ソフトウェア工学 73−14
（1990. 7. 20）

構成・版管理ライブラリ LifeLine を用いたC プログラム開発環境

坪谷 英昭、岸 知二、入交 晃一、鈴木 美和子*、猪狩 錦光*

日本電気 (株) ソフトウェア生産技術開発本部
* 日本電気技術情報システム開発 (株)

ソフトウェア開発支援環境は、開発過程で生み出される成果物を管理するための機能を提供しなければならない。我々は既に、ソフトウェア開発環境を構築する際のプラットフォームとして、成果物や設計情報を管理するためのライブラリ LifeLine を開発している。本稿では、LifeLine を用いて構築したC プログラムの開発環境 LifeStudio/C について報告する。LifeStudio/C は、複数人での共同開発を支援し、成果物管理機能に加えて、構成管理、版管理の機能を提供する。LifeStudio/C を用いることにより、実行形式の作成といったインテグレーション作業が容易になるとともに、システムバージョンの管理の手間を軽減させることができる。

LifeStudio/C: A Software Development Environment for C Programs

Hideaki TSUBOTANI　Tomoji KISHI　Koichi IRIMAJIRI
Software Engineering Development Laboratory, NEC Corporation
Igarash building, 11-5, Shibaura 2-chome, Minato-ku, Tokyo, 108, JAPAN

Miwako SUZUKI　Kanemitsu IGARI
NEC Scientific Information System Developement Corporation

A software development environment must present facilities to manage artifacts which are produced in software development process. We have developed LifeLine library for managing artifacts and design information, as a basis for constructing software development enviornments. This paper presents LifeStudio/C, a software development environment for C programs, which are constructed using LifeLine. LifeStudio/C supports software projects of many people and provides such facilities as configuration management and version control. By the use of LifeStudio/C, a user can easily create a target such as a load module and control system version.

1 はじめに

大規模なソフトウェア開発において、開発過程で生み出される様々な成果物を効果的に管理していくことは非常に重要である。

近年様々なCASEツールに関する研究・開発が盛んになっているが、成果物管理の機能はどのようなツールにも必要とされる基盤機能であると考えられる。

このような観点から、我々はソフト開発環境を構築する際のプラットフォームとして、成果物や設計情報を管理するための機能LifeLineを開発した[2]、[3]。LifeLineは、UNIX上の種々のCASEツールに組み込まれて使用されることを目的としたCライブラリである。

今回我々は、LifeLineを実際のツールに組み込み、その機能の有用性の実証ならびに評価を行なうためにCプログラムの開発環境LifeStudio/Cを構築した。適用例としてCプログラム開発環境を選んだ理由は、Cの開発支援環境に対する要望が強いということと、環境自身をブートストラップ的に開発・強化していくことができるためである。

本稿では、LifeLineにおける情報管理の考え方を示しながら、LifeStudio/Cの提供する機能について述べる。まず、第2章で、LifeLineライブラリが提供する基本機能について概説する。第3章以降では、LifeStudio/Cについて述べる。

2 LifeLineが提供する基本機能

LifeLineライブラリは成果物としてのファイルと、それに関連する設計情報を管理する機能を持っている。プロジェクトは成果物や設計情報の管理空間である。すべての情報はプロジェクト中で管理される。プロジェクト中ではフォルダによって階層的な管理構造が定義される。フォルダはUNIXのディレクトリと1対1に対応している。フォルダ中には成果物であるファイル、構成を表すコンポジション、ロードモジュールの様に他のファイルから生成されるファイルを表すターゲット等が管理される。これらの概念について、より詳しくは[2]、[3]を参照されたい。

2.1 成果物の種類

成果物には、最終成果物としてのソースコード、ロードモジュール、各種ドキュメントなどだけでなく、中間成果物としてのドキュメント、オブジェクトコード、テストデータ等々も含まれる。

