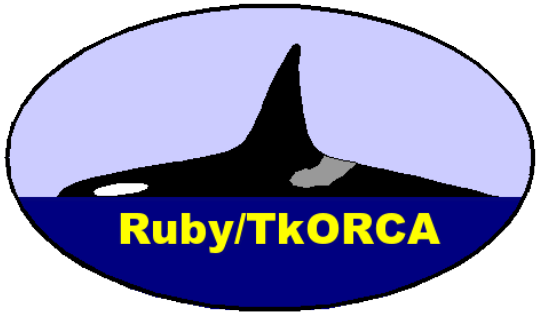
「どこでもGUI!!」

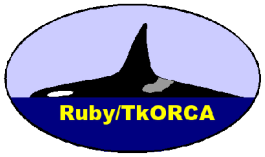
～ Ruby/TkORCAが目指すもの ～



永井 秀利

九州工業大学情報工学部知能情報工学科

2005/12/10



現在の状況．．．

A: 『こんなGUIで研究データ分析用ツールを作ってみたけど，どうかな？』

B: 『へぇ，いいねぇ．マルチウィンドウでグラフや可視化したデータが表示されるってわけか』

A: 『うん．いいライブラリを見付けたんでね．可視化データもインタラクティブに操作できるよ』

B: 『なるほど．あ，これってさぁ，デモシステムとして外部に公開できないかなぁ？今はやりのAjaxとかでやれば，何とかなるんじゃないの？』

A: 『えっ？それじゃ頭から作り直すのと同じじゃん．せっかく見付けたライブラリも使えなくなりそうだし...』

B: 『そうかぁ... ならさぁ，VNCで直接公開しちゃったら？』

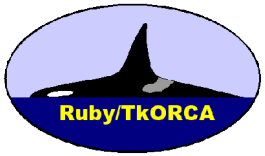
A: 『これ，セキュリティについては考えてないし，第一，サーバのウィンドウシステムを誰かもわかんない人に直接操作させんの？』

B: 『やっぱり無茶かなぁ．じゃぁ，ソースを公開するしかないか』

A: 『仮にソースを公開するにしても，データなしじゃ意味がないし，データ自体は全部公開するってわけにはいかないでしょ』

B: 『うーむ．どうにもならないかなぁ...』

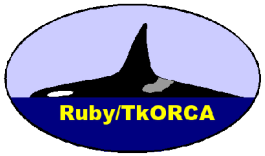
A: 『そうだねぇ...やっぱり公開用は別に作んなきゃいけないかなぁ...』



Ruby/TkORCA（Ruby/Tk On RFB Canvas）とは？

- ローカルのウィンドウシステムで稼働している
- マルチウィンドウアプリケーションを，
- 表示や操作性をそのままに，
- 再プログラミングの必要なしに，
- 簡単かつ安全に，
- 必要なら実行の監視や制約の機能を付加できて，
- サーバの負荷はできるだけ少なく抑えて，
- PDA 等の低能力端末を含む多種多様なクライアントで使えるように
- ネットワーク公開アプリケーション化

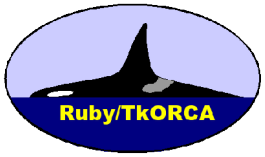
ができるようににするためのフレームワーク



一言で言うなら．．．

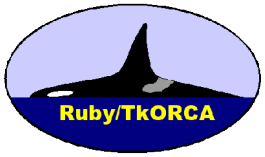
どこでもGUI!!

(GUI,Anyware!!)



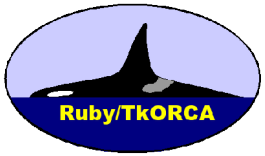
なぜそのようなものが欲しいのか？

- せっかく作ったGUIなんだから，そのままにローカルでも外からでも使えるようにしたい
- 外に公開するためにGUIの操作性を犠牲にしなきゃならないなんて，冗談でしょ
- 外から使いたくなるかどうかわからないような段階から公開のことを考えて作成するなんて，労力の無駄
- 公開用に新たに作り直すなんて，そんな面倒なことはしたくない
- ローカルのGUIならグラフ化なんかもライブラリで一発なのに，外向けには画像化などの細工がいるなんて馬鹿馬鹿しい
- Ajax？ Coolなのが作れるのはわかるけど，クライアントの能力への依存が大きいし，サーバとクライアントとのそれぞれ用にスクリプトを作ったりとお手軽じゃないよなぁ
- ローカルでちょっと使うだけの時にまで，重たい環境や重たいブラウザなんて立ち上げたくない
- 外に公開するときは，動作状況の監視くらいはしたいよね



