ネット公開マルチウィンドウGUIアプリ用フレームワーク
Ruby/TkORCA におけるウィジェット操作権限の分離手法

九州工業大学 情報工学部 知能情報工学科
永井 秀利 (nagai@ai.kyutech.ac.jp)

Ruby/TkORCAとは

- 2005年度下期未踏ソフトウェア創造事業「ネット公開を目的としたマルチウィンドウアプリ用フレームワーク」での開発物
- 正式名称　Ruby/Tk On Remote-Frame-Buffered Canvas
  （Ruby/Tk On RFB Canvas）
- 通称(略称)　Ruby/TkORCA 〔ルビー・ティーケー・オルカ〕
- 開発目的
  ローカルのウィンドウシステム上に限られていたGUIプログラミング技術の対象領域を，広くネットワークGUIアプリにまで拡張する
  ⇒ ローカルからネットまで「お気軽に」GUIプログラミングを楽しめるようにしたい
- コンセプト
  「どこでもGUI！」（GUI, Anywhere！）

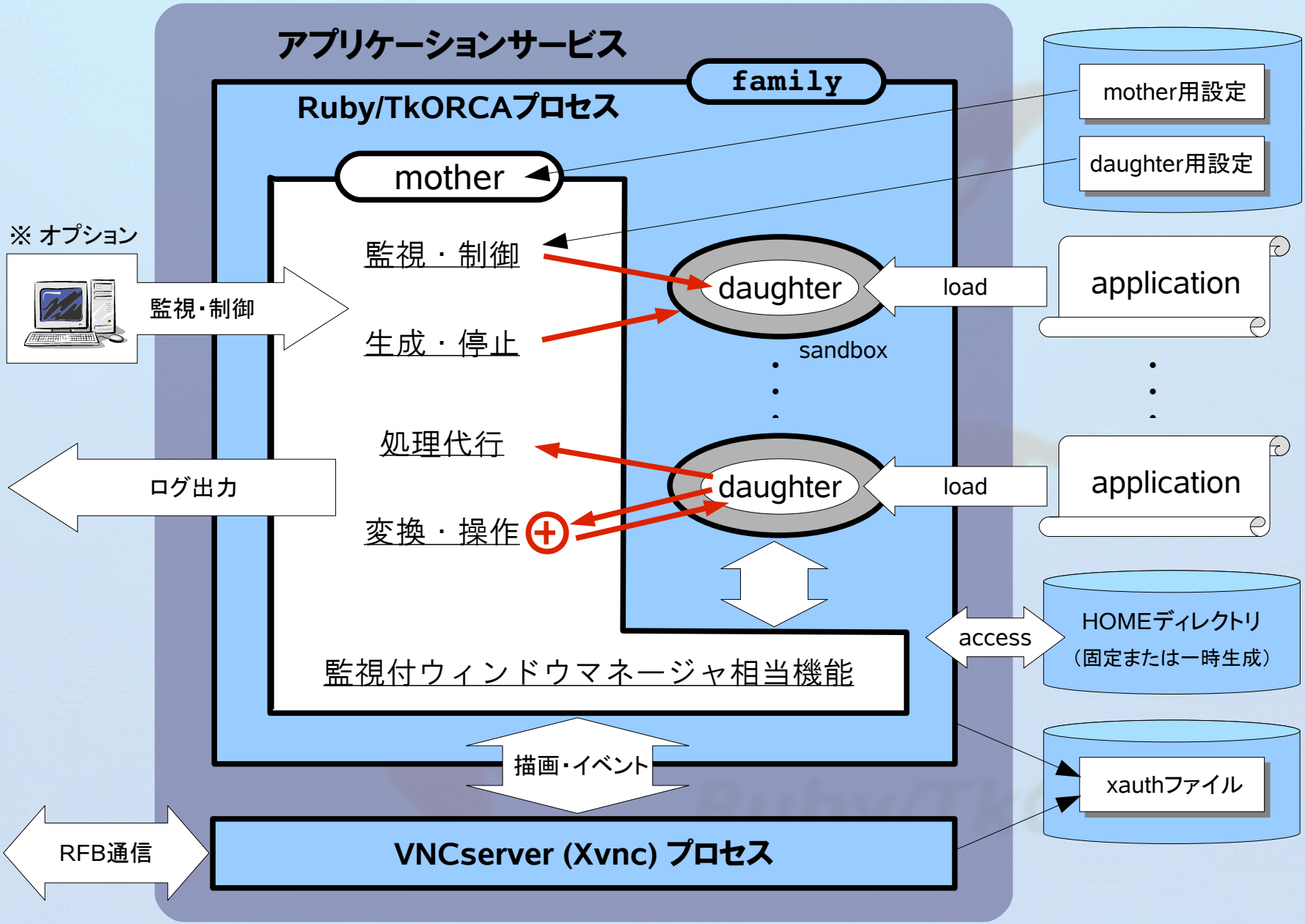
現状でのネットGUIアプリ作成時の問題点

- 基本傾向として，能力が高い枠組ほど制約も厳しく，想定範囲を越えようとすると実現が非常に困難だったり不可能だったりが多い
- Flash等の独自フレームワーク
  - 開発者に特有の技量を要求
  - インストールされたクライアント用プログラムのバージョンに依存するため，適用可能範囲が狭く，発展や機能改良も鈍重
- Ajax（ Asynchronous JavaScript + XML ）
  - サーバ/クライアントの連携やクライアントの差異への配慮が面倒
  - 機能向上に対し，開発コストや情報漏洩リスク等が急激に増大
- VNC（ Virtual Network Computing ）
  - 接続を許可した相手を信用し過ぎており，監視や制御ができない
  - 不特定多数をサービス対象とする仕組みが欠落している
- X window system (X protocol)
  - 初期化が非常に重く，LANクラスの速度でなければ起動を待ち切れない
  - アプリケーションサービス側からの攻撃には脆弱