これらの成果物は大きく二種類に分けることができる。一つは作業者がエディタ等を用いて直接作成するものであり、各種ドキュメント、図式、ソースコード、等々がこれに含まれる。もう一つは他の成果物から機械的な手続きによって生成できるものであり、オブジェクトコードやロードモジュール等々はこれに含まれる。LifeLineでは前者をファイル、後者をターゲットと呼ぶ。

2.2 構成情報の管理

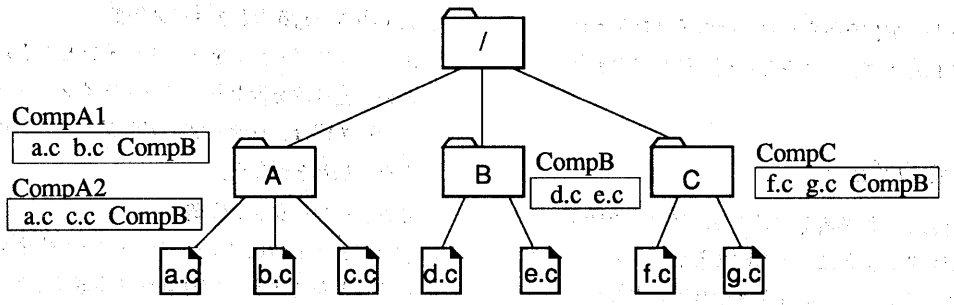
成果物の管理においては、その構成情報を管理する必要がある。一般に構成は成果物の管理構造とは異なることが多く、例えばファイルシステムの提供するディレクトリ構造などを用いて構成を表現することは、一般に困難であるといわれている[5]。LifeLineでは管理構造を表現するフォルダとは別に、構成を表現する手段としてコンポジションを用意している。コンポジションは、ファイル、ターゲット、他のコンポジションの名前のリストであり、フォルダによる管理構造とは独立に、階層的に構成を表現することができる。

例えば、図1(a)のようなフォルダ構造が作られており、フォルダA、B、C中にコンポジションCompA1、CompA2、CompB、CompCが存在している場合には、図1(b)に示すように3つのコンポジション構造が存在している。

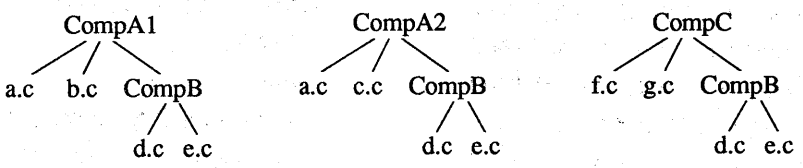
前述したターゲットの定義にはその生成に必要となる成果物の集合を指定する必要があるが、その指定にはコンポジションを用いる。構成情報はソフト開発の進行に伴って変化していくが、その構成に対応するコンポジションを修正さえすれば、ターゲットの定義にもそのまま反映される。

2.3 版管理

成果物の版管理機能としてはSCCS [6]やRCS [7]が広く用いられているが、ソフト開発環境でも、個々のファイルの版管理には同様の差分管理の技法が用い



(a) フォルダ構造



(b) コンポジション構造

図 1: フォルダ構造とコンポジション構造

られることが多い。一般に版管理の対象となるのはソースコード等のファイルであり、オブジェクトコードやロードモジュールは対象とならない。構成管理においてはファイル単体の版管理だけでなく、構成情報そのものの版管理をする必要がある。そこで LifeLine はコンポジションに対する版管理機能を提供している。従ってファイルに対しては SCCS や RCS と同様な差分管理を適用するが、ターゲットに対してはそれに対応付けられたコンポジションとその生成手続きの記述に対して版管理を行なうことにより版を管理する。

