テスト、設計、自動化と。

seki@ruby-lang.org

author of dRuby, Rinda, ERB and Drip

なぜ私の話を聴くとお得か

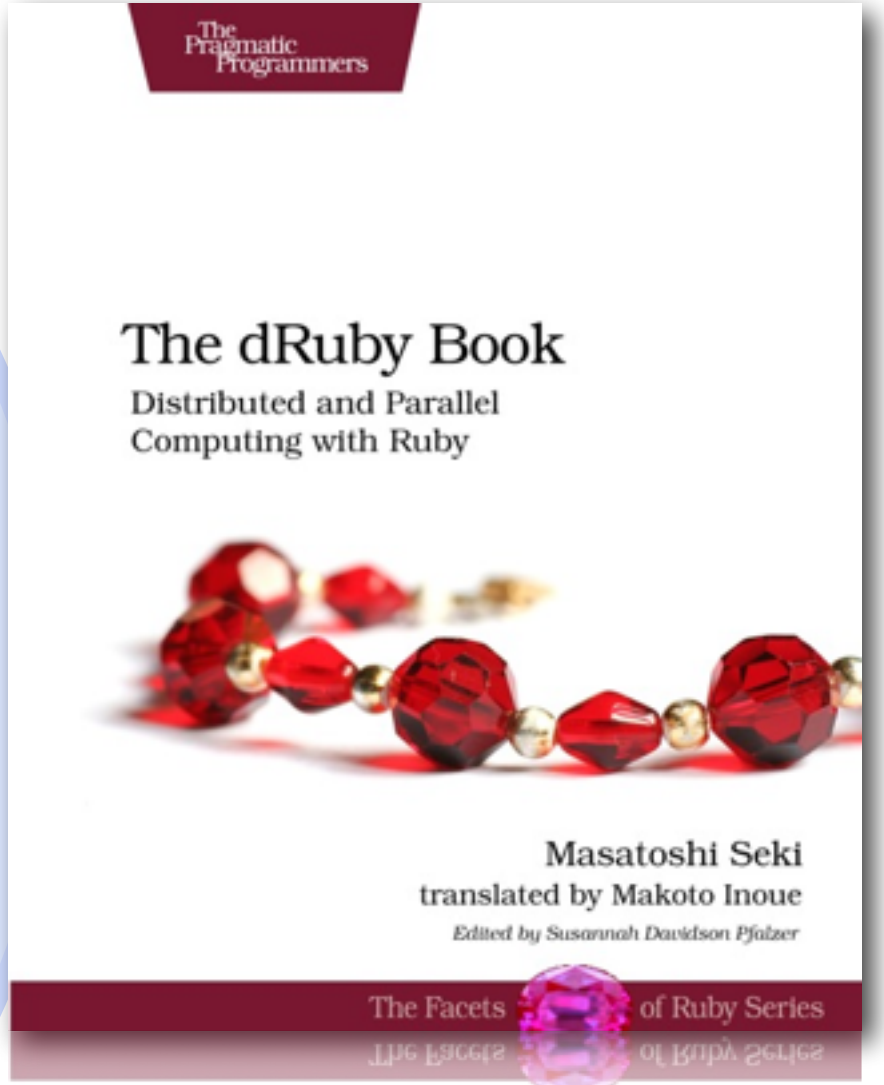
- XP長いよ
  - JaSST'04 忍者式テスト
  - 組み込み/C/C++
  - もうすぐ550イテレーション
- 浅い探究心
- 伝説上の存在

伝説上の存在

Learn from **legendary** Japanese Ruby hacker Masatoshi **Seki** in this first English-language book on his own Distributed Ruby library. You'll find out about distributed computing, advanced Ruby concepts and techniques, and the philosophy of the Ruby way---straight from the source.

The dRuby Book

- PragProgでbuy now!



とちぎテストの会議3

- いかすテスト天国（仮）
  - 2014-10-04 （仮）
  - 栃木のどこか
- わりといかす予定

今日の話

- 設計とテスト
- TDDのおさらい
- テストの自動化・計算機支援
- TDDの変形と開発ライブ

設計とテストは別なの？

- 旧世紀からの強い疑問
- どこからどこまでが設計なのか
  - 設計とは企み全てじゃないのか
- 残す価値のある設計情報はなにか
  - テストぐらいじゃないか

設計とテスト

- 最近買いました!!!
- 俺の設計とテスト
- 気が合いそう

ソフトウェアの開発

ソフトウェアの開発は，２種類の作業を含んでいる．
設計とテストである．

(ソフトウェアの信頼性 p.33)

設計とはなにか

「計画に従って形を整えること．」
この意味をとれば，要求や目的を定めることに始まり，プログラムを書くことに至るソフトウェアの仕事全体にわたり，それぞれの場面に応じて，設計があると考えることができる．

(ソフトウェアの信頼性 p.33)

設計とはなにか

- あらゆる行為は設計である
  - 要求、仕様、コーディング…

テストとはなにか

エラーを発見する目的でプログラムを実行する過程がテストである.

(ソフトウェアの信頼性 p.194)

よいテスト

- エラーを発見できたら成功
- エラーを発見する確率の高いものが良い

テストの心理学

- エラーがあることを示そうとすれば、エラーはたくさんみつけられる
  - エラーがないことを示そうとすれば、エラーはあまりみつからない

ソフトウェアの開発は

- 二つの態度でできている
  - 整理し世界を構築していく設計
  - それを疑い破壊するテスト

おまけ あるチームの心理

- 理想と現状との差を埋めよう
  - バグとか仕様とかあんまり気にしない
  - (想像上の) よりよい製品との差
    - 使いやすさ、見た目、性能、手触り…全て
  - 差を探す行為 - テストかなぁ
  - 差を埋める行為 - たぶん設計

エラーとは何か

1.2 エラーとは何か

よくある定義の一つは，仕様の通りにソフトウェアが実行されない時…

(ソフトウェアの信頼性 p.5)

根本的な欠陥がある

1.2 エラーとは何か

よくある定義の一つは，仕様の通りにソフトウェアが実行されない時にソフトウェアのエラーが生じたとするものである．この定義には根本的な欠陥がある．

(ソフトウェアの信頼性 p.5)

知ってた

仕様が常に正確であることを前提にしている．この前提が正しいことのほうが，むしろ稀である。

(ソフトウェアの信頼性 p.5)

こんな話意味あんの？

- 自分たちの行動がなんなのか理解してみる
  - 理解の枠組み
  - やり方を変える指針に

しばらくお待ちください

- ✓ +10:00すぎ
- ✓ マイヤーズの設計とテストを自慢
- ✓ つぎは「TDDおさらい」

テストの自動化

- キーノートで与えられたテーマ
- テストの自動化を想像できますか？
- 設計の自動化はどうですか？
- そもそも自動ってなに？
  - 自動実行
  - 計算機による支援

