機能追加と障害修正の実施制御方式

1R-9

新田将人
富士通　(株)

1．はじめに

ソフトウェア製品（特に汎用機のベーシック ソフトウェア）の寿命は長く，絶え間ない機能追加を行う必要がある．既に顧客に導入された製品に機能追加を行うため，開発では定期的に変更データ を提供している．一方，製品は顧客ごとに様々な版数が使用されており，一度障害が発生すると開発では各版数に対して修正データ を随時提供する必要もある．

ここでは，これら２種類の変更・修正データ を提供することでどのような問題が発生するか，そしてそれらの問題をいかに解決したかについて述べる．

2. 機能追加と障害修正

機能追加が発生した場合には，変更データ を提供する．汎用機のソフトウェア製品は大規模であるため，変更データ は製品全体ではなく，機能追加で変更された部分だけ，即ち変更のあるロードモジュールだけを提供し，顧客先でこれを置き換える．

例えば，製品の各版数が図2.1 のように変更されていくとすると，VL２にレベルアップする時の提供物としてはmod2’ とmod3’ を提供し，VL３にレベルアップする時の提供物としては，mod1’,mod3’’を提供する．

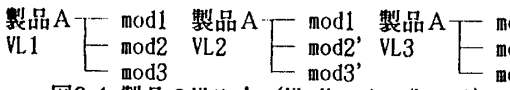

図2.1 製品のVLアップ　(VL:Version/Level)

一方，ある顧客で障害が発生した場合，この障害に対して障害修正データ を提供する必要がある．修正データ は修正実施の早さや情報流通性のよさ（例えば，電話やFAX で修正データ を教えることができる）からパッチ で提供されることが多い．

3. 変更データ と修正データ 提供の問題点

これら２種類のデータ を提供するときにどのような問題が発生するかについて以下に示す

① 顧客先の変更実施で順序性が発生する．

変更部分だけの提供を行うため，顧客では開発された順番で変更を適用する必要がある．

例えば，VL１からVL３にレベルアップする時に，VL３の提供物だけでシステムの変更を行うとmod2’ に置き換わらず製品構成が図2.2 のようになる．このように，論理的に矛盾した製品Ａが作成されることになる．

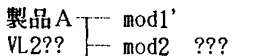

図2.2 製品変更ミスの例

② 修正データ の提供漏れが発生しやすい．

障害が発生した場合には，他の版数を使用している顧客でも同一障害が発生する可能性が高い．このため，他の版数にも，モジュール 変更の状態を意識しながら修正データ を提供する必要がある．

例えば，mod3のように各VLで変更されているロードモジュールには，全版数に修正を提供する必要がある．

③ 顧客の改版時に障害が再発する可能性がある．

変更データ は既に提供されたものであるため，次のVLアップ で変更を行うと前のVLでの修正データ が消える．このため，アップ されたVLに対応した修正を実施しないと障害が再発する．

例えば，図3.2 のようにVL１に障害が発生し，mod3に障害修正（●）を実施したとする．

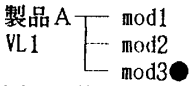

図3.1 障害修正実施例１

この後，この顧客がVL２に アップした場合には，mod3が新しいmod3’ で置き換わってしまい，障害が再発する．このため，図3.2 のように再度VL２のmod3’ 用の修正（▲）を実施する必要がある．

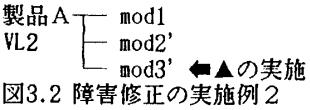

図3.2 障害修正の実施例２

④ 障害修正の反映忘れ

障害修正を最終的には開発中のソース に反映する必要がある．これを忘れると障害が再発する．

例えば，図3.3 のように，障害がVL３の提供後，VL4 の開発中に発生した場合には，いままで述べたようにVL3 までの各版数に障害修正を提供する必要

The Process Control of Enhancement and Modification
Masato Nitta
FUJITSU,Ltd.