2.4 アクセス制御

ソフトウェア開発環境が支援すべきひとつの機能として、複数人の共同開発時における情報のアクセス制御の機能がある。これはベースラインとワークスペースでの情報のやりとりによって解決することが一般的である。すなわち作業者はベースライン中にある成果物等にロックをかけ、それを自分のワークスペースにコピーしてその修正を行なう。その間他の作業者はその成果物に対して修正作業を行なうことはできない。修正作業が終了すると、作業者はその成果物をベースラインに返し、ロックを解除する。LifeLine ではこの操作をリザーブ・デポジット [4] と呼ぶ。

LifeLine ではこうしたベースラインやワークスペースを表現するために、プロジェクトを用いる。プロジェクトにはベースラインに相当するルートプロジェクトと、ワークスペースに相当するサブプロジェクトの二種類がある。開発作業に対しては一つのルートプロジェクトと、その子供として定義される複数のサブプロジェクトが用意される。一般にサブプロジェクトは作業者毎に作成される。サブプロジェクトがさらにサブプロジェクトを持つこともある。サブプロジェクト中で成果物等の修正作業等を行なう時には、リザーブ・デポジットの操作を用いて行なう。リザーブ・デポジットは、フォルダ、ファイル、コンポジション、ターゲットに対して行なうことができる。

LifeLine では、リザーブの他に二重化という機能を提供している。二重化はオブジェクトの修正権をとらずに、オブジェクトのコピーだけをカレントプロジェクト中に作成する処理である。二重化は自分だけが参照する目的で過去の版をチェックアウトしたり、コンパイルを行なったりする際に用いられる。二重化されたオブジェクトに対しては、リザーブしたオブジェクトと同様に修正作業を行なうことができるが、その結果を親プロジェクトに返すことはできない。二重化している間に、親プロジェクト中の該当オブジェクトが修正されても、あくまでカレントプロジェクト中のオブジェクトの内容しか見ることができない。またある

プロジェクトが二重化を行なうと、その子プロジェクトは二重化されたオブジェクトしか見ることができなくなる。

2.5 作業者のビュー

開発作業時には、作業者毎に異なったビューで情報を見る必要がある。例えば一人の作業者がソースコードを修正している場合、その作業者は修正途中のソースコードを見ながら作業をしなければならないが、他の作業者はその修正が完了するまでは、修正途中のソースコードを見ることは望ましくない。また修正が終了してデポジットを完了した時点では、二人の作業者は同一のソースコードを参照しなければならない。LifeLine ではリザーブ・デポジットの状況を見て、プロジェクト毎のビューを一意に決定している (図2参照)。

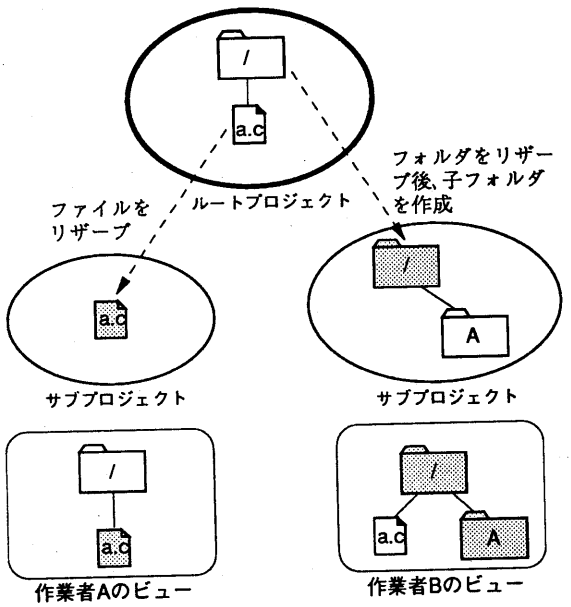


図 2: 作業者のビュー

3 LifeStudio/C の概要

LifeStudio/C は、前述した LifeLine ライブラリを用いて構築した C プログラム用の開発支援環境である。LifeStudio/C は主に以下のような作業を支援することを目的としている。

