人工知能学会
第12回知識流通ネットワーク研究会
SIG-KSN-012-04

設計情報との関連を考慮した要求定義文書の抜け漏れ防止手法

A prevention method of undefined requirements by backward traceability from design to requirements

穴田 啓樹，山本 修一郎
キャッツ株式会社，名古屋大学

Keiju ANADA，　Shuichiro YAMAMOTO
CATS CO.,LTD.，　Nagoya University

概要

組込みシステム開発の品質を確保するため，要求定義の段階から検証することが注目されている．しかし，これまでは単独の工程を対象としており，工程を完了する基準を明確化できなかった．本稿では，要求工学の国際規格である IEC29148 で紹介された観点を基に，設計工程で必要となる情報と紐付けを行った上で要求定義の文章を解析し，抜け漏れを防止する手法を紹介する．

Abstract

To ensure the quality of the embedded systems, it is necessary to verify from the stage of the requirement definition. However, there are no criteria to complete the process. In this paper, we propose a method to analyze the text of requirements definition in terms of IEC29148 and avoid undefined requirements.

1. はじめに

組込みシステム開発の品質を確保するために，要求仕様や設計の抜け漏れを防ぐことが重要である．これまで筆者は，組込みシステムの振る舞い設計の抜け漏れを防ぐために，状態遷移表を利用した設計手法を推進してきた．開発現場では状態遷移設計できない技術者も多く，また，属人性が高いのも実情で，状態遷移設計の教育から始めることが多かった．しかし，状態遷移設計ができない理由を調査すると，状態遷移設計を習得していないだけでなく，前工程である要求定義において設計に必要な情報が十分に抽出できていないこと，要求定義から設計への移行の判断ができないことに問題があることがわかった．

本研究では，状態遷移設計の改善のために，要求定義の段階で設計に必要な情報を抜け漏れなく抽出できたかを検証する手法の確立を目標とした．具体的には，状態遷移設計に必要な情報から要求定義の文章へと遡る方向に情報の紐付けを確認した上で，要求定義の文章の抜け漏れをチェックする手法を開発し，要求定義から設計への移行の判断とすることを試みた．

2. 前提とする設計プロセス

組込みシステム開発では，産業で共通するハードウェアプラットフォーム，ソフトウェアプラットフォームが存在しないため，開発プロセスは各企業，部門毎に異なる．本稿では独立行政法人情報処理推進機構・ソフトウェアエンジニアリングセンターが推進する組込みソフトウェア向け開発プロセスガイド（ESPR）を前提とした[1]．ESPR では，V 字モデルを前提としたシステムエンジニアリングプロセス，ソフトウェアエンジニアリングプロセスが定義されており，さらに，各プロセスのアクティビティのガイドが定義されている．ESPR はガイドであるため，実際の開発における各アクティビティのタスクは独自に定義する必要があり，Object-Oriented System Engineering Method (OOSEM)を参考にタスクとその成果であるモデルと文書を定義した[2][5][6]（図 1）．

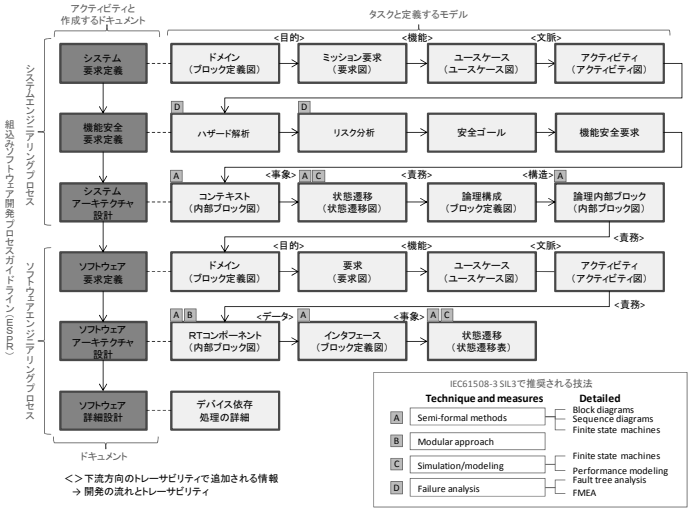


図 1 本稿で前提とした設計プロセス

本設計プロセスでは，まずシステムで扱う範囲を決めた後，要求，ユースケースの順に導出し，構造と振る舞いを設計する．構造設計では制御ユニットを導出し，インタフェースを設計する．振る舞い設計では状態遷移を設計する．本稿で対象としている状態遷移設計は組込みシステム開発で特徴的な設計であり，重要な設計成果となる．

