

反復プロセスと欠陥モデリングによるソフトウェア要因分析の改善

アジャイルな RCA の導入とその効果

永田 敦

ソニー株式会社

atsushi.nagata@jp.sony.com

要旨

RCA (Root Cause Analysis) やその手法の一つであるなぜなぜ分析は、障害の要因を明らかにし、再発防止、未然防止を行い品質改善するために非常に有効な手法である。その一方で、実践の際、時間がかかり、障害の担当者が時に責められることがあった。そうすると、行う回数は少なくなり、分析のスキルは上達しない。結果として、期待した結果が得られないことがしばしばある。本論文では、欠陥を表現するモデルを採用して分析の見える化を行い、短い分析をイテレーティブに行うことで、負荷を緩和することで、より効果的、効率的に障害のメカニズムを明らかにし、そのモデルから対策を策定するプロセスを提案し、その結果を紹介する。

1. はじめに

RCA はソフトウェア品質を改善するもっとも有効な手法の一つである。その目的の一つは、ソフトウェアの欠陥による問題を分析して、問題発生メカニズムおよび欠陥が成果物に入り込むメカニズムを明らかにし、その問題の再発や未然防止をすることである。また、RCA の結果を、*fault-prone* の箇所の推定に使うことによって、優先的および重点的にレビューやテストをすることにより、レビューやテストの効率や効果を上げることが期待できる。当初、我々は、RCA の一つであるなぜなぜ分析[2] を使って要因分析を行ってきた。そのテンプレートを図1に示す。

2. RCA の課題

なぜなぜ分析はその期待結果がある一方で、我々がソフトウェアの障害に対して実行した際に、以下のような問題があった。

- 実施に時間がかかる
- 分析の際、障害に関係する担当者(以下、担当者と呼ぶ)を責めてしまう傾向がある。
- 分析のスキルが上がらず、期待する効果が出ないことがある

ID	Symptoms	Issue				
		Why ①	Why 2	Why 3	Why 4	Why 5
		Root Cause				
		Cause of defect installation				
	References					

図1 なぜなぜ分析テンプレート

2.1. 実施に時間がかかる

なぜなぜ分析では、二つのプロセスを持っていた。

一つは、分析の準備で、担当者に分析のための説明資料や、場合によってはなぜなぜ分析まで行い、その結果を準備してもらう。

もう一つは、用意された資料を基に、関係者を集め、レビューするプロセスである。

まず、分析の準備で、数時間の時間がかかっていた。そして、レビューには、2時間かかっていた。しかし、1回のレビューでは終わらず、追加の資料作りを含めて、複数回のレビューが必要だった。それにより、担当者にかかる負担はトータルで8時間以上かかっていた。

2.2. ストレスのある分析

なぜなぜ分析は、要因分析の中で非常によく知られ、行われているプラクティスである。“なぜなぜ”分析という名の通り、要因分析のカギになるのは“質問の技術”である。Eric E. Vogt [3] は、“なぜ”という問いは、他の疑問詞よりも、よりよく考え、深いレベルの議論を引き出そうとする傾向があると述べている。それは、“強力な問いかけ”だからであり、それは、思慮深い検討を引き起こし、

創造的な思考を喚起するからである。図2に示すように、”なぜ”は、他の疑問詞に比べ最も強力な疑問詞である[3]。よって、なぜを使えば、質問の力も強い。

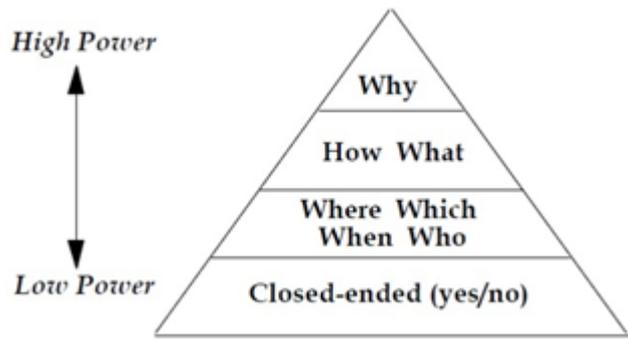
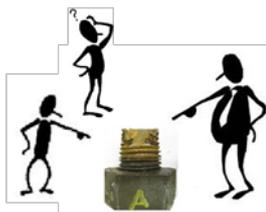


図2 The Art and Architecture of Powerful Questions

もともと、なぜなぜ分析は、トヨタ生産システム(TPS)における生産ラインの要因分析のために生まれたものである。TPSでは、この強力な質問を用いて工場のラインの要因分析に用いて、非常な効果を上げてきた。(図3参照)

TPSオリジナル：製造ライン



欠陥：物理的な問題
対象：現場のモノ

図3 TPSでのなぜなぜ分析

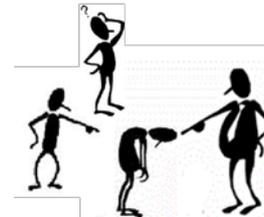
生産ラインにおける欠陥は、物理現象、化学現象による要因で物理的な問題で、分析の対象は、現物である。もちろん、その要因には人間のミスもあるが、それをポカと呼び、ポカよけと言って、ミスを起こさない工夫をしている。

