次世代ソフトウェア開発技術
Software Development Environment for Next Decade

平山　雅之 HIRAYAMA Masayuki
樺沢　博 KABASAWA Hiroshi
深谷　哲司 FUKAYA Tetsuji

システムの巨大化・複雑化に伴い，ソフトウェアに対する要求もより高度なものになっている。そのなかで，ソフトウェア開発技術もさまざまなOSやライブラリの充実，新たな開発方法論の提案など，この数年で大きな進歩を遂げている。ここでは，システム・ソフトウェアへの要求の高度化に対応して研究・開発を進めている次世代のソフトウェア開発環境を紹介する。当社の次世代ソフトウェア開発方式U-SDM(U-shape Software Development Method)とその支援環境U-SDE(U-shape Software Development Environment)は，設計情報の共有化とノウハウやIPなどの設計情報の有効活用を核に，ソフトウェア開発効率とその品質の向上を実現するものである。

Higher levels of software functions are required as the size and complexity of systems increase. With this as a background, software development methods have made remarkable progress in such areas as operating systems, software libraries, and new development methodologies.

This paper presents a software development environment for the next decade, which Toshiba has been researching. Our development environment, U-SDE(U-shape Software Development Environment), includes design information sharing and effective use of know-how and intellectual property, with the aim of achieving high development efficiency and high quality of software.

1 まえがき

従来のソフトウェア開発は，ウォーターフォールモデルやスパイラルモデルなどの開発プロセスモデルに準拠し，主な部分の大半をプロジェクト内で開発するソフトウェア工場的開発スタイルを実現してきた。しかし，ソフトウェアの規模や複雑さの増大，システムのオープン化などの影響を受け，従来型の開発スタイルが必ずしも十分に機能しなくなりつつある。

こうした背景の下，当社ではシステム・ソフトウェア開発技術の革新を試みており，次世代ソフトウェア開発方式として，U字型ソフトウェア開発方式(U-SDM)を開発し，一部は社内での利用を開始している。ここではU-SDMの概要について述べる。

大規模・複雑システムの開発では，ソフトウェアでカバーする領域が増大し，その安全性や信頼性の重要度が従来以上に高くなる。したがって，信頼性をソフトウェア開発プロセスの中でどのように作りこみ，保証するかが大きな課題となる。また，システム製品のライフサイクルは概して短くなっており，短納期化に対応したコンカレント開発や周辺知識・ノウハウの効率的利用を目ざすIP(Intellectual Property)，コンポーネントを利用した開発といった手法が採用されている。U-SDMはシステム・ソフトウェアの信頼性ファクタの取込みとコンカレント開発，IP，コンポーネント利用開発などに対応する。

2 次世代ソフトウェア開発の全体像

2.1 U-SDM

U-SDMは次のような特長をもっている。

(1) ソフトウェア開発における知的作業の効率化を実現　複雑システムの開発効率は，熟練者がどのように役割を果たすかで大きな差が生ずる。U-SDMではこれらのベテランが行う作業の効率化を重視している。

(2) 情報共有化への柔軟な対応を実現　設計情報，実装の情報，テストの情報などソフトウェア開発の過程で生まれるさまざまな情報を有機的にリンクして，開発プロジェクト内の情報共有を促進する。

(3) さまざまな開発形態に対応　新規開発だけではなく，保守や流用などの開発形態あるいはソフトウェアIP／コンポーネントなどを利用した開発にも対応できるさまざまなしくみをもたせている。

(4) さまざまな開発プロセスに適用可　フォワード方向，リバース方向の作業を円滑に流すことにより，ウォーターフォールやスパイラルなどの既存のプロセスモデルを含むさまざまな開発プロセスに適用できるしくみをもたせている。