テストの自動化

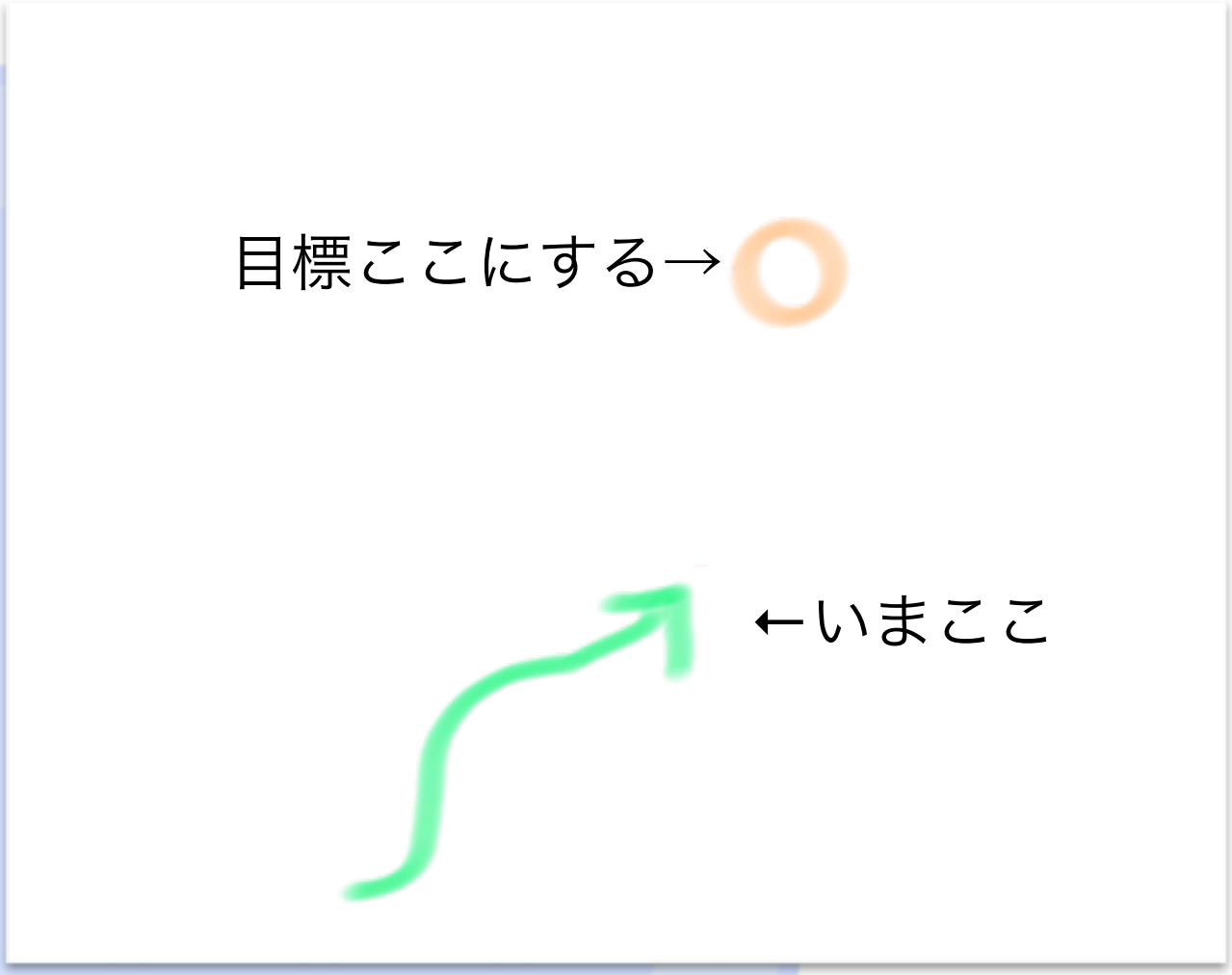
- てゆかテストの自動化がゴールなの？

TDDのおさらい

- テストの自動化の例としておさらいしたい
  - テスト駆動開発
  - テストに導かれたプログラミングの技法
  - 設計のテクニック

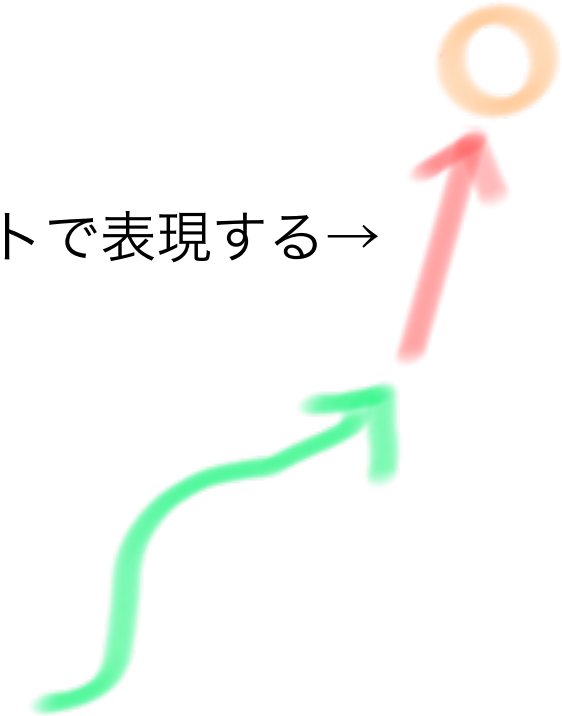
TDDのサイクル

- 次の目標を考える
- その目標を示すテストケースを書く
- コードを書き、テストをパスさせる
- テストが通るままで、リファクタリング
  - 繰り返し

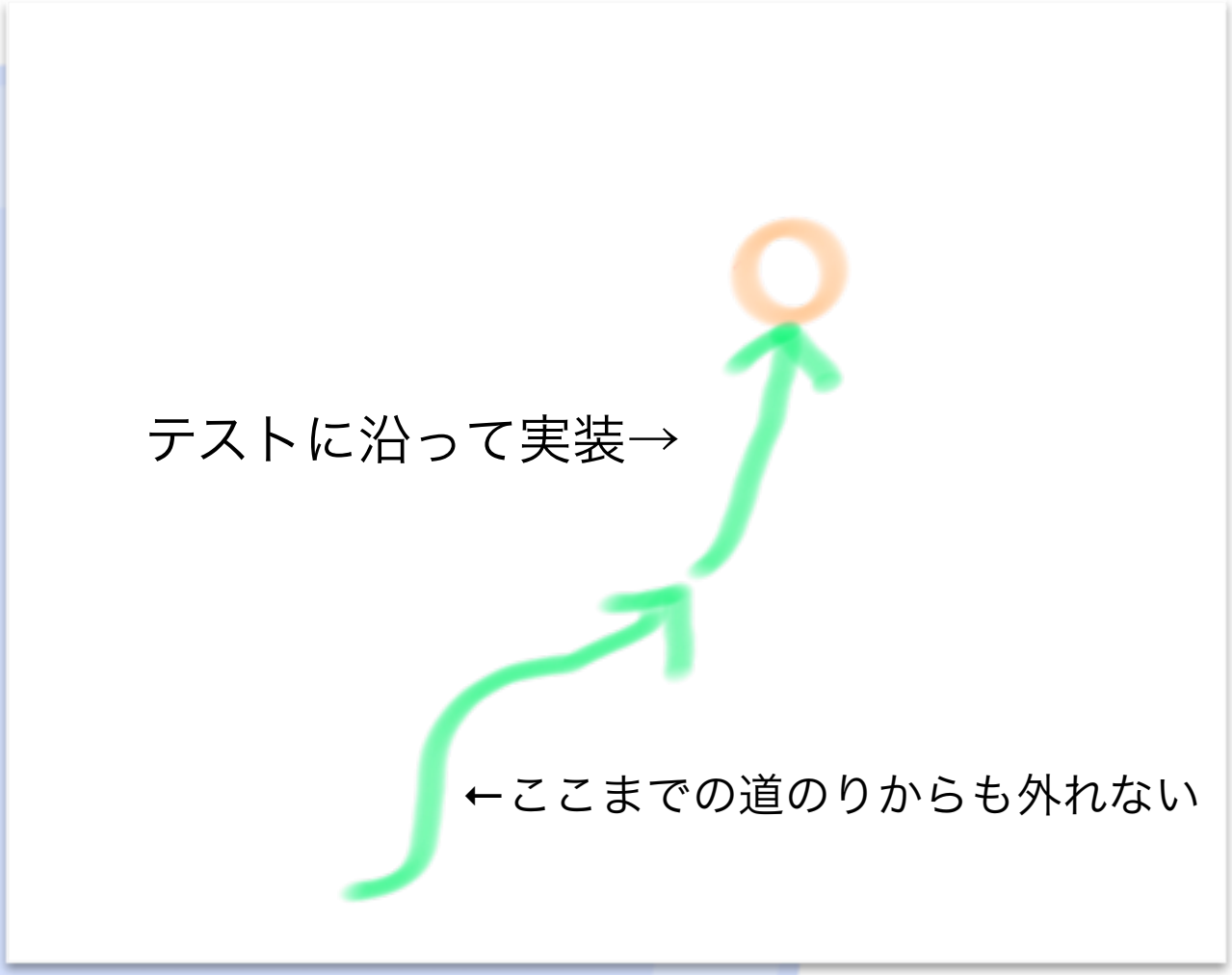


目標を考える

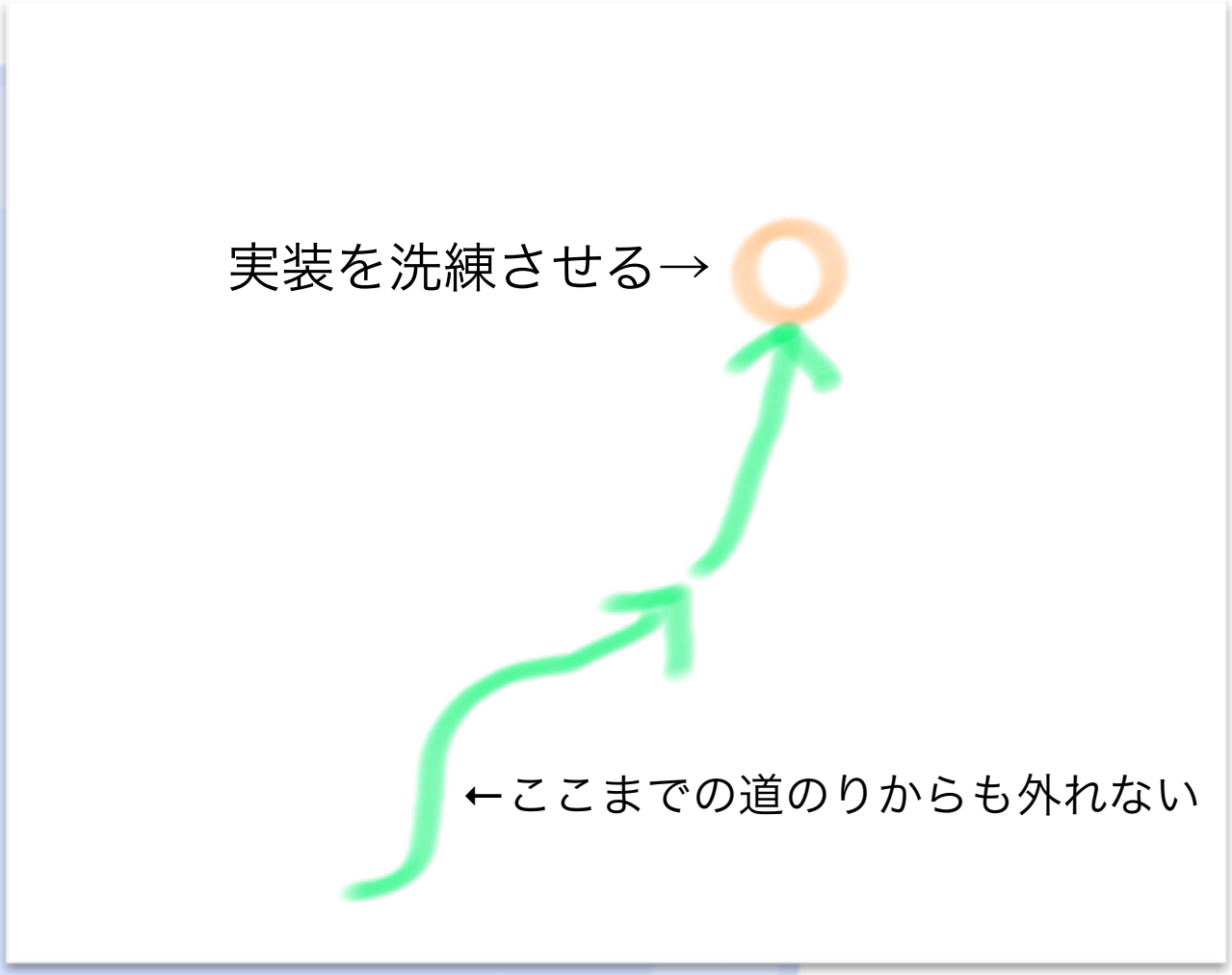
目標をテストで表現する→



Red

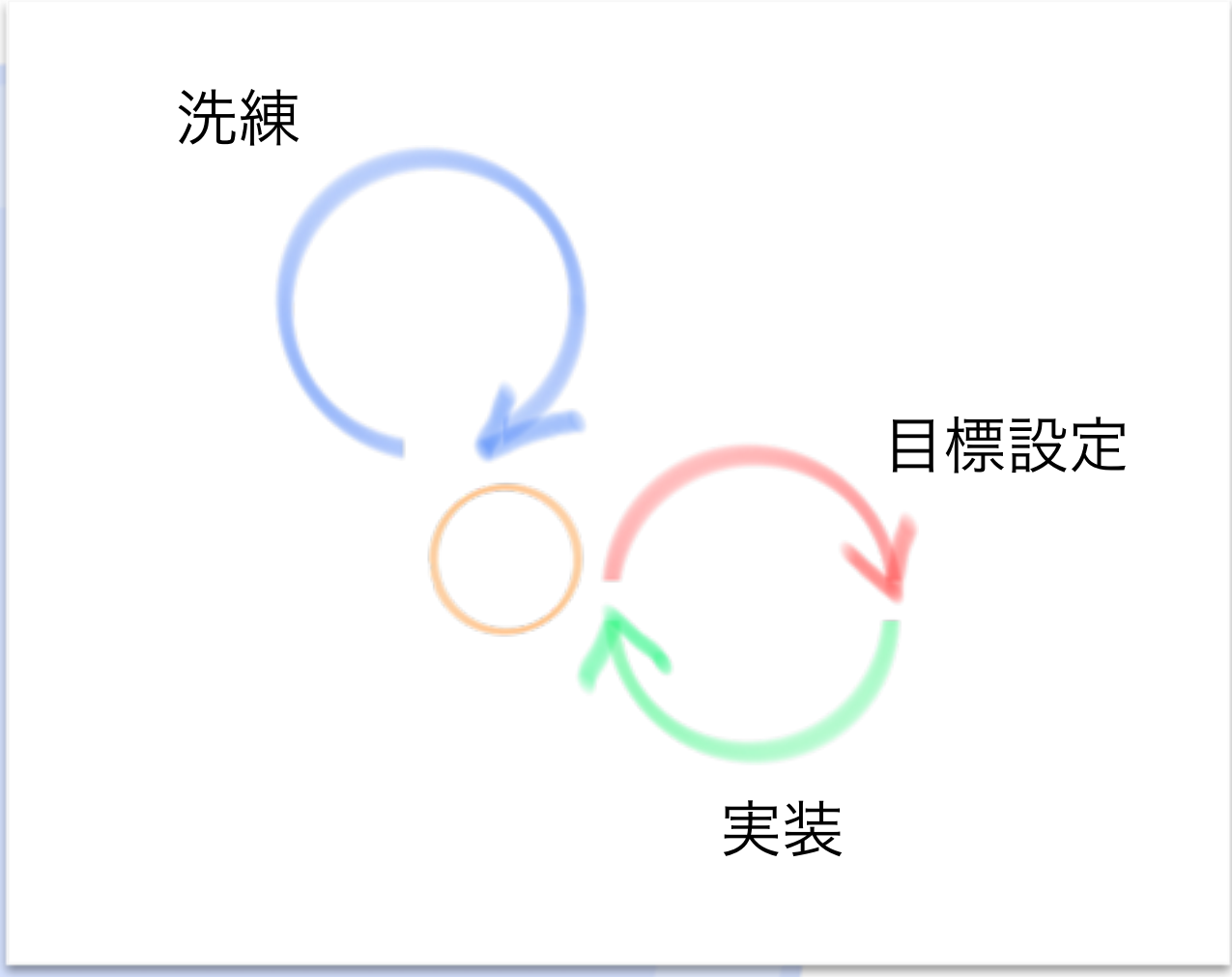


Green

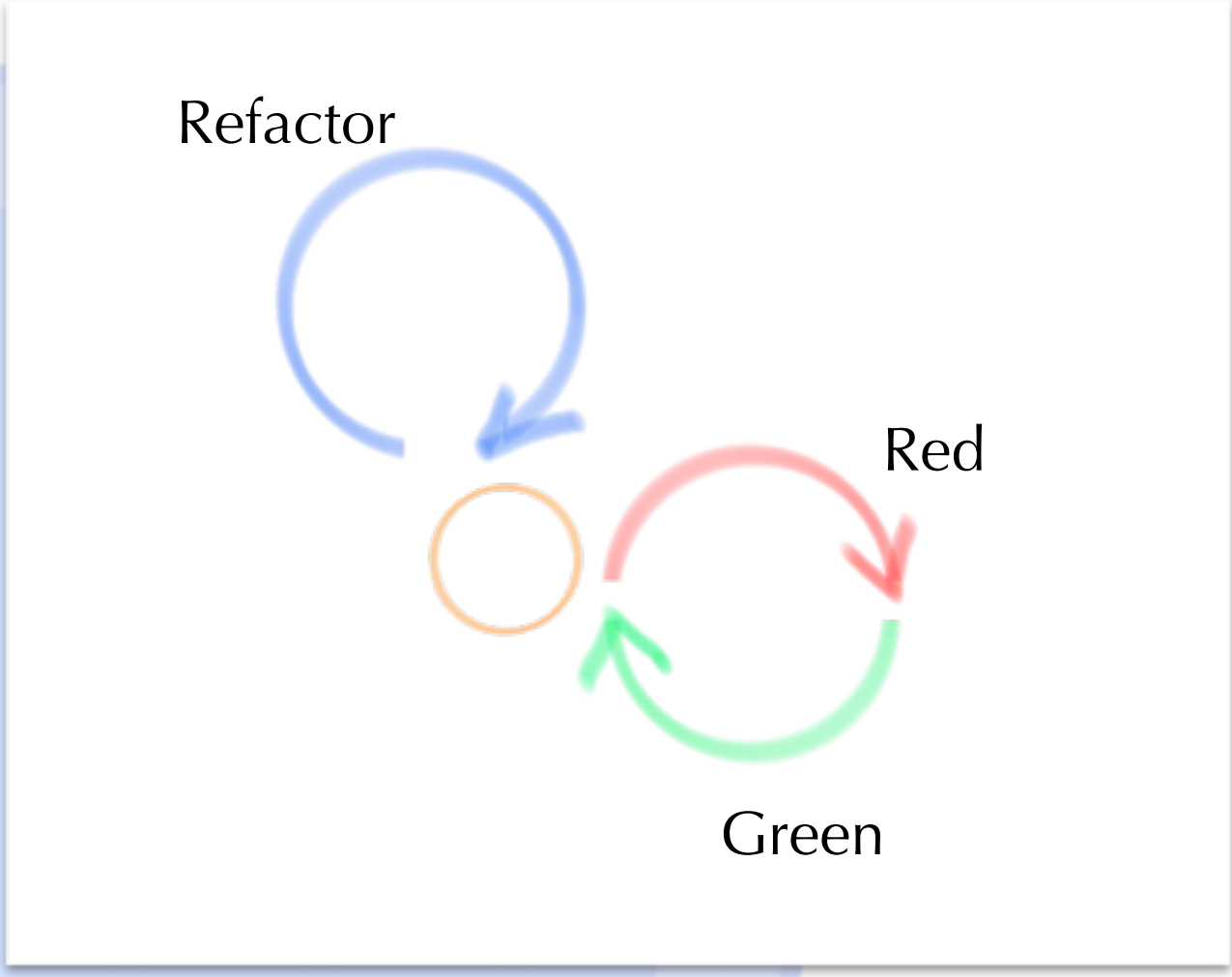


Refactor

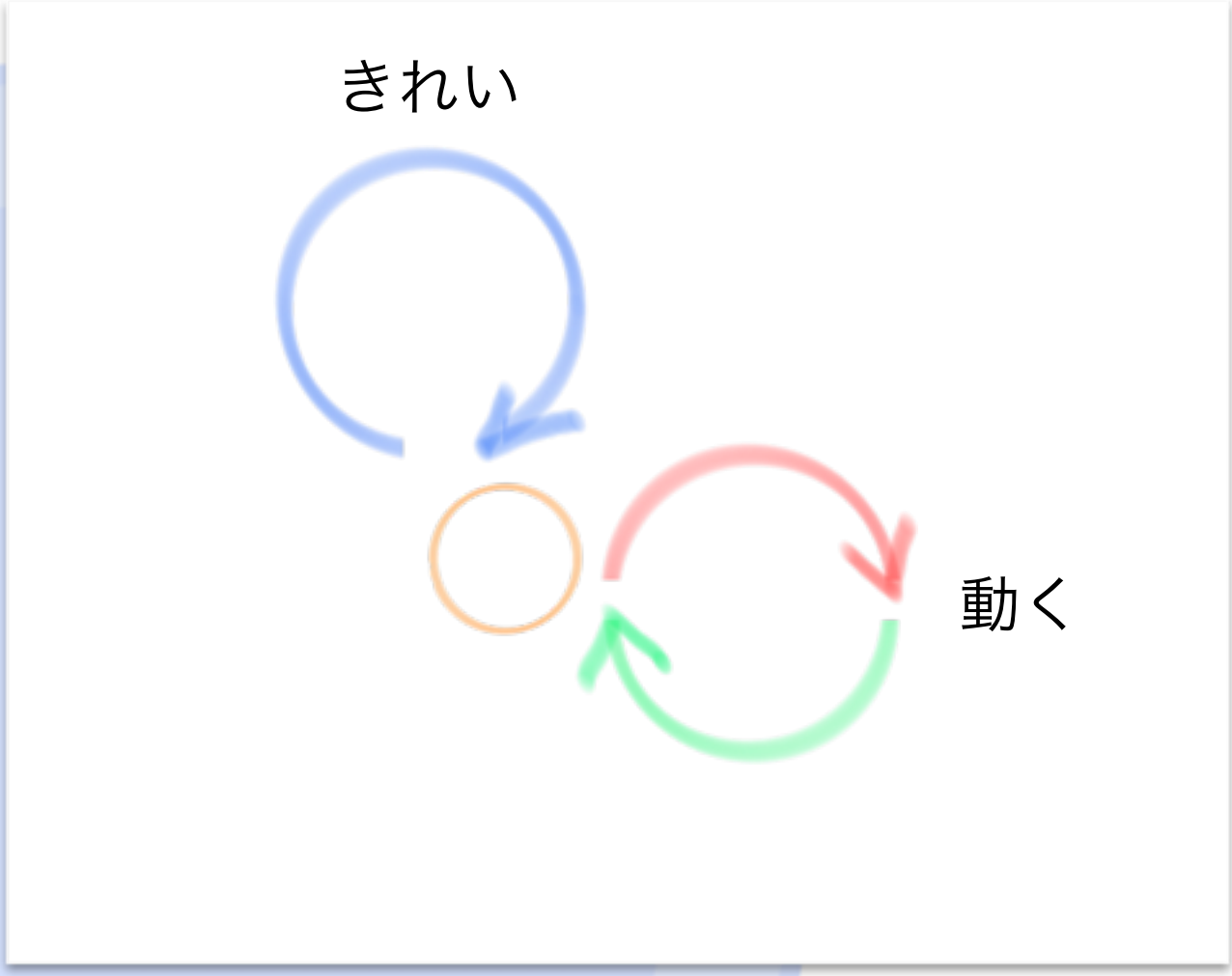
またくりかえす



状態のサイクル



状態のサイクル



気分

TDDのテストケース

- TDDは設計技法
  - ゴールをテストケースで表現
  - テストケースを考えること自体が設計
  - 設計とテストなら設計に近い

TDDのテストケース

- よいテストか？（エラーを発生する確率）
  - 一度目はRed ⇒ これって成功なの？
  - 二度目以降は低確率
  - ケアレスミスを待ち伏せ

自動テストっぽい？

- 計算機がテストを実行してくれる
  - ここまで辿ってきた各ステップのゴールを何度でも実行できる