2.3. 分析のスキルが向上しない

一方で、ソフトウェアの欠陥においては、その調査の対象は人間の過ちである。ポカよけを考えるには、人間の過ちそのものを、より分析しなくてはならない。そこで、人間に“なぜ”を投げかけることになる。Eric E.Vogtは、次のように述べている。“なぜ”という疑問文を人に使うときは、気を付けないと、簡単に防御的な反応を引き起こしてしまう(図4参照)。それは、調査をしていこうという

意思よりも、彼らの答えを正当化しようとするからである。たとえば、“なぜ思っていることをちゃんと伝えてくれないんですか”とか“なぜ、そのようにしたのですか”という問いは、新たな情報を得る可能性を広げる代わりに、相手をして自らの保身と過去に下した決断を正当化することに駆り立ててしまう。[3]

ソフトウェア開発現場



欠陥：バグ
対象：エンジニア

図4 単純にソフトウェア障害でなぜなぜをすると

つまり、防御モードになった時に出される答えは、図5のように、本当に求めている方向からずれてしまう恐れがでてくる。そのずれた答えに対し、さらに機械的に”なぜ”という質問をしまつと、さらにずれてしまう。強力な質問のために、分析の方向が制御できていないのだ。そこから出てくる結果は、真の原因をとらえていないため、そこから出てくる施策も効果的なものが出てこないことがしばしば起こっている。

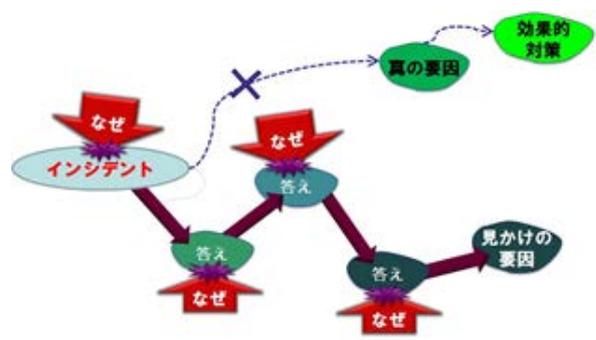


図5 防御モードでのなぜなぜ分析

つまり、RCAで必要なのは、“なぜ”という強力な疑問詞を使った質問を多用するのではなく、状況にあった適切な質問を考え出すことだ。これは、“なぜ”を使ってはいけないということではない。いかにして、人の心をオープンにしなが、本当に知りたいことを聞き出すことができるかということだ。それでは、どうしたら適切な質問を生み出すことができるだろうか。それには、知識、

知見や分析の見える化の仕組みも必要だが、質問のスキル、ファシリテーションスキルも必要になってくる。

2.4. 分析のスキルが向上しない

RCA の実行に非常に時間がかかるということは、RCA を頻繁に行うことができなくなる。ましてや、そのたびに責められるストレスを担当者が受けるのであれば、モチベーションが上がらないため、ますます RCA を行う頻度は少なくなる。2. 2節で述べているように、RCA のカギになるのは、適切な質問を作り出すことである。いかに深く分析できるかは、この質問力にかかっている。しかし、“適切”な質問を出していくことは難しく、机上で向上できるものではない。つまり、できるだけ訓練する場が必要なのだが、肝心の RCA を実施する機会が少なければ、実践をし質問のスキルを磨くことができない。そうすると、RCA をしても期待する効果、結果が出ないのでますますやらなくなってしまふ。

3. 改善1 適切な質問を出していく工夫

オリジナルのなぜなぜ分析テンプレート(図1)では、表現するのは答えだけである。質問は、“なぜ”がデフォルトだからである。もちろん、なぜを使ってはいけないというのでは決してない。その時、その状況で最適な質問は何が良いのかを考えることが重要なのだ。回答者から多くの情報を引き出すためには、回答者がオープンな気持ちになっている必要がある。そのためには、質問する我々は相手から信頼されていなければならないだろう。そのような関係を保ちつつ、核心に掘り進んでいくためには、ファシリテーションの中心である質問を吟味していかなければならない。質問をテンプレートに明記することは、質問が適確なものをレビューすることを可能にし、その質問自体にフィードバックをかけ、質問の改善ができるのだ。もちろん、回答する人にとっては、質問を正確に理解することができる利点もある。

そこで、テンプレートを図6のように、質問と回答とを組にして表現するようにした。質問は、“できるだけ”なぜ”を使わないようにし、前の質問の答えから、戦略を立て、狙いをつけて質問を考えるようにした。つまり、強い質問によって翻弄されることなく、論理の通った質問と答えの連鎖ができるよう心掛けた。

ただ、適確な質問を立て続けに出していくことは難しい。質問と答えの連鎖が長く伸びてしまうと、その論理性を保ちながら、さらに次の質問をリアルタイムに出していくことは次第に困難になる。それを無理に続けようとしても、質問の質が落ちれば分析を掘り進めることができない。しまいには結論を焦り、生煮な状態で対策案

1st Question	2nd Question	3rd Question	4th Question	5th Question
ID	ID	ID	ID	
Question	Question	Question	Question	
Answers	Answers	Answers	Answers	ID N/A
ID		ID	ID	
Question		Question	Question	
Answers	ID N/A	Answers	Answers	ID N/A
		ID	ID	
		Question	Question	
		Answers	Answers	ID N/A

