタスクを割り当てられません。

completed（完了）状態 は、終了したタスクに使用されます。タスクが *completed* 状態にある場合、タスクはチェックインされており、CM アドミニストレータだけが修正できます。タスクに関連付けられたすべてのオブジェクトバージョンは、タスクの完了前にチェックインされている必要があります。タスクの担当者だけがタスクを完了させることができます。担当者には、開発者またはビルドマネージャがなることができます。

プロジェクトおよびプロジェクト グルーピング（グループ化）

プロジェクト（*project*）は、関連ファイル、ディレクトリ、他のプロジェクト（サブプロジェクト）から構成されるユーザー定義のグループです。プロジェクトは、通常はライブラリや実行形式ファイルなどのソフトウェアを集めたものを表現しており、ファイルのディレクトリ構造を保持しています。たとえば、編集ツールを実装しているソフトウェアは、**editor** という名前のプロジェクトに保存されます。

他のオブジェクトと同様に、プロジェクトもバージョン管理されます。あるプロジェクトの別のバージョンは、別のバージョンのメンバーオブジェクトを含んでいたり、さらには異なるメンバーさえ含んでいる可能性があります。たとえば、**editor** プロジェクトの異なるバージョンは、それぞれ、最初のリリース editor/1.0 と継続リリース editor/1.1、editor/1.2、editor/2.0 を表す場合があります。リリース editor/2.0 の **editor** プロジェクトには、同じオブジェクトの新バージョンとともに、バージョン 1.0 には存在しなかった新規オブジェクトを含む可能性もあります。また、バージョン 2.0 には存在せずバージョン 1.0 にのみ存在するファイルもありえます。

ある一つのオブジェクトバージョンが複数のプロジェクトのメンバーである可能性もあります。同じオブジェクトバージョンが異なるプロジェクトに表示されていても、Telelogic Synergy データベースに存在するオブジェクトバージョンは一つです。

プロジェクトには他のプロジェクトを含むことができます。他のプロジェクトに含まれているプロジェクトをサブプロジェクト（*subproject*）と呼びます。一つのソフトウェアを別々のプロジェクトにグループ化して整理できます。たとえば、実行形式ファイルごとに

プロジェクトを 1 つ用意して、それら全部をアプリケーション全体を現しているプロジェクトのサブプロジェクトとして構成することができます。

どのようなプロジェクトでも、通常はいくつかの異なるバージョンが存在します。

- 開発プロジェクトは、開発者が使用するプロジェクトであり、必要な開発を行ったり変更点をテストするためのプロジェクトです。
- ビルド管理プロジェクトは、ビルドマネージャが使用するプロジェクトであり、テストやリリース用にソフトウェアを準備するためのプロジェクトです。
- *released* プロジェクト（リリース済みプロジェクト）とは、リリースされたソフトウェアまたは特定のマイルストーンに到達したソフトウェアのバージョンです。

プロジェクトグルーピングは、*working* およびビルド管理プロジェクトを「リリース」および「目的」にしたがってグループ化します。また、プロジェクトグルーピングは、自動的に作成、維持管理され、一連のプロジェクトを参照するための便利な方法を提供します。たとえば、**My 1.0 Insulated Development Projects** （「自分の 1.0 Insulated Development プロジェクト」）プロジェクトグルーピングは、開発者のすべてのプロジェクトを "Insulated Development" 目的の "1.0" リリースというくくりでグループ化します。プロジェクトグルーピングは、プロジェクトを更新する際に使われるタスクとベースラインも示しています。したがって、あるプロジェクトに存在するように選択されたメンバーは、そのプロジェクトグルーピング内のすべてのプロジェクトに常に存在することになります。

プロジェクトグルーピングを、タスク、ベースライン、リリース、プロジェクト目的といったさまざまな要素（プロパティ）を入れた上で、正しいメンバー構成を保持しているプロジェクトのグループを作成するための「容器」と考えてみます。一度プロジェクトグルーピングを作成しておけば、プロパティの追加 / 削除 / 変更、プロジェクト更新時に更新したくないタスクの指定、最新データを取得するためのプロジェクトグルーピング内のタスクとベースラインのリフレッシュ、あるプロジェクトグルーピングから別のプロジェクトグルーピングへのプロジェクトの移動、といった作業を簡単に行うことができます。つまり、プロジェクトを正しくグループ化しながら、必要に応じて柔軟に変更できます。

ディレクトリと候補

Telelogic Synergy は、ファイルと同様にディレクトリも制御します。ファイルシステムのディレクトリとは違って、Telelogic Synergy が作成するディレクトリ（*directory*）は、どのファイルがそのディレクトリに属しているかを把握しています。ディレクトリは、そのディレクトリ内の各ファイルについて、ディレクトリエントリ（*directory entry*）と呼ばれるプレースホルダを保持しています。ディレクトリエントリには、そのディレクトリに属するファイルの名前が記述されます。ファイルのバージョンではありません。たとえば、**delete.c** 用のディレクトリエントリは、**delete.c** という名前のファイルが必要であることを知っていますが、そのファイルの特定のバージョンを期待しているわけではありませ

ん。ディレクトリエントリ内に記述されるファイルとして適格なすべてのファイルバージョンを候補（*candidates*）と呼びます。

オブジェクトをディレクトリに追加したり、ディレクトリから削除するには、ディレクトリオブジェクトを書き込み可能にする必要があります。修正不可能な状態のディレクトリを修正（メンバーの追加や削除など）しようとすると、Telelogic Synergy は自動的にそのディレクトリの新規バージョンをチェックアウトします。カレントタスクが設定されていれば、その新規ディレクトリは自動的にカレントタスクに関連付けられ、タスク完了時に他の変更とともにチェックインされます。

ソースオブジェクトと同様にディレクトリでもパラレルバージョンが発生します。Telelogic Synergy は、パラレルバージョンのディレクトリのマージをサポートします。ディレクトリをマージする際、ディレクトリエントリ間の差分を比較し、マージ先のバージョンの中に盛り込むべきディレクトリエントリを選び出します。たとえば、あるユーザーが **sources** ディレクトリをチェックアウトして、**open.c** という名前のオブジェクトを追加し、別のユーザーがパラレルバージョンをチェックアウトして、**select.c** という名前のオブジェクトを追加した場合、マージ操作では両方の新しいディレクトリエントリが表示され、マージ先のバージョンに両方を含めることができます。

ワークエリア

ワークエリア（*work area*）は、プロジェクトをチェックアウトしたときに Telelogic Synergy がそのプロジェクトデータを書き込む、ファイルシステム内の場所です。ワークエリアは、ネットワークファイルシステム内であれば任意の場所に置くことができます。ワークエリア内のプロジェクトのディレクトリツリー構造は、Telelogic Synergy データベース内のプロジェクトのツリー構造と同一になります。

Telelogic Synergy は、ワークエリアとデータベースの間の同期を維持します。UNIX システムでは、ワークエリアにあるファイルは、データベースファイルに リンクされています。Windows システムでは、ワークエリアのファイルは、データベースファイルのコピーです。

プロジェクトの作成や変更を行うと、Telelogic Synergy は自動的かつ透過的にワークエリアを更新します。つまり、メンバーをプロジェクトに追加する場合は、Telelogic Synergy はワークエリアをその新しいファイルで更新します。また、プロジェクトからメンバーを削除する場合は、Telelogic Synergy はワークエリアから該当ファイルを削除します。また、ネットワークファイルシステム内のファイルに直接手を加えて、つまり Telelogic Synergy の制御を離れて、手作業でワークエリアを更新することもできます。

同期

Telelogic Synergy の操作でプロジェクトデータを変更した場合は、ワークエリアと Telelogic Synergy データベースの両方がその変更で更新され、同期が取れた状態になっています。しかし、Telelogic Synergy の操作を使わずにファイルをワークエリアで直接変更

した場合は、ワークエリアの内容が Telelogic Synergy のデータベース内のプロジェクトの内容と異なってくることもあります。この場合には、Telelogic Synergy の同期（*synchronize*）操作を行うと、データベースとワークエリアを比較して選択的に更新することができます。Telelogic Synergy は、データベースとワークエリアの内容が異なる場合はその旨を通知します。そこでユーザーは差分を確認し、必要な更新を選択できます。また、ファイルのマージも選択できます。マージを選択した場合、マージツールが変更点を並置して表示するので、行ごとに差分をマージできます

Telelogic Synergy データベース外で作業する必要がある場合、データベースオブジェクトバージョンがチェックアウトされている、されていないにかかわらず、ワークエリア内のファイルを修正できます。データベースに再接続して、プロジェクトを同期させると、Telelogic Synergy はデータベースからオブジェクトの新規バージョンを自動的にチェックアウトし、ワークエリア内のオブジェクトに加えた変更を新規バージョンに追加します。

同期操作は、データの整合性に重要です。ワークエリアファイルの変更を Telelogic Synergy データベースに保存しないと、その変更の維持管理はワークエリアのファイルシステムの責任になり、そのシステムの信頼性に依存します。同期操作を行えば、ファイルは Telelogic Synergy データベースの一部となり、データベースがバックアップされる際に必ずバックアップされます。

オブジェクトの使用、作成、追加、削除、消去

本章の初めの方で、既存バージョンからオブジェクトの新規バージョンを作成するチェックアウト操作について説明しました。オブジェクトはファイル、ディレクトリ、ドキュメント、または他のデータの集合でも構わない、ということを思い出してください。プロジェクトの内容を修正する方法は、他にもいくつかあります。

- **別バージョンのオブジェクトの使用**：プロジェクト内の他のオブジェクトバージョンを選択したい場合、たとえば、以前のバージョンに戻したい場合、そのオブジェクトのバージョンを使用（*use version*）できます。使用（*use version*）操作は、特にデバッグの際に役立ちます。テストが失敗する場合にはオブジェクトを以前のバージョンに戻してトラブルシューティングを行いたいことがあります。他のユーザーがチェックアウトしたオブジェクト（つまり、*working* 状態のオブジェクト）以外であれば、どのバージョンのオブジェクトでも使用できます。

- **オブジェクトの作成**

  既存オブジェクトの新規バージョンをチェックアウトするのではなく、まったく新しいオブジェクトを作成したい場合は、オブジェクトを新たに作成（*create*）します。この操作は、PVCS **vcs -I** コマンド（Windows）や SCCS **create** コマンド（UNIX）の操作に似ています。オブジェクトバージョンは、明示的に作成しないと（つまり Telelogic Synergy 制御下で作成しないと）、チェックインできません。オブジェクトを作成すると、プロジェクト内のディレクトリに作成され、プロジェクトのワークエリアに表示されます。

