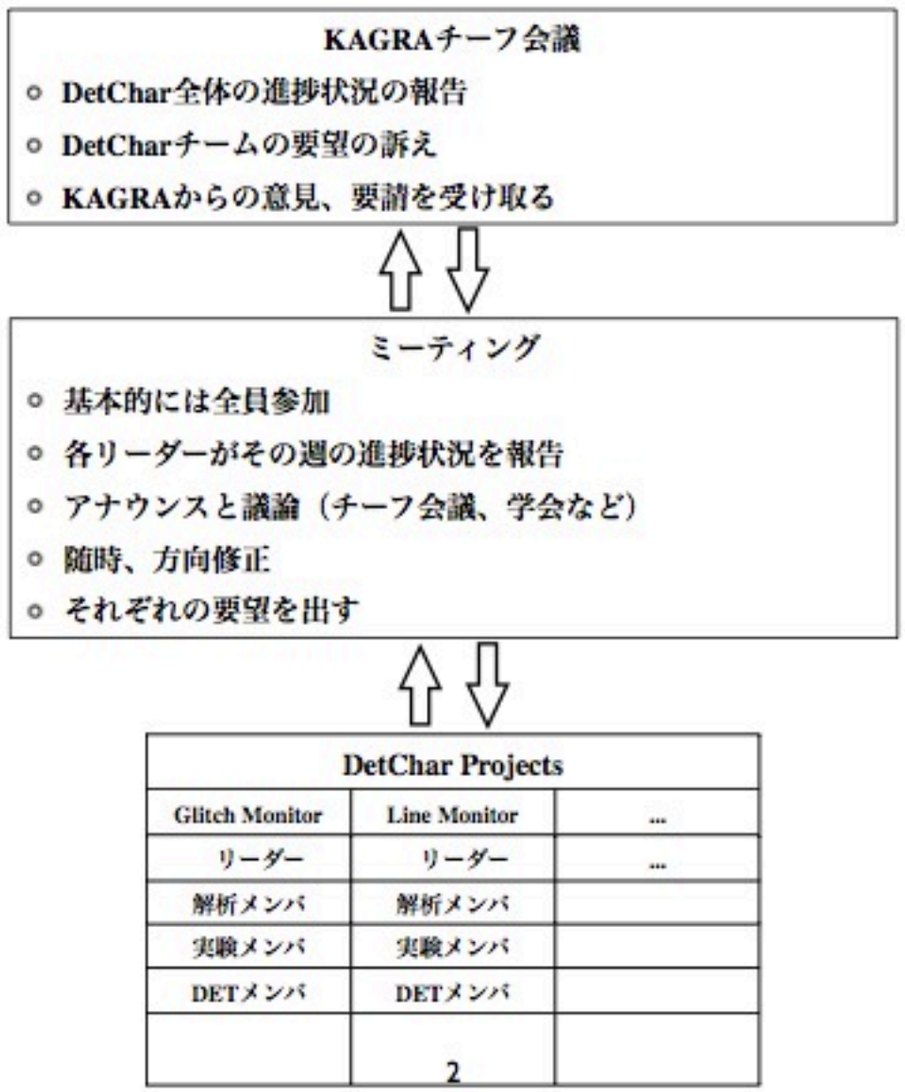
Detector Characterization groupの体制

反省会

端山

Detector characterizationサブシステム構成

- 大体このような構成で進んだ。
- Detcharからの要望の訴えはあまり成功していない。
  - マンパワーの増強（開発ソフトウェアのエキスパート村主さん、小野くんらが参加。が、自力の要素が強い。）
  - テクニカルノイズを含めた感度曲線など。
- KAGRAからの要請：リスク表の提出、プレゼン、特にKAGRAメンバに目が触れるようなGUIなどについてのプレゼン等、随時行ってきたが、こちらに来た要請に答えただけの状態。今後はこちらからのモーションも必要。
- ミーティング参加率は悪い。ミーティングで決定していくので、出なくなると進展についていけなくなる。



- ツールの方はKAGRAメンバが使いやすいようにGUIで構築。現在の開発の中心。モニタはレンジモニタ、グリッチモニタ、Gaussianityモニタが開発されGUIで動く。
- プロットツールについてもROOTを使ったものを中心として開発されてきている。
- ノイズバジェッドは、LIGOのmatlabスクリプトを入手、メーリングリストに参加まで。
- webベースまでたどり着いていない。設計もまだ作り込まれてはいない。（どこまでが理想でどこまでが現実的なのか等）