図6 改善1: 質問を明示的に表現する

を考えたり、テストで漏れてしまった要因を探ろうとしてしまうことがある。このような状態では、満足する分析結果は得られない。

4. 改善2: 分析ミーティングを短くし、繰り返し行う

4.1. 改善1での問題

テンプレートを担当者に渡して分析をまかせ、その結果をレビューするというプロセスをとる場合がある。すると担当者は、横方向にある一つの枝をいっぺんに掘り進めようとする傾向がある。それでは改善1で質問が見える化の工夫をしても、最初の質問が適切でなかったばあい、後の分析は無駄になってしまう。

4.2. イテレーティブ RCA

この改善は、ミーティングの時間を短くし、それを定期的に繰り返すようにしたことである。この理由は二つあり、一つは、回答者の負荷を軽くするため。もう一つは、前の節で述べた、適確な質問を出していくためである。

ミーティングは、1日1回で長さは15分から最長30分、なるべく、毎日同じ時間に行う様にする。

事前資料を要求せず、最初のイテレーション内で、何が起こったかのみを話してもら(インシデント分析)。それをシートに書いて、次の質問を2回目のイテレーションまでに考える。

2回目のイテレーションから、考えてきた質問を皮切りに時間内に分析を進めていく。イテレーション内では、質問の答えに対して、不明点やその背景を聞き、回答を整理しまとめていく。3回目以降、これを繰り返してい

く、そのイメージを図7に表す。

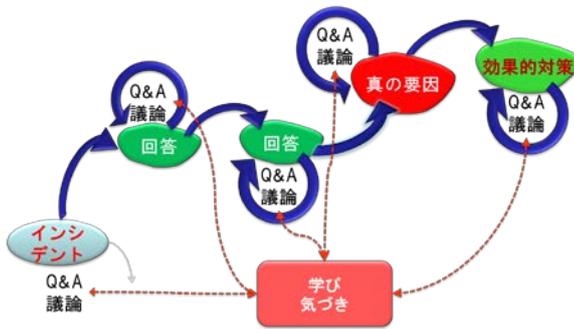


図7 イテレーティブ RCA の流れ

4.3. 改善点と課題点

この施策は、担当者にもこの施策により、回答者の負担が改善したが、より大きなことは質問の改善だった。分析ミーティングでのレビューアは、質問を出し、その答えを吟味して適切な次の質問を繰り返していかなければならない。

適切な質問を策定する方法は二つ考えられる。一つは **Heuristic** な方法で、もう一つは分析的な方法である。前者はすぐに質問を出せるが的を外れる場合もある。後者は的を外すことは少なくなるが、それなりの時間がかかる場合がある。分析ミーティングでは、レビューアは担当者を待たせてじっくり分析をする余裕はないので、**Heuristic** な方法をとってしまうことが多い。しかし、ミーティングが 30 分を過ぎると、レビューアは集中力を失っていき、そう簡単に **Heuristic** に適確な質問を続けられなくなる。やがて質問の質が落ちていき、的はずしたり真の原因にたどり着く前に質問が出せなくなったりすることがしばしば起こっていた。すると、矛先を原因追求から、そのトラブルがなぜ漏れたのかというテストのほうに向けることになる。しかし、真の原因にたどり着いていないので、結局適切な対策にたどり着かない。

それに対し、分析ミーティングにタイムボックスを使うことによって、質問の質を落とさず集中した分析で終え、そのあとに、分析的な方法で質問を考える時間をとることができる。タイムボックスでイテレーティブに行うことにより、質問の質も改善することができた。さらに、質問と回答の場は、参加者に気づきや知見の伝達ももたらしていた。

一方、このように質問に時間をとっても適切な質問が出せないことがしばしばあった。

その原因として、分析の見える化の仕組みを考えた。図6のようなテンプレートでは、原因と結果が木構造でために利用しているのだ。

連鎖したモデルになる。しかし、実際は、背景や、引き金になるものなど、属性と関係を持っており、その組み合わせで障害のメカニズムを表す必要を感じていた。まとめると次の3つのようなになる。

- 障害のメカニズムは思ったより複雑である。
- 質問から得られる応答は、要因ばかりでなく、いくつかの属性を持った因子になる。
- それぞれの因子は、関係を持ち影響を与えている。

5. 欠陥モデリング

2013年、ソフトウェアテストシンポジウム東京2013 (JaSST Tokyo 2013)において、細川 宣啓氏、野中誠氏、西 康晴氏、原 佑貴子氏、嬉野 綾氏によるプロジェクトファブル(Project Fabre)によって、欠陥エンジニアリングの紹介があり、そこで欠陥モデリングが提案された。[4]

なぜなぜ分析では、障害の真の原因を求めるために、表現されているのは質問とその答え(その質問に対する原因)であった。欠陥モデルは、真の原因の代わりに図のような”欠陥”とそれに関連する因子で組み合わせたモデルで、障害の起こるメカニズムと、それを引き起こす欠陥が入り込むメカニズムを表している。

