ソフトウェア工学 125－10
（２０００．１．１４）

プラントソフトウエアの要求定義用ソフトウエア部品の開発

高橋 正和　津田 和彦

筑波大学大学院 企業科学専攻

あらまし

本論文では、プラントの動作制御や運転支援を行うソフトウエア（PS: Plant Software）の開発を目的とした「ソフトウエア部品ベースのPS要求定義支援システム（PRDS: Plant software Requirements Definition System）」を提案する。PRDSはPS開発用のソフトウエア部品（PSC:Plant Software Components）、要求定義支援ツール（RDT: Requirements Definition Tool）、動作確認用シミュレータ（MRS: Motion Review Simulator）から構成される。PRDSを用いたPS要求定義プロセスは、はじめにRDTを用いて要求定義を行い、それをパラメータとしてPSCに与えて目的PSの要求を特化する。次に、MRSを用いてPSの動作確認を行い、要求の妥当性を確認する。さらに、要求が妥当となるまで、パラメータ修正を繰り返し、PSに対する要求を洗練していく。提案手法の性能を確認するため幾つかの事例に対してPRDSを用いたPSの要求定義を行った。その結果、従来の一般的な開発手法であるウォーターフォール型PS開発プロセスと比較して開発時間が30～37[%]に短縮できることを確認した。

The development of software components for requirements definition of plant software

Masakasu Takahashi　Kazuhiko Tsuda

Graduate school of systems management, the University of Tsukuba, Tokyo

Abstruct.

This paper proposed the software component based Plant Software (below PS) requirements definition support system (PRDS: Plant software Requirements Definition System) that aimed at the development of the PS that is used for motion control and operation support of the plant. PRDS is composed by the software components to build PS (PSC : Plant Software Components), the support tool to define the PS requirement (RDT: Requirements Definition Tool), and Simulator that confirms the PS's motion (MRS: Motion Review Simulator). In the PS requirements definition process that PRDS is apply for following. At first steps, the requirements of PS is defined by using RDT. At second steps, those are given to PSC as a parameter. At the third steps, a target PS is specified. Next, the motion of PS is confirmed by using MRS, and the validity of the requirement is confirmed through that. Furthermore, until a suitable requirements can be defined, a parameter is revised many times, and a requirements of PS are refined. We simulated some cases of PS's requirements definition of PS using PDRS. Consequently the reduction of development time that become 30 - 37 [%] in comparison with usual water-fall type PS development process was confirmed.

1. はじめに

化学薬品の製造や材料の加工を行う機械設備をプラントと呼ぶ。プラントは、顧客ニーズの多様化や材料加工の精密化に伴って、機能や性能が高度化してきた。そのため、手動運転が困難となり、コンピュータシステムによる動作制御や操作支援が必要となっている。プラントの動作制御や操作支援を行うソフトウエアをプラントソフトウエア(以下、PS: Plant Software)と呼ぶ。

通常、プラントはその目的に特化したものであるため、仕様は個々のプラント毎に異なっている。プラントの開発は顧客と製作会社の間で仕様を決定した後に実施されるため、PS開発には一般的にウォーターフォール型の開発プロセスが採用されている。この開発プロセスは各プロセスの完了時点で審査会やテストを実施し、不具合があれば、その場で修正する。つまり、原則として後戻りの修正は生じない。しかし近年、プラントの複雑化・精密化に伴い、下記の問題が生じている。

1) プラントの運用方法の決定が遅れるため、PS開発の開始時点でPSへの要求が明確化できない。
2) プラント本体の設計仕様が不確定なため、変更が生じ、PSへの要求の変更が生ずる。
3) プラント設計者のPSへの要求が十分検討されていないため、不整合や漏れがある。
4) PS開発者の技術レベルがまちまちなため、不適切な設計が行われたり、検討項目に抜けが生じる。
5) 顧客の要望によりPS完成間際に機能追加がされる。そのため、開発したPSと追加したPSの間で矛盾が生じる。

この結果、適合性や信頼性の低下、期間の長期化、コストの増加等の問題が発生している。この原因は、近年のプラント開発が従来のウォーターフォール型に適合しない状況になっているためである。

本論文では、このような開発プロセスに対応する手法としてソフトウエア部品(以下、部品)を用いてPSに対する要求を効率良く定義する手法を提案する。

2. 要求定義のアプローチ

要求定義を厳密にかつ効率良く行う手法として、一般的に①CASE手法、②プロタイプ手法、③代数仕様手法、④ドメインモデル手法が知られている。以下にこれらの手法の概要を示す。

2.1 CASE手法

CASE[1]を用いるには、構造化手法やオブジェクト指向等の開発方法論が不可欠である。CASEは、全領域のソフトウエアに対して適用できる利点がある。一方、方法論の習得が困難、決定する項目や作業量が多いという欠点がある。