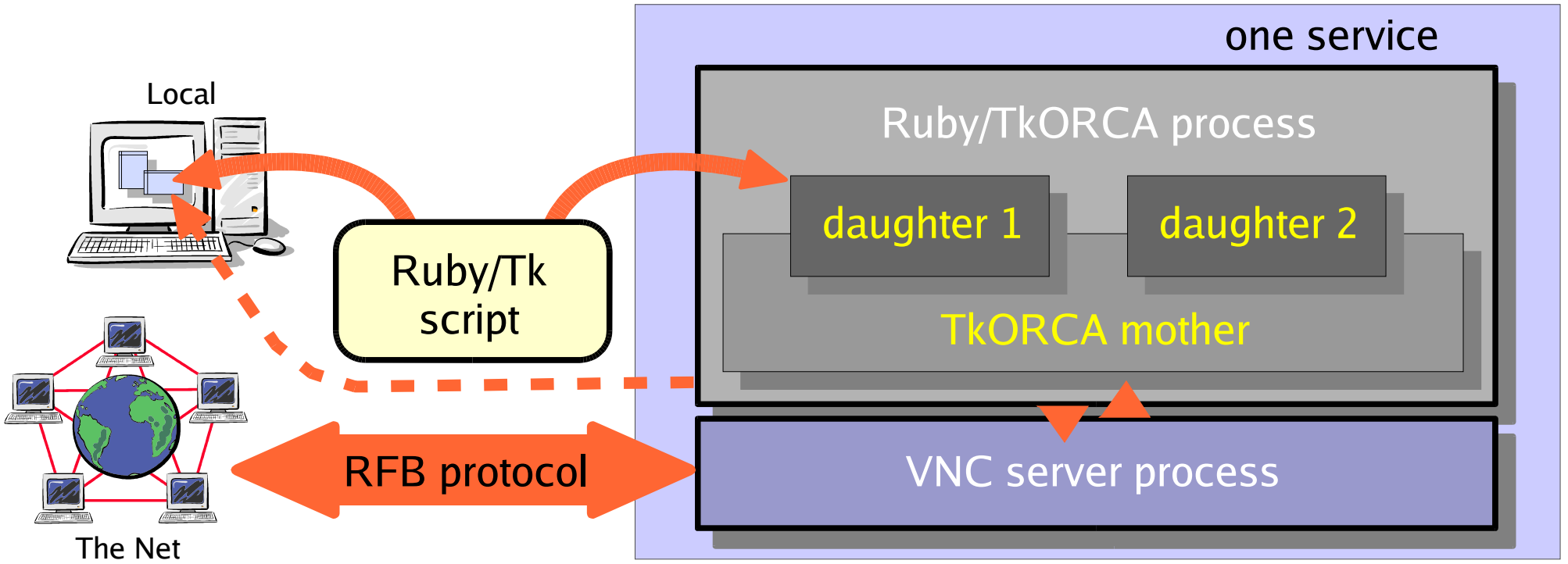
求められる要素は？

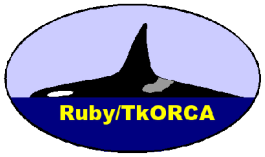
- ローカルのウィンドウシステムで単独で動かせるGUIアプリケーションを，そのまま変更なしに実行環境で動かせること
- 実行環境上では，自動的にセキュリティ配慮がなされること
- アプリケーションが他プロセスを起動しない限り，すべて単一プロセス(＋VNCサーバプロセス)上で完結していること
- アプリケーション実行環境とウィンドウマネージャ機能とが単一のプロセスの上に実装されていること
- 単一プロセス上ではあっても，うまく分離ができていること
  - 例えばウィンドウマネージャを構成するウィジェットをアプリケーションから直接いじれないようになっているなど
- 分離はされていても，アプリケーションの実行の監視や制約は可能であること
  - 例えばアプリケーションが異常な数のウィンドウを生成してしまうのを阻止できるようになっているなど