3. 研究課題の定義

状態遷移設計において，状態の抽出は設計者の時間粒度と区間のとらえ方に依存するので，特に属人性が高いとされる．しかし，属人性の計測法が確立していないため，明確な対策が立てられなかった．本研究では，状態遷移設計の改善のために，要求定義の文章を対象に，属人性の可視化，属人性を低減手法の確立，手法の完了基準の確立を課題とした（表 1）．特に，3 点目の「完了基準」については，実開発において悩むケースが多く，指標が得られるとメリットが大きい．

表 1 要求定義における課題

① 属人性の可視化	要求定義の成果物である要求仕様書の何に属人性が現れるのかがわからない．
② 属人性の低減手法の確立	要求仕様書作成の属人性を低減するには何をすべきかがわからない．
③ 要求定義の完了基準の確立	要求定義をいつまで続けるのか完了基準がわからない．

4. 課題の解決手順

本研究の課題解決にあたり，要求定義の文章において，設計に必要な情報の抜け漏れを最小化することを目標とする（図 2）．そのために，要求定義の文章の抜け漏れをチェックする方法を開発する．抜け漏れチェック法の開発にあたり，まず，要求定義手法と状態遷移設計手法の詳細を確認する．次に，状態遷移の構成要素に対応する文章の構成要素を定義する．それらの文章の構成要素が要求定義の文章に記述されれば，状態遷移設計に必要十分であるとし，これをチェックすればよいとする（図 3）．また，状態遷移に必要な文章の構成要素と状態遷移設計の属人性の関係を検討し，要求定義の文章チェックの自動化を検討する．