2.2 プロトタイプ手法

プロトタイプ[2]を作成するには、一般に4GL(4th Generation Language)等が必要となる。プロトタイプは、その挙動が可視化されるため、要求の理解が容易という利点がある。一方、プロトタイプ作成が開発のオーバーヘッドになる、指摘された項目を要求定義に確実に反映することが難しい等の欠点がある。

2.3 代数仕様手法

代数仕様[3]を使用するには、Z言語等が必要となる。代数仕様は、対象となる問題を厳密に定義でき、定義した内容を機械的に検証できるという利点がある。一方、高度な数学の知識が必要となる、代数仕様から目的ソフトウエアの挙動をイメージすることが困難である等の欠点がある。

2.4 ドメインモデル手法

ドメインモデル[4]を使用するには、事前に対象システムの開発に有効な、用語、業務の特徴、開発文書およびシステム構造等を明確化し、整理することが必要となる。ドメインモデルは、繰り返し利用が可能であり、事前に整理した有効な情報を適用することで開発効率が向上する利点がある。一方、その作成や理解に時間を要する等の欠点がある。

これらの手法は、それぞれ一長一短があり、顧客や開発者が容易に使用することが困難であった。特に、カスタマイズの容易さと、挙動のイメージアップを両立することが困難であった。ところで、PSはドメインモデルを作成した結果、類似した構造や機能を有していることが判明した。そこで、それらを標準的な部品として準備し、カスタマイズすることで目的ソフトウエアのプロトタイプを作成し、その挙動から要求を確認して定義する手法を提案する。要求定義段階では、ソフトウエアの構造や機能を明確にすることが重要であり、詳細なカスタマイズは必要としない。そこで、カスタマイズの範囲を限定する代わりに、カスタマイズの容易性と実行結果の可視性を重視した部品を開発し、要求定義に適用する。

3. PSCを用いたPS開発方法

PS用部品(以下、PSC:Plant Software Components)は、3.1～3.4に示す一般的な部品作成手順[5]で開発した。

3.1 PSドメインの定義

本論文で取り扱う代表的なプラントを表1に示す。

表1に示すプラントのPSに関係するハードウエアやソフトウエアの設計文書を検討した結果、以下の特徴が得られた。以降では、次の特徴を持ちプラントの動作制御や運転支援を行うソフトウエアをPSとして再定義する。

1) PSの構成に関する特徴
・リアルタイム性(応答時間、処理時間)が要求される。
・一定の時間間隔(周期)で繰り返し、同じ処理をする。
・計測データや外部からの動作指示により処理を行う。

・PSはプラントの制御装置に搭載される(組込システム)。
・PS構成はプラント機器構成に応じて変化する。

2) PSの機能に関する特徴

・「機能の実行管理」、「機器の動作制御」、「機器とのデータ入出力」、「外部とのインターフェース」、「状態表示」、「異常時の処理」をする機能を持つ。
・「機能の実行管理」は、PSの持つタスクの実行を管理する。
・「機器の動作制御」はプロセス制御とフィードバック制御に大別される。
・「機器とのデータ入出力」は、アナログとデジタルの入出力に大別される。入力はセンサーからのデータ計測、出力は機器への動作指示をする。
・「外部とのインターフェース」は、動作指示用の制御盤やネットワークへの接続に大別される。
・「状態表示」は、画面、ペンレコーダー等に計測したデータを表示する。
・「異常時の処理」は、機能縮退、冗長系の切替、緊急停止等がある。

表1. 代表的なプラント

プラントの名称	プラントソフトウエアのタイプ	代表的な機能
航空エンジン	動作制御	・エンジン出力のフィードバック制御 ・中央制御コンピュータとのデータ交換
発電用ガスタービン	動作制御	・ガスタービン出力のフィードバック制御 ・自動機器点検と故障発生時の安全化
	状態監視	・データ収集、表示および蓄積 ・故障診断と故障対策ガイダンス
貯蔵タンク(液化天然ガス、石油等)	動作制御	・貯蔵物の流量のフィードバック制御 ・設備の準備や運転のためのプロセス制御 ・操作室からの動作指示の受信やデータの表示
物流倉庫	状態監視	・入出庫の指示 ・の表示と入庫指示 ・操作室からの動作指示の受信

3.2 PSドメインモデルの作成

PS はプラントの動作制御や運転支援をすることが目的であるため、PSドメインモデルを作成するには、対象となるプラントの機器とPSの両方について検討する必要がある。[6]

はじめにプラントの機器について検討する。プラントの機器の動作制御を行うには、機器の種類や個数に応じて PS の構成を変更する必要がある。しかし、プラントを構成する機器の種類は、数百種類に及ぶ上、次々に新しい機器が開発されるため、全機器をリストアップすることは不可能である。しかし、本論文の目的は PS の要求定義であるため、機器の詳細な動作制御の情報は必要としない。従って、PS 要求定義が可能な程度まで機器を抽象化して最小限の機器の種類でプラントの構成を記述できるようにする。必要最小限の機器を明確にするため、主要プラントの機器を表2において比較する。 その結果、機器が取り扱う信号に着目することで、図1に示す3つのグループに分類できた。