  また、プロジェクトとディレクトリも作成できます。プロジェクトを作成すると、そのワークエリアは即時に作成されます。ディレクトリを作成すると、オブジェクトを作成するか、貼り付けるまでディレクトリは空のままです。

- **オブジェクトの追加**

  既存のファイル、ディレクトリ、プロジェクトを、プロジェクト内のディレクトリにコピーアンドペーストまたはドラッグアンドドロップによって追加できます。プロジェクトに追加するためにオブジェクトをチェックアウトする必要はありません。

- **オブジェクトの削除**

  オブジェクトを削除（*delete*）操作または 切り取り（*cut*）操作によって削除できます。オブジェクトを**削除**すると、データベースから永久に取り除かれます。データベース内の他のプロジェクトがそのオブジェクトを使用している場合は削除できません。オブジェクトを**切り取る**と、プロジェクトから削除されますが、データベースには残ります。後で必要になった場合にプロジェクトに戻すことができます。

更新、ベースライン、タスク、プロセスルール

更新（*update*）とは、プロジェクトまたはディレクトリ内のオブジェクトバージョンを更新するプロセスのことです。プロジェクトまたはディレクトリ内の各オブジェクトバージョンが評価され、適切なバージョンが Telelogic Synergy の使用可能な候補から選択されます。通常、開発者は新しいタスクの作業を開始する際には必ずプロジェクトを更新します。プロジェクトの最新メンバーを取り込むためです。

特定リリースと目的をもつ一連のプロジェクトのメンバーは、特定のベースラインに基づいています。ベースラインは、静的プロジェクトとタスクをグループ化したものです。ベースラインは 1 つ以上のプロジェクトとそこに含まれるタスク のある時点でのスナップショットと考えてください。したがって、特定のビルド、マイルストーン、またはリリースを表すこともあります。

プロジェクトがプロセスルールを使用している場合、どのベースラインを使用するかはプロセスルールが識別します。そのプロセスルールを参照するプロジェクトは、そのベースラインを使用して、更新時に使用するベースラインプロジェクト（*baseline project*）を見つけます。(ベースラインプロジェクトは、プロジェクトの開始点です。つまり、各プロジェクトはベースラインを確認してその開始点（つまり、ベースラインプロジェクト）を見つけます)。たとえば、カレントリリースの **Insulated Development** （個別開発）プロセスルールが **Integration Build 20020913** ベースラインを使用すべきであると指定していて、静的プロジェクト**toolkit-int_20020913** および **calculator-int_20020913**を含んでいる場合、開発者の **calculator-bob** プロジェクトは、そのベースラインプロジェクトとして **calculator-int_20020913** を選択します。

このように、プロセスルール（*process rules*）は、プロジェクトをどのように更新するかを定義した「パターン」です。プロセスルールは、プロジェクトを更新する際に使用するベースラインと一連のタスクを選択するための規則を指定しています。プロセスルールを使用することで、ソフトウェア開発 / テストプロセスをパターン化し、調整可能にできます。

更新操作では、ベースライン（*baseline*）内のタスクも使用します。この仕組みによって、タスクの評価作業の能率が上がり、更新操作のパフォーマンスが向上します。ベースラインを使用した更新の場合、そのリリース全体にわたってすべてのタスクを調べるのではなく、直近のベースライン以降追加されたタスクのみを調べるだけで済みます。

通常、ビルドマネージャがベースラインを作成してプロセスルールを設定し、開発者が特定のマイルストーンやリリースタイミングで利用できるようにします。

フォルダ

フォルダ（*folder*）は、ビルドマネージャが設定する、タスクを集めた名前付きのグループです。フォルダは、論理的に一まとまりにする（たとえば、タスクの状態、リリース、所有者、またはこれらのプロパティの任意の組み合せに基づいて）必要のあるすべてのタスクを集めるために使用します。フォルダの例としては、**All Tasks Assigned to Jane**（「ジェーンに割り当てられたすべてのタスク」）や **All Completed Tasks for Release editor/2.0**（「リリース **editor/2.0** 用のすべての完了済みタスク」）といったものがあります。

フォルダにタスクを追加する方法には、以下の 2 つがあります。

- 追加する各タスクを手動で選択します。この方法を使用すると、再度手作業でフォルダを修正するまで、フォルダにはその一連のタスクしか含まれません。
- データベースクエリを指定します。たとえば、リリース **editor/2.0** 用のタスクをすべて選択するクエリを設定できます。この場合、フォルダにアクセスすると Telelogic Synergy がデータベースのクエリを実行して選択基準に合致したタスクを集めます。クエリの利点は、新規タスクを作成するたびにフォルダを手作業で更新する必要がないことです。タスクがクエリの基準に一致した場合、Telelogic Synergy は自動的に新規タスクをフォルダに追加します。

もう 1 つのフォルダの便利な点は、複数のユーザーが一連のタスクを共有できることです。たとえば、ソフトウェア構成が統合テストに合格した後、テストに合格したフォルダ内のタスクを開発者が利用できるようになります。

ビルド、製品、Makefile

ビルド（*build*）とは 、ツールやコンパイラやコード生成ソフトを使用して、既存のソースファイルからファイルを生成するプロセスのことです。Telelogic Synergy マニュアルでは、ビルドと *make* という用語を同義として使用します

他のファイルを処理することによって構築されるファイルを製品（*products*）と呼びます。いかなる種類のオブジェクトも潜在的には製品です。最も一般的な製品は、実行形式ファイル、ライブラリ、再配置可能なオブジェクトファイル（**.obj**）です。

プロジェクトには、任意の数の *makefiles*（製品をビルドするための手順を含むファイル）を含めることができます。ビルドを実行するためにサードパーティ製品を使用できます。

製品は管理されているか、または管理されていないかのいずれかです。管理されていない製品がワークエリア内に存在する場合はありますが、Telelogic Synergy データベース内には存在しません。管理された製品とは、Telelogic Synergy 内でオブジェクトバージョンとして制御される製品ファイルです。管理された製品は Telelogic Synergy データベース内に存在するため、ユーザーはその製品を共有できます。

オブジェクトは手動でそのオブジェクトバージョンを製品とマーク付けすると製品になります。

注記：makefile のターゲットとして使用されないプロジェクト、シンボリックリンク、ディレクトリを除く、すべての種類の管理されたオブジェクトは、製品になり得ます。

4 Telelogic Synergy の方法論

本章では、情報を順番に紹介します。紹介順に読み進めることをお勧めします。

タスクベースの方法論

方法論とは、ソフトウェアを管理する上で使用するプロセスと方針のことです。Telelogic Synergy では、この方法論が、開発からテスト、リリース、保守までにわたる開発サイクルを通してソフトウェアの作業フローを制御します。

Telelogic Synergy は、作業の基本単位である「タスク」を使用して、ソフトウェアアプリケーションへの変更を追跡する「タスクベースの方法論」をサポートしています。タスクとは、ある意味を持つ一まとまりの変更のことです。タスクベースの構成管理の利点は以下のとおりです。

- タスクベースの構成管理は直感的に使用できる。

  一般に開発者は、ある機能のための変更を各ファイルにどのように割り当てるかを、あらかじめ計画します。大半の構成管理システムでは、開発者が、変更した各ファイルを忘れずにチェックインしなければなりません。タスクベースの構成管理方法論では、すべての関連する変更を自動的、継続的に追跡して、そのすべての変更を一回でチェックインできるような仕組みを用意することで、開発者にとって自然な考え方で作業ができるようにしてくれます。

- タスクベースの構成管理は、リリースを作成するにあたっての推測を要する作業を排除する。

  タスクベースの構成管理を使用すると、アプリケーションをベースラインと一連のタスクから構成できます。したがって、次のアプリケーションのリリースを作成するにあたり、最新のマイルストーンまたはリリースを起点にして特定の修正や機能拡張を追加するという合理的な方法を取ることができます。

- タスクベースの構成管理はファイル間の潜在的なコンフリクトを警告する。

  タスクベースの構成管理システムは、非タスクベースのシステムと比べると、ファイル間の関係を細かく把握しているので、テスト実施前にソフトウェア構成内のコンフリクト（矛盾）を検出できます。Telelogic Synergy は、構成を更新する際に失われたタスクや部分的に失われたタスクを検出します。

- タスクベースの構成管理は、リリースについて、より詳しい情報を提供する。

  リリースの内容を記述するにあたってリリースに含まれるタスクの説明を利用できるので、単なるソースファイルのリストよりも優れた情報を提供できます。

- タスクベースの構成管理は変更依頼を実際に修正を行うタスクと結び付ける。

変更依頼システムと実際のソフトウェア変更の間を緊密に結びつけることで、顧客が提示してきた不具合報告や改善依頼とタスクを関連付けられるようになります。

ユーザー

Telelogic Synergy ユーザーは、開発者またはビルドマネージャとして、それぞれの役割に適した操作を行うことができます。CM アドミニストレータは、この役割をデータベースレベルで各ユーザーについて設定します。この設定によって、そのデータベースで実行できる操作が決まります。また、作業対象のデータベースによって設定が異なる場合もあります。たとえば、ユーザーの Jane は、**main_product** データベースでは開発者およびビルドマネージャの操作を実行できても、**integrations** データベースでは開発者の操作しか実行できないかもしれません。CM アドミニストレータがユーザーを開発者とビルドマネージャの両方の操作ができるように設定しておけば、ユーザーは、設定を変更したりセッションを再起動せずに必要な操作を実行できます。

本章では開発の方法論について説明しているので、主に開発者およびビルドマネージャが実行できる操作を中心に説明します。

プロジェクトと作業フロー

プロジェクトは、ある特定のメンバーオブジェクトを含み、他から分離された作業環境を提供します（プロジェクトに関する基本情報については、19 ページの「プロジェクトおよびプロジェクト グルーピング（グループ化）」を参照してください）。同じプロジェクトの別バージョンを他の目的に使用できます。以下の例を見てください：