5.1. 欠陥モデリングの要素

Project Fabre では、欠陥モデルの要素として以下のように定義されている。

- 表出現象
欠陥によって引き起こされる不具合・障害
- 欠陥
成果物に含まれた、人間の思考の過ちが具現・表出化したもの
- 過失因子
人間の思考や判断の誤りそのものこと。
- 誘発因子
成果物の中に含まれる、人間の思考の誤りを誘発する“トリガー”となる要素のこと
- 増幅因子
過失の連鎖を助長し、欠陥の混入確率を増幅させる要素

ここで注意しておきたいのは、アジャイル RCA の成果物として作られた欠陥モデルは、Fabre で求めている欠陥モデルではないということである。このモデルの機能の一部を使って、あくまでも障害分析の見える化のために利用している。

6. アジャイル RCA

4.3 で述べた、イテレーティブ RCA の問題を解決するため、図7のようにそれぞれのイテレーションの間にモデリングプロセスを置いた。これをアジャイル RCA と呼ぶことにした。図8はそれをフローダイアグラムとしてあらわしたものである。

プロセスには4つのループがある。

- インシデント分析ループ
- 探索的分析ループ
- アジャイル RCA ループ
- 対策ループ

である。

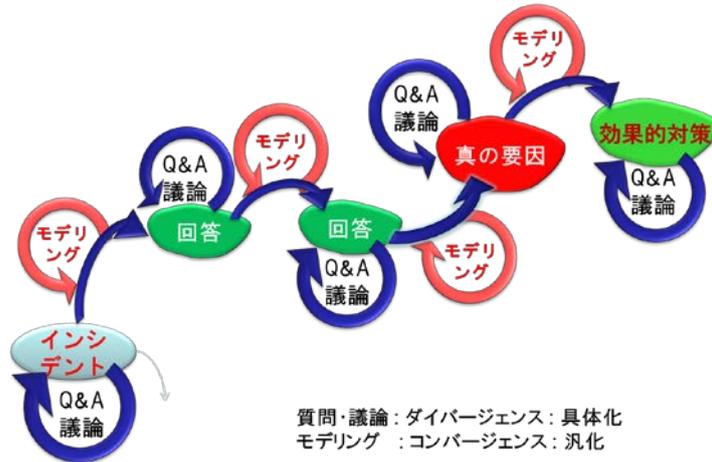


図7 アジャイル RCA のイメージ

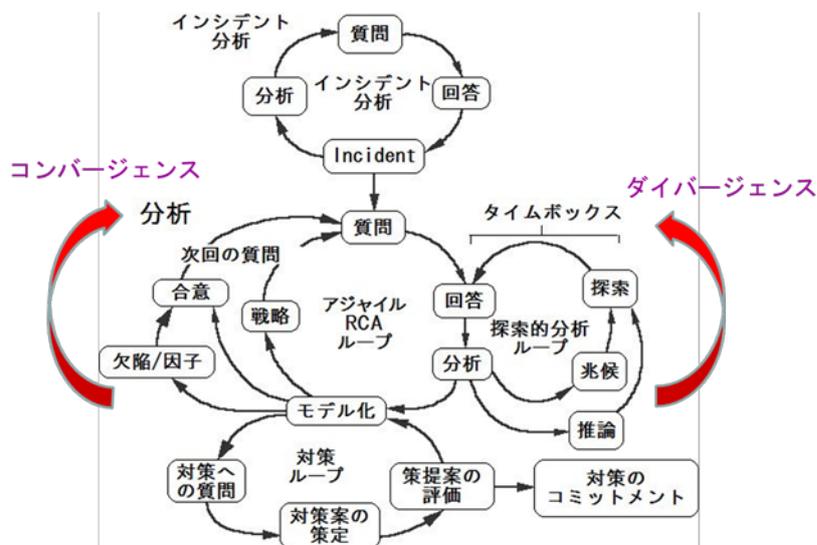


図8 アジャイル RCA プロセス

6.1. ロール

アジャイル RCA のプロセスの説明の前に、プロセスでのアクターであるロールを説明する。3つのロールがある。

- モデレータ
- 担当者

- レビューア

モデレータは、アジャイル RCA のプロセスとルールを理解している。質問を作成し実際に質問をし、分析、モデルの作成を行っていく。

担当者は、障害に関係する直接の担当者である。

レビューアは、モデレータとともに、質問の作成や質問をし、モデル作成を行う人である。

6.2. インシデント分析ループ

最初のイテレーション(初日)は、イテレーティブ RCA と同様にインシデント分析を行う。担当者から、インシデントの情報を正確に聞き出す。図9参照。

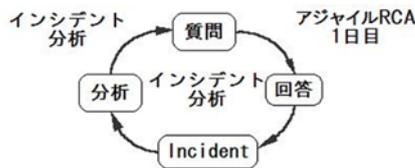


図9 インシデント分析ループ

注意することは、ここでは分析に踏み込まないことだ。インシデントを明確にしていくと、なぜそうなったのか、分析の強い疑問が湧いてしまうことがある。そこで、分析の質問を始めてしまうと、その質問が適切であるかのチェックなしに進んでしまう。それでは、以前と同じことになってしまい、効果がでない。しかも、インシデント自身の時間が短くなり、インシデント分析が不十分になるかもしれない。もし不十分な場合、例えば、その障害の起きた背景、必要なドメイン知識の詳細、そこで使われているプロセス、リソースの状態などが理解されていない場合には、分析者の経験や知見でのバイアスが影響してしまい、欠陥の探索の範囲が、狭まったり違ったりするために、適切な質問ができなくなってしまう。