次に PS の機能構成について検討する。代表的な PS の機能構成を表3に示す。PS は個々の機器の動作制御をする必要があるため、機器の種類や個数の影響を受けるが、必要な機能は共通であることが分かる。そこで、プラント機器が3種類に分類できる性質を用い、入出力、動作制御、外部とのインターフェースの各機能を抽象化し、共通化する。

表2. 代表的なプラントの機器の構成(一部抜粋)

プラントを構成する機器の名称 / プラントの名称	アナログ値を入力出力する機器		…	オン/オフ信号を入力出力する機器		…	コマンドやデータの授受をする機器		…
	温度勾配炉	流量調節弁	…	電磁弁	各種電源	…	操作盤	データ表示装置	…
宇宙ステーション搭載曝露環境基盤技術実験装置	○		…	○	○	…		○	…
宇宙ステーション搭載与圧環境材利用実験装置	○		…	○	○	…		○	…
航空エンジン制御装置		○	…	○	○	…			…
発電用タービン制御装置		○	…	○	○	…	○		…
貯蔵用プラント(タンク)加圧制御装置		○	…	○	○	…	○		…
貯蔵用プラント流量調節装置		○	…	○	○	…	○		…

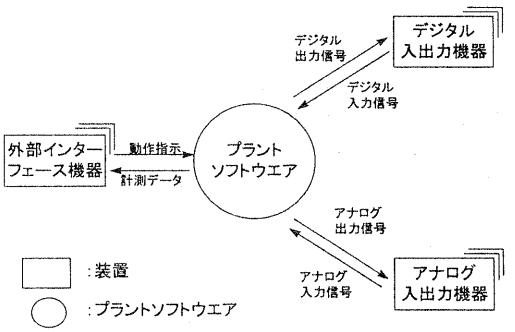

図1. 標準化された機器を用いたプラントの記述

表3. PSの機能構成(一部抜粋)

機能の名称 / プラントの名称	タスクの実行管理	プロセスの実行管理	機器に出力する操作量を計算する			機器の対して動作指示や計測信号の入出力をする			…
	タスク管理	プロセス管理	フィードバック制御	アナログ制御	デジタル制御	アナログ出力	デジタル出力	アナログ入力	
宇宙ステーション搭載曝露環境基盤技術実験装置	○	○		○	○	○	○	○	…
宇宙ステーション搭載与圧環境材利用実験装置	○	○	○	○	○	○	○	○	…
航空エンジン制御装置	○		○	○	○	○	○	○	…
発電用タービン制御装置	○	○	○	○	○	○	○	○	…
貯蔵用プラント(タンク)加圧制御装置	○		○	○	○	○	○	○	…
貯蔵用プラント流量調節装置	○		○	○	○	○	○	○	…

以上の結果、PSの機能を実行管理、動作制御、入出力、外部機器とのインターフェース、状態表示、異常処理の6種類に大別した。以下にそれらを記述する。

1)実行管理機能

実行管理は、PS の各機能を適切な起動方法、順序および周期で起動する機能である。起動方法は、周期的な繰り返し起動(以下、周期起動)と割込による起動(以下、割込起動)に大別される。周期起動する機能は、動作制御機能、入出力機能、外部表示機能および異常処理の一部(定常点検)である。割込起動する機能は、外部機器とのインターフェースおよび異常処理の一部(緊急停止)である。

2)動作制御機能

動作制御は、決められた手順で機器の操作要求をする

(プロセス制御)、プラントの状態を目標状態に近づけるための出力計算をする(フィードバック制御)機能である。

プロセス制御は、使用目的がプラント毎に異なるため、機器の操作順序(プロセス)は、類似点がない。しかし、ある動作が完了するまで次の動作を行わないというプロセス管理の方法は類似している。通常、プロセスは状態遷移図やインターロックブロックダイヤグラムを使って記述される。

フィードバック制御は、機器毎に最適な制御をするため、詳細な制御ロジックには類似点がない。しかし、本論文の対象となる要求定義段階では詳細な制御ロジックは必要とならない。従って、入力データ(制御量)と制御目標から出力データ(操作量)を計算する基本的な機能は同様となる。

3)入出力機能

入出力は、機器に対する出力や、センサーからデータを入力する機能である。プラントを構成する機器は100種類以上ある。しかし、それらは、前述の議論によりアナログ入出力機器とデジタル入出力機器に大別される。これらの機器には、データ入出力のタイミングや操作方法の違いがあるが、要求定義を行う際には重要でない。従って、アナログおよびデジタルの信号を入出力する基本的な機能は同様となる。