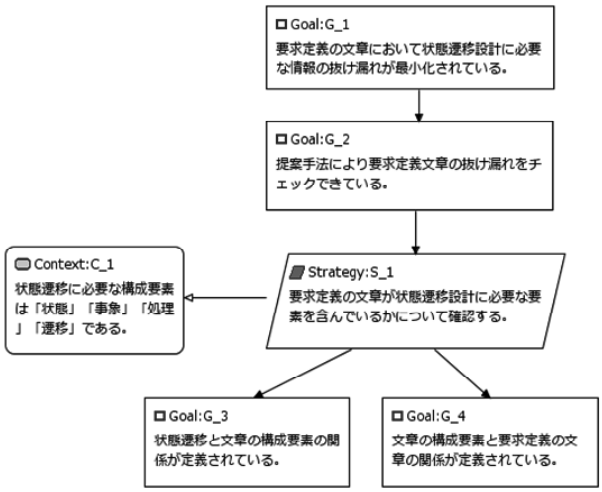


図 2 論証の流れ

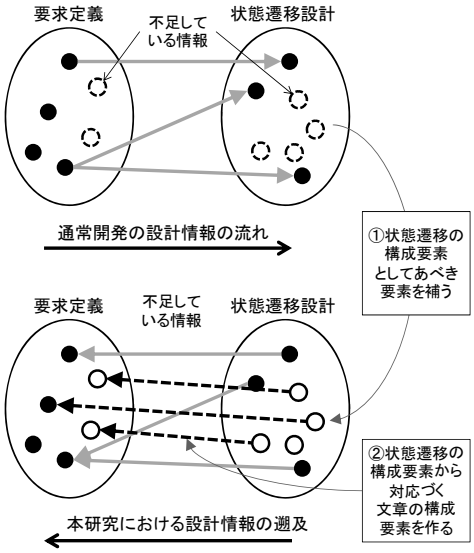


図 3 設計情報の流れ

4.1. 要求定義の確認

要求定義は，システムに求められることを明確にするアクティビティである．本稿の開発プロセスにおいて，要求定義の成果物は要求仕様書である．要求仕様書に含める情報は「要求」と「仕様」である．「要求」には，何をしたいのか，システムに何をさせたいのかを書く．「仕様」には要求を実現する手段を書く．多くの要求仕様書において，仕様のみが書かれることも多いが，仕様は実現手段で変化しやすく，本来は何をしたいのかをおさえておくことが重要であるので，本稿では要求と仕様を区別する．要求定義の手順としては，まず，何をしたいのか，システムに何をさせたいのかを要求として，思いつく範囲で洗い出す．次にそれらの要求を分類し，構造化する．分類して構造化するのは，システムで扱う範囲の要素を見渡すためである．

分類は，要求工学の国際標準規格である ISO/IEC/IEEE 29148 で紹介されているドキュメントの記載項目を参考にした[3]．IEC29148 では要求工学に必要なプロセスと文書構成のガイドが示されている．機能要求，非機能要求を洗い出し，分類するのに参考になり，本稿でも活用した（表 1）．

表 1 要求仕様書に記載する項目の例

システム要求	ソフトウェア要求	
機能	外部インタフェース	異常対応
ユーザービリティ	機能	品質
性能	ユーザービリティ	安全
インタフェース	性能	標準準拠
操作	論理データベース	
モードと状態	設計制約	
物理特性	システム属性	
環境条件	保守	
セキュリティ	初期化	
法規制	メモリ	
継続性	タイミング	

ISO/IEC/IEEE29148 から抜粋

構造化は，分類した要求間に関係がある場合に，より本質的な上位の要求を見出すために行う．この際，下位の要求は上位の要求を実現する「仕様」として従属関係を持たせる（図 4）．

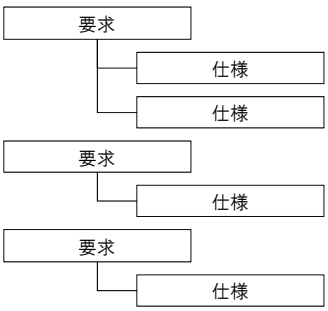
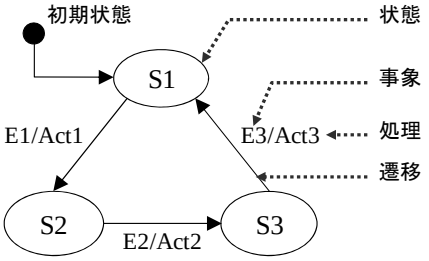


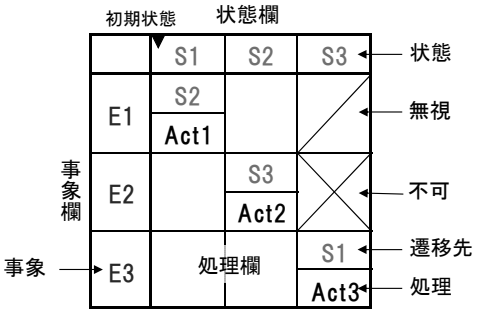
図 4 要求と仕様の従属関係

4.2. 状態遷移設計の確認

状態遷移設計は，対象システムの動的な側面を状態と事象を基にとらえるアクティビティである．状態は，設計対象の意識すべき時間粒度において，ある区間を他の区間と区別するために名づけしたものである．事象は設計対象に加わる刺激であり，処理は事象発生を契機に開始する振る舞いであり，遷移は状態の切り替わりである．本稿では，この「状態」「事象」「処理」「遷移」の一つの組み合わせを「状態遷移の構成要素」と定義する．状態遷移設計では，これらの構成要素を状態遷移図や状態遷移表にまとめる（図 5）．概要段階では状態遷移図，例外や異常への対応を網羅的に設計する段階では状態遷移表が適しているとされる．