6.3. 初回の探索的分析ループ

初回の探索的分析ループ、つまり2日目のアジャイル RCA のイテレーションである。

分析チーム、つまりレビューア、担当者、モデレータが集まり、15分ないし30分のタイムボックスの中で質問と回答、議論を繰り返して分析を行うプロセスである。(図10参照)モデレータやレビューアは、担当者の答から、欠陥やそれに関係する因子の兆候をとらえ、または、それらに対する推論をして、次の質問をしていく。イテレーションの間に、質問と応答の連鎖をできるだけ多く回して、図8にあるようにダイバージェンスしていく。

たとえば、ツールを使い、モデル上に、どんどん書き込んでスクリーンに映し出したり、模造紙に、付箋で質問と答えを書きながら貼り付けたりして見える化いくとよい。これにより、耳で聞いて認識、理解をするだけでなく、目でイメージから認識をして理解していくので、記憶を思い出し、思考を助長する効果があると考える。また、記録として残せることにもなる。

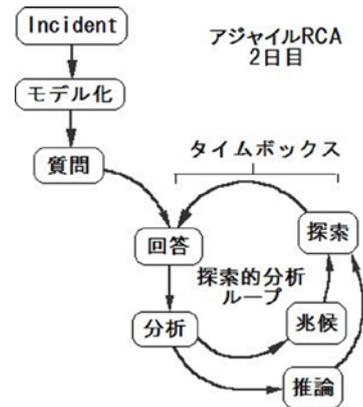


図10 探索的分析ループ

6.4. アジャイル RCA ループ

探索的分析ループで出た出力を持ち帰り、それらを因子に分類整理し、関係をリンクで表現してモデルを作っていく。因子どうしの論理性と、全体の論理の流れを確認することでより根本的な問題の兆候が見えてくる場合がある。その問題を仮説として、もしそうだったらどのような兆候がほかに出てくるかを推論して、それを質問事項としていく。図11参照

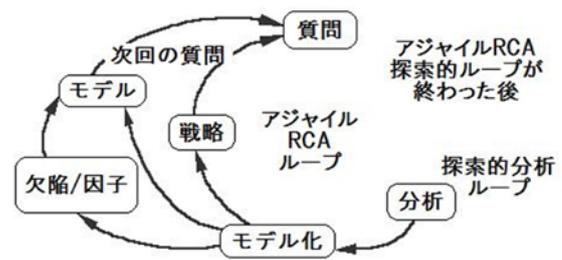


図11 アジャイル RCA ループ

探索的分析ループでは、回答をできるだけ漏らさぬようにメモをしていくのがよいが、内容としては重複していたり冗長であったり重要でないものもあり、それらを整理して、できるだけ少ない因子でシンプルに表現する。因子の内容表現も、より簡潔で正確なものにしていく。図8のように、モデリングをするときは、コンバージェンス、つまり収束させる方向でまとめていく。

6.5. 2回目以降の探索的分析ループ

アジャイル RCA ループで作られた欠陥モデルや質問を用意し、2回目の探索的分析ループに入る。2回目以降はこれと同じである。

まず、担当者にモデルを見せて、その合意をとることから始まる。違いの指摘を受けることもあるが、すぐに終わる。この欠陥モデルの合意をしたことにより、新たな視

点で分析をすることができる. 図12参照

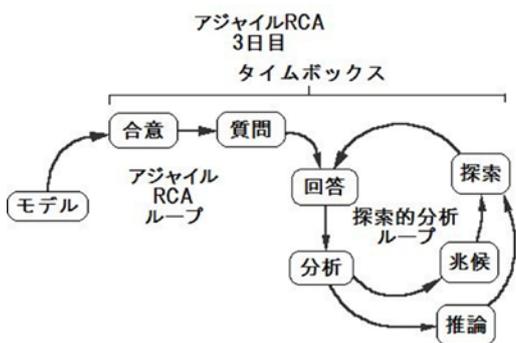


図12 2回目以降の探索的分析ループ

それ以降は、用意していた質問をきっかけに、1回目と同様に探索的分析ループを実行していく。

6.6. 対策策定ループ

何回か、探索的分析ループとアジャイル RCA を交互に行った後、モデルは障害のメカニズムを表すようになる。そこで担当者に対し、対策の質問をして、対策案を提案してもらう。その妥当性を議論しながら、実行していく対策を担当者にコミットしてもらい、プロセスを終える。図13参照。

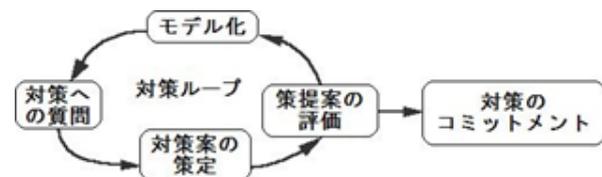


図13 対策ループ

7. 事例

6章で示したプロセスによって得られる欠陥モデルが、イテレーションごとに変化しながら、欠陥のメカニズムを表し、対策策定まで持っていく様子を、事例を用いて日別に示したものを以下に示す。