4)外部機器とのインターフェース機能

外部機器とのインターフェースは、制御盤やネットワーク経由で接続された制御室等との間で動作指示を授受する機能である。 プラントで用いられるインターフェースにはRS-422、GPIB、MIL-STD1553B、構内LAN等がある。通信プロトコルは、送信する動作指示の頻度や通信データ量 等により、規格で定められたものが採用される。従って、外部機器とのインターフェースには類似点が少ない。しかし、前述の議論により、要求定義段階ではインターフェースや通信プロトコルの正確な模擬は必要とならない。

5)状態表示機能

状態表示は、機器から入力したデータを端末等に表示する機能である。状態表示の方法としては、数値データ、トレンドグラフ、メータ表示等の方法がある。状態表示の方法やレイアウトは顧客の要求や表示画面の大きさにより決定されるため、類似点は少ない。

6)異常処理機能

異常処理は、機器から入力したデータを用いて定常点検、機能縮退および緊急停止等をする機能である。定常点検の内容は、計測データが安全基準内にあるか確認し、故障コードを出力することである。点検の方法は類似しているが、入力データの種類、安全基準および故障コードについては類似点はない

機能縮退は故障した機器を使用しないでプラントを稼働させることである。これは機器の構成に影響をうけるためプラント間で類似点はない。

緊急停止は機器の安全化を行い、プラントを停止させることである。これは機器の構成に影響をうけるためプラント間で類似点はない。

3.3 要求定義用ソフトウエア部品

PS ドメインモデルを参考にして要求定義用部品(以下、PSC: Plant Software Components)を作成する。対象をPS要求定義に限定することで部品の数や機能を限定し、容易にカスタマイズができる様に配慮する。PSドメインモデルを作成した結果、プラント間で類似性を持つ機能と多様性を持つ機能があることが分かった。類似性を持つ機能は、事前に部品を準備して使用する。多様性を持つ機能は、機能レベルで共通なものと機能自体が異なるものに大別し対応する。

機能レベルで共通なものは、部品をカスタマイズして使用する。カスタマイズの方式には、ソースコードの修正、クラスの継承関係を利用した差分修正等があるが、今回は容易にカスタマイズができるという観点からパラメータ方式を採用する。パラメータ方式は、カスタマイズに必要な固有情報をパラメータとして部品に与え、使用する方式(以降、パラメータ部品)であり、一般にソースコード修正が不必要である。

機能レベルで固有なものは、事前に部品を準備できないため新規に作成する。しかし適用事例が増加し、部品群が充実すれば、新規に作成する部品は減少していくと考えられる。けれども、これらの部位は入出力、外部機器とのインターフェース、状態表示等であり、PS の要求を定義するには必須な部位ではない。そのため、例えば簡単なスタブモジュールを作成し、代用することが可能である。

部品作成方針に則って作成した PSC の構成を図2に示す。図2中の実線四角形は無修正で使用する部品、点線四角形はパラメータ部品、一点鎖線四角形は、共通化ができないため新規に作成する部品を表す。

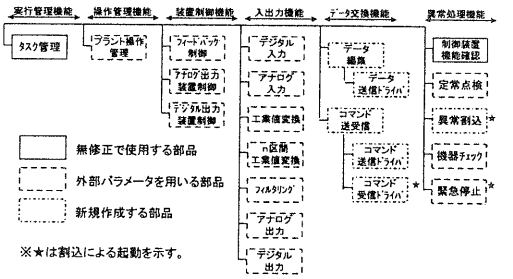

図2. PSC の機能構成

ところで PSC は一定の周期で起動されるが、その起動順序は、応答時間や計算精度等に影響を与える。これらを満足する起動順序を決定するため、PS の代表的な状態(アイドル、通常、動作指示受信、異常、緊急停止の各状態)において、PSC 起動のシーケンス図を記述し、比較した。その結果、PSC 起動順序を図3の様に決定した。ところで、PSC 起動周期はプラントの性能要求から決定されるため、事前

に特定することはできない。しかし、周期が変更されても、PSC 間の情報授受の依存関係は変わらないため、図3のPSC起動順序は、そのまま使用することが可能である。

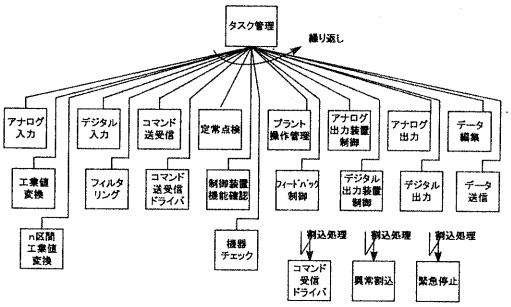

図3. PSC の起動順序

3.4 要求定義支援ツール試作

本論文で提案する要求定義プロセスを図4に示す。提案する要求定義プロセスでは、PS プロトタイプを稼働させ、挙動を確認する。不具合が確認されれば要求定義した内容へのフィードバックをする。この作業を繰り返すことで、顧客やプラント設計者のイメージに合致した要求へと進化させていく。この方法を実現するための要件を以下に示す。