(a) 状態遷移図



(b) 状態遷移表

状態遷移図と状態遷移表の読み方

・初期状態は S1

・S1 のときに E1 が発生すると Act1 を実行して S2 に移る

・S2 のときに E2 が発生すると Act2 を実行して S3 に移る

・S3 のときに E3 が発生すると Act3 を実行して S1 に移る

図 5 状態遷移図と状態遷移表

4.3. 状態遷移と文章の構成要素の関係

状態遷移表と文章を関連づけるために，4.2 で確認した状態遷移表を文章に近い形に変形する．状態，事象，処理，遷移に記述される要素は，

状態：この場合に
事象：こうなったら
処理：こうして
遷移：こうする

のように，文章の構成要素として見ることができる（図 6）．このように状態遷移表と文章の構成要素（文節）の関連がわかる．

要求	State 1 （この場合に）	State 2 （その場合に）
Event 1 （こうなったら）	Action 1 （こうして） State 2 （こうなる）	不可 （あり得ない）
Event 2 （そうなったら）	無視 （無視する）	Action 2 （こうして） State 1 （こうなる）

図 6 状態遷移表と文節の対応

4.4. 構成要素の詳細化（要求仕様整理表）

実際の製品開発における状態遷移設計では，事象や処理に条件が設定される場合がある．また，事象と処理には主体者が存在するので状態遷移表の構成要素を詳細化する（表 2）．また，詳細化した 12 の状態遷移の構成要素を表にまとめたものを要求仕様整理表と定義する（表 3）[4].

表 2 詳細化した状態遷移の構成要素

		主体	対象／要素
状態	遷移前の状態	ムーバー（事前）	ステータス（事前）
事象	事象発生の条件	イベントコンディショナー	イベントコンディション
	事象	イベンター	イベント
処理	処理の条件	アクションコンディショナー	アクションコンディション
	処理	アクター	アクション
遷移	遷移後の状態	ムーバー（事後）	ステータス（事後）

表 3 要求仕様整理表

文書	事前状態		事象				処理				事後状態・遷移先	
	ムーバー	ステータス	コンディショナー	コンディション	イベンター	イベント	コンディショナー	コンディション	アクター	アクション	ムーバー	ステータス

4.5. 要求仕様整理表と要求仕様文章の関係

例文を用いて，要求仕様整理表から要求仕様の文章へ遡及してみる．要求仕様整理表と「タイマが起動している場合，タイマボタンを長押ししたら，ブザーを 1 回鳴らした後，タイマ値はリセットされ，タイマが停止する.」という要求仕様の文章を関連づけることができる（図 7）.

文書	事前状態		事象				処理				事後状態・遷移先	
	ムーバー	ステータス	コンディショナー	コンディション	イベンター	イベント	コンディショナー	コンディション	アクター	アクション	ムーバー	ステータス
タイマが起動している場合、…タイマが停止する。	タイマ	起動			タイマボタン	長押し			(1) ブザー (2) タイマ値	(1) 1回鳴らす (2) リセット	タイマ	停止

タイマが起動している場合、タイマボタンを長押ししたら、ブザーを1回鳴らした後、
タイマ値はリセットされ、タイマが停止する。

図 7 要求仕様整理表から文章への遡及

4.6. 状態遷移表を作成するのに必要な文章要素

これまでの状態遷移表から要求定義の文章への遡及から，要求仕様の文章が要求仕様整理表の要素を包含していれば，状態遷移表を作成できると言える．これまでの支援においては，事象と処理は抽出できる開発者が多かった．属人性が高いのは「状態」の抽出であり，状態の抽出に自信が持てない，状態の抽出が漏れるなどのケースが見受けられた．また，「事象発生の条件」「処理の条件」は，状態遷移を構成する必須要素ではないが，設計の完成度に関わるものである．

以上より，要求仕様の文章において，12 の状態遷移の構成要素の網羅度を示すことは「課題①属人性の可視化」に貢献できるものと考えられる．特に「状態」に該当する「ムーバー」の有無や，条件の有無を指摘できると，要求仕様の文章としての抜け漏れを検証することが可能であり，「課題②属人性の低減」に貢献できる手法であると考えられる．