1. 成果物の一元管理とアクセス制御
   ルートプロジェクトをベースラインとする成果物の一元管理を行なう。ベースラインとワークスペース間は、リザーブ・デポジットによりアクセス制御を行なう。
2. インテグレーション作業
   プロジェクト毎にインテグレーション作業を行なうことができる。Makefile を生成する方式でインテグレーションを行なう。
   複数のマシン機種に対するターゲットの開発も支援する。
3. 版管理
   ファイル単体の版管理の他に、システムの構成に対する版管理も行なうことができる。

LifeStudio/C において C 言語に依存した機能はターゲットの作成機能だけであり、その他の機能は言語独立な機能である。

図 3に LifeStudio/C の構成を示す。成果物管理部に LifeLine、ユーザインタフェース部に X ウィンドゥならびに当社で開発したユーザインタフェース構築部品 "鼎" を用いている。

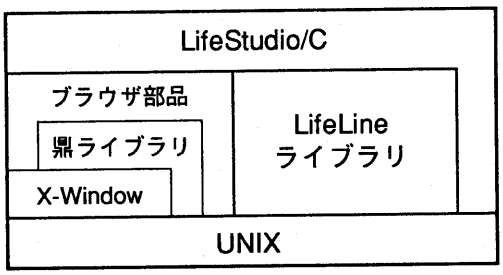


図 3: LifeStudio/C の構成

LifeStudio/C は、以下のような 3 つのサブツールから構成されている。

- ファイルマネージャ (FM)
  コンポジション構造やフォルダ構造のブラウジングとオブジェクトの作成を支援。
- ターゲットマネージャ (TM)
  ターゲットの作成を支援。
- バージョンマネージャ (VM)
  ファイルとコンポジションの版管理を支援。

以降の章では、これらのサブツールについてその特徴的な機能を述べる。

4 ファイルマネージャ (FM)

ファイルマネージャは、コンポジション構造をブラウジングするためのコンポジションブラウザとフォルダ構造をブラウジングするためのフォルダブラウザを提供する。

ファイル、フォルダ、コンポジションの作成など、ほとんどの作業はコンポジションブラウザ上で行ない、フォルダブラウザはオブジェクトの実際の削除のためだけに用いる。

4.1 コンポジションブラウザ

プロジェクトに対して LifeStudio/C を起動すると、最初に図4のようなコンポジションブラウザが表示される。表示のモードには、アイコンとリストの2種類があるが、図の例ではアイコンモードで表示されている。

4.1.1 オブジェクトの状態

コンポジションブラウザ、フォルダブラウザともに、オブジェクトが以下のいずれかの状態にあることが分かる。

- 修正可能 (修正権がありかつリザーブ中である)
- 二重化 (修正権がありかつ二重化中である)
- 修正不能 (修正権がない)

アイコン表示モードのブラウザ上では、アイコンの外形によって上の3つの状態が区別される。コンポジションの名前は、コンポジション名 [親フォルダ名] のように表示される。ただし、フォルダと同名のコンポジションはコンポジション名のみ表示される。

4.1.2 支援作業

コンポジションブラウザ上で行なうことができる主な操作について説明する。

- コンポジションエレメントの表示
  コンポジションに登録してあるエレメントをブラウザ上に表示する。
- コンポジションエレメントの登録・削除
  コンポジションに新しくエレメントを登録したり、既に登録してあるエレメントを登録から外す (実際にオブジェクトを削除するわけではない)。
- コンポジションの切替え
  現在登録してあるコンポジションをエレメントから外し、新しいコンポジションを登録する。これにより、コンポジション構造を変更することができる。
- コンポジションの追加
  既に存在しているフォルダ中に新しくコンポジションを作成する。
- コンポジションの新規作成
  新しくフォルダを作成し、その中にコンポジションを作成する。
- ファイルの生成
- リザーブ
- デポジット
- 二重化
- リザーブ / デポジット / 二重化のキャンセル
- 他ツール (フォルダブラウザ、TM、VM) の起動