ソフトウェアの開発

システムとツールに分けて開発

システム

DetCharシステム上で常時動作している。基本的に一度立ち上げたら、触らない。なるべく安定動作、リアルタイム性を重視。

ツール

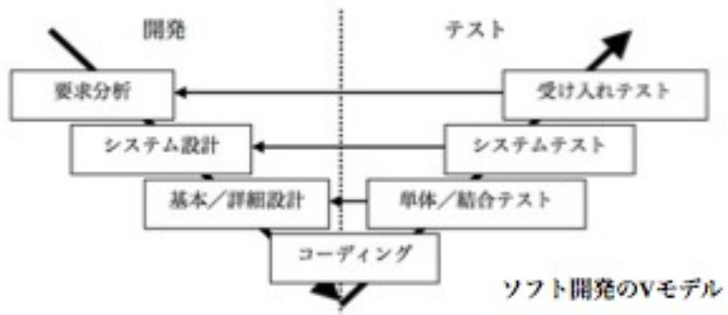
DetCharシステム、または単独でユーザが手動で用いる。

始めにツールとして開発を進めて、それから一部の基本的で重要なものをシステムとして組み込む。

開発プロセス

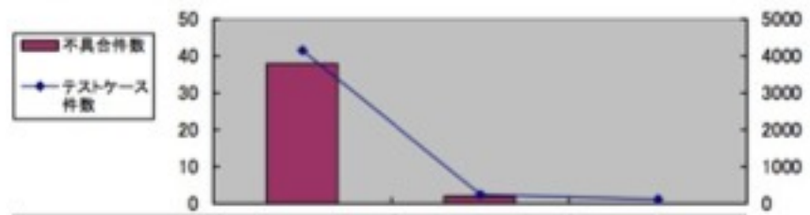
- 我々はプロジェクト制は完全なる素人という認識からスタート。最も大きな枠組みを決めて他はトライアンドエラーで経験を積む。
- モチベーションも大切なので、難解ではあるが可能性が高く、それ自体が研究対象の言語Haskellを使用。それによって、新規参入者が入りづらくなるので、基礎ゼミを定期的に開き、一から勉強する機会を持つ。開発方法を含めて指導し、その後スムーズに開発中心メンバになれるよう期待。
- コード規約は、基本作らないことが規約だがどうしてもなければいけないものをボトムアップ式に作っている。
- チェックリストの作成。
  - 大体１０未満のリストで、コードを投稿するとき、コードレビューをするとき、といった具合に場合に分けて複数のチェックリストを作りたい
- テストコードの作成
- コードレヴューの体制