一方，複数の文章で 1 つの状態遷移の要素を構成することもある．そこで，「要求と仕様の従属関係」（図 4）で定義した構造化された要求と仕様のグループ内において，状態遷移の構成要素を充足したかを確認することで「③要求定義の完了基準」とする．ただし，状態遷移表には複数の「状態遷移の構成要素」があるので，必要な全ての状態遷移の構成要素を満たしたかは判断できない．そのため，要求定義段階でも状態遷移表を作成して確認することを提案する．

4.7. 要求定義の抜け漏れ確認の効率化

要求定義の抜け漏れを防止するには，要求仕様書の文書から要求仕様整理表を作成して，12 の状態遷移の構成要素の網羅度を確認すればよいことがわかった．しかし，文書中の全ての文章について確認することは多大な労力であり，ミスも想定されるため，ツールによる自動化を検討したい．幸い，状態遷移表の構成要素は特徴的な言葉を持つことが多い．例えば，次のような言葉から要求仕様整理表に分類して記載すべき要素を抽出することができる．

状態：「～の場合に」「～であれば」
事象：「～のとき」「～したら」
処理：「～する. 」「～を行う. 」
遷移：「～となり」

要素の抽出にあたり，まず，文章から要求仕様整理表に適した文節を作る必要がある．文節の作成には，オープンソースの形態素解析エンジンである Mecab を利用した．Mecab で文章を品詞に分解した後，品詞の前後関係から要求仕様整理表

の作成に適した文節に組上げた．以上により，文章を要求機能整理表の項目に分解し，状態遷移の構成要素の網羅度を確認するツールを試作した．

5. 結果

本手法と試作した支援ツールは開発支援の”支援”に活用するものとして評価中である．実際の開発プロジェクトに適用しており，支援対象となる開発者からは「はじめから抜け漏れのない網羅的な文章を書くことは苦痛であるので，文章を書いた後に自分で確認できるのがよい」という評価を得ている．プロジェクト終了後に定量的な評価を行いたい．

6. 考察

要求工学の国際標準規格であるIEC29148では要求の表現文法の例を紹介している（図 8）．本手法と試作ツールも日本語においてこれに準じた分割を行う．このように文章の意味の単位で分割して見せることは，要求定義の文章を検証する上で有用であると考えられる．

[Condition] [Subject] [Action] [Object] [Constraint]

EXAMPLE: When signal x is received [Condition], the system [Subject] shall set [Action] the signal x received bit [Object] within 2 seconds [Constraint].

ISO/IEC/IEEE 29148:2011(E)から一部を引用

図 8 IEC29148 で紹介する要求の表現法

また，規格書では条件と制約の違いについて触れており，制約は全ての要求に渡って適用される可能性を指摘している．組込みシステム開発においては機能要件に目が向きがちであるが，後工程で制約が顕在化すると，それを充足するための修正がソフトウェアの構造に想定外の影響を及ぼすことも珍しくない．よって，制約の抜け漏れの可能性を気づかせる工夫が必要だと考えられる．

7. 今後の課題

結果で述べた通り，本手法と試作ツールは実際の開発プロジェクトで効果を検証中である．プロジェクト終了後には定量的な評価を行い，改良していきたい．

参考文献

[1] 独立行政法人情報処理推進機構ソフトウェア・エンジニアリング・センター, “改訂版 組込みソフトウェア向け開発プロセスガイド”, 翔泳社，2007 年

[2] Alan Moore, Rick Steiner, “A Practical Guide to SysML”

[3] INTERNATIONAL STANDARD ISO/IEC/IEEE 29148, "Systems and software engineering — Life cycle processes — Requirements engineering"

[4] 松井恭, “要求仕様の漏れ・抜け検証の手法について”, 2011

[5] Geoffrey Biggs，坂本武志，藤原清司，穴田啓樹，"SysML によるモデルベースディペンダブルロボット開発"，ロボティクス・メカトロニクス講演会(Robomec2012), 2012 年 5 月，浜松

[6] 藤原清司, 中坊嘉宏, 水口大知, Geoffrey Biggs, 穴田啓樹，"ディペンダブルロボット車両のハードウェア試作"，ロボティクス・メカトロニクス講演会(Robomec2012), 2012 年 5 月，浜松