4.2 フォルダブラウザ

フォルダブラウザはフォルダ中のすべてのフォルダ、コンポジション、ファイルを表示する。

フォルダブラウザ中で行なうことができる操作は、オブジェクトの実際の削除だけである。

5 ターゲットマネージャ (TM)

TM は、実行形式やライブラリ (.a ファイル) の作成作業 (インテグレーション) を支援する。

5.1 Make の利用

UNIX 上でインテグレーション作業を支援するツールとして Make [1] がよく知られており、広く用いられている。そこで、LifeStudio/C でも、UNIX 環境への移行性などという観点から、実際のインテグレーションは Make を呼び出すことによって行なうことにした。

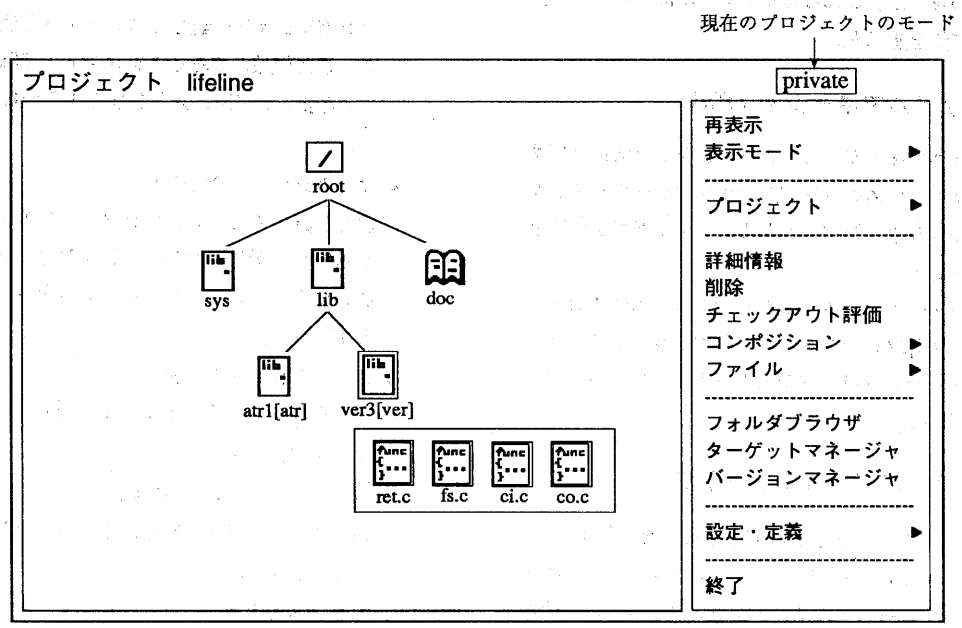


図 4: コンポジションブラウザ

Make を使用するためには、ユーザは Makefile というファイルを作成しておかなければならない。Makefile 中には、例えば実行形式を作成する場合には、リンクするために必要なオブジェクトファイル名の一覧やヘッダファイルへの依存関係といった情報を記述しておかなければならない。また、実行形式を作成するのに必要な構成ファイルが追加・削除されるたびに、Makefile を修正しなければならない。この手間を軽減するために、TM は Makefile 生成の機能を提供している。

5.2 Makefile の生成

各ターゲットに依存した Makefile は、予め用意されているテンプレートファイル中に以下のような定義情報を埋め込み、プリプロセッサ (cpp) で処理することによって生成される。

1. プロジェクトで共通の定義
   参照するライブラリのディレクトリ名、コンパイルフラグなど。この定義は各プロジェクトで予め用意しておく。
2. コンポジションに依存した情報
   そのターゲットを作成するために必要なオブジェクトファイル名の一覧など。オブジェクトファイル名の一覧は、コンポジションに登録されているソースファイルの名前を変換することによって得ることができる。
3. ユーザが記述した Makefile
   ユーザは、TM を用いて、生成される Makefile に含ませたい定義を記述することができる。テンプレートには予めいくつかの Make のターゲットが記述されているが、それら以外のターゲットを作成したい場合などには、ユーザが直接 Makefile の一部を記述することができる。
4. 依存関係の情報
   各オブジェクトファイルがどのヘッダファイルに依存しているかに関する情報。