テストプロセス



ソフト開発のVモデル

【事例１：プロジェクトA】
使用言語：C++　　開発規模：新規１２．１Ｋｓ
開発期間／人員：計画 … ５ヶ月／７人、実績 … ４．５ヶ月／７人

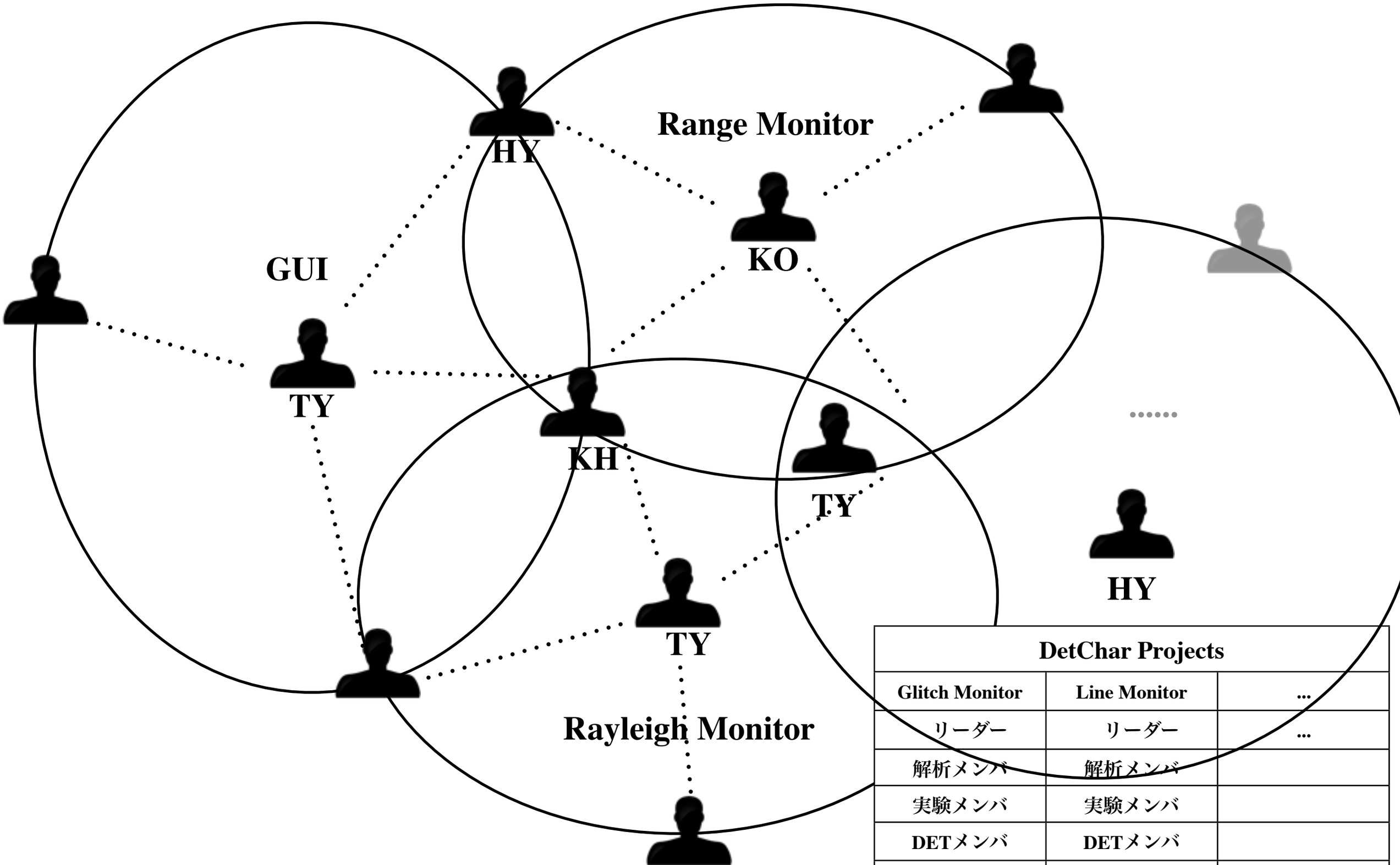


テスト工程	単体テスト	結合テスト	システムテスト
テストケース件数(件)	４１５０	２４２	１２０
不具合件数(件)	３８	２	０
不具合摘出率(件)	９５％	５％	０％