- 自動テストっぽい

コストと確率

- 作成コストはわりと安い
  - 少しずつ書くので面倒じゃない
- 実行コストはとても安い
- 問題を発見する確率は低い

Checking vs. Testing

- Checking
  - 既知の情報の確認
- Testing
  - 新しい情報を探す。未知の物。
    - JaSST'11 Tokyo、Lee Copelandが言ってた誰かのblog

Checking vs. Testing

Michael Bolton

- **Checking** is something that we do with the motivation of *confirming existing beliefs*
- **Checking** is a process of *confirmation, verification, and validation*. When we already believe something to be true, we verify our belief by checking
- **Checking** is a highly automatable process

http://www.developsense.com/blog/2009/08/testing-vs-checking/

8

Checking

Checking vs. Testing

- **Testing is something that we do with the motivation of *finding new information***
- **Testing is a process of *exploration, discovery, investigation, and learning*. When we configure, operate, and observe a product with the intention of evaluating it, or with the intention of recognizing a problem that we hadn't anticipated, we're testing**
- **Testing is not automatable. It's using our brain in real time, and it's really fun**

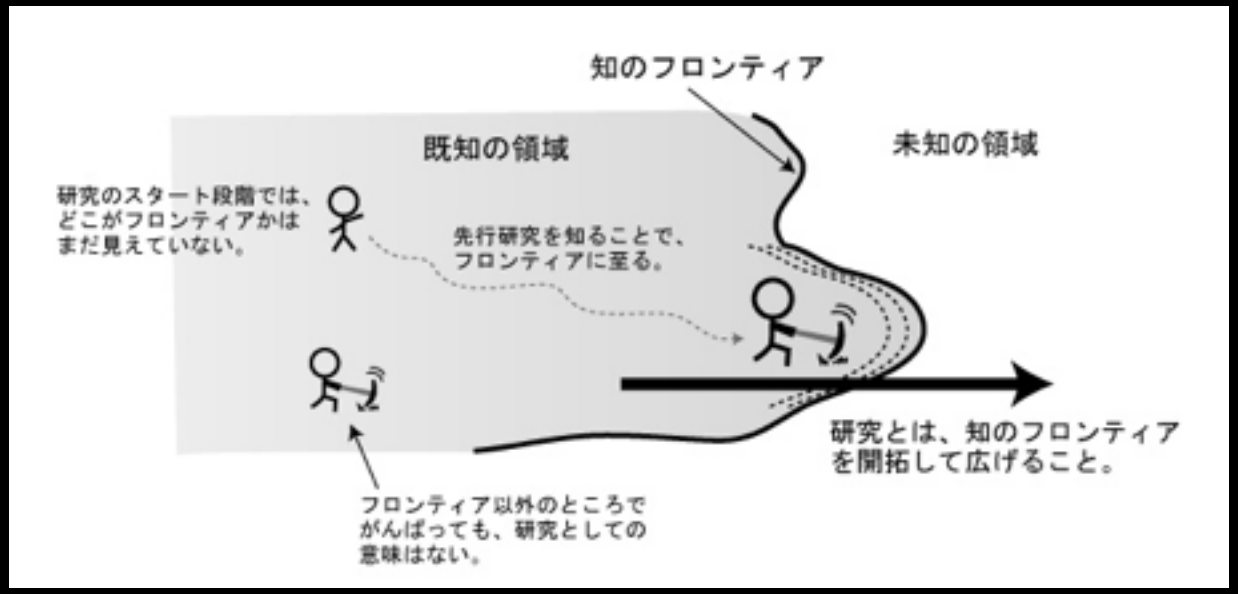
Testing

既知と未知

- そういえば

http://web.sfc.keio.ac.jp/~iba/sb/log/eid75.html

IBALOG - Concept Walk 日本語版 | 「研究」と「勉強」の違い



テストケースの自動実行

- Checking
- 自動化しやすい
- 既知の情報の確認
- 待ち伏せ

TDD以外の自動実行

- 作成コスト高そう（TDD比）
  - わざわざ作るのめんどくさい気分
- WinRunnerとか
- 実行コストは安いけど

Checkingの自動実行

- テストケースの自動実行
- 想像しやすい
  - いかにもできそうな気がする
- じゃあ他のものはどうなの？

しばらくお待ちください

- ✓ +20:00すぎ
- ✓ TDDおさらい
- ✓ 次はTestingやその他の計算機支援

それ以外の領域

- 未知の問題の発見を手伝ってもらうには..
- テストケース自動実行以外に何かないの？

ネタ

- 不具合からの不具合自動生成
- テストスイートのリコメンド
- つづきはみんなが考えて!