7.1. 一日目

図14に示すように、インシデントとそこからわかる二つの欠陥を示している。この例では、インシデント分析をやっていく中で、欠陥がわかってしまった。設計は最初 QA の報告を、仕様であると返したが、仕様を作っているチームが、これは仕様には載っていないうえに、設計の実装が違っていることを指摘していた。

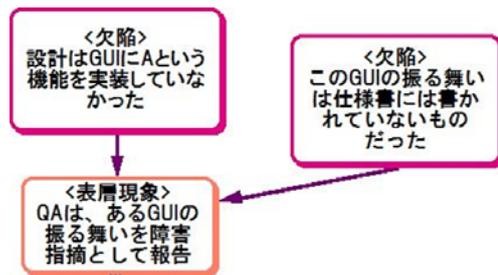


図14 第一日目の欠陥モデル

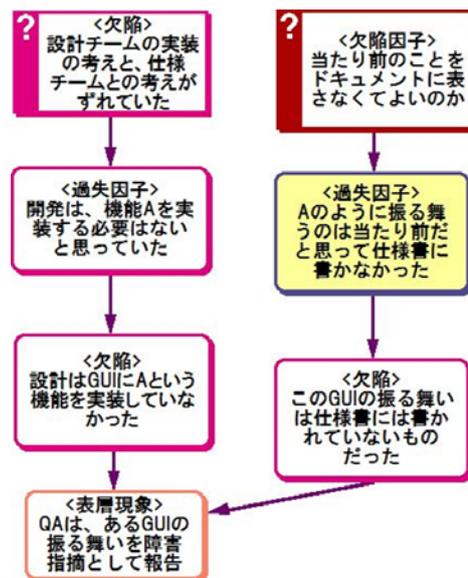


図15 二日目の欠陥モデルの結果

7.2. 二日目

図15のように、“当たり前のことは仕様書には書かない”という話を得られた。これは暗黙知で共有している知識や判断基準は、あえて書く必要がないという心理から来ているが、それにより、認識の齟齬が起きていることに気づかないところを過失とした。その先に、このような“当たり前”という暗黙知をどうした良いかを質問としてモデルに書いている。

7.3. 三日目

図16のように、さらに分析が進んだ。暗黙的な仕様について共有出来ていないことがわかり、設計と仕様チームとの判断基準が違うことが分かった。

7.4. 最終日

最終的には図17のようになった。担当者は、振る舞いについてのポリシーがないことがわかり、施策として、仕様チームがポリシーを書くことにした。また、仕様についての理由や背景も書くことをコミットした。