また，U-SDMのプロセスを支援するために次世代ソフトウェア開発環境として，U-SDEを準備している。

2.2 U-SDE

U-SDEはU-SDMに基づいたソフトウェア開発を進め

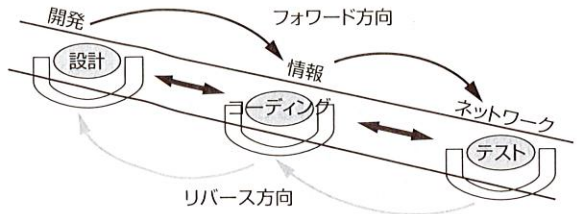


図1．次世代ソフトウェア開発環境U-SDEの概念　設計，実装，テストの各フェーズに対応するワークデスク上に開発者の知的作業を支援するツール群を提供する。
Overview of U-SDE

る環境であり，個々の作業プロセス(例えば，設計プロセス，実装プロセスなど)ごとに，それらを円滑に進めるための開発テクニックや支援ツール群をU-PASE(U-shape Process Activity Software development Environment)として整備している。

図1に示すように，設計プロセスではU-PASE for Design，コーディングプロセスではU-PASE for Coding，テストプロセスではU-PASE for Testingというように，各作業に対するワークデスクとして開発環境が用意されている。各U-PASEの情報は互いに交換可能であり，U-PASE for Designで作成した情報をU-PASE for Codingで参照したり，利用することができる。したがって，ソフトウェア開発段階での情報の共有化と参照性が向上し，さらに個々人のノウハウに相当するレベルの情報が随時相互に利用できるようになる。また，情報の参照や利用の方向はいわゆる開発の上流から下流の向きに相当するフォワード方向とその逆向きのリバース方向があるが，U-SDEでは双方の情報の流れを実現している。

近年の開発ではシステムをまったく新規に開発するケースは概して少なく，ベースとなるシステムの改良や部分利用などにより開発効率の向上を図ることが多い。このような開発では，リバース方向の情報流が特に重視される。

U-SDMではこのような既存システムをベースにした開発を適用指向型ソフトウェア開発(ASD：Adaptable Software Development)と称しており，U-SDEはASD型の開発のサポートもねらっている。各U-PASEはベースとなるフレームワーク部，フロントエンドに相当するビューア部，その中間に個別に提供する各種ツールエンジンを配置する構造をとっており，既存のツールやベンダー提供の開発支援ツールなどが容易にプラグインできるようになっている。これにより開発作業者は独自の開発手法やツールを組み込んで作業できる形がとれる。

3 ソフトウェアIP指向設計

ソフトウェア開発の効率・信頼性向上には，既存のソフトウェアの活用が有効である。近年では市販のライブラリ群も整備され，ソフトウェアに関してもIPの考えかたが急速に広がっている。ここでソフトウェアIPとは，ソフトウェアオブジェクトだけではなく，その周辺知識・ノウハウを含めたものを指す。これらのソフトウェアIPを有効活用するには，ソフトウェアIP利用を前提としたソフトウェアの設計環境が必要である。U-SDEにおけるソフトウェア設計ワークデスクU-PASE for Design(図2)では，設計情報の共有化を実現する開発プラットフォームEdieを提供している。

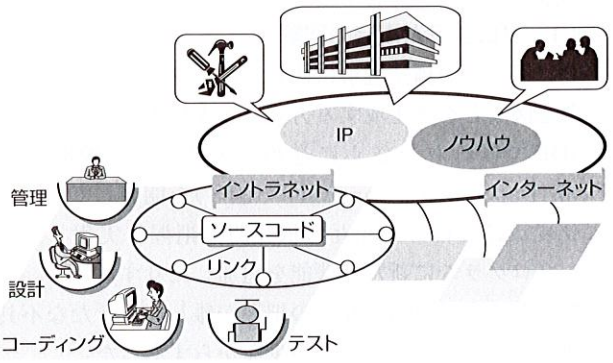


図2．ソフトウェア設計支援環境U-PASE for Design　設計書，ソースコードなどの情報を情報リンク技術によって関係付け情報共有化を加速する。
Design phase support：U-PASE for Design