どうやって実現するのか？

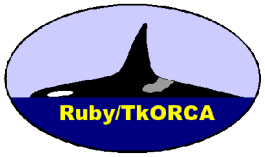
- Ruby/Tk によって次のものを一つのプロセス上に実現
  - mother　：ウィンドウマネージャ機能とプロセスの総括
  - daugher ：公開用アプリケーション実行環境 (複数可)
- 基本はこのプロセスと VNC サーバプロセスとの二つで 1 サービス
- 公開したい Ruby/Tk スクリプトを daughter に読み込んで実行





なぜ「Ruby/Tk」なのか？

- mother & daughter を実装できる機能の存在
  - MultiTkIp クラスによる複数の Tk インタープリタ (IP) の駆使
    - IP は enclose された ThreadGroup で分離
    - Ruby/Tk スクリプトを IP の文脈で評価可能
  - ウィジェット管理が IP ごとに分離
    - ウィジェットオブジェクトは IP の情報を持たない
      → 他の IP のウィジェットオブジェクトを入手しても操作不能
  - safe slave IP を生成可能 (Ruby と Tk との両方でのセーフ機構)
  - master IP から slave IP の監視・操作が可能
- ウィンドウマネージャの代わりを務めることができる機能の存在
  - frame ウィジェットのコンテナ機能
  - canvas ウィジェットの埋め込みウィンドウ機能
  - canvas のスクロールによる表示可能サイズよりも広い画面領域
  - Tk への埋め込みが可能な非 Tk ライブラリも多い
    - 埋め込みさえできれば，非 Tk の画面出力でも支配下に置ける
    - safe Tk での稼働を想定していないものも多い点には要注意