不具合からの不具合自動生成

- 計算機によるTestingの支援
- 架空の不具合を生成するツール
- 想像できない問題を想像するために使う
- 不具合データベースとマルコフ連鎖

不具合データベース

- ふつう毎日読み合わせをするもの
- 不具合から危険を感じる脳内配線を育てる
  - 反射神経
  - 脊髄反射

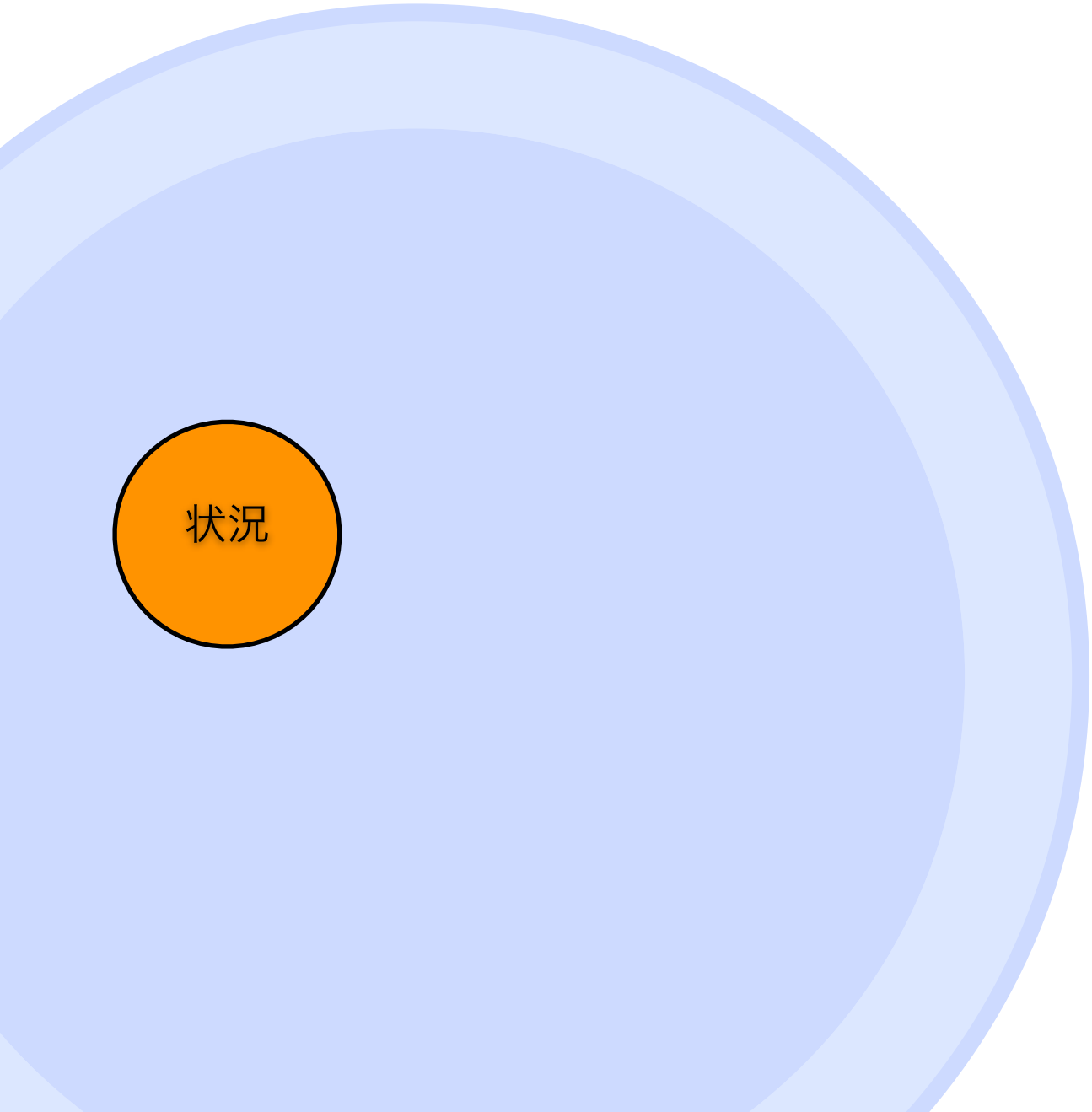
不具合を想像してテストする

- いつものTestingを思い出してみて!
- 無から生まれるわけではない
- これまでの学習で得られたパターン
- そこからありそうなイヤなことを選択

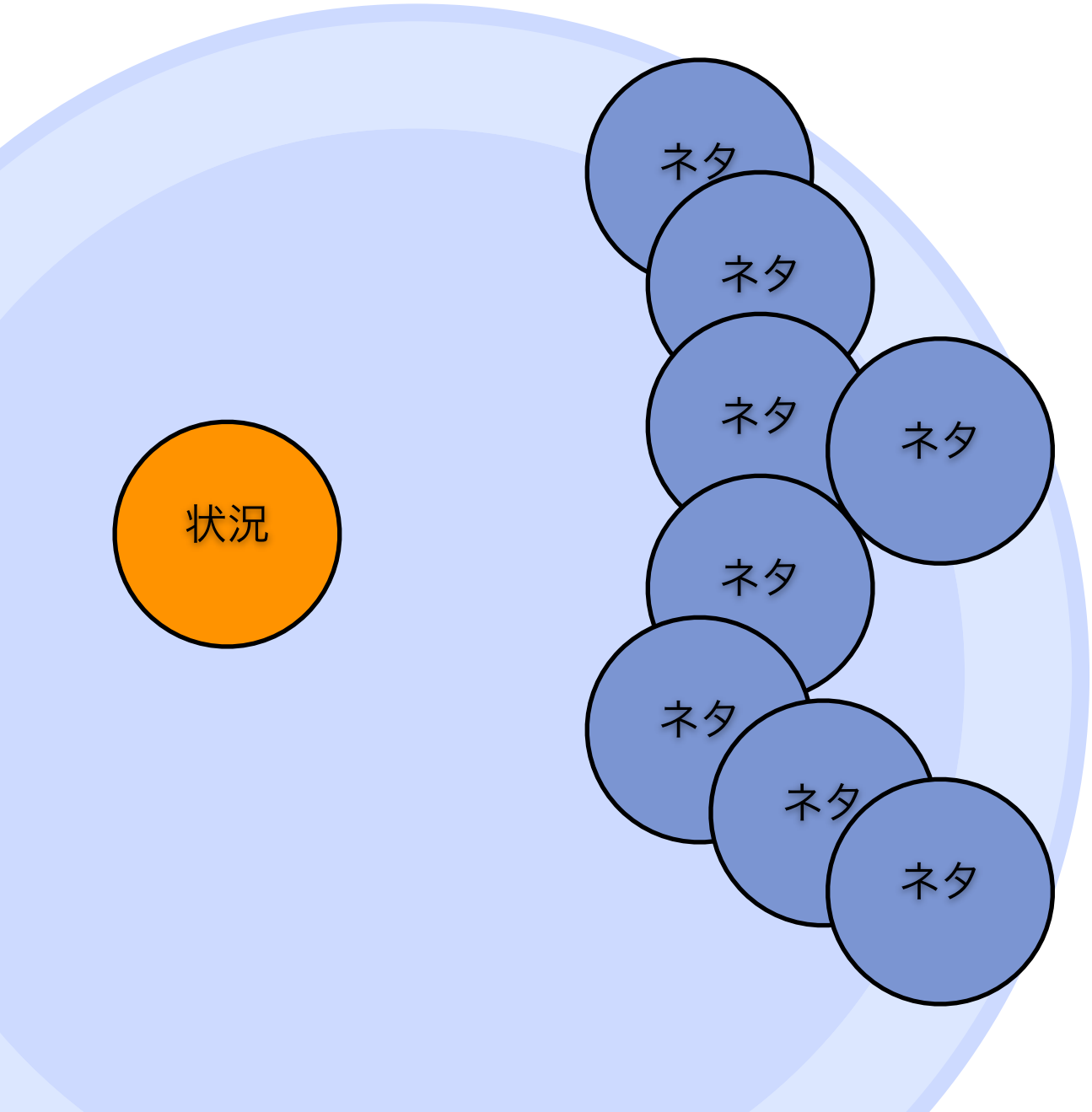
脳内では

- 現在の状況から無数のパターンを生成し見込みのありそうな数個を選ぶ、という繰り返しを一瞬で計算している
  - ちょっと嘘かも

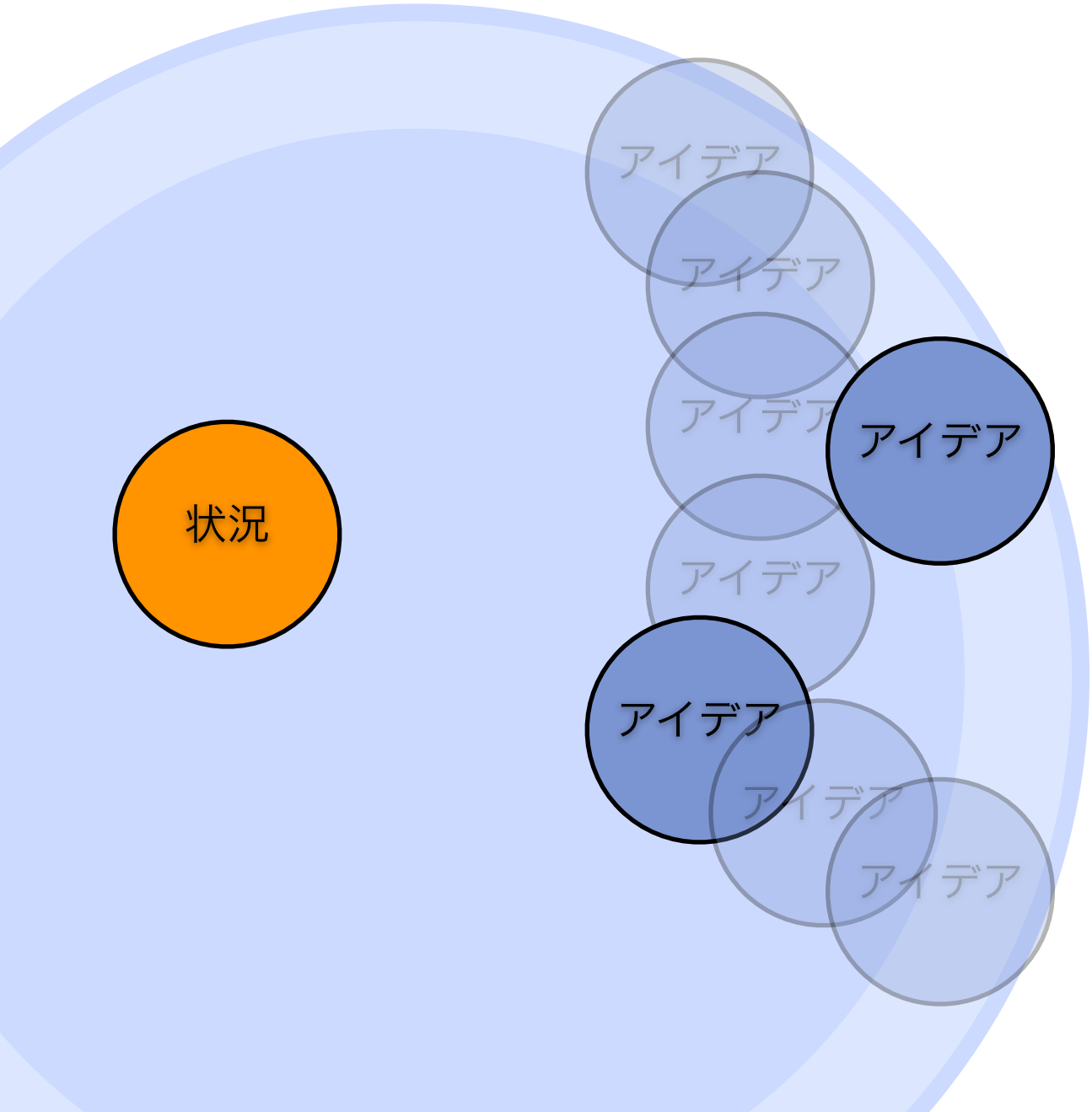
脳内では



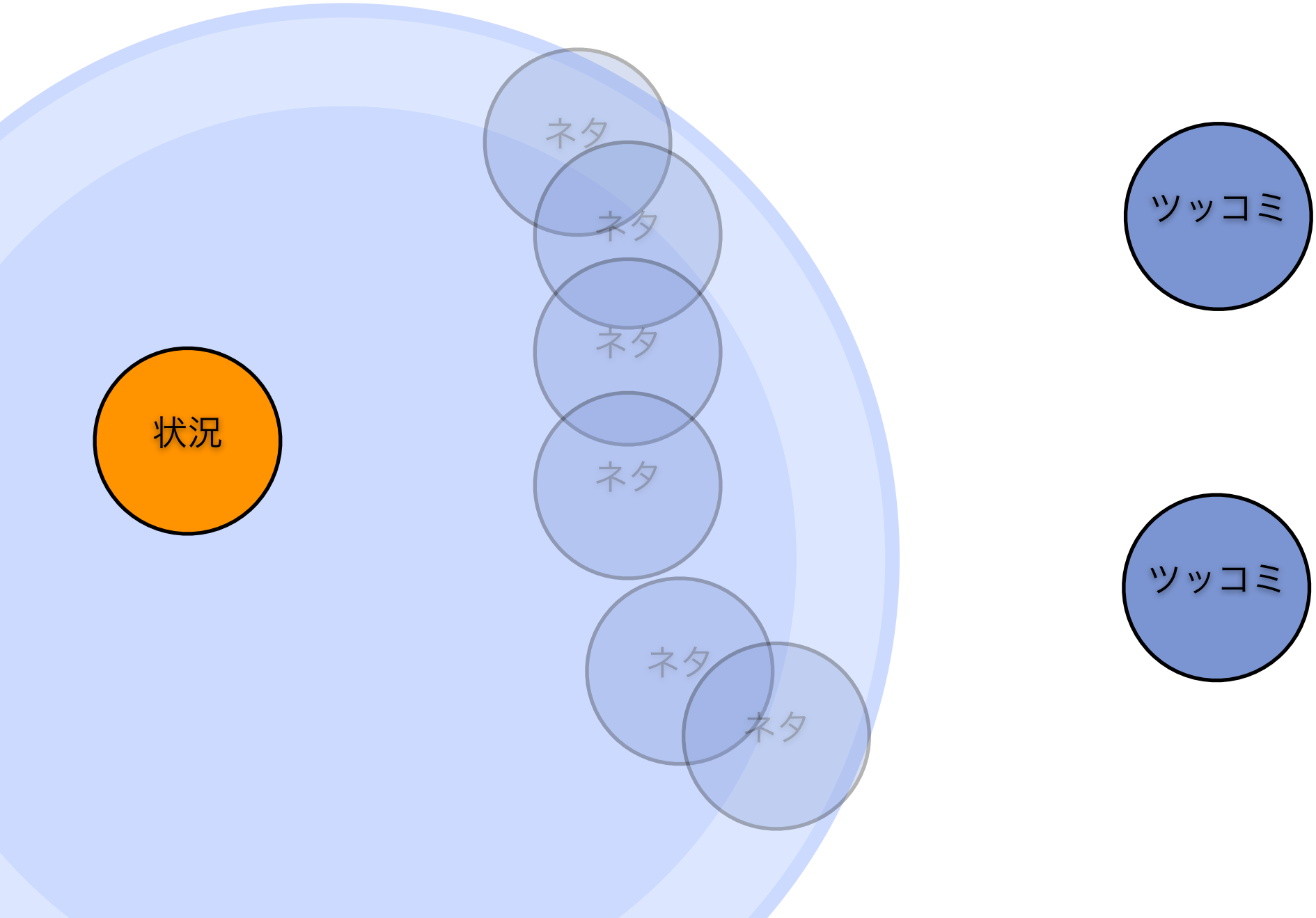
いろいろ思いつく



見込みのあるのはどれかな



バグに出会えそうなのを採用



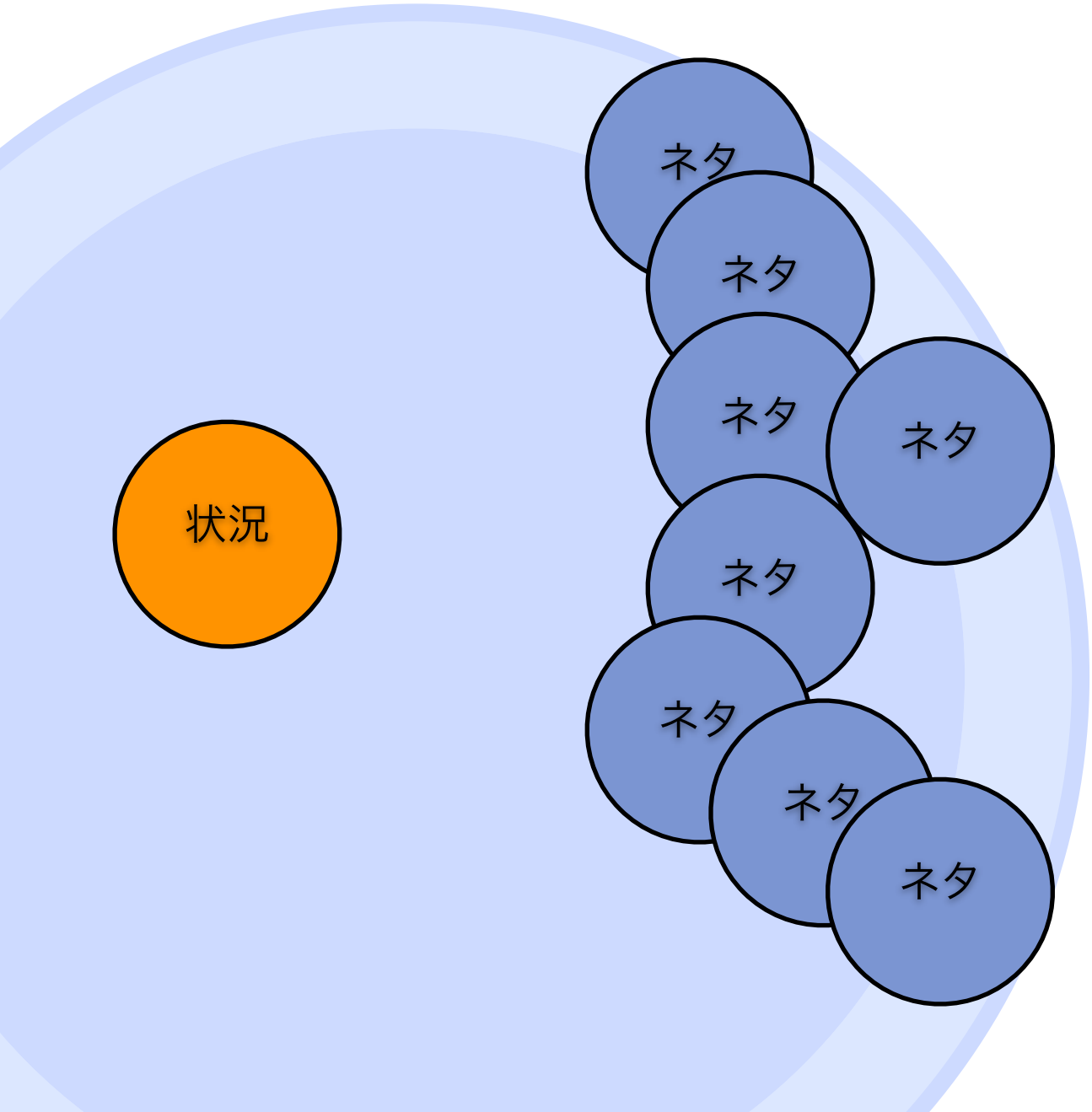
作成したシステム

- 過去不具合を学習し、ありそうな不具合を自動生成するプログラム
- ツッコミのネタを生成する

実装は簡単