テンプレートファイル中には上の順番で埋め込まれていくので、同じものが異なる場所で定義されている場合には、下にあるものの定義が使われる。

これらの情報はファイルとしてコンポジションに登録されて管理されるので、コンポジションを版管理することにより、任意の版の Makefile を再生成することが可能である。

なお、テンプレートはコンポジションの型ごとに用意することができる。

5.3 複数機種への対応

実際のソフトウェア開発においては、複数のマシン機種に対して同時に開発を行なっていく場合が多い。このような場合、ソースファイルは一元管理の観点から一ヶ所にまとめておくことが望ましいが、オブジェクトファイルは機種ごとに異なる場所に置かなければならない。

そこで、LifeStudio/C では、各フォルダに対して機種ごとにオブジェクトファイル格納用ディレクトリを用意している。ユーザは、Make を行なう前に、どのマシン用のターゲットを作成するかを指定するだけでよい。Make を行なった後、作成されたオブジェクトファイルは機種ごとに用意されている格納用ディレクトリに移される。

6 バージョンマネージャ (VM)

VM はファイルとコンポジションの版管理を支援する。

6.1 チェックイン・チェックアウト

図5はコンポジションに対して VM を起動した場合に表示される画面である。画面上半分にはそのオブジェクトの版の履歴が木構造で表示される。この中からひとつの版を選ぶと、その版を含む版のパス上にあるすべての版のリストが画面下半分に表示される。

版をチェックインしたりチェックアウトする場合には、リスト表示の中からひとつの版を選んだ後、メニューから“チェックイン”または“チェックアウト”を選択する。チェックインが選ばれた場合には、ダイアログボックスが表示されるので、それを用いて新しい版の版識別子の設定、コメント文の入力、キーワードの設定を行なう。

コンポジションをチェックインすると、そのコンポジションが含んでいるすべてのファイルが自動的にチェックインされる。コンポジションがコンポジションを含んでいると、そのようなエレメントのコンポジションに対してもチェックイン操作が適用される。つまり、指定されたコンポジションをルートとするコンポジション構造に含まれるすべてのオブジェクトがチェックインされる。チェックアウトの場合も同様に、コンポジション構造に含まれるすべてのオブジェクトがチェックアウトされる。

コンポジションの版管理機能を用いることによって、あるロードモジュールを作成するために必要なソースファイルの集合をまとめて版管理したり、仕様書と設計書を対応させて版管理する、といったことが容易に行なえるようになる。

6.2 チェックアウトのタイミング

コンポジションをチェックアウトした場合には、コンポジション構造に含まれる大量のファイルがチェックアウトされる可能性がある。そのため、チェックアウトが指示された時点で実際にチェックアウトを行なうとシステムの応答性が悪くなる恐れがある。また、コンポジションに含まれるエレメントをすべてチェックアウトしたとしても、実際にアクセスが行なわれるファイルの数は少ないと考えられる。

そこでチェックアウトのタイミングに次の二つのモードを用意した。

- Eager モード
- Lazy モード

Eager モードではチェックアウトが指示されるとすぐに実際のチェックアウト処理が行なわれる。Lazy モードでは、必要になるまでチェックアウト処理は行なわれない。

Lazy モードでコンポジションがチェックアウトされた場合には、そのコンポジションと子コンポジションがリカーシブにチェックアウトされる。この際ファイルはチェックアウトされない。

また、Lazy モードでチェックアウトされる場合には、ユーザが誤って目的の版以外にアクセスすることを避けるため、カレントのファイルの実体はフォルダ中から削除しておく。