ソフトウェアIPを活用した設計では，次のような情報を設計情報として残すことが重要である。

(1) どのようなソフトウェアIPを利用したか。
(2) 対象ソフトウェアの仕様レベルの制約をどの部分でどのようなメカニズムで吸収したか。
(3) ソフトウェアIPに関する動作制約や組み込む際の制約はどのようになっているか。

設計情報の一部は個々の設計者のノウハウであり，これらの情報を第三者からも見える形で残し共有化することにより，設計作業効率の数段の向上が期待できる。Edieでは，これら設計情報の共有をXML(eXtensible Markup Language)を応用し，ドキュメントリンクの技術によって実現している。

従来方式でも仕様書や設計書などさまざまなドキュメントを規定してきたが，これらのドキュメントではシステム設計の結果は記載されても，その理由(どうしてそのような構成やアーキテクチャ，アルゴリズムを採用したかなど)が残されず，システム不具合発生時の保守などの際に大きな障害になることが少なくなかった。Edieでは，こうした設計の理由やノウハウなどを簡単な電子ポップアップメモとしてリンクしたり，ドキュメント内の情報を複数ドキ

ュメント間で相互リンクし，情報参照を容易にすることで情報共有化を支援している。

4 適用指向型ソフトウェア実装技術

ソフトウェアIPに代表される既存のソフトウェアを利用する開発においては，利用すべきソフトウェアやそれに対して新たに機能追加するソフトウェアの動作理解の正確さと効率が重要になる。U-SDEでは，既存ソフトウェアの利用を意識したASD技術の中心に再利用対象のプログラムの構造と動作の理解を位置付けており，U-PASE for Designにプログラム理解支援ツールとして実現している。

4.1 プログラム構造の理解

プログラムの構造理解に関しては，プログラムモジュール関連図などを逆生成する方式などが提案されてきたが，U-SDEではこれをさらに進め，プログラム理解ツールDean(Dependency analysis)[1](図3)を開発して，関数および変数に着目した相互の依存関係把握を実現した。従来，プログラムに新たな機能を追加したりする場合，その影響がどこまで及ぶかなどの把握が難しく，新たな不具合を埋め込むといったケースが少なからずあった。

Deanではプログラム スライシング技術を応用して，どの関数がどの関数を呼び出しているか，どの関数がどのグローバル変数の書込み参照をしているか，などを解析し，その結果を関数・グローバル変数グラフの形でユーザーに提供する。関数・グローバル変数グラフは従来の関数コールグラフに変数を介した依存性情報を付加したもので，従来方式で把握が難しかったグローバル変数を介した関数のつながりの把握を可能としている。

4.2 プログラム動作の理解

プログラムの再利用では，そのプログラムがどのような動作をするかを正確に把握することが必須(す)である。それにはプログラムを動作させた際の，ある時点での内部の状態とその変化のようすを知ることが有効であり，従来はデバッガなどを利用して適切な時点にブレークポイントを設定してこれらの情報を入手することが多かった。しかしこの方法では，プログラムの動作の一断片を把握することは可能であっても，プログラム全体の動作を流れとして把握することは難しい。

U-SDEでは前述のプログラム スライシングの結果を利用して，プログラムの進行に添って，着目すべき特定の変数や関数など内部状況をレポートするためのログ出力文を自動挿入するツールDean-DataWatcherを提供している。従来方式においてもプログラムの内部状態を把握するために，print文などのプローブを内部に挿入する手法はとられてきたが，print文を挿入する場所の特定が難しく，またログ出力文を数多く入れるとプログラムの実行タイミングや速度に影響が現れる問題があった。Dean-DataWatcherでは特定の変数や関数に影響する範囲に絞り込んだ内部状態の把握を可能にするものである。Dean-DataWatcherにより得られたプログラム動作の内部状態は，専用のログビューワによって時系列ごとに可視化され，ユーザーのプログラム動作理解を支援する。