- 形態素分析とマルコフ連鎖を用いる
- 文章をたくさん学習させるとそれらしい文章を生成できる
  - 要約システム
  - 偽論文生成など

ネタの生成を機械にやらせる



🍣🍣🍣🍣実行中にcloseしたらロックした際の挙動が変

🍰🍰🍰で再生表示を行うと変な残像が見える

全部消したらInsertもResetも出来ない

🍜🍜🍜されず、ページ変更も行なえなかった。

3D🍣🍣🍣🍣の1枚だけ引き継ぐとダメな場合がある

🐱🐱🐱🐱🐱の設定が反映されない

☕☕☕のボタンの下に表示されていないときがある

ある🍣🍣🍣🍣名にマイナスが入力できない

Saveぼたんがなんか変。

平均を求める時に🍣🍣🍣🍣🍣の視点位置が変更すると原画の断面線が出ない。

読み込みをabortするときれいに並ばない

🍟🍟🍟🍟のオブジェクト１だけを選んでもプリセット3が適用される場合がある。

🍣🍣🍣🍣ページで読み込むと、🍜🍜🍜🍜が閉じる。

オブジェクト登録/削除を実行したら、🍫🍫🍫が死んだ

複数🍣🍣🍣🍣🍣選択しても🍫🍫🍫フォーマットを変更したら、📷📷📷📷ウィンドーが消えた。

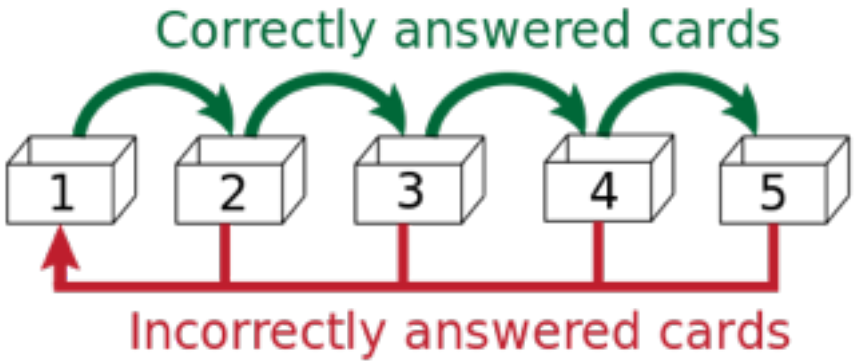
📚📚📚📚が最小値と💩💩💩💩💩値を加えるとエラーにならないときにマウスカーソル⭐⭐⭐⭐⭐補し
のに角度が付く

嘘でも報告されたら

- プログラマはすぐに再現試験しようとしたり発生する可能性を考えたりする…
  - 運が良ければそこから新しい不具合を発見するかも!!!

