2005－SE－147（15）
2005／3／18

列車統合管理システムの ソフトウェアプロダクトライン

吉田 実　　遠藤 義雄
三菱電機 (株) 先端技術総合研究所

列車統合管理システム (Train Integrated Management Systems, TIMS) は冗長構成の分散監視制御システムで，営業運転中でもシステムの構成が変更される．本論文では，以下の特徴を持つ TIMS のソフトウェアプロダクトライン化手法を提案する．(1) 冗長構成、かつ、動的再構成に容易に対応可能な列車および機器の構成方式を持つ．(2) 冗長系システムの無矛盾性を実現するためのソフトウェアアーキテクチャを持つ．(3) 制御ソフトウェアモジュールの入出力から通信と機器の配置などを隠蔽したデータアクセス機構により，制御ソフトウェアモジュールが機器の配置と個数の影響を受けない．(4) 設計支援ツールにより，通信パケットのデータフォーマット，制御ソフトウェアモジュールの配置とデータ参照に関するプログラムの生成が可能である．以上により，制御ソフトウェアモジュールの再利用性を高めるとともに，動的なシステム再構成に容易に対応できる．

A Software Product Line of Train Integrated Management Systems

Minoru Yoshida　　Yoshio Endo
Advanced Technology R&D Center, Mitsubishi Electric Corporation

Train Integrated Management Systems(TIMS) are distributed fault-tolerant control systems. The configurations of them are dynamically changed during normal operations of them. In this paper, a method to construct a software product line of TIMS is proposed, which has the following features. (1)The structure that represents train and equipments makes redundancy and dynamic reconfiguration easy. (2)The software architecture keeps fault-tolerant systems to be consistent. (3)Because the data access mechanism encapsulates communication and allocation of equipments, control software modules are insulated from allocation and quantity of equipments. (4)The design-support tool produces program of data formats of communication packets and allocation and data access of control software modules. These features improve the reusability of control software modules and make dynamic reconfiguration easy.

1　はじめに

ある対象ドメインに対してあらかじめ作られた共通のコア資産から製品ファミリを開発するソフトウェアプロダクトライン (Software Product Lines，以下，SPL)[1] が，ソフトウェアの生産性を高める手法として注目されている．本論文では，多種多数の機器を監視制御

する冗長構成のシステムである列車統合管理システム (Train Integrated Management System, 以下，TIMS) の SPL の構築と特徴的な点について説明する．本 SPL では，制御ソフトウェアモジュールの入出力を通信と機器の配置などを隠蔽したデータアクセス機構で結合し，列車や機器の構成情報を設計支援ツールによって管理する．第 2 章で背景とドメインの特性を説明し，第 3 章で本 SPL において可変性を実現する機構について説明する．第 4 章ではソフトウェアの構成と開発方法を説明する．第 5 章で関連研究について議論し，第 6 章でまとめを述べる．

2　背景とドメインの特性

TIMS は，列車内の多種多数の機器を監視制御し，少人数の乗務員による列車運用を支援する分散制御システムであり，現代の高密度鉄道輸送のキーとなるシステムで高い信頼性を要求される．また，営業運転中に列車と列車が結合する併結，分離する分割があり，動的なシステム再構成が必要になる．これらの要求は TIMS のソフトウェア製作を難しくする要因となっている．一方，列車の車体は路線，用途，開発時期別の特注品となっているのが現状であり，TIMS のソフトウェアはこれらに対応するために少量多品種となっている．ソフトウェア生産性を改善するためには，ソフトウェアプロダクトラインの開発が有望であると考え，TIMS の SPL である PLATINA(PLAtform for TIms Nucleus with Advanced technology) を開発した．

TIMS の要求仕様，システム構成などの分析の結果，機能要求[1] は単純なものが多くを占めることがわかった．ソフトウェア開発の難しさは，個々の機能要求より，むしろ，機能要求の数と信頼性の確保の方法，列車構成と機器の多様さ，実時間性の保証など非機能要求が強い影響を与えていた．例えば,「ドア開」スイッチを押すとドアが開くという機能要求は今日の複雑化したソフトウェアから見ると単純なことのように思える．しかし，ドアの誤った開閉が起きず，逆に，正しいドア開の操作で確実にドアが開くために冗長構成を用いており，ドア制御装置と TIMS との通信網のトポロジと故障時の各故障モード[2]における通信の確保の方法などに多くの設計努力が注がれている．