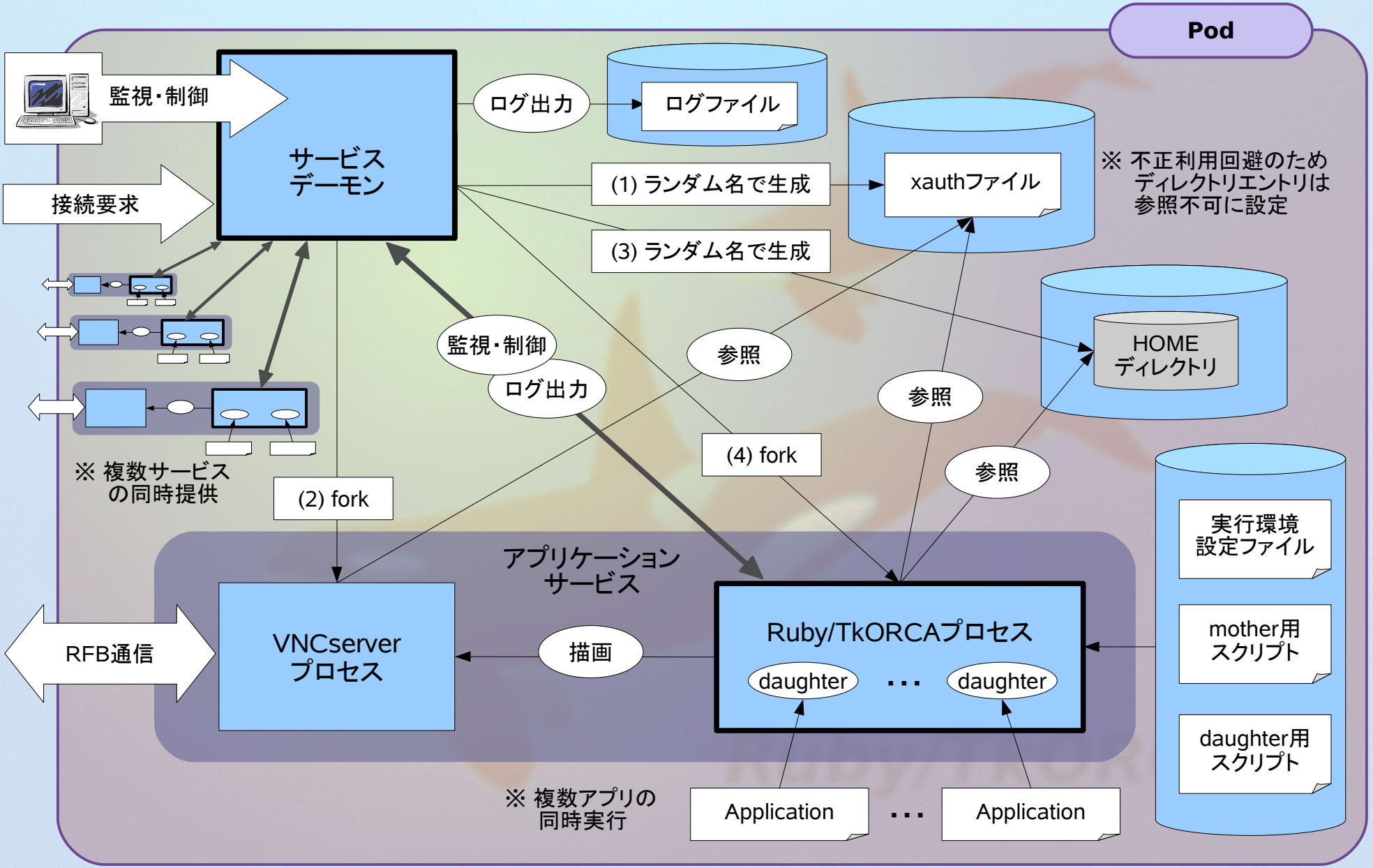
Ruby/TkORCAの思想

- VNCの利点を活かしつつ，その問題点を回避した枠組とする
  - 高度なインタラクティブ性や，プログラムやデータをクライアントに送らないことによる低い情報漏洩リスクなどの利点はそのままに享受
  - 問題の源になる一般的ウィンドウマネージャは動かさず，相当機能の実装によりウィンドウ操作も監視や制御の対象とする
- 不特定多数の他者へのサービス提供に利用できるものとする
- 開発者に対して，特有で特殊な技量を要求しない
- ローカル用とネット用とでソース変更をほとんど必要としない
- 監視や制御の機構をあらかじめ組み込んでいなくても，アプリケーションの監視や制御を可能に（改良・強化も容易に）する
- 援用する他のソフトには一切手を加えない
  - 導入コスト低減や，他ソフトの強化・改良からの最大限の受益のため
  - 広く使われているものに「似て異なるもの」を開発する愚は避ける

