1L-2

ソフトウェア工学における要求定義の効率化について

上田　翔太†　石野 正彦†　石田秀信†　寺田 郁二†
福井工業大学大学院工学研究科†

1. はじめに

現状のソフトウェア開発において、要求定義は非常に重要である。ソフトウェア開発のプロジェクト崩れでは多くの原因が要求定義に原因がある。また、要求定義の精度と速度の向上は全体の作業の効率化やコストの削減である。本稿は要求定義について投稿したものである。

2. 要求定義が注目されている理由

ソフトウェア開発では様々なユーザ要求が重なり修正が発生する。その原因として上流工程の要求定義フェーズでの仕様の確定が不十分である。この要求定義フェーズでの仕様の欠落は、顧客の要求を正しく定義できていない。それは顧客の言い忘れや、意識していなかった部分である漏れの要素[1]。また、定性的な情報や相対的な情報であるあいまいさといわれる要素がある。この二つの要素が必要以上の修正とコストを出している。さらに、漏れやあいまいさは以下のような害を及ぼす。

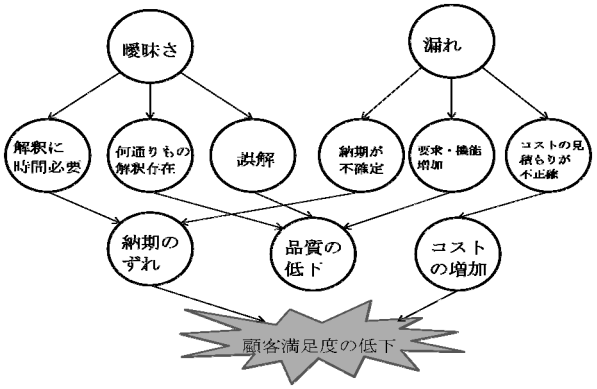

図１　漏れとあいまいさの害

3. ソフトウェア開発におけるプロジェクト崩れ

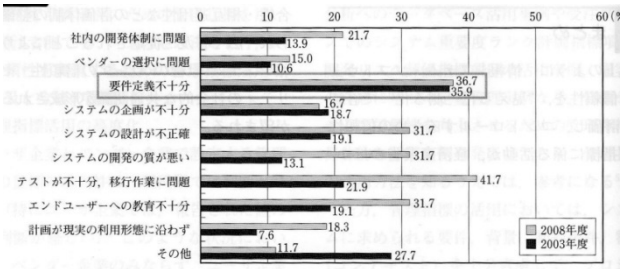

図２　プロジェクト崩れの統計的データ

図 2 から示されるようにプロジェクトの遅れの主な原因 は下記のとおりである[2]。

- 要求定義不十分[36.7%]
- テストが不十分 [41.7%]
  - システムの開発の質が悪い[31.7%]
  - システムの設計が不正確[31.7%]

国内のシステム開発プロジェクトの成功率は26.7%である。このようにソフトウェア開発では約７割の失敗があり、その原因として約４割が要求定義の原因である。

4. 要求定義で抽出される要求

要求分析では抽出されたか、要求の正しさ、望まれているかという成分で構成される要求がある[3]。表１、図３で示す。

表１　抽出される要求

記号	望ましさ	抽出	正しさ	説明
U	x	x	–	望ましくない要求が抽出できていないので、正しいのか確認できない
V	O	x	–	要求の漏れである。抽出できていないので、正しいかどうか確認できない
W	O	O	x	望ましい要求が摘出されているが正しくないので、修正対象である。
X	O	O	O	望ましい要求が正しく抽出されている
Y	x	O	O	正しいが不要な要求である。要求漏れではないが、要求変更の対象となる。
Z	x	O	x	誤った不要な要求である。要求漏れではないが、修正対象である。

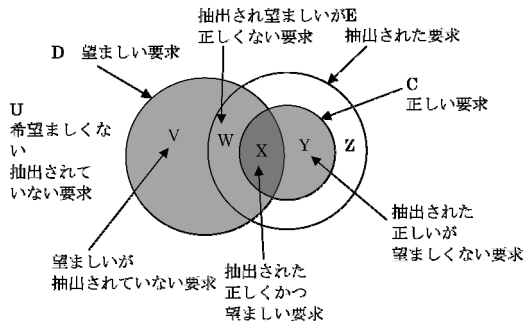

図３　抽出される要求

5. 現状の問題

プロジェクト崩れが起こる理由として、以下に挙げられる[4]。

①顧客の問題

・業務上何を欲しているかわからない
・ソフトウェアの知識がないため、アルゴリズムに近づこうとしない。

②開発の問題

・顧客との意思の疎通がうまくできていない
・技術者が要求分析を行うための業務知識に欠け、正確な要求分析ができていない。

Efficient of requirements definition in software Engineering
†Syota Ueda,Masahiko Ishino,hidenobu Ishida,Yuji Terada
†Fukui University of Technology