3　可変メカニズム

SPL では，コア資産からあらかじめ決められた方法によって複数の製品を開発する．そのためには各製品の要求仕様にあわせてコア資産を変更する方法が必要である．本章では，コア資産から個々の製品を開発するために用いる可変メカニズムについて検討する．ただし，コア資産の一部である PLATINA の共通部ソフトウェアは，以下のクラス設計に基づく方法を用いて設計されており，コア資産自体の長期的な変更に耐えられるよう設計されている．以下に，検討した可変メカニズムをあげる．

クラス設計に基づく方法 オブジェクト指向設計の広く一般的に用いられている方法としては，文献 [9, 6] のように集約/委譲，継承，パラメータ化などの方法がある．これらはデザインパターン [10] を用いることで再利用性を高めることができる．

コンポーネント技術による方法 分散プログラミングでは，コンポーネントベースのソフトウェア再利用が利用される場合がある [7, 8]．コンポーネントのパラメータやコンポーネント間を結合するグルーコードによりシステムを作り上げることができる．コンポーネント間はメッセージパッシングで通信するのが一般的である．

[1] 機能要求はシステムの基本的な目的を達成する機能性の仕様である．非機能要求は機能性が持たなければならない性質で，信頼性，速度，運用環境などを含む [16].

[2] 故障の原因や影響に対する類別を故障モードという．例えば，通信ドライバ素子が断線故障の場合と短絡故障の場合で影響が異なることがある．

状態遷移に基づく方法 組込み分野では，状態遷移に基づきイベント駆動で設計する場合がある [13, 14]．イベント，状態の種類，状態遷移表を変更してシステムの動作を変えることができる．

制御モジュールを組み合わせる方法 制御分野では，制御モジュールを組み合わせてシステムを作る場合がある．制御モジュールの選択と配置，ある制御モジュールの出力データを入力する制御モジュールの選択によりシステムを作り上げることができる [3]．再利用可能なソフトウェアモジュールを組み合わせてシステムを構築するという点でコンポーネント技術と同じであるが，制御モジュールはメッセージパッシングを用いず，周期的に起動され，実行時に入出力データを読み書きする．

クラス設計に基づく方法は，比較的高機能なコンポーネントには有効であるが，多数の単純な制御がある TIMS の SPL の可変点の実現方式としては効果が限定される．例えば，単純な制御をするクラスを継承してやや複雑な制御をするクラスを作った場合，もともと単純な処理なので大きな効果は期待できない．

コンポーネント技術と状態遷移による方法は，メッセージパッシングやイベントを使っている．高信頼性が要求される冗長構成のシステムでは，イベントの欠落時の処理が複雑になり，また，実時間性の維持も難しくなる．このため，TIMS には適していない．

制御モジュールを組み合わせる方法は，制御の種類の選択とその入出力情報の配置・構成に注目したもので，これらの変更が多いシステムに適している．

TIMS では，単純な機能要求が多く，列車構成と機器の多種多様さ，信頼性の確保などが問題になっていたため，これらの検討結果から，PLATINA では制御モジュールを組み合わせる方法を主要な可変メカニズムとした[3]．

[3]ただし，クラス設計に基づく手法も有効な箇所では利用している．

3.1 機器

機器の種類と数が多いのが TIMS の特徴である．パンタグラフや車輪を駆動するモータ，ドア，空調機，トイレ，自動販売機など様々な機器が存在する．TIMS と機器の通信方式は，リレーなどの接点入出力によるもの，および，RS485 を用いて独自のプロトコルとフォーマット定義に基づく伝送とに分けられる．列車においては，これらの機器の配置は列車中の車両の種類によって決まる．例えば，モータは動力車両にのみ存在し，ドアは全ての車両に 4 つずつ存在するなどである．TIMS における最も変更される要素が機器との通信内容，および，機器の配置である．そのため，PLATINA では機器を中心的な概念の一つとしている．