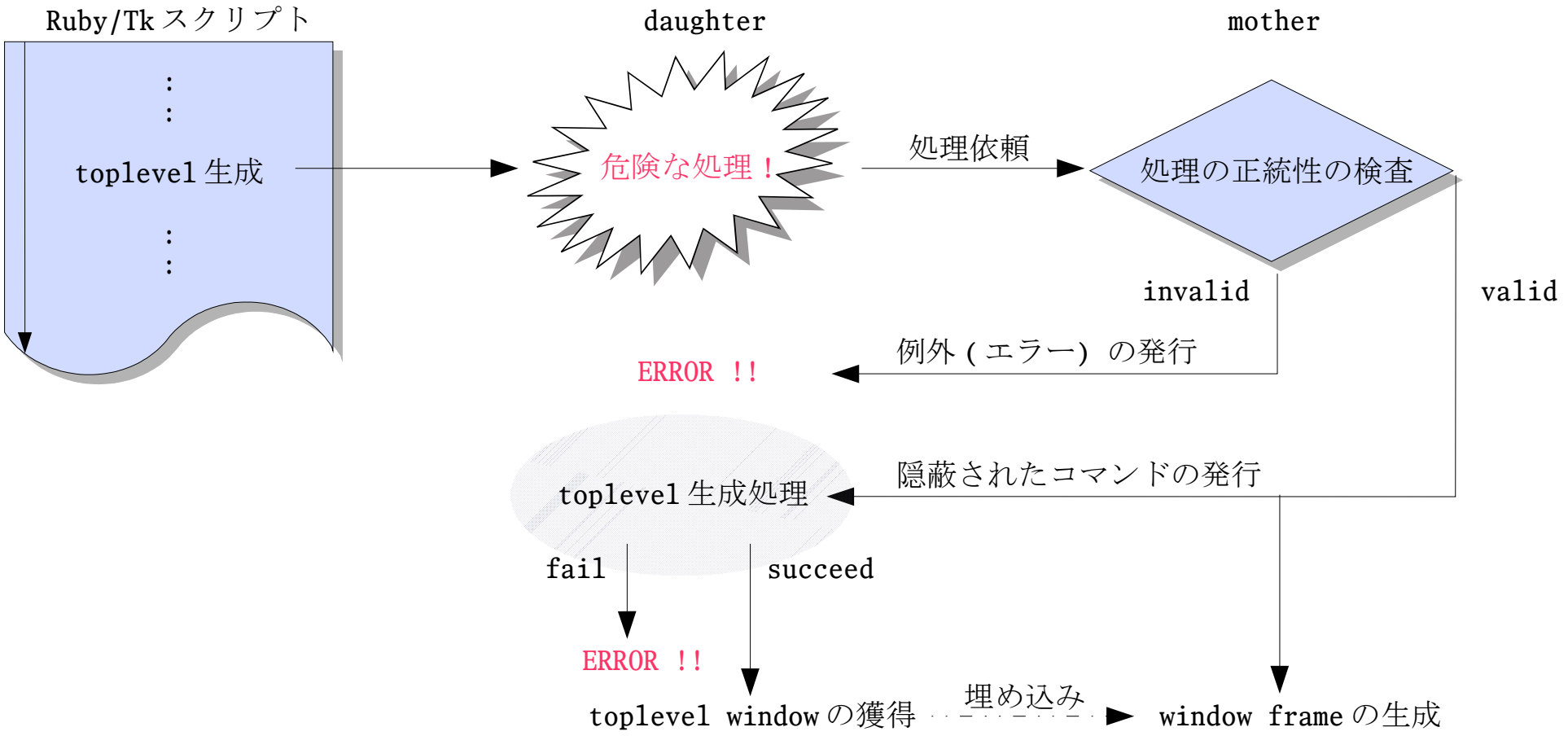


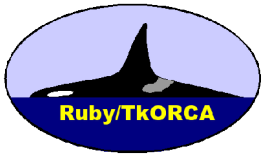
例：トップレベルウィンドウの生成

トップレベルウィンドウの生成を無条件に認めるのは危険

↓

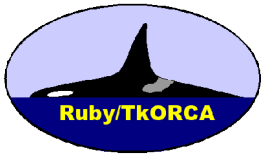
生成は監視の下で





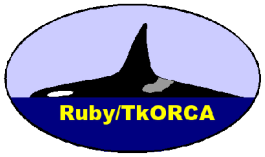
どんな人にとって有益か？

- GUI を持ったデモシステムやサービスを，ネットワークを介して広く公開したい人
- マルチウィンドウアプリケーションを，ローカルからネット公開までシームレスに開発したい人
- 高度なインタラクティブ性を持つネットワークアプリケーション (Web アプリを含む) を簡単かつ気軽に製作し公開したい人
- それなりのインタラクティブ性は持たせたいが，Ajax のようなものは面倒だと考えている人
- クライアントに情報を提示はしたいが，データを直接クライアントに送りたくはない人
- 自分で作成した GUI 付きのちょっとしたツールを，ローカルでも外からでも同じに使えるようにしたい人
- 他，GUI + ネットワーク公開というキーワードに興味を引かれる人



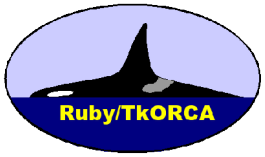
どのような GUI が作れるのか？

- GUI 部分が Ruby/Tk で作られているものなら，どんなものでも
- GUI のインタラクティブ性はそのまま維持可能．フレームレートがさほど高くなければアニメーション表示も十分可能
- 各種の Tk 拡張ライブラリの使用に制限なし
- Tcl/Tk のコンテナ（フレームウィジェットやトップレベルウィジェット）への埋め込みができるライブラリも可
- 画面表示を行わないライブラリなら，利用の制約は全くなし
- 表示サイズよりも広い仮想画面（スクロール可）が利用可能
- サーバ・クライアント間でのファイルやデータの転送機能はなし（←将来の課題の一部）
- リアルタイム性が求められるものやサウンドの送出などは対象外（←外部コマンドと連携する手段としては提供すべきか?）



どのようなスキルが要求されるのか？

- 必須となるスキル
  - GUI の最表層部は Ruby/Tk となるため， Ruby および Ruby/Tk (Tcl/Tk でも可) の最低限の知識は必要
- 推奨されるスキル
  - 通常は $SAFE==4 の sandbox での実行となるため， $SAFE==4 での制約や決まりごとを理解していることが望ましい
  - 基本の監視機能の範囲以上の細かい監視や制御の実現を望むなら， Ruby/Tk での Tk インタープリタ操作のスキルも必要
- VNC などの基板となる部分は隠蔽されるため，それについての知識は必須ではない