6.解決策の提案

これらの問題を解決するためには必要な物をして、①自動化と②視覚的にわかりやすいツールや手法が挙げられる。

①自動化

自動化を行う方法[5]として、オブジェクトやノードなどの一定のデータを入力し関係図を作成する。そうすることによって顧客側にも視覚的に分かりやすく、あいまいさや誤解の回避につながる。誤解に関しては、顧客と開発側の誤解による要求分析の失敗という部分がなくなるので精度は高くなる。また、機械的に分析されるので時間的コストは非常に少なくなる。具体的な方法として上流 CASE ツールなどがある。

②視覚的にわかりやすいツールや手法

視覚的に分かりやすくすることにより、技術者以外の人や顧客に分かりやすくする。また、これで簡単に確認できるようになり漏れやあいまいさからの誤解を回避することができる。しかし、簡略的にすることにより分かりやすさの向上、時間的コストを削減はできるが、詳細な仕様については省かなければならない。また逆に詳細な仕様書の作成は時間的コストの増加や漏れ等に気付かなくなる恐れがあるというトレードオフの関係にあるため、簡易的な確認のための仕様書と詳細な仕様書の双方をそろえることが必要である。現状の方法として統一モデリング言語がある。図 4 のようなシーケンス図を代表とするようにシステムの表面的な部分を視覚的に分かりやすくしている。また、手法も 13 種類あり使い分けることができる。

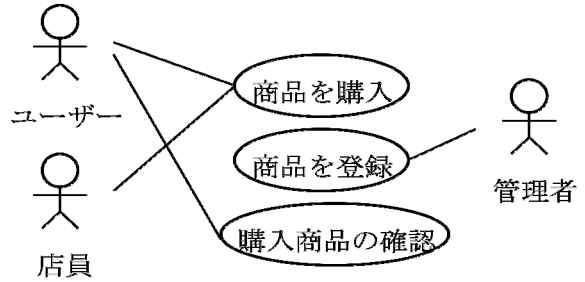

図 4　ユースケース図の例

7.考察

問題解決の方法として、ユーザ側に立った手法が望ましいと思われる。重視する点として、①1日のデータの流れ図、②月次のデータの流れ図、③ユーザの入力画面の 3 次元表記である。

しかしデータを詰め込みすぎた仕様書は見づらく、注意が必要である。予想効果として、①と②で年間のシステムの動きを詳細に把握できる

①と③でユーザ側になった説明ができ、保守的な役割も持てる。

②と③で季節変動するようなシステムの説明ができる。また、仕様変更前、変更後という 2 種類の仕様書を使うことで、変更おける要求定義におけるミスを軽減することができると思われる。同時表記ということで変更前と変更後の仕様を色の濃艶などを使用し同時に表記することにより、より分かりやすくなる。また、最初にユースケース図のような簡易的で大まかな流れを表示し、その部分をクリックすることにより詳細なデータや時系列などを表記するといったツールが出来れば、顧客側、開発側の理解度の差を軽減することができ要求分析としては不十分かもしれないが、確認用ツールとして使用できる。これにより、より正確な要求定義ができる。図 5 は一例として示す。

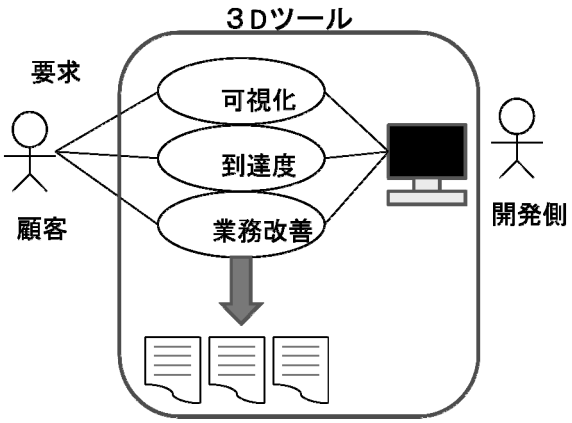

図 5　三次元ツール

8.まとめ

本稿ではソフトウェアの問題点に指摘した。

しかし、実際どのような手法で解決していくかなどの具体的な手法には至っていない。今後、可視化と自動化について研究し、アンケート調査などを実施することにより、さらに研究を進めていきたいと思う。

参考文献

[1] 佐川　博樹:”要求定義の基本と仕組み”,秀和システム,(2010)
[2] (社)情報サービス産業協会:”情報サービス産業白書 2010”,(2010.6)
[3] 山本　修一郎:”～ゴール思考による!!～システム要求管理技法”,ソフト・リサーチ・センター(2007)
[4] 山岸　耕二:”要求開発”,要求開発アライアンス,(2006)
[5] 河村 美嗣, 他, UML を入力とするソースコード自動生成ツールの開発, 情報処理学会, 第 72 回全国大会, (2010)