列車は 1 つまたは複数の編成の (順序を持った) 列であり，編成は 1 つまたは複数の車種の列で構成される．車種ごとに搭載機器の種類と個数が決められている．各編成のその構成車種の列および車種ごとの搭載機器の種類と個数は静的に決められている．これらは，4.1 節で説明する自動生成プログラムとして実機上のラインタイムプログラムに埋め込まれる．一方，列車の編成は，動的に変化する可能性がある．TIMS は現在列車がどの編成から構成されているかを監視し，構成に変更のあった場合は，システムを止めることなく新たな構成に移行する必要がある．これを実現するために，現在の列車構成とシステム全体の機器の個数と配置などを管理する機構を設けている．

3.2 制御モジュールの配置

分散制御システムではある制御を実行する場所を何らかの方法で決定する必要がある．PLATINA ではこれを制御モジュールが扱う機器を指定することで実現する．制御モジュールは指定された機器のあるユニット[4] で機器の個数，もしくは，1 個だけ配置する．制御モ

[4]管理する機器を持ち，他のユニットと通信し，自律的に動く TIMS の構成要素．列車の各車両に 1〜2 個ある．

ジュールが機器の数だけある場合は，個々の制御モジュールは対応する 1 個の機器の制御に専念する．制御モジュールが 1 個の場合，そのユニット内の全ての機器を同時に制御する．

制御モジュールの配置は，車種ごとに静的に決まる．システムの構成が変更されても，制御モジュールが新たに生成されることはない．制御する機器の数の変更は，次節のワイヤ機構による．

3.3 ワイヤ機構

制御モジュールまたは機器の出力から，制御モジュールまたは機器の入力へデータを運ぶ仕組みをワイヤと呼ぶことにする．ワイヤは機器の個数と配置を隠蔽し，機器の変更による影響を制御モジュールのプログラムに与えないようにする．これにより，制御モジュールの再利用性を高めている．ワイヤは異なる CPU 間にデータを運ぶ場合は通信を用いて実現され，同一 CPU 内の場合は CPU の主記憶中で実現される．ワイヤは，任意のビット数のデータを，出力側機器と入力側機器から決定される個数だけ送る仮想的な線である．

ワイヤは，情報の伝達される通信の種類と入出力機器を属性として持つ．通信の種類は，システム全体，ユニット内，CPU 内など通信の範囲などを指定する．システム全体で通信される情報の場合，各ユニットはそのユニットの接続された機器の数だけの情報を出力する．システム全体に存在する機器の個数だけ情報が存在することになる．一方，ワイヤは，入力側機器に対して個別に指令することも，全体を同一の値で指令することもできる．出力側機器の個数を n，入力側機器の個数を m とすると，システム全体で $n \times m$ 個のデータが存在することになる．しかし，通常このような例はわずかであり，n または m の一方，または，両方が 1 である場合が多い．

$n \times m$ 個のデータが必要になるのは，指令機器が個別機器を指定してデータを出す場合である．例えば，エアコン温度設定器が列車中に n 個あり，そこから m 個あるエアコンに個別に温度を設定する場合を考える．この場合，エアコン設定器はエアコン個別に m 個の指令温度を出し，エアコン設定器が n 個あるため，システム全体に存在する情報の個数は $n \times m$ 個となる．ワイヤから情報を受け取る制御モジュールは自らの制御対象のエアコンの指令温度しか必要としない．その数はエアコン温度設定器の数と同じで n 個である．ワイヤにより各制御モジュールが必要とする n 個のデータが個々の制御モジュールに送られる．

制御モジュールはワイヤから情報の数と情報への参照方法を与えられ，静的または動的に機器の数や配置が変化しても，制御モジュールのプログラムは影響を受けない．よって，異なる機種，機種の仕様変更に強く，また，列車の結合による構成変更を自動的に反映するという利点を持っている．

3.4 例

図 1 に制御モジュール，ワイヤと機器の指定の関係の例を示す．右に先端のある 5 角形は機器からの情報出力であり，左に先端のある 5 角形は機器への情報入力である．長方形は制御モジュールである．それらの下に付いている小さな長方形は，機器名である．機器からの情報出力，制御モジュール，機器からの情報入力をつなぐ線はワイヤを表す．機器名の「/」の後にある 1 と*はそれぞれあるユニットに機器が複数あった場合，制御モジュールを 1 個だけ作るか，機器個数分作るかを指定するものである．