- 各開発者はある変更についての開発とテストをするための作業バージョンを所有する。
- プロジェクトの別バージョンを使用して、統合テスト用の最新の完了済みタスクを集めることができる。
- プロジェクトのあるバージョンを使用して、システムテスト用の特定の変更を含めたビルドができる。
- プロジェクトのさらに別のバージョンを使用して、特定の構成をリリースまたはマイルストーンとして保管できる。

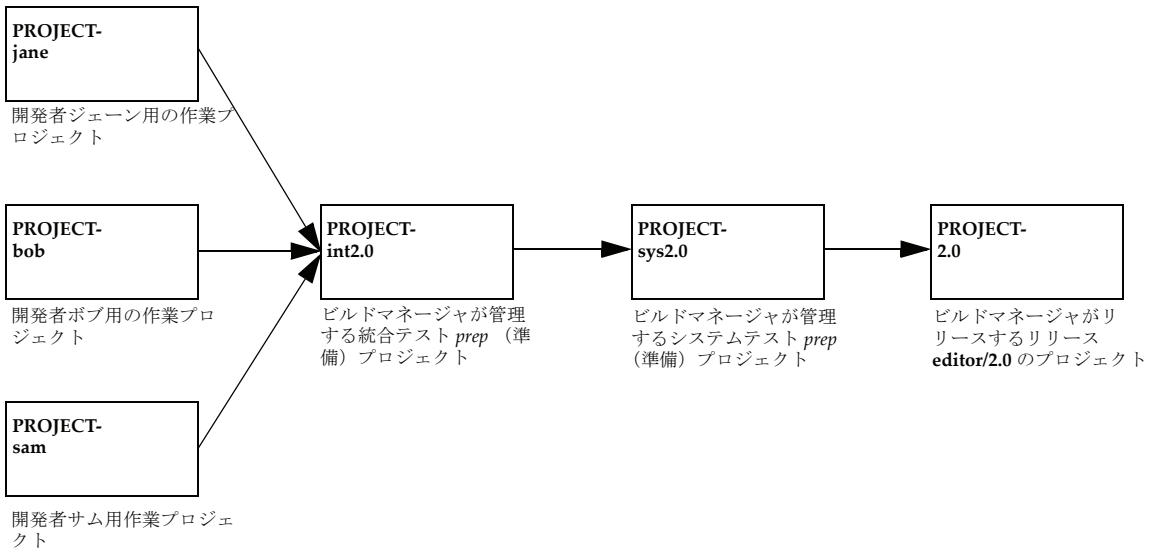
こういったさまざまな目的を持つプロジェクトを使うことで、1 つのアプリケーションに対して、開発チームが共同して作業にあたることができます。開発チームの作業フローは、プロジェクトとそのプロジェクトがどのように変更を選択するかによって決定されます。

Telelogic Synergy が提供するデフォルトの作業フローは、以下の段階から構成されます。

- 開発者は、自分の開発プロジェクト内で開発を行い、変更点をテストします。1 つのタスクを完了すると、そのタスクは統合テストプロジェクトに盛り込むことができるようになります。開発者が自分たちのプロジェクトを更新すると、自分専用のチェックアウトされたバージョンを保持できます。さらに、そのバージョンは統合テストに合格した最新のバージョンです。

- 統合テストプロジェクトは、現在までに完了したすべてのタスクを集めます。このプロジェクトは、ソフトウェア開発における「最良の実践（best practice）」である「毎日のビルドとスモークテスト」を実現するために利用されます。統合テストの最終目的は作り込まれてしまった問題をできる限り早期に発見することです。ビルドマネージャが統合テストプロジェクトを管理します。
- ビルドマネージャは、ビルドが統合テストに合格したらベースラインを作成します。
- システムテストプロジェクトは、より詳細なテスト向けに特定の変更を盛り込むために利用されます。ビルドマネージャは、変更のリストを定義、更新して、開発者がまだ作業をしている変更からシステムテストプロジェクトが確実に隔離されるように配慮します。各修正は、プロジェクトがチームの品質基準を満たすまで、追加、ビルド、再テストされます。システムテストプロジェクトがよく利用されるのは、リリースやマイルストーンの準備のためです。
- ソフトウェアがリリースされた後、またはマイルストーンに到達した後、ビルドマネージャはプロジェクトをベースライン化するかリリースして、その構成を保管できます。リリースされたプロジェクトは、新規リリース用のベースラインとして使用できます。

下図は、Telelogic Synergy のデフォルトの作業フローを実現するために、プロジェクトがどのように利用されるかを示しています。矢印はプロジェクトを通したタスクの流れを示します。



Telelogic Synergy は、購入後すぐにこの方法論を使用できるように設定されています。また、Telelogic Synergy のプロセスモデルには柔軟性があり、自分の開発チームのプロセスに適合するようにデフォルトの方法論をカスタマイズできます。

リリース

Telelogic Synergy では、開発者は必ず特定のリリース（*release*）に携わることになります。リリースは、ソフトウェアアプリケーションのバージョンを現す「ラベル」です。たとえば、ソフトウェアの最初のリリースが editor/1.0、2 番目のリリースが editor/2.0、または editor/1.1 となります。

プロジェクトをチェックアウトする場合、使用するリリースを指定します。同じように、タスクを作成する場合にも、そのタスクが含まれる先のリリースを指定します。リリースは非常に重要です。なぜならば Telelogic Synergy は、リリースを使用してタスクやプロジェクトを体系付け、タスクが自分のリリース値の合ったプロジェクトで使用されるようにするからです。

Telelogic Synergy は、ソフトウェアアプリケーションのリリースを保管します。ある 1 つのリリースが、特定のリリース向けのプロジェクト、タスク、フォルダにマーク付けできます。また、リリースは、各リリースでどのオブジェクトバージョンが開発されたかを追跡するのにも役立ちます。

リリースの作成と変更を実行できるのは、ビルドマネージャだけです。ビルドマネージャは、**リリース**エクスプローラまたは **ccm release** コマンドを使ってリリースを確認できます。Telelogic Synergy データベースごとに一連の固有のリリースがあります。分散型変更管理を使用すると、データベース間でのリリースの転送もできます。

典型的なリリースとして以下のものがあります。下表の例はビルドマネージャが作成したリリースです。このリリースは、コンポーネント名とコンポーネントリリースから構成されています。

リリース	コンポーネント名	コンポーネントリリース
1.0		1.0
2.0		2.0
2.0_patch		2.0_patch
Telelogic Synergy/6.5	Telelogic Synergy	6.5
editor/2.0	editor	2.0
editor/2.1	editor	2.1

リリースは、コンポーネント名（オプション）、リリース区切り文字、コンポーネントリリースで構成されます。コンポーネント名は、**Telelogic Synergy** または **editor** のように

アプリケーションやコンポーネントの名前を示します。コンポーネントリリースは、そのアプリケーションやコンポーネントの特定のリリースを示します。

注意してほしいのは、コンポーネント名はリリースの必須な部分ではありません。上表の１行目では、**1.0 リリースには**コンポーネント名がないので、Telelogic Synergy は空白のままにします。

オブジェクトをチェックアウトする際には、Telelogic Synergy がカレントタスクから新規オブジェクトにリリースをコピーします。

プロジェクト目的

プロジェクト目的（*purpose*）は、プロジェクトの用途を指定して、更新処理のための一連のルールと結び付ける設定です。Telelogic Synergy が提供する定義済みのプロジェクト目的は、以下のとおりです

目的	状態
Insulated Development（個別開発）	*working*（作業中）
Collaborative Development（共同開発）	*working*（作業中）
Custom（カスタム）	*working*（作業中）
Integration Testing （統合テスト）	*prep*（準備）
System Testing（システムテスト）	*prep*（準備）
Shared （共有）	*shared*（共有）
Visible（可視）	*visible*（可視）

Insulated Development（個別開発）、Integration Testing（統合テスト）、System Testing（システムテスト）の各目的は、前節でも説明したように、デフォルトの方法論で使用されます。Shared（共有）、Visible（可視）、Collaborative Development（共同開発）の各目的は、より密な共同作業のために標準的な方法論のバリエーションを必要とする開発チーム向けです。Custom（カスタム）目的は、開発者がカスタムプロジェクトのベースラインとタスクを指定できるようにします。

開発者が通常使用する目的は、Insulated Development（個別開発）、Collaborative Development（共同開発）、および Custom（カスタム）です。Insulated Development（個別開発）目的は、開発者が新しいプロジェクトを作成するときに使われるデフォルトの目的です。開発者が自分のプロジェクトを更新すると、現行リリースの最新 Integration Testing（統合テスト）ベースラインに加えて、自分が完了した現行リリースのすべてのタスクを取得できます。

Collaborative Development（共同開発）目的は、Insulated Development（個別開発）の代わりとして使用できる目的です。この目的の典型的な使用例は、小規模の開発チームで、メンバーの行った変更がビルドを破壊する可能性が低い場合です。開発者がCollaborative Development（共同開発）目的を使って自分のプロジェクトを更新すると、現行リリースの最新 Integration Testing（統合テスト）ベースラインに加えて、自分が完了した現行リリースのすべてのタスク、および他の開発者が完了した現行リリースのタスクをすべて取得できます。

ビルドマネージャが通常使用する目的は、Integration Testing（統合テスト）と System Testing（システムテスト）です。Shared（共有）と Visible（可視）目的は、デフォルトであるタスクベースの方法論ではない、他の方法論を採用したい開発チーム向けです。

プロジェクトの作成、またはプロジェクトのコピーを実行するときに、その目的を指定します。すると、Telelogic Synergy が自動的にプロジェクトの更新プロパティを設定し、その目的向けに新規プロジェクトを準備します。

また、［状態］カラムにはプロジェクトの状態が表示されます。この状態は、プロジェクトが指定された目的で作成されたときのデフォルトの状態です。たとえば、プロジェクトを作成した結果 *visible*（可視）状態になるのは、目的が Visible（可視）の場合のみです。作成されたプロジェクトが *working*（作業中）状態になるのは、作成時に Insulated Development（個別開発）、Collaborative Development（共同開発）、または Custom（カスタム）目的を選んでいた場合です。また、状態は、更新時にプロジェクトが正しいメンバーを選択できるようにします。

更新プロパティ

プロジェクトを更新する際に、プロジェクトは「更新プロパティ」（*update properties*）と呼ばれる一連のプロパティを使って、どのオブジェクトバージョンを選んでワークスペースに持ってくるかを自動的に決定します。更新プロパティは、プロジェクトグルーピングごとに保管されており、1 つのベースラインと、タスクおよび／またはフォルダのリストで構成されています。