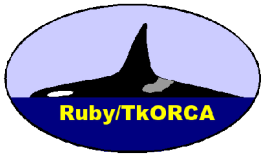


どの程度の手間が必要か？

- ローカルで動くものを作るまでは，通常の GUI アプリ作成と同じ
- sandbox 内で動くことをチェックし，必要なら修正をするという手間は必要
  - 「sandbox 内で動かない」 == 「そのまま公開するのは危険」であろうから，この作業の必要性はやむを得ない
  - 「sandbox 内で動く」 == 「最低限の安全性は確保」と言えるようにするのが目標
- 詳細な監視がしたければ，監視スクリプト作成が必要
  - ある程度の監視スクリプト用 API は提供する予定
- チェックが完了したら，サーバの設定に追加して公開へ
  - 必要な手間はわずか．例えば，外部からの要求に対して

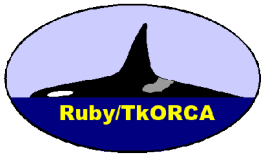
    tkORCA_mother [-a <監視スクリプト(必要なら)>] <GUI アプリ> ...

    などというような起動がなされるように設定する程度



実際に実現できるのか？ 試作品はあるか？

- ローカルでのサンプル実行の実例



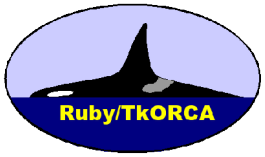
実際に実現できるのか？ 試作品はあるか？

- ローカルでのサンプル実行の実例
- 遅いマシンだが，コンセプト実証用実験サーバが存在
  - Web ブラウザによるアクセス
    (Java アプレットが実行可能であること)

    http://131.206.154.81/
  - VNC ビューアによるアクセス

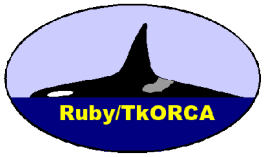
    IP addr:131.206.154.81 Port:5933

※ いずれの場合も IP addr:131.206.154.81 , Port:5933 へのパケットをファイアウォール等が通過させることが必要



試作品が動いてるなら，もうほとんど完成か？

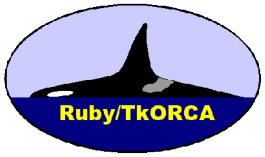
- まだまだ，全然ダメ！
  - ウィンドウコントロールの実装済み機能はほんのわずか
  - アプリケーションからの要求への対応皆無
  - 入出力コンソールの代用品なし (shell は動かさない！)
  - 危険な命令の wrap 不十分
  - 監視や実行制約の機能も未実装
  - 複数 daughter への対応も未完
  - カスタマイズ機能もなし
  - 当然ながら，ドキュメントもなし
- こういう目的に合った VNC サーバがない
  - 試作品では特殊な細工を施して実現
  - パッチを提供する？　VNC サーバ自体を提供する？
  - サーバ設定ツールも必要か



将来は？

- 監視機能やカスタマイズ機能等のさらなる強化や追加
- Tk の描画要求を処理する部分で直接に RFB プロトコルを喋らせたい
  - 公開用アプリケーションが pure Ruby/Tk スクリプトであれば，１サービスにつき１プロセスだけでサービス提供が可能
  - サーバ負荷の軽量化に直結
- サーバ・クライアント間データ転送機能を提供できるようにしたい
  - RFB プロトコルでの通信と並列にソケットを開設すれば現在でも可能だが，利便性が低い
  - RFB プロトコルによる通信への寄生かプロトコル自体の拡張によって，ソケットを増やさずに対応したい
- 実現に条件が付くとは思うが，セッションの継続 (GUI の永続化) も視野に入れたい

ご清聴，ありがとうございました



既存のアプローチでは？

- Ajax 等の thick client アプローチ（個別に独自のプロトコルを使用する場合を含む）
  - 気をつかう部分が多く，開発が面倒！
  - ローカルで使うときには実行環境が不必要に重い！
  - クライアント部がオープンなのは，嬉しくないことも多い？
- X プロトコル
  - クライアント制約が大きい上，プロトコル自体が重すぎ！
- VNC（RFB プロトコル）
  - 有力候補だが，ウィンドウマネージャが癌！
    - 操作権限の与えすぎるにもかかわらず，動作監視ができない
    - 稼働プロセス数が増えるなど，結構重い
    - GUI 操作性維持には必須（Win では置き換えすらできない）