図の左上に示された「ドア開スイッチ」は，列車の先頭車両，および，最後尾車両の運転台にあり，どちらかでスイッチが押されると，ドアのあるユニットに置かれた制御モジュールで論理和がとられ,「ドア開押下」が真になる.「ドア開指令」は,「ドア開押下」かつ「列車速度が 5km/h 以下」の条件で真になる．

この例のロジックを実現する上で機器などの個数は必要なく，TG(回転速度計) 搭載車種，各車種のドアの数などが変更されても，車種ごとの搭載機器情報を変更するだけでよい．

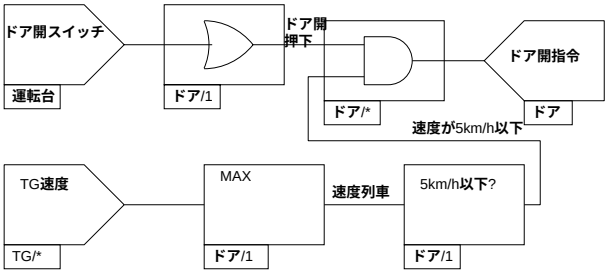

図 1: 制御モジュール，ワイヤと機器の指定

4 ソフトウェアの構成と開発方法

4.1 製品開発のワークフロー

組込みソフトウェアの開発でも，近年，開発ツールによる製品化支援の範囲が広がってきており，プログラムの自動生成を使う例が増えつつある [2]．PLATINA では，機種依存プログラムの多くは列車構成，制御モジュールの配置，制御モジュールと機器の間のワイヤの配線であるため，これらのプログラムを生成するツールを開発した．図 2 に製品設計のワークフローを示す．図中で破線で囲った部分をツールが支援する．標準要求仕様から適切な仕様を選択し，適切なパラメータを与え要求仕様とする．要求仕様は，設計支援ツールを用いて入力され，それらの構成，配置，配線情報は，一旦 XML 形式でファイルに保存され，プログラム生成時にこれらの情報に基づき実行可能なプログラムを生成する．

生成プログラムは，制御モジュールライブラリ，共通プログラムとファイルレベルで完全に分離されており，基本的に開発者が生成プログラムに修正・追加することはない．

4.2 ソフトウェアアーキテクチャ

PLATINA のソフトウェアアーキテクチャは図 3 のように層状になっている．最下層がハードウェア層であり，1 つ上の層がドライバ層である．ドライバは，割り込みや一定周期でハードウェアをチェックし，ハードウェア層のデータを分散共有メモリに反映させ，また，

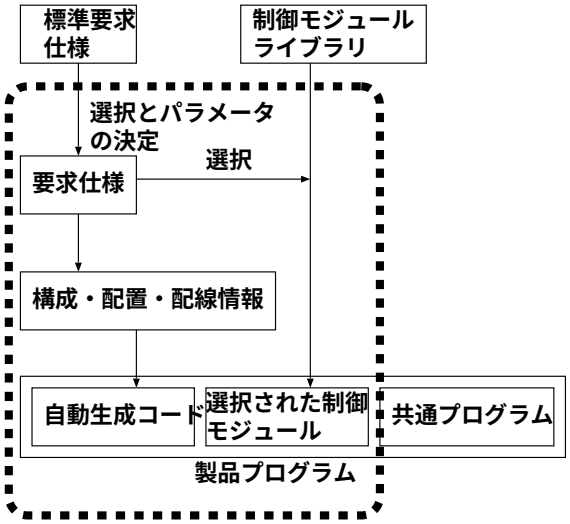

図 2: PLATINA の製品開発ワークフロー

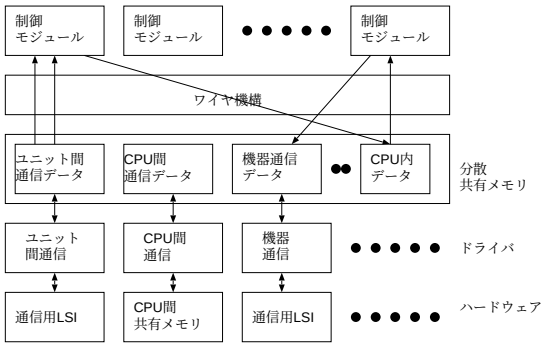

図 3: ソフトウェアアーキテクチャ

分散共有メモリの内容に基づきハードウェア層にデータを与える．分散共有メモリには，ドライバやワイヤ機構から要求されたデータが入出力されるが，そのインタフェースは統一されている．そのため機種やアクセスする分散共有メモリの種類やアクセス箇所が異なっても制御モジュールのプログラムを変更する必要はない．