テストスイートのリコメンド

- 手動でやるテストの話
- 現在の開発状況で、バグの発見できる確率、発生した時に損害の大きなものを推薦
- あと1週間ならなにをするべきか
- ライトナーシステムの応用



ライトナーシステム

復習間隔が次第に延びる間隔反復学習

オープンクラス向き

その他の計算機支援

- 静的解析
- 動的解析
- コンコリックテスト
  - デバッグと違わない気もするけど細かい分類は気にしない
- もっといいのがあるとおもうよ、みんな!

中締め

- マイヤーズの設計とテスト
- TDDのおさらいとChecking
- Testingの自動化はみなさんで

- 次は開発ライブの前に…

しばらくお待ちください

- ✓ +30:00すぎ
- ✓ 中締め
- ✓ つぎは開発ライブの前ふり。時間がなければ飛ばす

TDDBC大阪でやったご提案

- Red Green Refactor
  - ちょっとずつ仮説の上に仮説を重ねて積みあげる気分
- + Destroy
  - そう言うけどどっか間違ってるんじゃ？
  - この実装ならここでcore dumpだろ

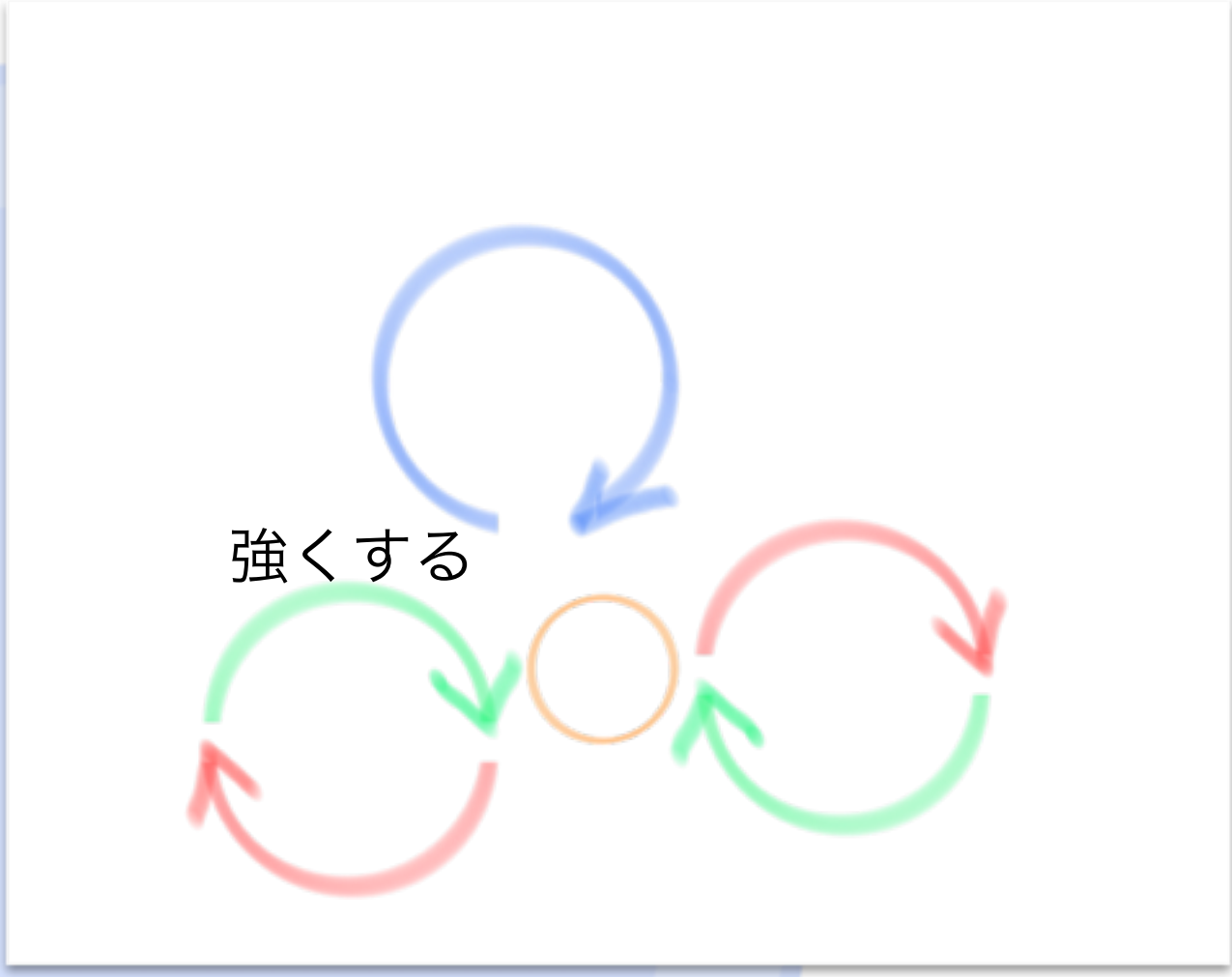
Destroy

- Red Green Refactorのサイクルにたまに混ぜる
- この実装ならこれでバグが出るだろ!みたいなのを二人で考えてテストケースを書き実行する
  - 当たればRed、外ればGreen