Ruby/TkORCAの構造

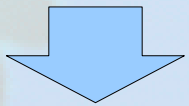


Ruby/TkORCAサーバの構成概略図



なぜ「Ruby/Tk」なのか

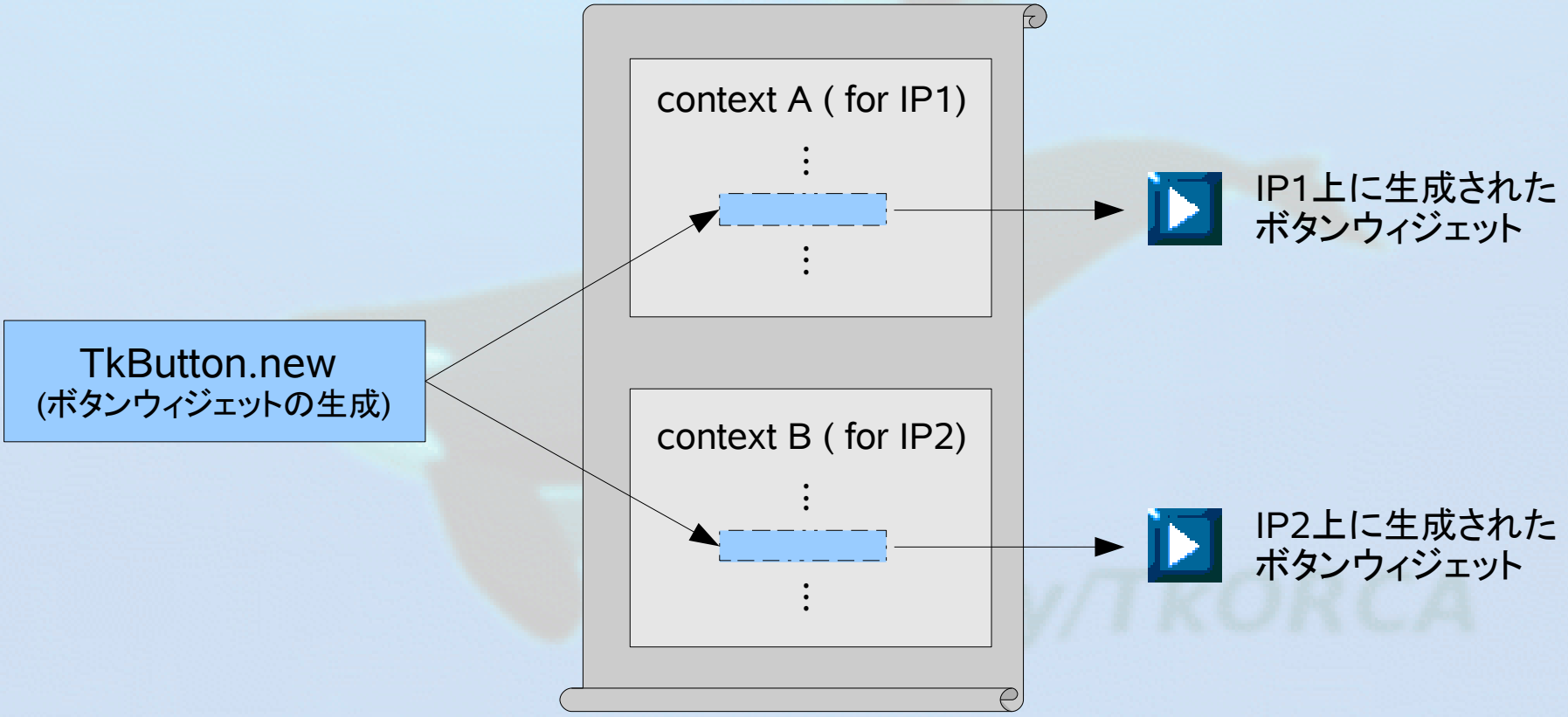
- 手軽に使えるGUIツールキットの一つ
  - 多くのOSで使える高いポータビリティ
  - 手軽にGUIを作れるTcl/Tkにオブジェクト指向の殻を被せてさらに手軽に
    ⇒ ウィジェットパス名に悩まされない，
    新しいウィジェットクラスの作成も容易に，等
- ウィジェットはTcl/Tkインタープリタに依存して存在
  - デフォルトのIPは，ライブラリのロード時に自動生成して定数に保持
  - ウィジェットオブジェクトは依存するIPの情報を持たない
    ⇒ オブジェクトを入手しただけでは操作できず，IPの操作権獲得も必須
- スレーブTkインタープリタの生成が可能
  - マスターからスレーブを操作することはできるが，逆は不可