更新プロパティはすべてのプロジェクトグループにあります。プロジェクトをチェックアウトするときに、選択されている目的（Insulated Development（個別開発）、Integration Testing（統合テスト）など）とプロジェクトのリリース値から、プロジェクトの更新プロパティがどのように設定されるかが決まります。または、手動で更新プロパティを設定することもできます。

更新の操作を行うと、Telelogic Synergy は以下のようにプロジェクトを更新します。

1. Telelogic Synergy は、どのタスクを取り込むかを決定します。更新プロパティにリストされている各フォルダを評価し、そこにリストされているタスクとともに、更新プロパティ内で直接指定したすべてのタスクを追加します。
2. Telelogic Synergy は、リスト内の各タスクでまだベースラインに含まれていないものを見て、オブジェクトバージョンのリストを算出します。このオブジェクトのリス

トとベースラインオブジェクトからのメンバーが、更新の候補リストになります。このリストのオブジェクトバージョンのみが候補と見なされます。

3. Telelogic Synergy は、一連の規則を使って各候補を評価します。この規則は、候補のプロパティとプロジェクトのプロパティを比較して、最も合致するものを選び出すためのものです。

Telelogic Synergy のデフォルト作業フロー

本節では、Telelogic Synergy のデフォルト作業フローについて説明します。

タスクの利用

タスクとは、ソフトウェアアプリケーションの問題や機能拡張のための作業を指します。ある 1 つのタスクは、特定の問題の解決や機能拡張のための修正対象である複数のオブジェクトをすべて取りまとめるので、タスクを完了すれば、それらのオブジェクトを個別にチェックインする必要はありません。つまり、タスクはユーザーのための多くの作業を肩代わりします。

デフォルトで、どの Telelogic Synergy ユーザー（開発者、ビルドマネージャなど）も、タスクを作成できます。また、顧客や技術サポートエンジニアから提出された変更要求に基づいて、タスクの生成やタスクの割り当てを行うこともできます。

ある作業者がタスクを作成したとします。彼がこの問題を解決できる人間を知っていれば、そのタスクは即座にその人に割り当てられます。ビルドマネージャは、タスクを自分自身や他のユーザーに割り当てることができます。つまり、タスクを作成したユーザーはそのタスクを自分自身に割り当てられます。タスクを割り当てるには、リリース値を設定して、そのタスクを盛り込むソフトウェアアプリケーションのバージョンを明らかにする必要があります。

タスクが割り当てられたら、開発者は以下の手順で作業をします。

1. カレントタスクとするタスクを選択する。

   割り当てられたタスクの中から、任意のタスクをカレントタスクとして選択できます。

2. タスクを完了するために必要な変更修正をすべて行う。

   Telelogic Synergy は、変更修正されたすべてのオブジェクトバージョンを自動的にカレントタスクに関連付けるので、ある特定の操作（オブジェクトのチェックアウトやオブジェクトの追加）の対象となったオブジェクトは、すべてカレントタスクと関連付けられます。

   ここで、ユニットテストを実行してください。このテストによって、さらに修正が必要かどうかが判明します。

3. カレントタスクを完了する。

   タスクの完了操作を行うと、Telelogic Synergy は、最初にそのタスクに関連付けられているオブジェクトをチェックインし、その後、タスクを完了します。完了されたタスクは、ビルドマネージャが統合テストに使用し、また、さらに以降の統合テストとシステムテストのために使用します。完了されたタスクが統合テストに合格した後、ビルドマネージャはそのタスクを他の開発者が使用できるようにします。

開発プロセス

プロジェクトに携わっている各開発者は、作業バージョン（*working version*）のプロジェクトを持っています。開発者のプロジェクトは、ビルドマネージャの統合テストプロジェクトからコピーしたものです。

通常、開発者は 開発プロジェクトをチェックインしません。開発者がプロジェクトで変更修正作業を行ってその変更をテストする場合、Telelogic Synergy がタスク完了時に変更された各々のオブジェクトバージョンを自動的にチェックインします。したがって、開発者はプロジェクト自体をチェックインしません。開発プロジェクトは一種の「コンテナ」であり、リリースからリリースへと使い続けられます。プロジェクトの内容は、開発者が更新操作を行うたびに変化します。

開発者は、新規タスクに取り組み始める時や、自分のプロジェクトを最新の変更で更新する準備ができた時に、更新操作をします。必要に応じて更新を行うことで、プロジェクトを最新の状態に保ち、プロジェクトのメンバーオブジェクトに対するあらゆる変更を 1 つのバージョンの開発プロジェクトで行います。

各開発者は、自分の開発プロジェクトを使って、自分が行った変更修正に対するユニットテストをする責任があります。各開発者は、自分のタスクをチェックインする前に、まずプロジェクトを更新してから自分が行った変更修正について再テストを行い、自分の変更が他の開発者の行った最新の変更と互換しているかどうかを確認する必要があります。ユニットテストの完了後、開発者はタスクを完了、つまり、相互に関連しているすべてのオブジェクトを一度にチェックインします。この方法によれば、統合テストのために必要なすべてのオブジェクトを、確実に、ビルドマネージャに渡すことができます。

デフォルトで、開発者が自分の開発プロジェクトを更新した場合、Telelogic Synergy は、その開発者に割り当てられて完了した、現在のリリース向けのすべてのタスクと、そのリリース向けのタスクで統合テストに合格した最新のものをすべて集めます。

統合テストサイクル

統合テストサイクルの最終目的は、開発サイクル内のできる限り早期に問題を見つけることです。ビルド管理プロジェクト がビルドマネージャがテスト用やリリース向けのソフトウェアを用意するためのプロジェクトであることを思い出してください。統合テスト準備プロジェクト（*integration testing projects*）では、最新のチェックイン済みの変更を集めて統合テスト用にビルドします。統合テストプロジェクトが数多くの新しい変更を含んでいる上に、ユーザーが絶えず新しいオブジェクトをチェックインしてくるため、統合テストエリアは、通常は不安定な状態にあります。このような「不安定さ」は、あらかじめ予想されたものといえます。なぜならば、このエリアで問題を発見することが目的だからです。

デフォルトで、ビルドマネージャがビルド管理プロジェクトを更新して統合テスト用のビルドを実行した場合、Telelogic Synergy は、現在のリリース向けのすべての完了済みタスクを集めます。ビルドマネージャは、開発者に割り当てられただけのタスクを取り込むことを望みません。それは、開発者が修正作業の最中であり、そういったオブジェクトはまだ統合準備が整っていないからです。

統合テストサイクルは、多くの場合、反復プロセスです。つまり、開発チームは、ソフトウェアが求められる品質基準に達するまで、ビルド、テスト、修正、タスク追加を何度も繰り返します。通常、ビルドマネージャが一連の統合テストプロジェクトを更新すると、最新の完了タスクを取り込むために、タスクのリストは自動的にリフレッシュされます。壊れたビルドを修正する必要がある場合は、ビルドマネージャは完了したタスクの統合テストプロジェクトへの自動取り込みを中止し、ビルド修正用のタスクのみが統合テストプロジェクトグループに含まれるようにします。

リリース **editor/2.0** の統合テストプロジェクトを更新するビルドマネージャについて考えて見ましょう。デフォルトの動作は以下のとおりです。

- 各プロジェクトのベースラインプロジェクトとして、最新の Integration Testing（統合テスト）ベースライン内の適切なプロジェクトが設定されます。
- **All Completed Tasks for Release editor/2.0**（「リリース **editor/2.0** 用全完了タスク」）というフォルダが、それぞれのプロジェクトに含まれます。このフォルダはクエリを使用して、*completed*（完了）状態のリリース **editor/2.0** 用のタスクをすべて集めます。

ビルドマネージャは自分の統合テストプロジェクトを定期的に更新し、統合テスト用にソフトウェアをビルドします。ビルドマネージャがプロジェクトを更新すると、**All Completed Tasks for Release editor/2.0**（「リリース **editor/2.0** 用全完了タスク」）フォルダはそのクエリを使用して、リリース **editor/2.0** に携わっている開発者が完了させたすべてのタスクをデータベースから集め、それらのタスクに関連付けられたオブジェクトバージョンでプロジェクトが更新されます。その後、ビルドマネージャはソフトウェアアプリケーションをビルドできます。

ビルドが成功しない場合、ビルドマネージャは次の 2 つのアクションを取ることができます。タスクを作成し、そのタスクをビルドを失敗させたオブジェクトを担当している開発者に割り当てる、または、ビルドを失敗させたオブジェクトを担当している開発者に修正が必要なことを伝えて、その後、その開発者が自分でタスクを作成します。

製品（*product*）が正しくビルドされた場合は、ビルドマネージャは新しいベースラインを作成します。開発者が自分のプロジェクトを更新すると、この新しいベースラインによって最新のテストを通過した変更が確実に取り込まれます。

通常、ビルドマネージャは統合テストプロジェクトをチェックインしません。統合テストプロジェクトは一種の「コンテナ」であり、リリースからリリースへと使い続けられます。プロジェクトの内容は、ビルドマネージャが更新操作を行って開発者の最新完了タスクを取り込むたびに変化します。

システムテストサイクル

安定したビルドや特定のマイルストーンの提供に向けて開発チームの作業が十分に進捗したら、ビルドマネージャは、品質保証（QA）チームがシステムテスト用に使用するソフトウェアアプリケーションをビルドします。システムテストプロジェクトの最終目的は、製品リリースのようなマイルストーン用のソフトウェアアプリケーションを準備することです。*system testing project*（「システムテスト準備プロジェクト」）は、システムテストを行う段階にまで達しているすべての開発者の作業結果を意味します。つまり、このプロジェクトには、システムテストやリリース準備で使用できるファイルバージョン、ディレクトリ、製品が含まれます。

開発者は、自分の開発を行ってその変更をテストし終わるたびにタスクを完了してゆくため、統合テストプロジェクトは、開発者による最新の変更を絶えず拾い上げていきます。ビルドマネージャは、開発者が新たにチェックインしてくる変更から切り離されたエリアとして、安定度の高いシステムテストプロジェクトを用意する必要があります。