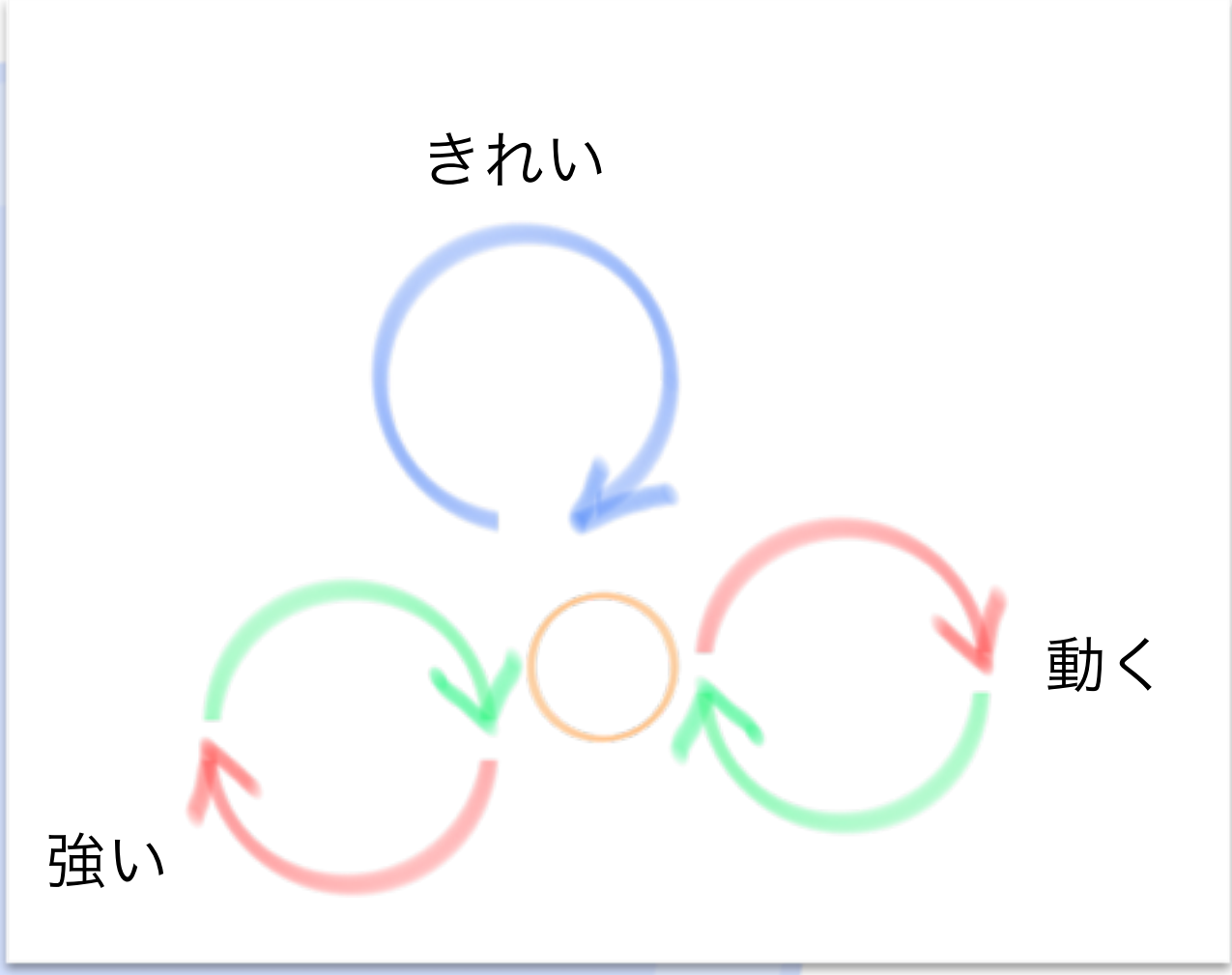
Destroyとは

- 自分たちのコードを破壊する
- TDDのサイクルに持ち込む
  - バグを見つける帽子

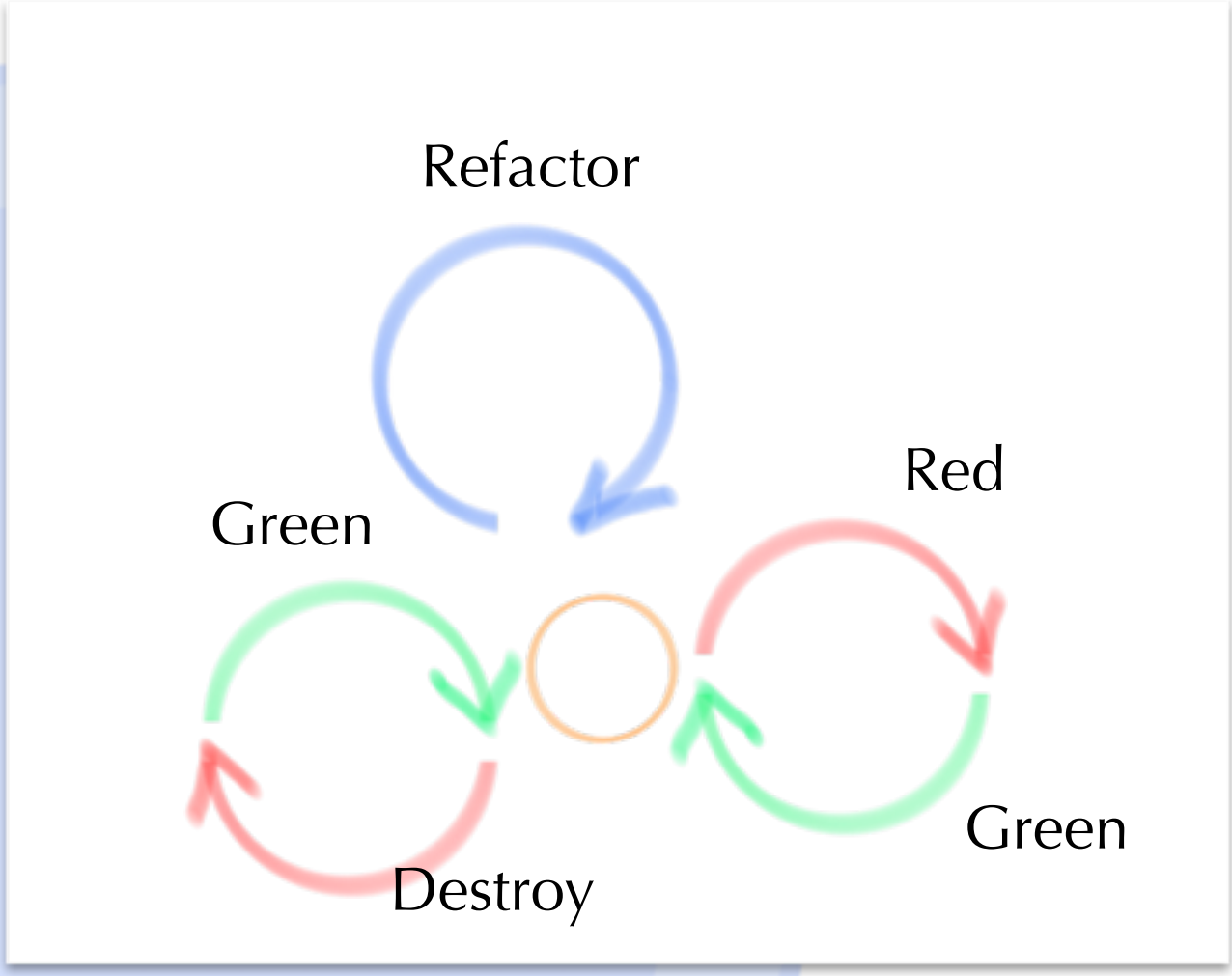
疑って壊してみる



より良く、より強く



新しい指針



Destroy

これはTDDか

- いつもは建設的な態度で臨む
- たまに破壊的な思考を持ち込む
- Checkingの中にTestingを

見つかった問題は既知じゃん

- Testingが輝くのは一瞬
- 一度解いた問題は明日のChecking

次は開発ライブ

- +DestroyするけどTDDのTest Firstしません
- 私がコードを書きます
- みんなでバグが出そうなテストケースを提案してね!