『手軽さ』+『mother/daughterを実装できる素地』を持つ

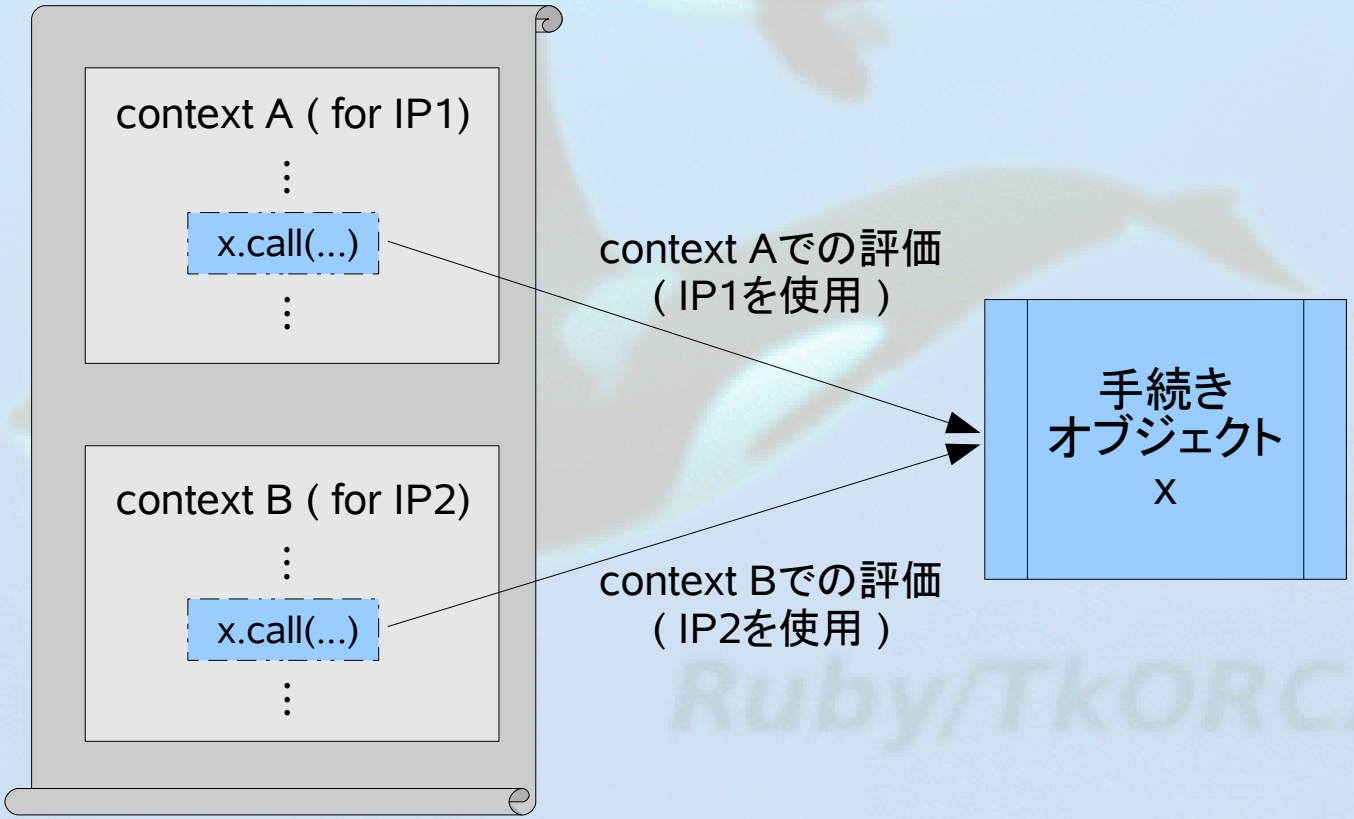
問題点(1)

- ウィジェットのコンストラクタやインスタンスメソッドには，Tkインタープリタを指定する引数がない
  - スクリプトのコンテキストに基づいて，対象となるTkインタープリタを自動選択する必要がある