図 5-1 新規開発における各テスト工程での不良摘出率推移

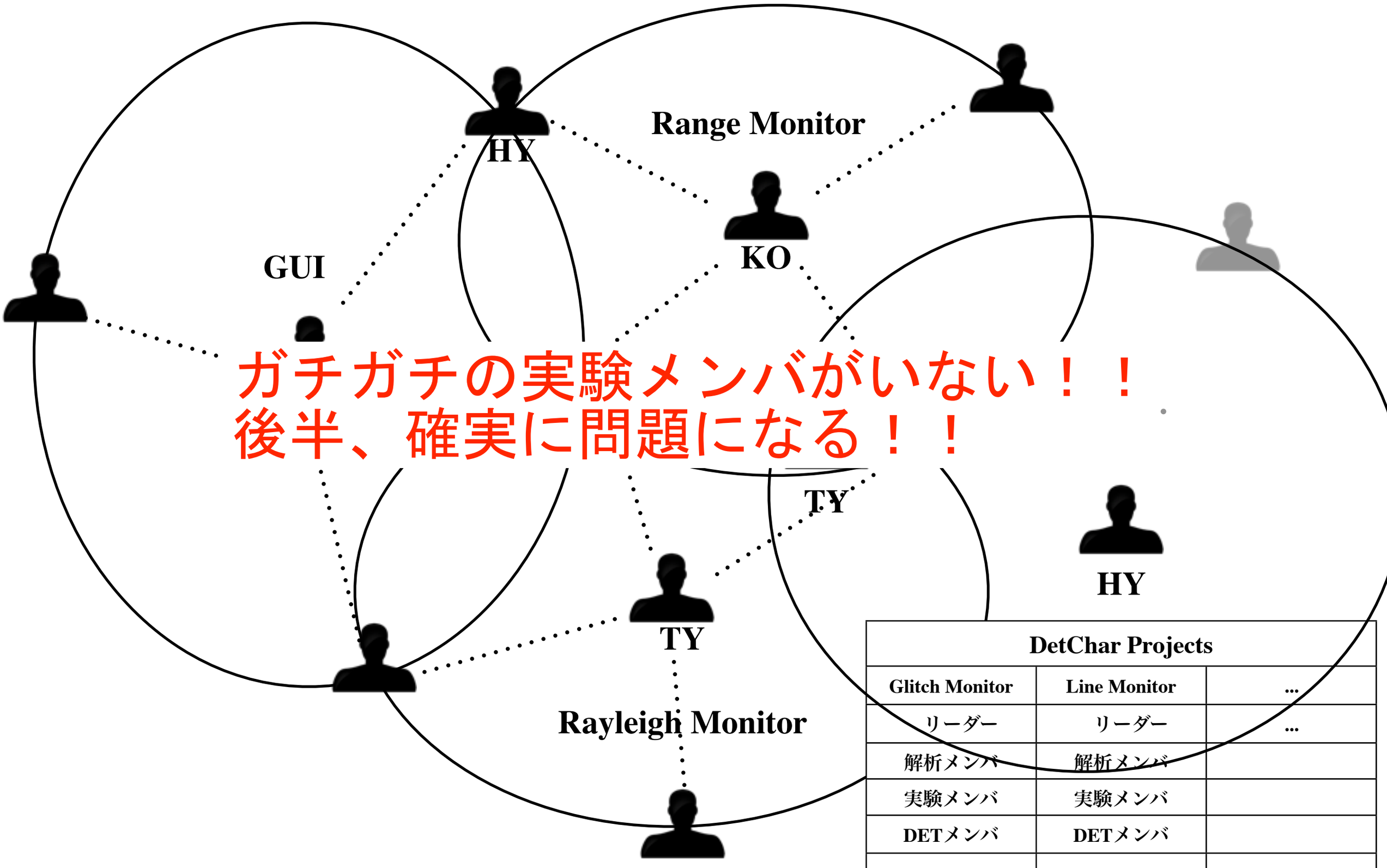
＊新規開発であり、単体テストレベルを網羅的に実施することで、早い段階で不具合を摘出できている。

開発プロジェクトの構成



DetChar Projects		
Glitch Monitor	Line Monitor	...
リーダー	リーダー	...
解析メンバ	解析メンバ	
実験メンバ	実験メンバ	
DETメンバ	DETメンバ	

開発プロジェクトの構成



ガチガチの実験メンバがいない！！
後半、確実に問題になる！！

DetChar Projects		
Glitch Monitor	Line Monitor	...
リーダー	リーダー	...
解析メンバ	解析メンバ	
実験メンバ	実験メンバ	
DETメンバ	DETメンバ	

開発に用いている言語のエキスパートがいること

と、おことわりした上での方策ですが、リストに補完関数をかませたいが、その関数はIOに依存している、という感じですね。

なので、1配列を扱うライブラリに乗り換えるか、　2 iteratorを使う　という策が考えられます。

たとえば、このへんの
http://hackage.haskell.org/package/vector/docs/Data-Vector-Unboxed.html#v:mapM
は、Vectorがヒープに確保されると思います(未検証)ので大丈夫だと思います。

次にiteratorですが、iteratorとはメモリにすら乗らないとか、あるいはネットワーク経由で次々に到着するとかのストリームデータに対する遅延IO処理をどう記述するのか、に対する回答として考えられた概念です。例えば、メモリより大きなファイルを読み込んで線形に処理して別のファイルに書き出したいとかの時、従来はHaskellのデフォルトの遅延IOに頼ることもありました。しかし、それでは評価順やメモリの使いかたを精密に制御するのが難しく、信頼の置ける解法とはいえませんでした。iteratorはこれを解決しようとして登場したものです。

Haskellのiteratorは実装も何種類もあり、最近の進展は追いきれていないんですが、資料は検索するといろいろでてきます。私の知る範囲ではpipesという実装が後発で、わりとまとまってて好きです。