分散共有メモリは，データ作成元とその内容ごとにデータフレームという単位で他の CPU と通信される．データフレームは，CPU 内でのみ使われるもの，ユニット内で使われるもの，システム全体で使われるものがある．データフレームは内部データのビット配置を決めたフォーマット情報を持つ．

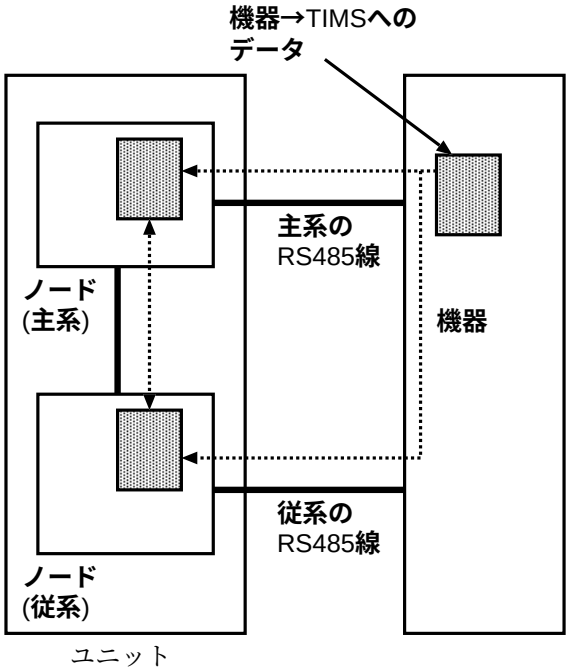

図 4: ユニット内の監視制御機器へのバックアップ処理の例

4.3 故障時の動作

今日の高密度鉄道輸送の安定した運行のため，列車はシステムとしての耐故障性が重要であり，TIMS は耐故障性実現の中心的な担い手となっている．図 4 は機器との RS485 を用いた通信を 2 重化している例である．どちらかの通信線が切断または短絡しても，他方で通信を続けることができる．機器と TIMS は相互にパケットを交換することで通信している．TIMS のユニットは，主系と従系の 2 つのノードからなり，主系が故障しても，ほとんどの処理を従系が瞬時に引き継ぐことができる[5]．ノード間は機器から TIMS へのデータを常時交換しあっている．図で，主系の RS485 の通信線が切れた場合は，機器からの情報は，従系の通信線を通じて従系ノードに伝達され，更にノード間の通信を経由して主系ノードに伝達される．

機器から TIMS へのデータは，各ノードでは，機器から直接のデータ，ノード間通信を経由したデータと常用データの 3 種類保持される．常用データはその時点で有効なデータのコピーである．直接機器から受信されるデータが有効な場合はそのデータがコピーされ，直接のデータが無効で，かつ，ノード間通信経由のデータが有効な場合はノード間のデータがコピーされる．両方無効である場合は，常用データは無効になる．なお，直接のデータとノード間のデータは，一定期間正しいデータが受信されない場合無効と見なされる．制御モジュールが機器からの情報を直接参照する場合，常用データを参照するのが一般的である．その場合，もし，直接的な通信が不通になっても，自動的に迂回経路のデータが取得できる．

図の機器が故障した場合などは，常用データが無効になる．データは有効かどうかチェックでき，また，無効時にデータを読み出すと 0 が返る．制御ロジックには無効時の処理を含んでいる．これは，リレーの A 接点，B 接点を意識した論理設計に類似した概念である．

TIMS の機器の接続形態は多種あり，機器の特性，要求される信頼性などから選ぶことになる．しかし，このように制御モジュールはシステムの故障を意識しないように設計できるため，制御モジュールの再利用性が高まる．

5 関連研究

IEC61131[4] と IEC61499[5] は，アルゴリズムと呼ぶ制御モジュールを配線するように設計するための規格で，ツール支援も考慮されており PLATINA と共通点が多い．IEC61131 は PLC(Programmable Logic Controller) の規格でイベントの扱いはない．一方，IEC61499 の分散制御のための規格で，イベントを扱う機能が追加されている．これは，分散制御システムでは何らかの同期機構がないと効率的に処理ができないとの考えに基づいているためと思われる．一方，PLATINA は分散制御システムを対象としているが，イベントを扱わない．IEC61499 の FAQ(Frequently Asked Questions)[3] では，重要なイベントの欠落に対しては，定周期でイベントを送るか，