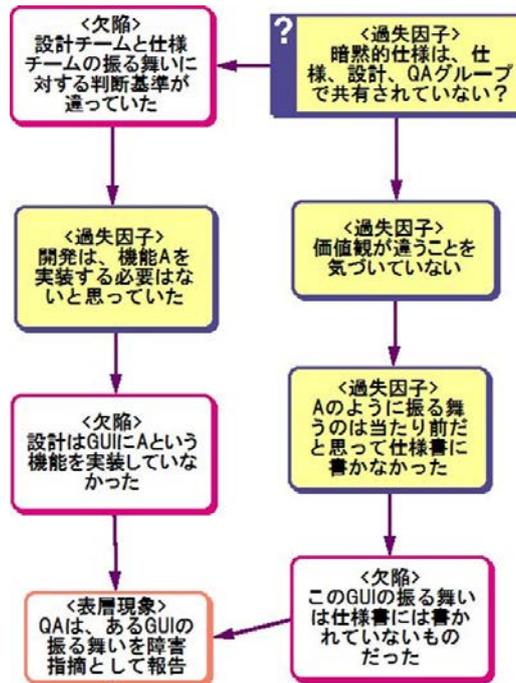


図16 三日目の欠陥モデルの結果

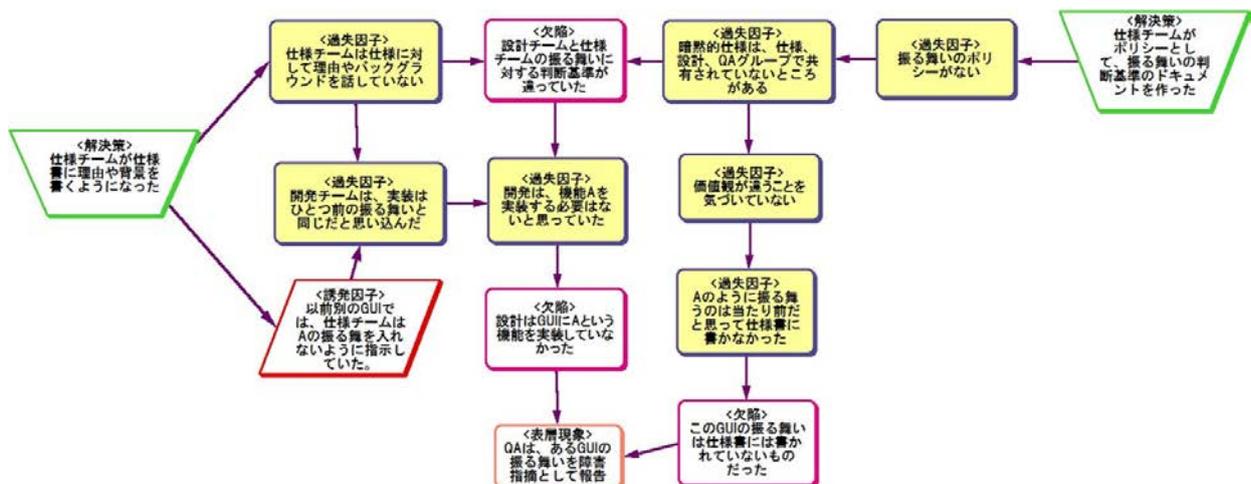


図17 最終日の欠陥モデル

8. 結論

8.1. 旧方式の方法での結果

図1のテンプレートを使っていたころの分析結果は表1のようになっていた。

表1 実行結果

件数	真の原因	分析ミーティング (平均 分)	準備時間 (平均 分)	トータル時間 (分)
6	1	340	200	540

分析ミーティングは、関係者が集まり、なぜなぜ分析を行い、質問の回答をテンプレートに書き込んでいく会議である。その当時は1回2時間の会議を、3回程度行う必要があった。それに伴い、担当者の準備時間が増えていった。トータル時間は、分析のために担当者がかわったすべての時間で、平均して9時間に及んでいた。

8.2. 実行時間の削減

対象は、アジャイル開発をしている A,B の2チームで行った。A チームは午後決まった時間、B チームは、夕食後決まった時間に行った。

まず、担当者が事前に準備時間は、必要なくなった。報告のために簡単な説明を別途作っている場合は、それを参考に使わせてもらう程度だった。

表2は、二つのチームで行った時の結果である。

表2 実行結果

	課題数	イテレーション	欠陥数	因子数	トータル時間 (分)
チームA	6	4	2	15	103
チームB	7	4	4	21	133

課題数 : 扱った課題の数
 イテレーション : 一つの課題に対するイテレーション回数の平均
 欠陥数 : 一つの課題に対する欠陥の数の平均
 因子数 : 一つの課題に対する因子の数の平均
 トータル時間 : アジャイルRCAのトータルの時間の平均。

4回から6回のイテレーションで済んでおり(対策ループを入れると、もう1回増える)トータルで2時間ないし3時間で終わっている。これは、今までの方法より、3倍から4倍速く終わっている。

欠陥は、平均して2個ないし4個程度であるが、それに関係する因子数の平均は、15および21であった。複数の欠陥(真の原因)が確実に見つかっており、それに伴う多くの因子がメカニズムを表していることがわかる。

8.3. 継続性

両チームとも、6か月以上継続して行っていった。半年で、13の障害をトータル56のイテレーションで分析した。大体、月あたり1回ぐらいのペースに見えるのは、相手のスケジュールの都合で休みが入ったりした為である。しかし、開発中の忙しい中で、このように継続できたのは、アジャイル RCA の軽量性と、確実に結果を出していく実績と、それによる信頼関係の構築によるものである。A チームはその後、自分たちで行うようになり、私の手を離れた。

8.4. 欠陥メカニズムの気づき

欠陥モデルにより、欠陥のメカニズムは、より複雑であることが理解できた。これは、最初から過失を防ぐことが容易くできるものではないことを、現場の人たちと共に学んだ。もし同じ条件、同じシチュエーションになったときは、同じように間違えを犯してしまうだろう。

8.5. 学習、伝達および記憶の改善

たとえ数日間、アジャイル RCA のイテレーションの間隔が空いたとしても、担当者に欠陥モデルを見せながら数分説明するだけで、状況を思い出して分析を再開

することができた。また、途中から入ってきた人に対しての説明にも有効に使うことができた。

アジャイル RCA で得られる知見は、他の組織のチームにおいて、警告としてガイドすることができた。たとえば、ある障害から、構成管理による問題の欠陥モデルを分析結果として得た。そのモデルの誘発因子から導かれる予兆の情報を使って、他の組織のチームにおける構成管理の欠陥の予兆を進捗会議でつかんだ。そこで、その欠陥から推定されるインパクトをもって警告することにより、構成管理の改善を促すことができた。

9. 考察

9.1. 分析の二つのタイプ

アジャイル RCA のプロセスでは、探索的分析ループとアジャイル RCA ループをイテレーションごとに交互に使っている。

図8のように探索的分析ループは、返ってきた答えに対して、直感的に予兆を探り、推測と仮説を使って次の質問を出して、短い時間に集中して情報を得ていく。そのため、発散的なダイバージェンスタイプの分析になっている。一方、アジャイル RCA ループは、発散した情報をじっくりとモデルに変換し、インスタンスとのリンクを取りながら汎化、抽象化していく。その際、重複したり、重要ではないものはできるだけそぎ落として、エッセンスを残していく。つまりコンバージェンスタイプの分析になっている。次の探索的分析ループを始める前に、ここでできた欠陥モデルを使って、担当者と合意をすれば、より分析するポイントが明確になり、より深い分析をすることが期待できる。これが、従来のなぜなぜよりも早くできる要因の一つだと考えられる。