・要求定義が容易にできること。
・定義した要求からプロトタイプが容易に作成できること。
・プロトタイプの挙動を可視化できること。

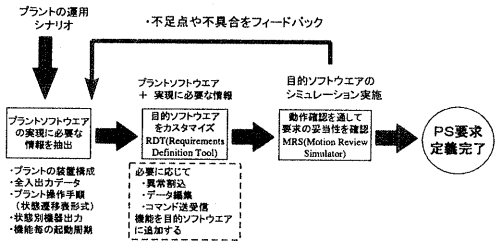

図4. PSC ベースの PS の要求定義

1)　要求定義支援ツール(RDT: Requirements Definition Tool)

図4の RDT は、PS を導出するために必要な機器構成、プロセス制御、起動周期、その他の固有情報（要求定義内容）を定義するためのツールである。RDT は、入力した要求の確認を容易にするため、図表形式による入力を採用した。例として図5にプロセス制御の手順の定義画面を示す。

さらに、PS 開発効率を向上させるため、RDT で定義した情報を用いて要求仕様書を自動的に作成する仕組みを作成した。これは定義した内容を事前に準備した要求定義書のテンプレートに転記することで要求定義書を作成する。

2) 動作確認用シミュレータ(MRS: Motion Review Simulator)

図4の MRS はプロトタイプをコンピュータ上で稼働させ、要求定義の内容確認や潜在要求の掘り起こしをするシミュレーション・ツールである(図6)。

一般にプラントの機器の構成は個々に異なるためデータ表示画面を事前に準備することは難しい。しかし、これらは市販のツールを適用することで、容易に作成できるため、事前準備できないことは問題とならない。ところで、画面作成の際には、計測データとしてプラントの内部変数が必要となるため、プロトタイプの内部データを取得する関数を準備し、画面作成ツールから利用できるようにした。これにより、MRS と画面表示ツールの独立性が増し、要求に応じてデータ表示画面をカスタマイズすることが可能となった。

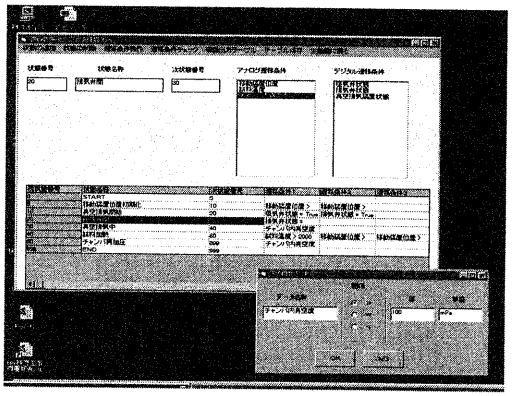

図5. RDT の画面　－プロセス制御手順の入力－

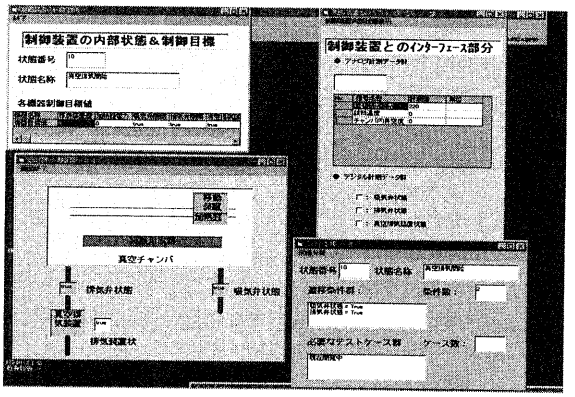

図6. MRS の画面

4. PSC の適用と評価

4.1 適用事例

5つの事例に対し、提案する要求定義プロセス適用した。

はじめに PSC の利用結果を表4に示す。ここで、適用 PSC とは動作制御、異常処理等の共通化できない PSC を指す。

次に作成した PSC のコード再利用率を表5に示す。なお、表5で使用する再利用率の定義は次のとおりである。[7]

再利用率 ＝ (再利用したコード数)
／(再利用したコード数＋新規作成コード数)

最後に、開発時間に対する実験結果を表6に示す。表6で使用する開発時間の定義は次の通りである。

開発時間 ＝ 要求分析から要求定義書を作成するまでに要した時間

開発時間は、従来の開発プロセスを適用した場合の開発時間(同じまたは類似のPSの開発実績)と比較した。

要求定義プロセス適用例として、事例Aの機器構成(図7)、PSの構成(図8)、プロセス制御の概要(図9)に示す。

表4. PSCの利用結果

	総PSC数	適用PSC数	状態遷移数
事例A	21	13	25
事例B	13	5	14
事例C	16	8	6
事例D	12	4	8
事例E	19	11	20

表5. 再利用率に関する結果サマリー

	再利用コード数	新規作成コード数	固有パラメータの行数	再利用率[%]
事例A	425	32	105	92.9
事例B	298	27	82	91.7
事例C	345	34	30	91.0
事例D	278	25	45	91.7
事例E	408	38	82	91.5