6.3 版の検索

チェックインされた複数の版の中から、特定の版を検索することができる。検索時に指定できる検索条件としては、

- 指示された日時より前あるいは後に登録されたもの

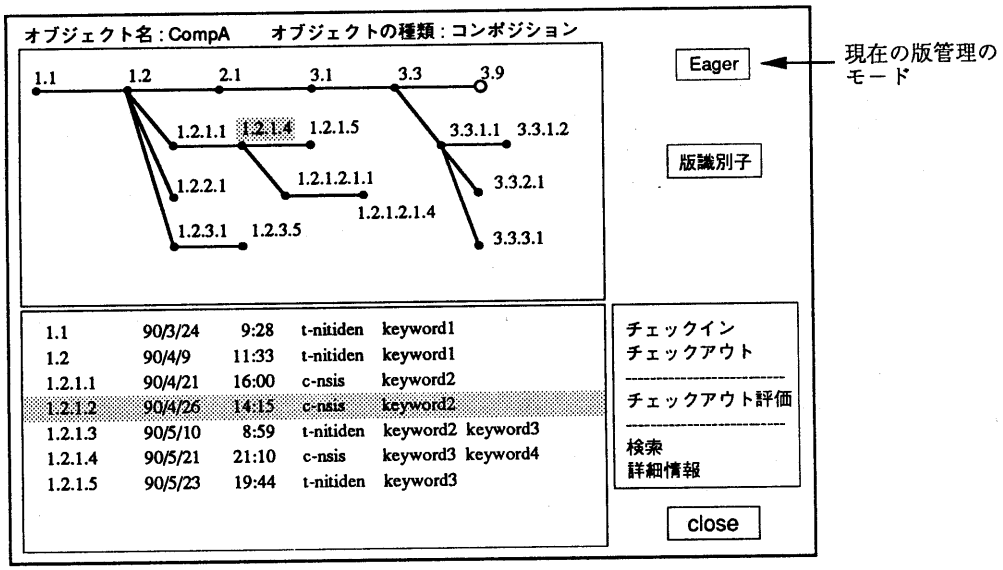


図 5: バージョンマネージャ

- 登録者
- キーワード

がある。上の条件の全てを指定する必要はない。複数の条件が指定された場合には、全ての条件を満たす版が検索結果として得られる。

ユーザは、チェックイン時にキーワードの中からいくつかを選んで、版に付与することができる。ユーザが選ぶことができるキーワードはカスタマイズ情報としてプロジェクトごとに設定することができる。

7　おわりに

本稿では、成果物や設計情報を管理するための機能 LifeLine と、LifeLine を用いて構築した C プログラム開発支援環境 LifeStudio/C について述べた。

現在、実際のソフトウェア開発に LifeStudio/C を適用し、その評価を行なっている段階である。

今後は、この評価を LifeLine へフィードバックし、ソフトウェア開発支援環境の基盤としてどのような機能を用意すればいいのか、さらに検討を行なう予定である。

参考文献

[1] S. I. Feldman, *Make – A Program for Maintaining Computer Programs*. Software – Practice and Experience, Vol.9, No.4, April, 1979.

[2] 岸知二, 入交晃一, 坪谷英昭, CASE 環境構築のためのファイル管理機能. 情処学会 CASE 環境シンポジウム, 1989 年 3 月.

[3] 岸知二, ソフト開発環境における情報管理について. 情処学会 データベースシステム研究会 76-5, 1990 年 3 月.

[4] Charles W. Krueger, *The SMILE Reference Manul.* The GANDALF System Reference Manuals. Carnegie-Mellon University, 1986.

[5] John Nestor, *Toward a Persistent Object Base.* Techinical Report SEI-86-TM-8, SEI, July 1986.

[6] Marc J. Rockhind, *The Source Code Control System.* IEEE Trans. on Software Engineering, SE-1(4), December, 1975.

[7] Walter F. Tichy, *RCS: A Revision Control System.* Integrated Interactive Computing Systems, North-Holland Publishing Company, 1983.