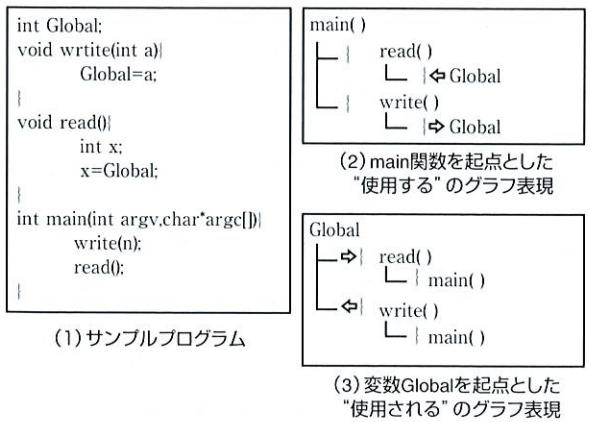


図3．プログラム理解支援ツール Dean　プログラム内の関数，変数間の依存関係を解析し，その情報をユーザーに提示する。
Program dependency analysis : Dean

5 システムシミュレーションによるテスト方式

近年のシステム開発では，開発期間の短縮に向けて随所でコンカレントな開発スタイルが採用されている。しかしながら，システムテストを行う時点で関連するすべてのパートが完成していることはまれであり，この結果コンカレント開発の効率が十分発揮できないケースも少なくない。

U-SDMでは，この課題を解決するためにシステムシミュレーションを応用したテスト技術を開発し導入している。ここでは，マイコンファームウェアを事例に，U-SDEにおけるシミュレーション テスト支援技術を紹介する。

5.1 システムテスト工程の分割

マイコンシステムの多くはファームウェア，ハードウェアデバイスなどの開発がコンカレントに進められる。ファームウェアの動作確認では，実機やICE(In Circuit Emulator)などが必要であるが，コンカレント開発ではファームウェア動作確認の時期に実機ハードウェアが準備できないことが多い。U-PASE for Testingでは，ファームウェア テスト デバッグフェーズを以下の三つのサブフェーズに分割し，これをスパイラルに回すことでテストフェーズ全体の効率向上をねらっている。

(1) 論理テストフェーズ　システムモジュールの内部状態が入出力に対して仕様どおりに変化するかどうかなど，論理面の確認・検証を主とする。

(2) 状態遷移テストフェーズ　マイコンファームウェアの多くは外部からのイベントに対応した制御動作を扱う。状態遷移テストフェーズはモジュール間インタフェースが仕様どおりであることを確認し，システムの内部状態が外部イベントにより仕様通りに変更されているかを確認する。

(3) タイミングテストフェーズ　内部状態の変更，状態遷移，モジュール間インタフェース動作が仕様どおりのタイミングで処理されるかを確認・検証する。

このようにテストフェーズを分割した結果，論理動作テストフェーズと状態遷移テストフェーズは，基本的にハードウェアとの依存性が低く，ハードウェアを用いた実機環境を用意しなくてもファームウェア単体でテストできる。