http://hackage.haskell.org/package/pipes

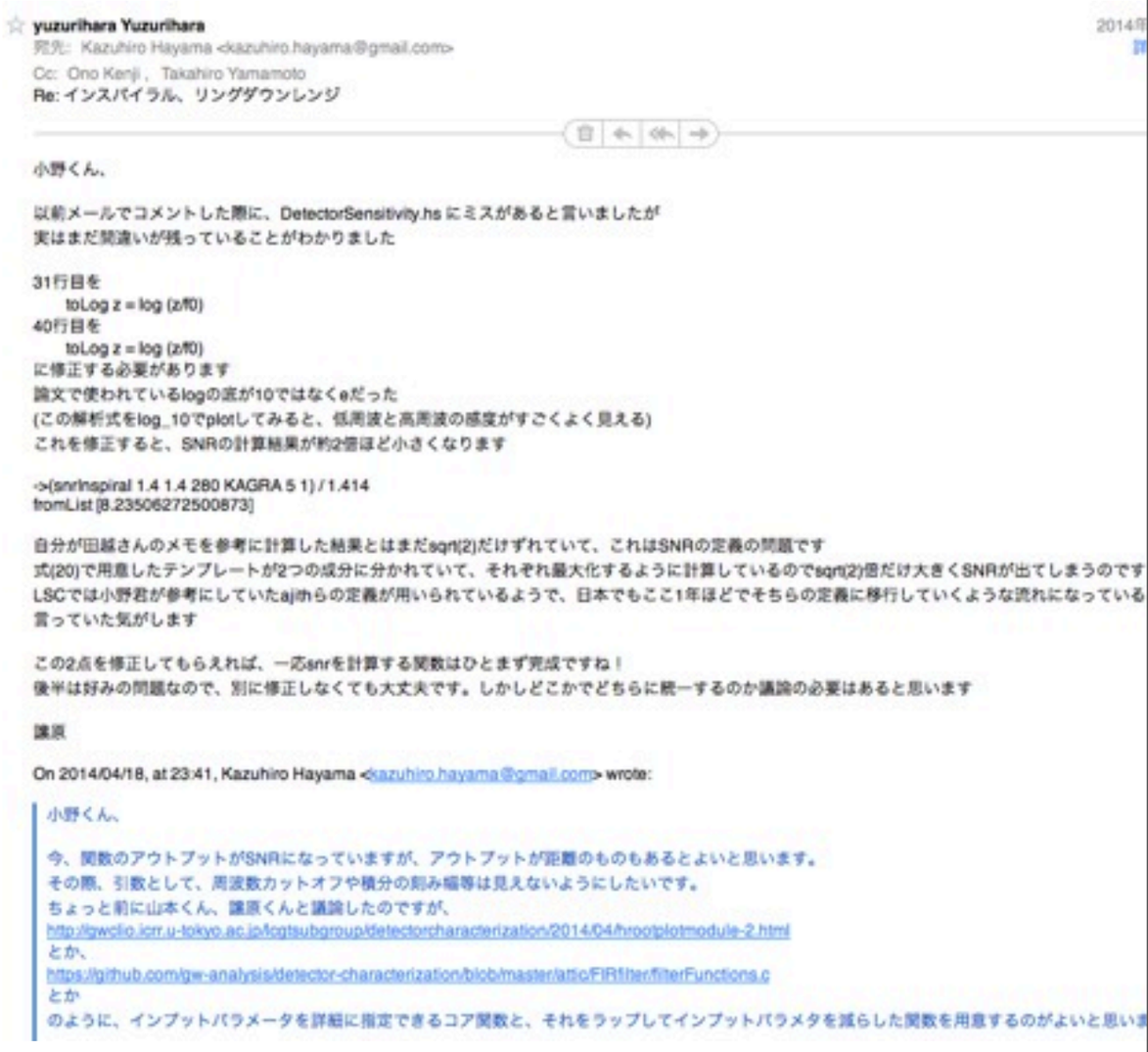
私の勘では、まずはVectorを試すのがいいと思います。結果がわかったら、また教えてください。

2014年4月25日 10:41 Kazuhiro Hayama <kazuhiro.hayama@gmail.com>:

> 村主さん、
>
> HasKALの開発で、一つ難題につきあたっています。
>
> 例えば多チャンネル解析を行う際に、
> １０００チャンネルのデータの相関解析などをするとき、
> 場合の数で1000_C_2の組み合わせがあり、
> 合計～GB程度といったかなり大量のデータを一度にキープしておかなければならない自体が
> 生じるのですが、その際haskellではスタックオーバーフローを起こしてしまうのではないかと懸念しています。
>
> 最近、僕がGSLの関数を使った内挿をするコード
> https://github.com/gw-analysis/detector-characterization/blob/master/attic/interpolation
> https://github.com/gw-analysis/detector-characterization/blob/master/attic/interpolation/test_Interpolation.hs
> 山本くんが、同じくGSLの関数を使った乱数リストを生成するコード
> https://github.com/gw-analysis/detector-characterization/tree/master/attic/gslFFI
> https://github.com/gw-analysis/detector-characterization/blob/master/attic/gslFFI/gslRand.hs
> の中で100万点のリストからforMで内挿点、乱数リストを計算したのですが、