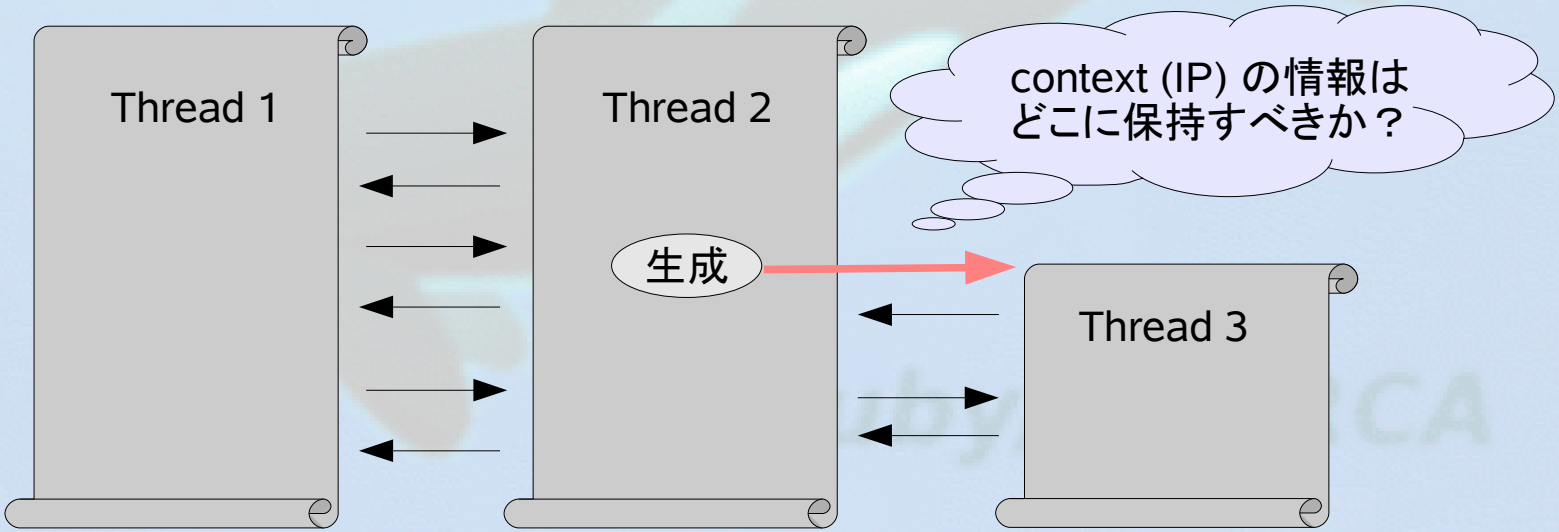
問題点(2)

- 同じ手続きオブジェクト を異なるコンテキストで評価できる必要がある（例えばコールバック処理などでの手続き流用)
  - 手続きオブジェクトの生成時に固定してしまうようなことはできず，動的に定める必要がある



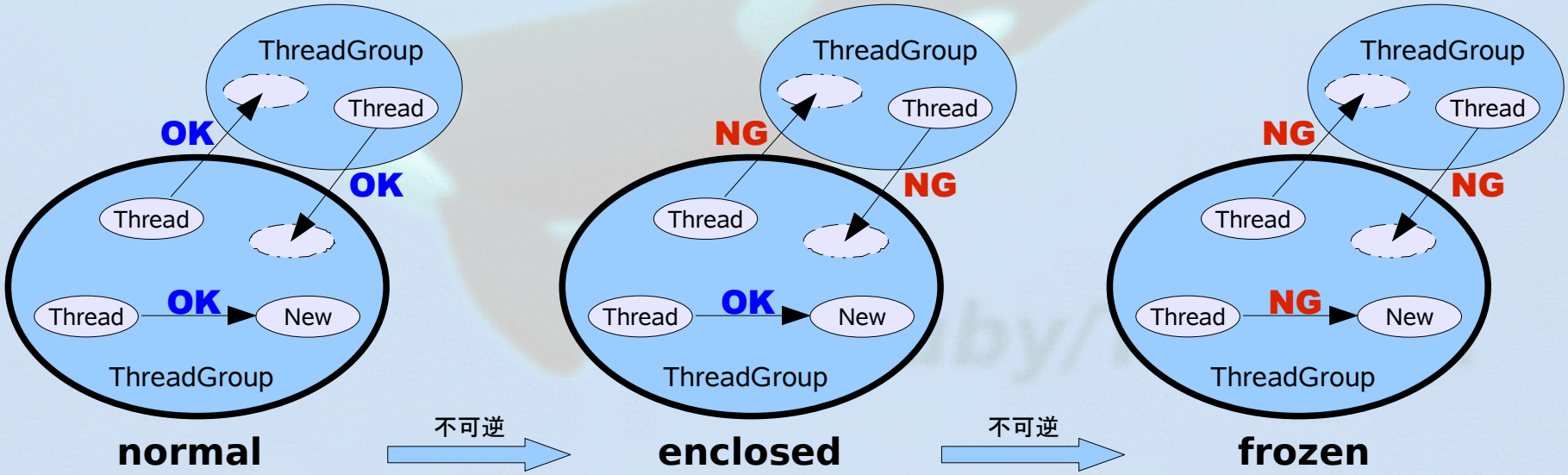
問題点(3)