ビルドマネージャがビルド管理プロジェクトを更新してシステムテスト用のビルドを行う時には、テスト対象のタスクのリストを正確に指定します。ソフトウェアに盛り込まれるタスクのリストを正確に管理することによって、開発チームは品質基準に達するまで、ソフトウェアの修正、ビルド、再テストを実行できます。システムテストサイクルは多くの場合反復プロセスです。つまり、開発チームは、ソフトウェアが求められる品質基準に達するまで、ビルド、テスト、修正、タスク追加を何度も繰り返します。

リリース **editor/2.0** のシステムテストプロジェクトを更新するビルドマネージャについて考えて見ましょう。ビルド管理プロジェクトは以下のように設定されます。

- 各プロジェクトのベースラインプロジェクトとして、最初は最新の Integration Testing（統合テスト）ベースライン内の適切なプロジェクトが設定されます。
- 以降の反復では、ビルドマネージャは、最新の Integration Testing（統合テスト）ベースラインがシステムテストプロジェクトに自動で取り込まれないようにします。ビルドマネージャは、システムテスト用のビルドが満足できる品質水準に達するように、必要に応じてタスクを追加します（ビルドマネージャは正確に決められた一連のタスクをテストし、自分が指定したタスクのみを必要とします）。

ビルドマネージャは、新しいシステムテストサイクルの準備として、システムテストプロジェクトグルーピング（*System Testing project grouping*）が最新の Integration Testing（統合テスト）ベースラインを取得するように設定します。システムテストはその後、以下のステップを繰り返します。

1. ビルドマネージャはシステムテストプロジェクトを更新し、ビルドする。
2. QA チームは、その結果のソフトウェアのテストを行う。
3. 開発チームは発見された問題をすべて見直し、このサイクルでどの問題を修正するべきかを決める。

4. チームはステップ 3 で修正することが承認された問題解決用のタスクを作成し、開発者に割り当てる。
5. ビルドマネージャは、System Testing（システムテスト）プロジェクトグループが最新の Integration Testing（統合テスト）ベースラインを取り込まないようにする。
6. 開発者が承認されたタスクを完了したら、ビルドマネージャはシステムテストプロジェクトグループにそれを追加する。プロセスはステップ 1 からもう一度始まります。
7. 他に修正すべき問題がなくなるまで、ステップ 1 ～ステップ 6 を繰り返す。
8. ベースラインを作成する。
9. ベースラインをリリースする。

システムテストベースラインのリリース

ソフトウェアが出荷されたら、ビルドマネージャは、今後のリリース用の起点とするために、ソフトウェアのビルドに使用したシステムテストベースラインを、直ちに *released*（リリース済み）状態に移す必要があります。リリース済み状態のオブジェクトは、すべて修正不可となります。したがって、ソフトウェアは保存状態となり、必要に応じて再作成可能であることが保証されます。

次期リリースの準備

ビルドマネージャがプロジェクトをリリースしたら、次期リリースのベースラインとして利用できます。以下のステップは、次期リリースの準備に必要です。

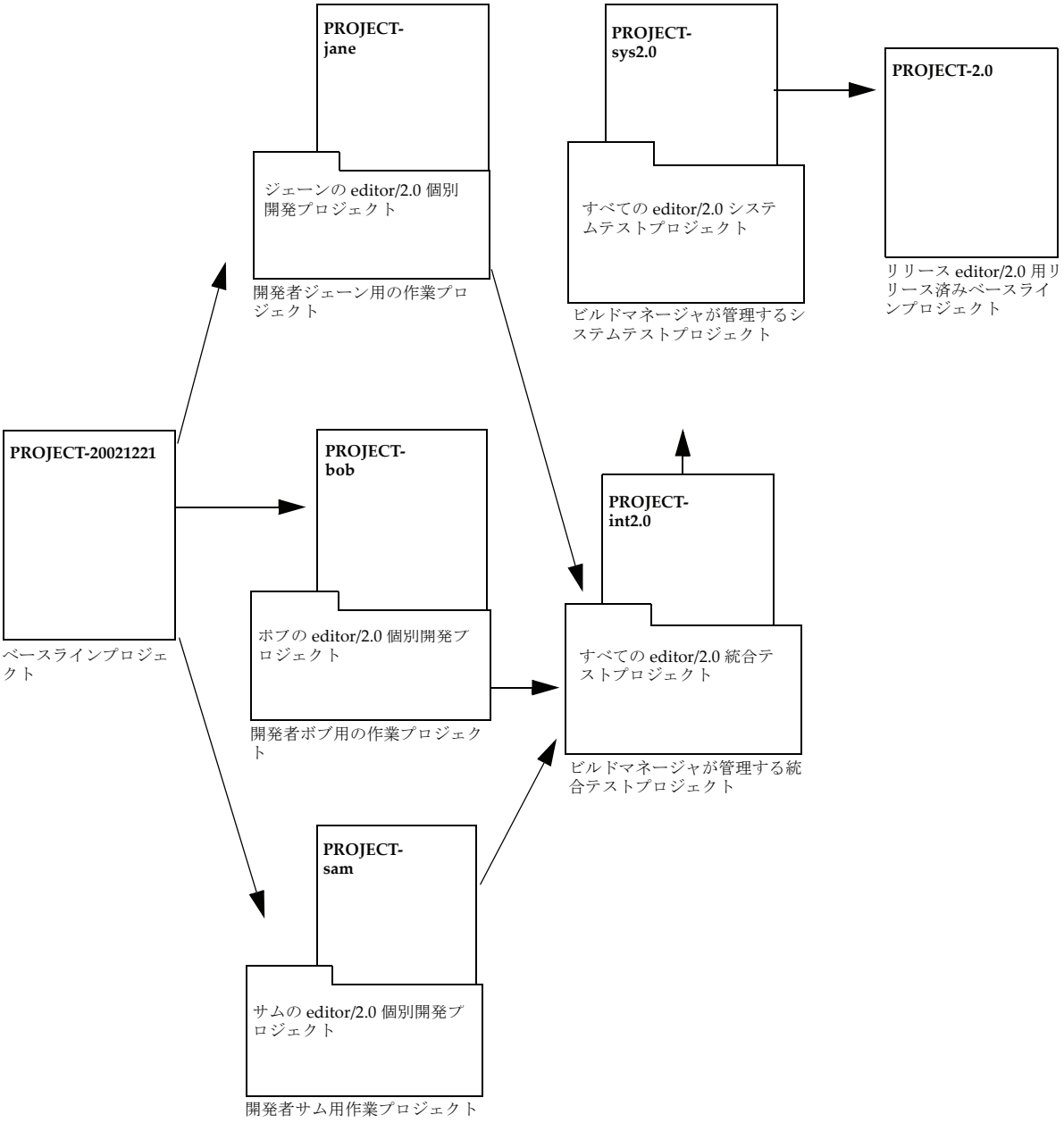
- ビルドマネージャは、リリースのリストに新規リリースを追加し、新規リリース用にプロセスルールを設定します。
- 開発者は、自分の開発プロジェクトを新規リリースを使用して更新します。つまり、プロジェクトは新規リリース用に再利用できます。
- ビルドマネージャは、新規リリースを使用するため統合テストプロジェクトを変更します。統合テストプロジェクトはチェックインされていないので、ビルドマネージャは、更新するだけで新規リリースの統合テスト用にプロジェクトを再利用できます。
- ビルドマネージャは、システムテストプロジェクトを変更して新規リリースを使用するようにします。

  開発チームが、現行リリースの作業を終了する前に次のリリースの作業を開始した場合、ビルドマネージャは、プロジェクトのコピーを行って新しい統合テストプロジェクトと新しいシステムテストプロジェクトを作成する必要があります（ビルドマネージャは、新しいプロジェクトを作成するために古いプロジェクトをコピーします）。

まとめ

ここまで、開発サイクル全体を見通してきました。開発者は自分の開発プロジェクトでタスクを完了し、ビルドマネージャはそのタスクを集めてテストに回し、そして、最終的なリリースが行われて、次期リリース用のベースラインが作成されます。下図に、作業フローを実装するためにプロジェクトをどのように使うかについて、まとめておきます。矢印は、作業が、どのようにしてさまざまなプロジェクトを通して流れてゆくかを示します。この図のキーポイントは以下の通りです。

- ベースラインプロジェクトとして示されたプロジェクトは、一般に開発者が開発プロジェクトにコピーするプロジェクトです。または、開発者は Integration Testing（統合テスト）または System Testing（システムテスト）プロジェクトをコピーすることもできます。通常は、開発者はベースラインのメンバーであるプロジェクトをコピーします。最後に示したプロジェクト **PROJECT-2.0** は、新しくリリースされたベースラインの一部です。この例では、説明を単純にするために 1 つのプロジェクトのみを示していますが、一般にベースラインには多数のプロジェクトが含まれます。
- 各開発者のプロジェクトには最新のベースラインが含まれています。完了タスクが統合テストに合格したら、ビルドマネージャは新しいベースラインを作成します。また開発者にはそれぞれ、個人のフォルダ（たとえば、**Jane's Assigned or Completed Tasks for Release editor/2.0**、つまり「**Jane** のリリース 2.0 用割り当て済みもしくは完了タスク」）があり、指定リリース用の自分のタスクを集めます。各開発者は自分のプロジェクトグルーピングにタスクを追加、削除できます。
- ビルドマネージャは、テスト用に統合テストプロジェクトとシステムテストプロジェクトを使用します。統合テストプロジェクトは、あるクエリを使用してタスクを集めた **All Completed Tasks for Release editor/2.0**（リリース editor/2.0 用のすべての完了タスク）という名前のフォルダを使用します。ビルドマネージャは、自分のプロジェクトグルーピングに、承認された変更を追加したり、承認された変更を削除したりします。
- ビルドマネージャは、適切な水準のテストに合格した各統合テストビルドとシステムテストビルドからベースラインを作成します。リリース作業の最後にビルドマネージャは、最終システムテストベースラインをリリースします。


PROJECT-
jane
ジェーンの editor/2.0 個別
開発プロジェクト
開発者ジェーン用の作業プロ
ジェクト
PROJECT-
sys2.0
すべての editor/2.0 システ
ムテストプロジェクト
ビルドマネージャが管理するシ
ステムテストプロジェクト
PROJECT-2.0
リリース editor/2.0 用リ
リース済みベースライ
ンプロジェクト
PROJECT-20021221
ベースラインプロジェ
クト
PROJECT-
bob
ボブの editor/2.0 個別開発プ
ロジェクト
開発者ボブ用の作業プロジェク
ト
PROJECT-
int2.0
すべての editor/2.0 統合テ
ストプロジェクト
ビルドマネージャが管理する統
合テストプロジェクト
PROJECT-
sam
サムの editor/2.0 個別開発プ
ロジェクト
開発者サム用作業プロジェクト