表6. 開発時間に関する結果サマリー

	本プロセス[時間]	従来プロセス[時間]	本プロセス／従来プロセス[%]
事例A	42	134	31
事例B	28	76	37
事例C	19	56	34
事例D	22	61	36
事例E	35	118	30

4.2 総合評価

1)PSCの再利用率に関する評価

PSCの再利用率について再利用コード数のばらつき、新規作成コード数のばらつき、再利用率、固有パラメータ数のばらつきの4つの視点から検討と評価をする。

ア. 再利用コード数のばらつきの検討

はじめに、再利用コード数を算出する論理式について検討する。通常PSは、基本PSと適用PSから構成されている。ここで、基本PSCとは、各事例で共通に使用されているタスク管理、プラント操作管理、入出力関連、データ編集関連、定常点検の8個のPSCを指す。従って、再利用コード数は式(1)で記述できる。

再利用コード数＝ 基本PSCの総コード数(定数) ＋ 適用PSCの総コード数 (1)

基本PSCのコード数を集計した結果、総コード数は201であった。次に適用PSCの総コード数について検討する。適用PSCは主にプラントの入出力を模擬するPSCであるため、要求定義段階では詳細な機能を実現する必要がない。

従って、プロトタイプを作成する場合には入出力と簡単な応答を模擬する適用PSC(一般にはスタブと呼ばれる)を使用する。適用PSCは、サブルーチン定義、変数宣言、入力データ変換、出力データ変換、応答計算の部分から構成される。過去の事例を調査した結果、適用PSCはサブルーチン定義(平均2行)、変数宣言(平均3行)、入力データ変換

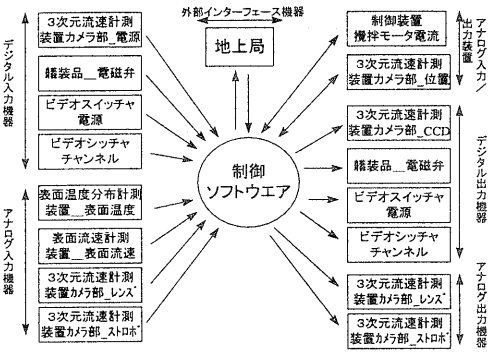

図7. 事例Aの機器の構成

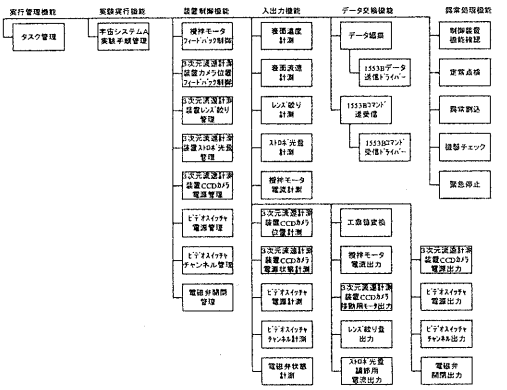

図8. 事例AのPS構成

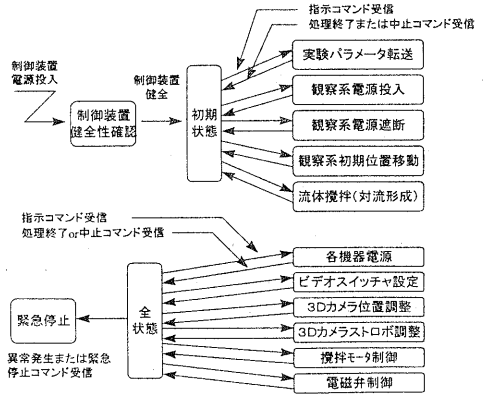

図9. 事例Aのプロセス制御の概要

(平均 4 行)、出力データ変換(平均 4.2 行)、応答計算(平均 7.2 行)から構成されており、平均 20.4 行であった。また、どの適用 PSC も機能やコード数に大差がない。そのため、適用 PSC の総コード数は使用した適用 PSC の個数に比例する。従って再利用コード数は理論式(2)で記述できる。

再利用コード数＝201＋20.4×適用 PSC の数　　(2)

次に適用事例の結果から、論理式(2)の妥当性を検討する。適用事例 A～E の再利用コード数と使用した適用 PSC の個数との関係を図8に示す。この結果を1次式で近似すると実験式(3)が得られる。

再利用コード数＝205.4＋17.9×適用 PSC の数　(3)