がある．

これと同時に現在開発中のVL4 には，この障害修正のｿｰｽ 反映を実施する必要がある．これを忘れると，VL４の変更を実施した顧客で，以前発生した障害の再発という③と同じ問題が発生する．

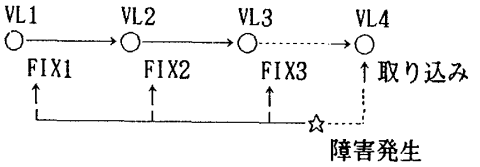

図3.3　障害修正の取り込み

4.　機能追加と障害修正の管理と実施制御

3.で述べた問題を解決するためには，開発で変更・修正の履歴を管理し，これらの情報を提供ﾃﾞｰﾀ とともに顧客に提供し，顧客先で変更・修正の管理及び実施の制御を行う必要がある．以下にその実現方法について述べる．

(1)　開発での管理

開発では，基本的にはﾛｰﾄﾞﾓｼﾞｭｰﾙに対して，版数の履歴管理を行い，修正が発生した時点で，修正の提供が必要な版数を指示する．また，次の版数の修正が行われた段階で修正反映を指示する．

例えば，2.で述べた製品Ａのmod3の場合を例にとると，以下のように開発で情報を管理する．

mod3の変更履歴

版数	状態	修正
VL1	提供	FIX1
VL2	提供	?
VL3	提供	?
VL4	開発	?

図4.1 開発履歴

即ち，mod3を変更した版数を記録しておき，FIX1の修正を作成した段階で，VL2 の修正の提供が必要なことを指示する．これにより，3.の問題②が解決できる．

また，3.の④のような問題に対しても現在開発中のVL4 に修正反映が必要なことを指示する．そして，FIX1の反映が行われるまでは，VL4 の開発が完了できないようにする．

(2)　開発での情報提供

(1)の管理を行っているだけでは，3.の①，③の問題は解決しない．これらの解決には，開発から提供先へ，変更・修正ﾃﾞｰﾀ の実施方法や検査の情報を付加する必要がある．

3.の①の問題に対しては，(1)の管理情報をもとに自分の変更が実施される前に，製品がどの版数まで実施されていなければならないかの情報を提供する．

例えば，VL2 の提供を行う時には，以下のような情報を変更ﾃﾞｰﾀ とともに提供する．

PREMISE　VL1

これは，VL2 の変更実施の前にVL1 が既に実施されている必要があることを指示する．

3.の③の問題に対しては，VL1 の顧客が，VL2 にﾚﾍﾞﾙｱｯﾌﾟした場合に，障害を再発させないような情報を修正ﾃﾞｰﾀ 提供時に提供する必要がある．

例えば，修正FIX1を提供する時に以下のような情報を修正ﾃﾞｰﾀ とともに提供する．

EXCHANGE　FIX2(VL2)
EXCHANGE　FIX3(VL3)

これは，VL2 の変更が実施された時には，FIX2，VL3の変更が実施された時にはFIX3が必要であることを指示する．

これらの情報は，修正ﾃﾞｰﾀ を作成すると，開発の履歴を管理しているﾂｰﾙ が自動生成する．

(3)　顧客での管理

変更・修正の実施は，各顧客により異なるため，各顧客の変更・修正の履歴を管理するﾂｰﾙ が必要である．このﾂｰﾙ は，(2)で生成された情報をもとに変更・修正を行う．

第１に，製品がどのVLになっているかを管理しておき，変更時に(2)のPREMISE 情報の前提ﾚﾍﾞﾙ と一致するかを検査する．

第２に，修正のEXCHANGEの情報を仕掛情報として管理しておき，変更が実施された時に仕掛情報を検査する．

例えば　下の形式で記録しておく．

仕掛製品名	仕掛変更	置換修正
製品Ａ	VL2	FIX2
製品Ａ	VL3	FIX3

図4.2 仕掛情報

VL2 の変更が実施された時に，FIX2の修正が実施されなければ変更が完了しないようにして，障害の再発を防止する．

5.　おわりに

このような仕組みを作ることにより，顧客での変更ﾐｽ，同一障害の再発という非常に大きな問題を解決することができる．■

【参考文献】

新田：「構成管理ﾂｰﾙ PRIMS-X によるｿﾌﾄｳｪｱ製品の品質管理」，『第５回ｿﾌﾄｳｪｱ生産における品質管理ｼﾝﾎﾟｼﾞｳﾑ報告文集』，P131-138，日本科学技術連盟，1985