並行開発

並行開発とはオブジェクトの複数バージョンの同時並行的な開発です。Telelogic Synergy では、デフォルトで任意のオブジェクトタイプ（例、**csrc**、**library** など）について並行開発ができるようになっています。パラレルオブジェクトバージョンは必ずお互いを区別するある特性（例、プロパティ）を持っている必要があり、その特性によって、Telelogic Synergy がプロジェクト内で正しいオブジェクトバージョンを選択できるということを理解してください。更新の際にオブジェクトバージョンの評価方法を決定するのは、こうした特性です。パラレルオブジェクトバージョンをそれぞれ区別するプロパティを並行開発プロパティといいます。

Telelogic Synergy がサポートする並行開発は以下の通りです。

- **並行同時開発**は、複数の開発者が同じオブジェクトから自分たちの *working*（作業中）バージョンをチェックアウトする時に発生します。多くの場合各開発者は、コードの別の箇所について修正作業をします。自分の作業を完了した段階で 2 つのバージョンのコードをマージする必要があります。
- **並行バリアント開発**（または、並行プラットフォーム開発）は、複数の開発者が異なるハードウェアプラットフォーム（一般に **「バリアント」**と称する）用に同じオブジェクトの別バージョンに取り組む時に発生じます。異なるバージョンのオブジェクト（たとえば、Windows 用のバージョンと UNIX 用の別バージョン）は一般にマージしません。
- **並行リリース開発**は、ソフトウェア製品の複数リリースを同時に開発する必要があるときに発生します。たとえば、それぞれ異なる開発者が担当する、次期リリース作業、現行リリースへのパッチ作業、保守リリース作業（すべて現行リリースをベースラインとする）が同時に発生するような場合です。典型的な並行リリースではいずれかのリリース作業が終了した後でマージが行われます。たとえば、現行リリースへのパッチリリースの終了後、このパッチリリースは保守リリースにマージされ、保守リリースの終了後は、この保守リリースは次期リリースにマージされます。

以下の項では、Telelogic Synergy が上記のようなタイプの並行開発を管理する方法について説明します。

並行同時開発

Telelogic Synergy は、状態（*status*）と所有者（*owner*）プロパティの値を使用して並行同時バージョンを管理します。オブジェクトバージョンが *working*（作業中）状態にある場合は、所有者のみがそのオブジェクトバージョンを自分のプロジェクトに含めることができます。更新プロセスは、*working*（作業中）オブジェクトバージョンの所有者を更新されるプロジェクトの所有者と比較し、確実に正しいバージョンを選択します。2 つのパラレル *working*（作業中）バージョン（およびその関連タスク）がチェックインされたら、すべての変更を含む単一のバージョンがないため、すぐにマージする必要があります。チェックインはしたがバージョンをマージしていない場合、両方のタスクを含んだプロジェクト更新は最新の作成日時を持つバージョンを選択し、プロジェクトはパラレルコンフリクトを表示します。

並行プラットフォーム開発

プラットフォーム（*platform*）プロパティは、特定のプラットフォーム向けに設計されたプロジェクトやオブジェクトを特定します。複数プラットフォーム向けにソフトウェアをビルドする場合、各プラットフォーム用にプラットフォーム固有のプロジェクトが必要であり、各プロジェクトバージョンにプラットフォーム（*platform*）プロパティを設定する必要があります。たとえば、UNIX と Windows 用に **snap** プロジェクトを作成する場合は、2 つのバージョン、**つまり**プラットフォームプロパティを **unix** に設定したものと、プラットフォームプロパティを **win** に設定したものが必要になります。

プロジェクトをベースライン、フォルダ、タスクを使用して更新する場合、これらの選択は、更新プロジェクトプロパティに指定されている候補に限られます。その候補に異なるプラットフォーム用のパラレルバージョンが含まれている場合、更新操作でプロジェクトのプラットフォームプロパティと一致するオブジェクトバージョンを選択できるように、この候補のプラットフォームプロパティを正しく設定する必要があります。

たとえば、**line.c** を含んだ **snap** プロジェクトについて考えましょう。このバージョンのプロジェクトのプラットフォーム値**は win32** であり、32 ビット Windows プラットフォーム用に作成されたことを示しています。**line.c** オブジェクトには、プラットフォーム値が **win32** と **unix** の 2 つのバージョンがあります。プロジェクトのフォルダやタスクに含まれていたことからこの両方が候補となっている場合、選択規則にしたがって、プラットフォーム値が一致する方（**win32**）が選択されます。

並行リリース開発

リリース（*release*）プロパティは特定のリリース向けのプロジェクトやタスクを識別するために使われることを思い出してください。複数リリース向けにソフトウェアを開発している場合、各リリース用としてプロジェクトのバージョンが 1 つ必要になり、それぞれのプロジェクトバージョンのリリースプロパティを該当リリースの値に設定する必要があります。こうすることで、各プロジェクトは更新の際に正しいサブプロジェクトとタスクを選択します。

自分のタスクを正しいリリースプロパティ値でマークして、正しいリリースのプロジェクトによって選択されるようにしてください。

> **注記**：複数のリリースに適用する変更は、リリースごとに個別のタスクで行う必要があります。

コンポーネントベースの開発

コンポーネントとは、1 つ以上のライブラリまたは実行可能ファイル、およびそれらの使い方を記述したサポートファイルです。サポートファイルには、ヘッダーファイル、ヘルプ、互換性と依存性に関する情報、ハードウェアまたはソフトウェア要件、設計情報、テストケースなどがあります。コンポーネントにソースコードを含めることもできます。

Telelogic Synergy では、コンポーネントは、個別のファイルまたはプロジェクトで表現できます。

Telelogic Synergy ではファイルバージョンが再利用可能なので、あるプロジェクトでファイルを作成またはビルドして、他のプロジェクトで使用できます。たとえば、**ccmscci.dll** ライブラリファイルは、そのソースコードがある **ccmscci** プロジェクトでビルドされますが、その同じファイルを **visual_studio_integration** プロジェクトと **va_java_integration** プロジェクトのメンバーにすることも可能です（各プロジェクトは、必要に応じて、そのファイルの異なるバージョンを持ちます）。

更に、ある 1 つのコンポーネントとして一まとまりで公開される複数のファイルを含んだ新しいプロジェクトを作成することもできます。たとえば、ファイル **ccmserver.jar**、**ccmserver.properties**、および **ccmserver.html** を含む、**ccmserver_ext**（**ccmserver** プロジェクトに似ているが外部用）というプロジェクトを作成できます。**ccmserver_ext** プロジェクトには多数のバージョンがあり、それぞれが互換ファイル付コンポーネントの公開バージョンを持っています。

コンポーネントのユーザーがコードを表示したり変更できる必要がある場合には、ソースプロジェクト全体を 1 つのコンポーネントと考えることもできます。

コンポーネントは他のコンポーネントからも作成可能なので、全プロジェクト階層で 1 つのコンポーネントを表現するということもあり得ます。

コンポーネントの管理

Telelogic Synergy では、リリース（*releas*）は、Telelogic Synergy/6.6a や Telelogic Change/5.1 のようなソフトウェアアプリケーションのリリース用ラベルを指します。リリースはバージョンに似ていますが、リリースは 1 つのソフトウェア製品全体にあてはまるものです。リリースは、出荷済みの製品を表現したり、現在開発中の製品を示すことができます。プロジェクトとタスクは、ある特定のリリース向けとしてマーク付けされます。

各コンポーネントは独自のリリースストリームを持つ必要があります。たとえば、1 つの GUI ライブラリと、2 つのアプリケーション calculator および editor を開発するチームを考えてみます。以下の例に示すようなリリースを設定します。

コンポーネント	リリースストリーム
GUI ライブラリ	gui_lib/1.1、gui_lib/1.2、gui_lib/1.3
editor アプリケーション	editor/1.0、editor/2.0、editor/2.1、editor/3.0、editor/4.0
calculator アプリケーション	calc/1.0、calc/2.0

各コンポーネントについて個別のリリースストリームを設定する目的は、異なるコンポーネントが互いに依存しないように分離するためです。したがって、コンポーネントのリリーススケジュールを個別に設定できるため、開発チームは異なるプロセスを使って作業できるようになります。

コンポーネントの公開

コンポーネントは、コンポーネントを現しているファイルやプロジェクトをリリース（または、変更不能な状態にチェックイン）することによって公開できます。コンポーネントを公開するといったとき、コンポーネント開発者はソフトウェアを開発して他の開発者のためにコンポーネントを公開し、一方ビルドマネージャは統合テストのためにソフトウェアを集めて、ビルドし、さらにシステムテストのためにソフトウェアを集めて、ビルドします。

コンポーネントが厳密なテスト手順を使用する組織化されたチームによって開発された場合、通常はビルドマネージャが公開します。逆に、コンポーネントが組織化されていない小チームや個人開発者によって開発された場合は、コンポーネント開発者が公開します。

コンポーネントが公開された時点で、公開した作業者（ビルドマネージャまたはコンポーネント開発者）は、他の作業者が参照できるように、コンポーネントをタスクに関連付けることも考えられます。コンポーネント公開時に開発チームが自動通知されることを希望する場合は、トリガを定義します。

コンポーネントの参照

コンポーネントはタスクに関連付けることができます。タスクを使うと、コンポーネントの消費者は自分で使用したいコンポーネントバージョンを指定できます。一般にコンポーネントの各バージョンは個別のタスクに関連付けられます。

コンポーネントを公開する人が、タスクを作成して適切なファイルやプロジェクトに関連付けることがあります。コンポーネントの消費者が、タスクを作成し、自分が必要とするコンポーネントファイルに関連付けることも可能です。あるコンポーネントを複数のタスクに関連付けることもできます。