[5]例えば，ブレーキをかけている時に主系ノードがダウンした場合などでも従系が，50ms 程度でブレーキの制御を引き継ぐ．

応答のイベントを返すかせよとある．一方，PLATINA では，基本的に定周期でデータを送るシステムとなっている．

CORFU[12] は，フィールドバスをルータでイーサネットで結んだヘテロなネットワークを対象としたソフトウェアフレームワークで，IEC61499 に基づく設計を支援する．アーキテクチャの差異はあるが，実現しようとしている全体像は PLATINA と似ている．

また，文献 [11] では制御システムを再構成可能にする方法としてイベント処理に基づく NFSM(Nested Finite State Machine) を使っている．イベントの取りこぼしと重複，冗長系システムでの無矛盾性などを真剣に考える必要の多いシステムの場合 PLATINA 方式がよいと考えている．逆に，1 重系で高度な信頼性が要求される処理が少ない場合は，イベントを中心にしたシステムの方が柔軟性があると思われる．

可変メカニズムについては，既に，3 章で議論している．その他の方法としては，アスペクト指向プログラミング [15] の概念も興味深い．PLATINA の可変メカニズムと別の観点で，試験や特殊なコンフィグレーションなどに使える可能性がある．

6　おわりに

冗長系分散制御システムにおいて，動的にシステム構成が変更される場合を扱うソフトウェアプロダクトラインを可能にする方法を提案した．イベントを排除し，一定周期で処理を行うことで，信頼性と実時間の予測可能性の高いシステムとなる．また，プログラム生成ツールを採り入れたワークフローを検討した．機種依存の仕様からプログラマがあらかじめ作るプログラムと分離した生成プログラムを作るツールにより，高い生産性が得られると期待できる．

参考文献

[1] Clements, P. and Northrop, L., “Software Product Lines - Practices and Patterns”, Addison Wesley (2002).

[2] Pasetti, A., “Software Frameworks and Embedded Control Systems”, LNCS 2231, Springer (2002).

[3] Robert Lewis, “Modeling control systems using IEC 61499 - Applying function blocks to distributed systems”, IEE (2001).

[4] International Electrotechnical Commission, “Programmable controllers Part 3: Programming Languages”, IEC 1131-3 (1993).

[5] “Function blocks for industrial-process measurement and control systems - Part 1: Architecture”, IEC-PAS 61499-1 (2000).

[6] Jacobson, I, Griss, M., Jonsson, P. “Software Reuse: Architecture, Process, and Organization for Business Success”, Addison-Wesley, (1997).

[7] Stefan Kettemann, Dirk Muthig and Michalis Anastasopoulos, “Product Line Implementation Technologies - Component Technology View”, IESE-Report No. 015.03/E (2003).

[8] D. Muthig, et. al., “Technology Dimensions of Product Line Implementation Approaches - State-of-the-art and State-of-the-practice Survey”, IESE-Report No. 051.02/E (2002).

[9] Michalis Anastasopoulos and Cristina Gacek, “Implementing Product Line Variabilities”, IESE-Report No. 089.00/E, (2000). (Proc. of the Symposium for Software Reusability, Toronto, Canada, 2001)

[10] E. Gamma, et. al, "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley (1995).

[11] S. Wang, K. Shin, "Constructing Reconfigurable Software for Machine Control Systems", IEEE trans. on Robotics and Automation, Vol. 18, No. 4 (2002).

[12] K. Thramboulidis, "Development of Distributed Industrial Control Applications: The CORFU Framework", 4th IEEE International Workshop on Factory Communication Systems, Vasteras, Sweden, (2002).

[13] Dougrass, B. P., "Doing Hard Time", Addison Wesley (1999).

[14] Gomma, H., "Designing Concurrent, Distributed, and Real-Time Applications with UML", Addison Wesley (2000).

[15] G. Kiczales, et. al., "Aspect-Oriented Programming", ECOOP'97 (1997).

[16] S. Robertson and J. Robertson, "Mastering the Requirements Process", Pearson Education, (1999).