9.2. 欠陥モデル

Project Fabre [4] で本来求められている欠陥モデルは、インスタンスを含まず、抽象的な表現になっている。しかし、現場における障害の分析というアジャイル RCA の活動の目的から、欠陥モデルにインスタンスも便宜上入れざるを得なくなっている。これは、欠陥モデルに対する目的が異なっているからである。

9.3. なぜ、アジャイルというか

図8のように、アジャイル RCA は、探索的分析ループと、アジャイル RCA ループが、欠陥モデルを用いてフィードバックを回しながら、分析をイテレーションごとに改善しながら、ゴールに向かっているプロセスといえる。この考えから、アジャイル RCA と名付け提案している。

10. 課題

10.1. 欠陥モデル作成がまだ時間がかかる

30分の探索的分析ループでの結果をまとめるのに1時間ほどかかることがあり、さらなる工夫が必要である。

パターンを使うとより早くなるので、パターンを作っていくことが次の課題になっている。また、欠陥モデルや因子の整理をし、アジャイル RCA 用の欠陥データベースを考える必要もある。

10.2. 欠陥モデルの表現と剪定

剪定の技術がまだ不足で、欠陥モデルが複雑になりすぎてしまう。一見するとマインドマップのように見えてしまい、担当者との間では、理解できても分析に参加していない人を見ると理解しづらくなっている。これは、一緒に共有しながら分析を行っているときに暗黙的な知識が共有され、補っていると考えられる。しかし、この欠陥モデルは、第三者にも知見として伝えていかなければならない。そのためには、もっと理解しやすい、よりシンプルな表現にしていかなければならない。

10.3. モデレータ、レビューアの実験知識やプロセスの知識

ドメイン知識やプロセスが違うところでアジャイル RCA をやる機会があったが、そのドメイン知識やプロセスの知識を知らないために、バイアスがかかってしまい、誤った方向の質問を出して分析がはずれてしまい、手戻りが起こってしまった。これは、インシデント分析の不足とも言える。違うドメイン、プロセスでは、インシデント分析でより多くの情報を聞き出すか、事前に調べて、バイアスがかからないようにしていかなければならない。

11. まとめ

何段階かの改善を経て、アジャイル RCA にたどり着き、一定の改善効果を得ることができた。

現在弊社では、一部のアジャイル開発 (SCRUM) の振り返りで分析が必要な問題に対して、アジャイル RCA を使っている。現場の設計者に負荷をかけずに分析を行い、適切な施策をフィードバックできるので、振り返りをイテレーションごとに行う SCRUM では、アジャイル RCA を継続して実施するようになる。これにより、開発の時点から、効果的な改善を継続的に行うために、アジャイル RCA を行うモチベーションが生まれた。

また、従来よく使われている市場問題の分析にも、アジャイル RCA が使われるようになってきた。人を責めず、効果のある分析を早く行うことが認められ、ソフトウェアばかりでなく、システム、ハードウェアの問題でも使われ、効果を上げている。

一方、欠陥モデリング自身は、因子や表現方法などに改良の余地が多くある。分析するための欠陥モデルと、報告のため、蓄積のため、第三者に知見として使うための欠陥モデルは、使う人によって変わっていかなければならないかもしれない。

また、モデレータの育成、適切な質問をどのようにやっていくかということも、今後考えていかなければならない課題である。

12. 謝辞

アジャイルの本質を教えていただいた、Tom Gilb, Kai Gilb に感謝する。

欠陥エンジニアリングおよび欠陥モデリングの知見を教えていただいた、細川 宣啓氏、西 康晴氏、野中 誠氏、嬉野 綾氏、原 佑貴子氏に感謝する。

参考文献

- [1] Ram Chillarege., Orthogonal defect classification-a concept for in-process measurements, Software Engineering, IEEE Transactions on (Volume:18, Issue: 11), <http://www.chillarege.com/articles/odc-concept,1992>.
- [2] 小倉仁志, なぜなぜ分析 10 則一真の論理力を鍛える” 日科技連出版社 (2009/03).
- [3] Eric E. Vogt, Juanita Brown, and David Isaacs, THE ART OF POWERFUL QUESTIONS: Catalyzing Insight, Innovation, and Action, Whole Systems Associates, CA, USA, 2003
- [4] Nobuhiro Hosokawa, Yasuhiro Nishi, Aya Ureshino, Makoto Nonaka, Yukiko Hara, 過失に着目した欠陥のモデリング, JaSST 2013 Tokyo – Project Fabre, 2013
- [5] Tom Gilb, Fundamental Principles of Evolutionary Project Management, INCOSE, 2005
- [6] Andersen, B., Fagerhaug, T.: Root Cause Analysis: Simplified Tools and Techniques. ASQ Quality Press (2006)
- [7] Tomomi KATAOKA*, Ken FURUTO and Tatsuji MATSUMOTO, The Analyzing Method of Root Causes for Software Problems, SEI TECHNICAL REVIEW · NUMBER 73 · OCTOBER 2011
- [8] Duke Okes, Root Cause Analysis, The Core of Problem Solving and Corrective Action, ASQ Quality Press, Wisconsin, 2009
- [9] Isao Hayakawa et al., Software Quality Symposium 2008, ‘Applying “ask why five times” method on software development,’ pp. 185-194