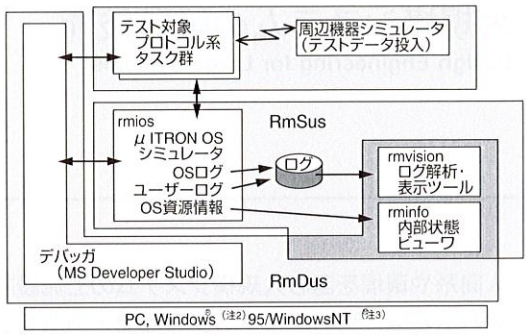
5.2 RTOSシミュレーションによるテスト技術

U-SDEでは，マイコンファームウェアがリアルタイムOS（RTOS：Real Time OS）上に構築されることに着目し，OSシミュレーション方式によるファームウェアテスト環境RmXUS(注1)(2)を提供している。RmXUSは通常マイコン上に搭載されるRTOSの動作を模擬実行するRTOSシミュレータをパソコン(PC)上に実装し，本来マイコン環境でしか動作しないマイコンファームウェアをPC上のRTOSシミュレータ上で動作させて，周辺の実機ハードウェアを準備しない環境でのテストを可能とするものである。前述の論理動作テストおよび状態遷移テストフェーズの多くのテストは，このRmXUS環境を導入することで早期に着手が可能となる。

図4は当社の携帯情報端末開発にRTOS対応テスト環境を適用した際の概要を簡略に示したものである。このケースでは，携帯端末のプロトコル部の動作試験をRmXUSで提供するRTOSシミュレータ上で行なっている。携帯情報端末のユーザーインタフェース(UI)部は同一PC上のUIシミュレータで代用し，基地局側は基地局シミュレータで模擬実行することで，周辺ハードウェアや機器がすべてそろわなくてもシミュレーション環境下で実際のプロトコルの論理動作および状態遷移テストを実施することが可能となった。図4に示した事例では，RmXUSによるRTOSシミュレーション方式テストを導入したことにより，プロトコル単体が完成した時点から随時システムとしての論理動作面のテストをシステム開発と並行して行うことが可能となった。また，ハードウェアとの結合以前に論理面，状態遷移面での不具合を除去することにより，ハードウェア結合段階でのテストのスリム化や不具合による後戻りが大幅に削減される効果も確認された。

(注1) RmXUS：Reactive & multi-task program testing eXecution environment by Using logical Senarios シミュレーションやシナリオを利用したリアクティブシステム向けのテスト環境の全体名。

(注2)，(注3) WindowsおよびWindowsNTは，Microsoft社の商標。



μITRON：μIndustrial the Real-time Operating system Nucleus
RmSus，RmDus：RmXUSの構成ツールの一つで，テスト実行エンジン，デバック支援ツール。

図4．リアクティブシステムテスト環境RmXUS　PC上にRTOSシミュレータを実現し，マイコンファームウェアを実機レスにテストする。
Reactive system test tool : RmXUS

6 あとがき

ここでは，当社の次世代ソフトウェア開発手法U-SDMおよびその支援環境U-SDEについて概要を紹介した。ソフトウェアの世界はこの数年飛躍的な進歩と革新を遂げてきており，ソフトウェアを提供するメーカーとして当社もこれらの変革に迅速に対応することが求められている。

U-SDM，U-SDEはこのような背景の下で実用化を目ざしている。この一部はすでに実際の開発フィールドでの適用が開始されており，当社ソフトウェアの品質向上と開発期間短縮に大きな成果を上げつつある。今後も引き続き研究・開発を継続し，顧客へのフィードバックを強化していきたい。

文　献

(1) 小山徳章，他．OSシミュレーションによる李アクティブソフトウェアのテスト工程の改善，情報処理学会SW研究会　120-14, 1998, p 93-100.
(2) 本間昭次，他．“プログラムの構造と動作の理解支援”．情報処理学会全国大会，1999.

平山　雅之　HIRAYAMA Masayuki
研究開発センター　システム技術ラボラトリー研究主務。ソフトウェア設計・検証テスト技術の研究に従事。情報処理学会会員。
System Engineering Lab.

椛沢　博　KABASAWA Hiroshi
電力システム社 府中電力システム工場 エネルギー共通技術システム部主査。ソフトウェア設計支援ツール，生産管理支援ツールの開発に従事。
Fuchu Operations-Power Systems

深谷　哲司　FUKAYA Tetsuji
研究開発センター　システム技術ラボラトリー研究主務。ソフトウェア開発環境の研究・開発に従事。情報処理学会会員。
System Engineering Lab.