コードチェック体制

- （例）小野くんがレンジモニタのコードを書き、山本くん、譲原くんにレビューを依頼。二人によってチェックがなされ思わぬところからバグが発見されたりして（端山コードでした）ライブラリが成長した。
- このコードを書き、２人にレビューを依頼という流れは良いと思うので、ぜひ体系化して開発システムに入れたらどうか？
- ブログに記事を書き、現在、誰がどのような問題に突き当たっているのか、誰がどんなコードを書いたか、そのコードはどう使うのかなどが記録されているようにする。（まとめてではなく、なるべくリアルタイムであるのが望ましい。）



☆ yuzurihara Yuzurihara 2014年

宛先: Kazuhiro Hayama <kazuhiro.hayama@gmail.com>
Cc: Ono Kenji , Takahiro Yamamoto
Re: インスパイラル、リングダウンレンジ

小野くん、

以前メールでコメントした際に、DetectorSensitivity.hs にミスがあると言いましたが
実はまだ間違いが残っていることがわかりました

31行目を
　toLog z = log (z/f0)
40行目を
　toLog z = log (z/f0)
に修正する必要があります
論文で使われているlogの底が10ではなくeだった
(この解析式をlog_10でplotしてみると、低周波と高周波の感度がすごくよく見える)
これを修正すると、SNRの計算結果が約2倍ほど小さくなります

->(snrInspiral 1.4 1.4 280 KAGRA 5 1) / 1.414
fromList [8.23506272500873]

自分が田越さんのメモを参考に計算した結果とはまだsqrt(2)だけずれていて、これはSNRの定義の問題です
式(20)で用意したテンプレートが2つの成分に分かれていて、それぞれ最大化するように計算しているのでsqrt(2)倍だけ大きくSNRが出てしまうのです
LSCでは小野君が参考にしていたajithらの定義が用いられているようで、日本でもここ1年ほどでそちらの定義に移行していくような流れになっている
言っていた気がします

この2点を修正してもらえれば、一応snrを計算する関数はひとまず完成ですね！
後半は好みの問題なので、別に修正しなくても大丈夫です。しかしどこかでどちらに統一するのか議論の必要はあると思います

譲原

On 2014/04/18, at 23:41, Kazuhiro Hayama <kazuhiro.hayama@gmail.com> wrote:

> 小野くん、
>
> 今、関数のアウトプットがSNRになっていますが、アウトプットが距離のものもあるとよいと思います。
> その際、引数として、周波数カットオフや積分の刻み幅等は見えないようにしたいです。
> ちょっと前に山本くん、譲原くんと議論したのですが、
> http://gwclio.icrr.u-tokyo.ac.jp/logtsubgroup/detectorcharacterization/2014/04/hrootplotmodule-2.html
> とか、
> https://github.com/gw-analysis/detector-characterization/blob/master/atticFIRfilter/filterFunctions.c
> とか
> のように、インプットパラメータを詳細に指定できるコア関数と、それをラップしてインプットパラメタを減らした関数を用意するのがよいと思いま

規約

- 今のところ、３つある。増やさない方向ではあるが、必要に応じて増やしていく。

- コードを投稿するときにチェックリストを参照する
- 使った参考資料はコードにコメントしておく。
- コードを書いたら、２人以上またはメールリストにコードレヴュー、動作確認を依頼する。

現在実効的に使っているチェックリスト

チェックリスト

- いくつかの段階でそれぞれチェックリストを作る。
- 一つのチェックリストは１０項目以内にする。
- 最大公約数的なもの。
- 一つの文で芋づる式に他ものものチェックできるもの。

- テストプログラムをコメントアウトで書いていますか？
- ブログに解説を書きましたか？
- インプット、アウトプットが一般的な変数ですか？

今後

- これからコードの規模が拡大していくので、それにどう対応するか（GUIなど）
- テストコードについて
- コードヘッダについての共通フォーマット
- コードレビュー体制
- 実験メンバの参加
- チェックリストの作成
- ブログを中心にした情報共有に負担はないか？
- どこかに見えない負担は生じていないか？
- Webベースdetcharの開発とノイズバジェットツールの開発に心配がある。