論理式(2)と実験式(3)を比較するとＹ切片、傾きともに誤差10[%]程度であり、良く一致している。従って、再利用コード数は論理式(2)で記述することができる。

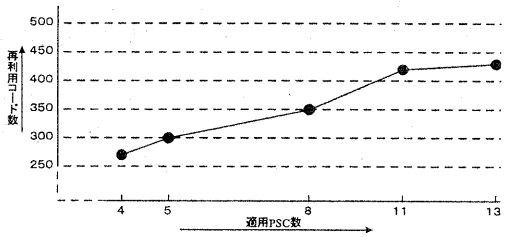

図10.　再利用コード数と適用 PSC の個数の関係

イ. 新規作成コード数の検討

事例 A～E では、プロセス制御の遷移条件に追加があり、PSC を修正した。一般に修正コード数については個別の要求に応じたカスタマイズとなるため、新規コード数を予測することは難しい。しかし、経験上、要求定義段階では30行程度の修正が生ずることが分かっている。主な修正は、遷移条件の追加、遷移の分岐追加等である。なお、今回の検討では、新規機能の PSC を追加するケースは範囲外とする。

ウ. 再利用率の検討

再利用率について検討する。再利用コード数は式(3)で記述できるため、再利用率は論理式(4)で記述できる。

再利用率 ＝　(201＋20.4×適用 PSC 数)
／(201＋20.4×適用 PSC 数＋30)　(4)

式(4)を用いて事例A～Eの再利用率を計算すると、94.0[%]、91.0[%]、92.4[%]、90.4[%]、93.4[%]であった。一方、実際の再利用率は、92.9[%]、91.7[%]、91.0[%]、91.7[%]、91.5[%]であり、論理式(4)との誤差 10[%]は以内となり、良く一致している。従って、再利用率は論理式(4)で計算することができる。

エ. 固有パラメータの検討

固有パラメータ数を算出する論理式について検討する。固有パラメータは、パラメータ部品をカスタマイズするために必要な機器の構成とプロセス制御の手順を記述するものである。機器の構成では機器の種類、入力データ、出力データを 1 行で定義する。過去の事例を調査した結果、プラント毎の機器個数の平均は 10.5 であった。次に、プロセス制御の手順を記述するには、全てのプロセスに対して「次のプロセス」、「遷移条件」、「その際のプラント機器への出力」を定義する。今回開発した PSC では「次のプロセス」、「遷移条件」、「プラント機器への出力」をそれぞれ1行で記述するため、遷移当たりの記述行数は3行となる。従って、固有パラメータ数は遷移の数に比例する。上記の検討の結果、固有パラメータは論理式(5)で記述することができる。

固有パラメータ数 ＝ 10.5 ＋ 3 × 遷移の数　(5)

次に実験の結果から、論理式(5)の妥当性を確認する。適用事例 A～E の固有パラメータ数と遷移の数との関係を図9に示す。この結果より式(6)を得る。

固有パラメータ数 ＝ 12.0＋3.68 × 遷移の数　(6)

論理式(5)と実験式(6)を比較すると Y 切片、傾きは誤差20[%]であり、比較的一致している。従って、固有パラメータ数は論理式(6)で記述することができる。

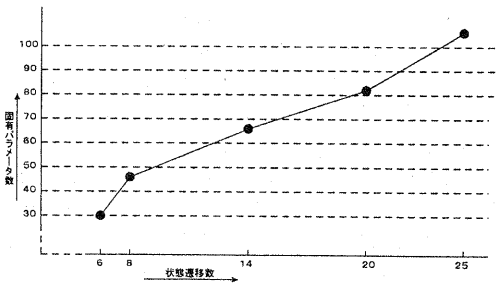

図11. 固有パラメータ数と遷移数の関係

PS の総コード数は、再利用コード数、新規作成コード数、固有パラメータ数の総和で求められるため、今までの議論により論理式(7)で記述できる。

総コード数 ＝213.0 ＋ 20.4 × 適用 PSC 数
＋3 × 遷移の数　(7)

2)PS の開発時間に関する評価

はじめに提案する要求定義プロセスにより、PS 開発時間が削減される理由について検討する。一般に、要求定義作業は PS 要求のリストアップ、PS 要求の妥当性確認、PS 要求定義書の作成に大別される。

PS 要求のリストアップでは、プラントの機器構成、PSC の起動周期、プロセス制御等の明確化をする。従来の要求定義プロセスでは、経験の少ない開発者は何を要求定義すれば良いか判断できず、試行錯誤するため時間を要していた。本プロセスで定義すべき項目を明確化したことで、試行錯誤が減少し、PS 開発時間が短縮した。ところでリストアップ項目の中でプラント機器の構成と PSC の起動周期は、設計

情報から特定できる上、ほとんど変更がない。従って、PS 開発時間差は、プロセス制御の複雑さにより生ずる。プロセス制御の複雑さは、図9からも明らかなように、遷移の数に比例する。従って、PS 要求のリストアップ時間は遷移の数に比例する。