消費者がタスクを作成する利点は、コンポーネントの新しいバージョンを自分で単体テストでき、必要なあらゆる変更を加え、またそれらの変更を同じタスクに関連付けできることです。これにより、新しいコンポーネントバージョンへのアップグレードのための変更をすべてまとめられるので、チームの他の人は新しいコンポーネントバージョンの影響を受けずにすみます（コンポーネントバージョンは複数のタスクに関連付けできます。つまり、1 つはコンポーネントを開発したチーム、もう 1 つはそれを使用する各チームといった具合です）。

Telelogic Synergy では複数のタスクをフォルダにまとめることができます。つまり、一まとまりとして使用できることが保証された、一連の互換性あるコンポーネントバージョンをグループ化するフォルダを作成できます。このようなフォルダは、同一セットのコンポーネントを再利用することが必要な、異なる消費者向けのアプリケーションで共有できます。

プロセスパターン

コンポーネントベースの開発は、現在よりも低コストで、高品質、顧客満足度の高いソフトウェアソリューションの提供を確実に促進します。この手法の実践は、ソフトウェア開発業界を席巻しつつありますが、現在のツールと技術の大半は、コンポーネントを管理、公開、共有するために開発チームがどのように協業するのかという重要な課題を見落としています。

Telelogic Synergy は、複数チーム間や個人の間でコンポーネントを共有するための強力なフレームワークとプロセスを提供します。この仕組みによってコンポーネントの管理、公開、再利用、配布が可能となると同時に、ビルトインの作業フローを使うことによって、ソフトウェア開発の「最良の実践」が可能になります。

Telelogic Synergy は、柔軟な方法でコンポーネントベースの開発を支援し、数多くのプロセスパターンを提供しています。プロセスパターンについては、IBM Rational Software Support site を参照してください。

5 用語解説

4 部名称

Telelogic Synergy データベース内のオブジェクトを識別するのための一意の識別子。4 部名称は、**name-version:type:instance** （名前 - バージョン：タイプ：インスタンス）のように記述されます（オブジェクトスペックまたは完全名称とも呼ばれます）。

CLI

「command line interface（コマンドラインインターフェイス）」の略。UNIXやWindows の CLI からほとんどの Telelogic Synergy の操作を行うことができます。

completed（完了済み）状態

終了したタスクに割り当てられる状態。

GUI

「graphical user interface」の略。GUI を使用して多くの Telelogic Synergy 操作を行うことができます。

integrate（統合）状態

開発者がチェックインしたバージョンに与えられる状態。統合状態にあるオブジェクトは、他の開発者がチェックアウトしたり、使用することができます。

prep（準備）プロジェクト

52 ページの「**ビルド管理プロジェクト**」を参照してください。

released（リリース済み）状態

リリース済みもしくはマイルストーンに到達したオブジェクトに与えられる状態。

shared （共有）状態

複数のユーザーが修正できるプロジェクトに与えられる状態。

Telelogic Synergy データベース

ソースファイルとデータファイル、プロパティ、相互関連を含む、すべての管理データを保管するデータリポジトリ。

Telelogic Synergy Distributed

世界中のいたるところにあるTelelogic Synergyデータベース内のソフトウェア変更を共有可能にする Telelogic Synergy モジュール。

task_assigned（タスク割り当て済み）状態

開発者に割り当てられたタスクを表す状態。

visible（可視）状態

他のユーザーにより使用可能だが変更不可能とするためにソースオブジェクトに与えられた状態。

working（作業中）状態

オブジェクトを開発者個人用にコピーして作業している状態。これが、開発者が変更を行う状態です。

インスタンス

同じ名前と種類を持つ複数のオブジェクト間を区別するために使用されるプロパティ値。しかし、お互いのバージョンではない。

オブジェクト

ファイルやディレクトリのような、データの集まり。オブジェクトの例には、ソースファイル、makefile、テスト結果、ディレクトリ、ドキュメントなどがあります。

オブジェクトスペック

47 ページの「4 **部名称**」を参照してください。

オブジェクトバージョン

オブジェクトの特定の改訂。それぞれのオブジェクトバージョンは、それをさらに定義するために一連のプロパティを持ちます（例、**名前**、**所有者**、**作成時間**など）。

開発プロジェクト

開発者が、開発や変更後のテストに使用するプロジェクト。

外部プロジェクト

製品を含む特定のプロジェクト。開発者はビルドするために使用するソースコードを含むプロジェクトをコピーせずに製品にアクセスできます。

概要

タスクの名前。

カレントタスク

開発者が現在作業しているタスク。開発者がカレントタスクを指定すると、チェックアウトするオブジェクトはすべて、自動的にタスクに関連付けられます。

完全名称

4 部名称を参照。

管理製品

Telelogic Synergy 内のオブジェクトバージョンとして管理される製品ファイル。

関連付けられた

タスクでグループ化されたオブジェクトを、タスクに関連付けられているといいます。

共同開発

Telelogic Synergyでファイルバージョンを再利用する機能。1つのプロジェクトでファイルバージョンを作成またはビルドし、他のプロジェクトで使用できます。

権限

権限により、データベース内で行える操作が決まります。また、作業するそれぞれのデータベースにより設定が異なる場合もあります。ユーザーが手動で権限を変更する必要は絶対にありません。ユーザーがある操作を行おうとすると、Telelogic Synergyはその操作のための該当権限が使用可能かどうかを判定します。

更新

プロジェクトの内容をそのメンバーオブジェクトの最新バージョンで自動的に更新するプロセス。プロジェクトまたはディレクトリ内の各オブジェクトバージョンが評価され、適切なバージョンが Telelogic Synergy の使用可能な候補から選択されます。

更新プロパティ

作業者がプロジェクトを更新する際に、選択するオブジェクトバージョンを決めるためにプロジェクトが使用するプロパティ（旧リリースでは、「更新」は「リコンフィギュア」と呼ばれていました）。

候補

プロジェクト内のディレクトリエントリで使用するのにふさわしいファイルバージョン。

個別開発

開発者は Telelogic Synergy を使用すれば、必要となるまで変更を他の開発者から取得せずに開発プロジェクトでの変更を開発しテストできます。開発者がタスクを完了すると、そのタスクを統合テストプロジェクトの中に含めることができます。開発者が自分たちのプロジェクトを更新する際、自分たち自身がチェックアウトしたバージョンを保持し、統合テストに合格した最新のバージョンを取得します。

コンポーネント

コンポーネントは、使用方法を示すサポートファイルを伴う、ヘッダーファイル、ヘルプ、互換性と依存性についての情報、ハードウェアまたはソフトウェア必要条件、設計情報、テストケースなど、1 つ以上のライブラリまたは実行可能ファイルです。

コンポーネント名

54 ページの「**リリース**」を参照してください。

コンポーネントリリース

54 ページの「**リリース**」を参照してください。

作業中プロジェクト

48 ページの「**開発プロジェクト**」を参照してください。

サブプロジェクト

他のプロジェクト内に含まれるプロジェクト。

システムテストプロジェクト

システムテストとリリース準備に使用するファイル、ディレクトリ、製品のバージョンを含むプロジェクト。

状態

ライフサイクルにおける段階のようなオブジェクトの特徴、および実行できる処理（たとえば、誰が修正できるか）を定義します。

製品

他のファイルを処理することにより、ビルドされるファイル。

製品タスク

製品を管理するために自動的に Telelogic Synergy が作成するタスク。

属性

プロパティを参照してください。

タスク

ソフトウェアアプリケーション内の論理変更を完了するために必要となるソフトウェア変更をすべてグループ化したもの。

タスクベースの方法論

開発組織が作業の基本ユニットとして、個々のファイルよりも、タスクを使用してソフトウェアアプリケーションに対する変更を追跡できるようにする方法論。

チェックアウト

Telelogic Synergy のデータベース内に保管されている既存バージョンからオブジェクトの新規バージョンを作成するプロセス。開発者は作業ができるようにオブジェクトをチェックアウトします。

チェックイン

他のユーザーも利用できる開発者のオブジェクトバージョンを作成するために使用される操作。

ディレクトリエントリ

Telelogic Synergy ディレクトリ内のプレースホルダで、そのディレクトリ内に属するファイルの記録を取ります。

統合テストプロジェクト

統合テスト用の最新の完了済みタスクをすべて集めるために使用されるプロジェクト。

統合テスト

離散ソフトウェア変更を一緒にビルドし、テストするプロセスで、それが正しく動作することを確認します。

同期

開発者が自分のワークエリアをデータベースと比較し、抜粋して更新するために使用する操作。

並行開発プロパティ

パラレルオブジェクトバージョンを互いに区別するプロパティ。

並行同時開発

通常コードの別セクションに取り組むために、複数の開発者が同じオブジェクトから自分自身の作業中バージョンをチェックアウトする際に発生します。

パラレルバージョン

単一のオブジェクトからチェックアウトされる複数のオブジェクト。

並行バリアント開発

複数の開発者が異なるハードウェアプラットフォーム用に同じオブジェクトの異なるバージョンの作業をする際に生じます（多くの場合、この状況をバリアントといいます)。

並行リリース開発

組織がソフトウェア製品の複数リリースを同時に作らねばならないときに生じます(たとえば、現行リリースをすべてのベースラインとして、別々の開発者が次期リリースや現行リリースのパッチ作業を行ったり、保守リリースの作業に取り組んでいるといった状況があります)。

バリアントプロジェクト

製品のプラットフォーム特有のバリエーション。

非管理製品

自分のワークエリアには存在するが、Telelogic Synergy データベースには存在しない製品。

ビルド

ツール、コンパイラ、コードジェネレータを使用して、既存のソースファイルからファイルを生成するプロセス。

ビルド管理プロジェクト

ビルドマネージャがテストやリリース用ソフトウェアの準備に使用するプロジェクト。

ビルドマネージャ

ソースファイルからプロジェクトや製品をビルドする、開発組織内の担当者。ビルドマネージャは、プロセス、リリース、フォルダ、フォルダテンプレートの作成、コピー、修正、削除を行えます。また、プロセスルール、目的、リリースの作成、修正、削除を行えます。

フォルダ

名前を付けたタスクのグループ。

フォルダテンプレート

フォルダを作成するために使用されるパターン。フォルダテンプレートは、テンプレートに基づく各フォルダに適用される以下のプロパティのセットを持っています。フォルダへの書き込みが可能、使用可能、手動とクエリ使用のいずれで更新するか、またタスクの選択で使用されるクエリ。プロセスルールは、プロジェクトの更新基準の一部としてフォルダテンプレートを使用できます。

プラットフォームプロパティ

特定のハードウェアプラットフォーム用のプロジェクトやオブジェクトを示す識別子。

プロジェクト

関連ファイル、ディレクトリ、他のプロジェクトのユーザー定義のグループ（サブプロジェクトと呼ぶ)。プロジェクトは通常、ライブラリや実行形式ファイルのような、ソフトウェアの論理グループを示し、ファイルのディレクトリ構造を持ちます。

プロジェクトグルーピング

Telelogic Synergy では、プロジェクトはその目的とリリースによって、たとえば **My 1.0 Insulated Development Projects** のようにグループ化されます。これをプロジェクトグルーピングと呼びます。また、プロジェクトグルーピングには、プロジェクトの更新時に使用されるタスクおよびベースラインが含まれます。

プロジェクト のコピー

開発者は使用するプロジェクトをコピーします。開発者は修正が必要な場合、そのプロジェクトをコピーする必要があります。

プロセス

複数のプロセスルールを名前を付けたグループにグループ化したもの。プロセスはリリースに使用可能なプロセスルールを指定するのに使用されます。

プロセスルール

プロジェクトをどのように更新するかを定義するパターン。ベースラインを決めるためのルールと、それに加えて、作業者がプロジェクトを更新する際に使用する一連のタスクとフォルダを指定します（旧リリースでは、「プロセスルール」は「更新テンプレート」または「リコンフィギュアテンプレート」と呼ばれていました）。

プロパティ

オブジェクトに与えられたプロパティ。オブジェクトの基本プロパティは、4 部名称（**名前**、**タイプ**、**インスタンス**、**バージョン**）により識別される項目で、**所有者**、**状態**、**プラットフォーム**、**リリース**も含まれます。

ベースライン

ある時点における一連のプロジェクトとタスクのセットのスナップショット。以降の開発の開始点として使用されることがあり、参照のために他のベースラインと比較されることがあります。

ベースラインプロジェクト

自分のプロジェクトのベースにするプロジェクトバージョンをそのベースラインプロジェクトと呼びます。たとえば、「**editor-2.0**」プロジェクトのベースラインプロジェクトは、「**editor-1.0**」です。プロジェクトの新規バージョンをチェックアウトすると、そのベースラインプロジェクトが自動的に設定されます。

方法論

ソフトウェア開発を管理するために使用されるプロセスと方策。

目的

プロジェクトの状態を指定し、更新時に確実に正しいメンバーが選択されるよう、その状態とプロジェクトのリリースのプロセスルールとを対応づけるための設定。

ライフサイクル

オブジェクトバージョンが遷移する一連の状態。オブジェクトバージョンの状態は遷移するため、任意の時間で必ずひとつの状態にある。

リコンフィギュアプロパティ

49 ページの「**更新プロパティ**」を参照。

リリース済みプロジェクト

リリース済みもしくはマイルストーンに到達したソフトウェアのバージョン。

リリースプロパティ

ある特定のリリースに特有のプロジェクトやタスクを識別するプロパティ。

リリース

リリースはコンポーネント名（オプション）とリリースデリミタ、およびコンポーネントリリースで構成されます。コンポーネント名はアプリケーションまたはコンポーネントの名前を示します。たとえば、**Telelogic Synergy** または **editor** などがあります。コンポーネントリリースは、そのアプリケーションまたはコンポーネントの特定のリリースを示します。**Telelogic Synergy/6.6** は、リリース名の一例です。

履歴

オブジェクトのすべての既存バージョンとバージョン間の関係。

履歴ダイアログボックス

オブジェクトの履歴を表示する Telelogic Synergy ダイアログボックス。上記の履歴を参照。

ロール

49 ページの「**権限**」を参照してください。

ワークエリア

作業者がプロジェクトを個人的使用のためにコピーする際に、Telelogic Synergy がプロジェクトを書き込むファイルシステム内の場所。

割り当てる

ある特定のタスクを、作業する開発者に割り当てるプロセス。

付録：特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の 製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの 評価および検証は、お客様の責任で行っていただきます。

IIBM は、本書に記載されている内容に関して特許権（特許出願中のものを含む）を保有している場合があります。本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 106-8711

東京都港区六本木 3-2-12

日本アイ・ビー・エム株式会社

法務・知的財産

知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。: IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示 もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

:

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム（本プログラムを含む）との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、製造元に連絡してください。

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, Massachusetts 02142
U.S.A.

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

商標

IBM、IBM ロゴ、ibm.com、AIX、Rational、Telelogic、Telelogic Synergy、Telelogic Change は、International Business Machines Corporation の米国およびその他の国における商標ま

たは登録商標です。これらおよび他の IBM 商標に、この情報の最初に現れる個所で商標表示 (® または ™) が付されている場合、これらの表示は、この情報が公開された時点で、米国において、IBM が所有する登録商標またはコモン・ロー上の商標であることを示しています。このような商標は、その他の国においても登録商標またはコモン・ロー上の商標である可能性があります。IBM および関連の商標については、www.ibm.com/legal/copytrade.html をご覧ください。

Microsoft、Windows、Windows 2003、Windows XP、Windows Vista、および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

UNIX は、The Open Group の米国およびその他の国における登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

:

索引

C

CLI、説明 2

E

Eclipse 8

G

GUI（グラフィカルユーザーインターフェイス）2

I

IBM Rational Application Developer 8
integrations
  Eclipse 8
  IBM Rational Application Developer 8
  Microsoft Visual Basic 8
  Microsoft Visual C++ 8
  Microsoft Visual Studio 8

M

Microsoft Visual Basic 8
Microsoft Visual C++ 8
Microsoft Visual Studio 8

P

prep（準備）プロジェクト 20

Q

QA テストサイクル 38

R

released（リリース済み）
  状態、およびベースライン 39
  プロジェクト、バージョン 20

T

Telelogic Synergy Classic インターフェイス、説明 2
Telelogic Synergy Distributed 8
Telelogic Synergy インターフェイス、説明 2

W

Windows 開発、統合 8
Windows 開発環境との統合 8

お

オブジェクト
  4 部名称 14
  完全名称 14
  切り取り 23
  更新 24
  削除 23
  作成 23
  修正 14
  使用 23
  状態 18
  新規バージョンの作成 14
  スペック 14
  他のユーザが利用できるようにする 14–15
  チェックアウト 14
  チェックイン 14–15
  定義 11
  バージョン、および 4 部名称 14
  バージョン、定義 11
  バージョン履歴 15
  複数の場所 14
  プロジェクトに貼り付け 23
  プロパティ 17

ライフサイクル 18
履歴 15
オブジェクトの切り取り 23
オブジェクトのコピー、説明 23
オブジェクトの削除 23
オブジェクトの作成 23
オブジェクトの使用 23
オブジェクトの状態 18
オブジェクトのプロパティ 17
オブジェクトの履歴、説明 15

か

概念、説明 11
概要（Synopsis）12
カレントタスク、説明 17
完了状態、定義 19
関連オブジェクト、定義 12

け

権限、定義 49

こ

更新
オブジェクト 24
プロジェクト 24
プロジェクトとオブジェクト
更新プロパティを参照。
更新プロパティ
オブジェクトの評価方法 33
定義 33
更新テンプレート、定義 24
構成管理
目的 5
利点と機能 6
構成管理の機能 6
構成管理の利点 6
候補、定義 20
個人用プロジェクトバージョン 36

さ

作業中
状態、説明 18
プロジェクト、定義 20
プロジェクト、およびタスク 36
作業フロー
タスク 35
デフォルト 35
プロジェクト 29
サブシステム属性 13
サブプロジェクト 19

し

システムテスト prep（準備）プロジェクト 30, 38
システムテストサイクル 38
システムテスト準備プロジェクト 38

せ

製品、定義 25

た

タスク
オブジェクトとの関連 12
開発者の修正プロセス 35
カレント、説明 17
関連オブジェクト 12
作業中プロジェクト 36
作業フロー 35
使用 35
定義 11
ライフサイクル、定義 19
割り当てる 35
タスクの割り当て、説明 35
タスクベース方法論、説明 27
タスク割り当て済み状態、定義 19

ち

チェックアウト

定義 14
リリース指定 31
チェックイン 14–15

て

定義
説明 11
用語解説 47
ディレクトリエントリ 20
ディレクトリ、定義 20
データベース
情報転送 8
定義 11
同期のバックアップへの影響 22
ワークエリアとの同期 21
データリポジトリ、説明 11
テストサイクル
QA 38
システム 38
統合 36
デフォルトの作業フロー 35

と

同期
ワークエリアとデータベース 21
統合状態 18
統合テスト prep（準備）プロジェクト 30, 36
統合テストサイクル 36

は

バージョン
オブジェクト、および 4 部名称 14
オブジェクト、定義 11
パラレルオブジェクトバージョンのマージ 15

ひ

ビルド
makefile を参照。
定義 25

ふ

フォルダ
タスクの追加 25
定義 25
プラットフォーム属性 13, 17
プロジェクト
システムテスト準備プロジェクト 38
prep 20
released（リリース済み）20, 39
オブジェクト変更 23
開発プロセス 36
更新 24
個人用 36
作業中 36
作業中、定義 20
作業フロー 29–34
サブプロジェクト 19
システムテスト prep（準備）プロジェクト 30, 38
定義 19
統合テスト prep（準備）プロジェクト 30, 36
バージョン付け方法 19
目的 32
リリース済み 30
プロジェクトグルーピング
説明 20
定義 52
プロジェクトのための開発プロセス 36
プロジェクトをコピーしてリリースを指定 31

へ

並行開発
異なるプラットフォーム用 42, 43
異なるリリース用 42, 44
属性 42
タイプ 42
同時 42, 43
ベースライン

更新 24
更新テンプレート 24
説明 24
定義 53
プロジェクトグルーピング 20
プロジェクト、説明 24
プロジェクト、定義 53

ほ

方法論、タスクベース 27

も

目的、プロジェクト 32

よ

用語
説明 11
用語解説 47
用語解説 47

ら

ライフサイクル
オブジェクト 18
タスク 19

り

リリース
属性 13, 17
次の準備 39
名前、定義 31
名前、例 31
プロジェクトをコピーするときを指定 31
リリース状態
状態、およびライフサイクル 18
リリース済み
プロジェクト、作業フロー 30

わ

ワークエリア
データベースとの同期 21
定義 21