PS 要求の妥当性確認では、定義した PS 要求をウォークスルーやプロトタイプにより確認する。従来プロセスでは、文書ベースで確認していたため、チェックもれが生じ、その結果、後戻りが生じ、時間を要していた。本プロセスにより視覚的に PS の挙動確認が可能となり、チェックもれが減少した。プロセス制御の確認に要する時間は遷移の数に比例するため、PS 要求の妥当性確認時間は遷移の数に比例する。

PS 要求定義書の作成では、要求から要求定義書を作成する。本手法により、リストアップする要求が状態遷移表に形式化されたため、要求仕様書が容易に作成可能となった。この作業は RDT により、機械的に実施されるため、開発者が実施しなければならない作業はない。

以上により、PS 開発時間は、プロセス制御の状態遷移の数に比例するため、論理式⑻で記述できる。

PS 開発時間=プラントの機器構成定義の時間
+PSC 起動周期の定義に要する時間
+PS 要求のリストアップと確認の単位時間×遷移の数⑻

上記の定数が明らかになれば、論理式⑻を用いて PS 開発時間を予測できる。ところで、上記の定数の中で PS 要求のリストアップと確認の単位時間は、個人の能力に依存しており様々である。よって、本論文では5つの適用事例から計算し、11.8[時間]および 1.22[時間]と仮定した。従って、式⑻は式⑼のよう記述できる。

PS 開発時間＝11.8+1.22×遷移の数　　⑼

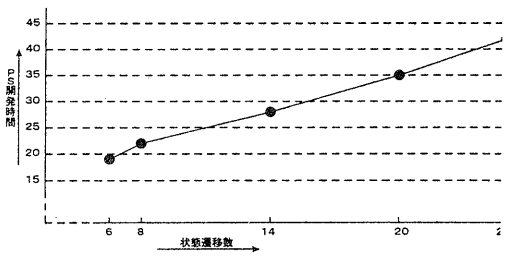

図12. PS開発時間と遷移数の関係

論理式⑼を用いて事例A～E の PS 開発時間を計算すると 42.3[時間]、28.9[時間]、19.1[時間]、21.6[時間]、36.2[時間]となる。一方、実際の PS 開発時間は 42[時間]、23[時間]、19[時間]、28[時間]、35[時間]であり、論理式⑼との誤差は 20[%]となり、良く一致している。従って、式⑼は PS 開発時間の計算に適用することができる。

最後に表5で示した提案する要求定義プロセスと従来の要求定義プロセスの PS 開発時間を比較すると、開発時間が従来の 30～37[%]となり、平均は 33[%]となった。この比率のばらつきは、提案する要求定義プロセスを適用した事例が小規模であったため、各事例共通のオーバーヘッドである論理式⑼の第 1 項による影響が大きいためと考えられる。この問題は、大規模な事例の場合には、論理式⑾の第 2 項による影響が大きくなるため問題とならない。

5. おわりに

PS 領域で要求定義を行うための PSC について、作成プロセス、作成方針および構成を報告した。さらに PSC を用いた要求定義プロセスを提案し、要求定義支援ツール(RDT)と動作確認用シミュレータ(MRS)について解説した。

PSC、RDT、MRS を実際の PS 要求定義に適用することで、従来の 30～37[%]の時間で PS 要求定義を実現できるようにした。これにより、短時間で PS の要求定義が実現でき、顧客要求との適合性が向上した。その結果、より品質の高い PS を短時間で構築できるようになり、プラント製作会社の競争力が向上した。

しかし、実際の PS 開発では要求定義の全体の開発時間に占める割合は2割程度である。従って、PS 開発の全工程を効率化させるには、基本設計、詳細設計、目的ソフトウエアの構築について対応することが必要である。

参考文献

1) Matsumoto M. et. al. :Specifications reuse process modeling and CASE study-based Evaluations, COMPSAC91, Proc. of the 15th annual international computer software & applications, 1991
2) Martin J. :"Rapid Application Development", Macmillian Publishing Company, 1991
3) 小林 久浩, 河田 恭郎, 前川 守, 川崎 章, 藪 彰文 and 小野川 公也: 実行可能な仕様記述によるプラント制御システムの環境のモデル化, 情報処理学会論文誌, Vol.35, No.7, pp.1402-1409, July 1994
4) 名取 万里, 加賀谷 聡 and 本位田 真一:ドメイン分析に基づく仕様再利用法, 情報処理学会論文誌, Vol.. 37, No.6, Mar 1996
5) W. Tracz and L. Coglianes: Domain-Specific Software Architecture Engineering Process Guidelines, IBM Federal System Company, ADAGE-IBM-92-02, 1992
6) J. DeBaud and K. Schmid: A Systematic Approach to Derive Scope of Software Product Lines, ICSE'99 Los Angeles CA, pp.34-43, May 1999
7) 松本 正雄 編:『ソフトウエアのモデル化と再利用』, 共立出版, 1995 と 1996