- スクリプトにおいて，デフォルトのTkインタープリタを保持する定数を直接参照してメソッド呼び出しを行うことがある
- スレッドの存在に注意が必要
  - 処理中のスレッド切替えを考えると，選択すべきTkインタープリタの情報はスレッド固有である必要がある
  - 新しいスレッドが生成される際，自動的に確実にTkインタープリタの情報を新しいスレッドの固有情報として引き継げる保証がない



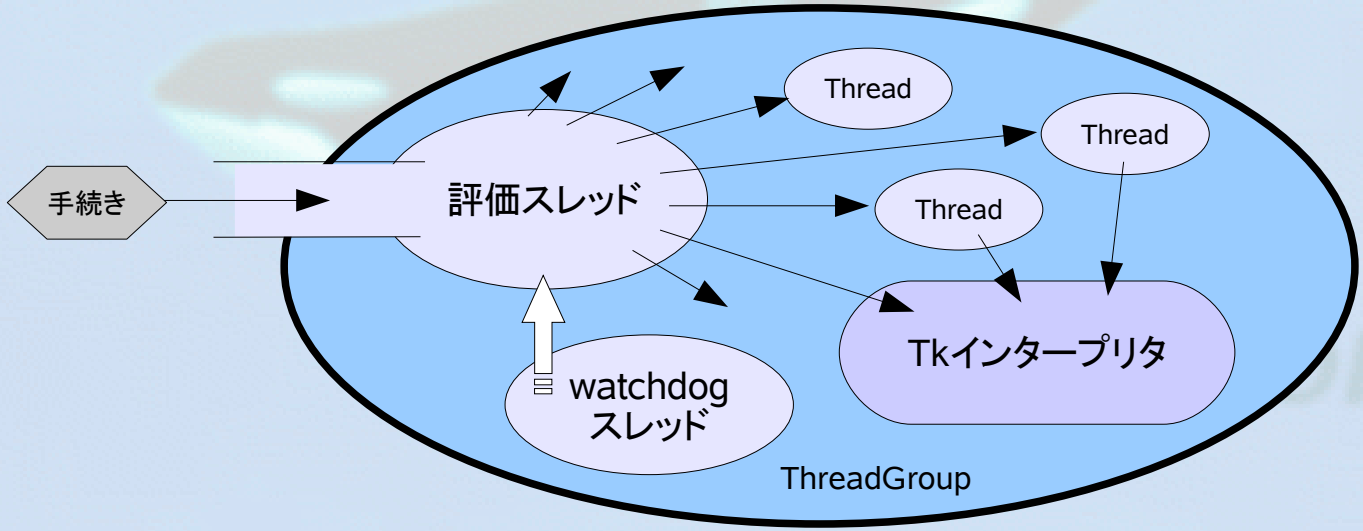
スレッドグループの活用

- Rubyのスレッドグループを活用
  - スレッドグループの「enclosed」という状態を活用
    - 以前に提案し，導入してもらったもの
    - 新たなスレッドを生成することはできるが，スレッドを別のスレッドグループに移したり，別のスレッドグループからスレッドを追加したりすることができないという状態
  - コンテキストはカレントスレッドが属するスレッドグループによって規定
    - 新たなスレッドを生成してもコンテキストの維持が保証できる



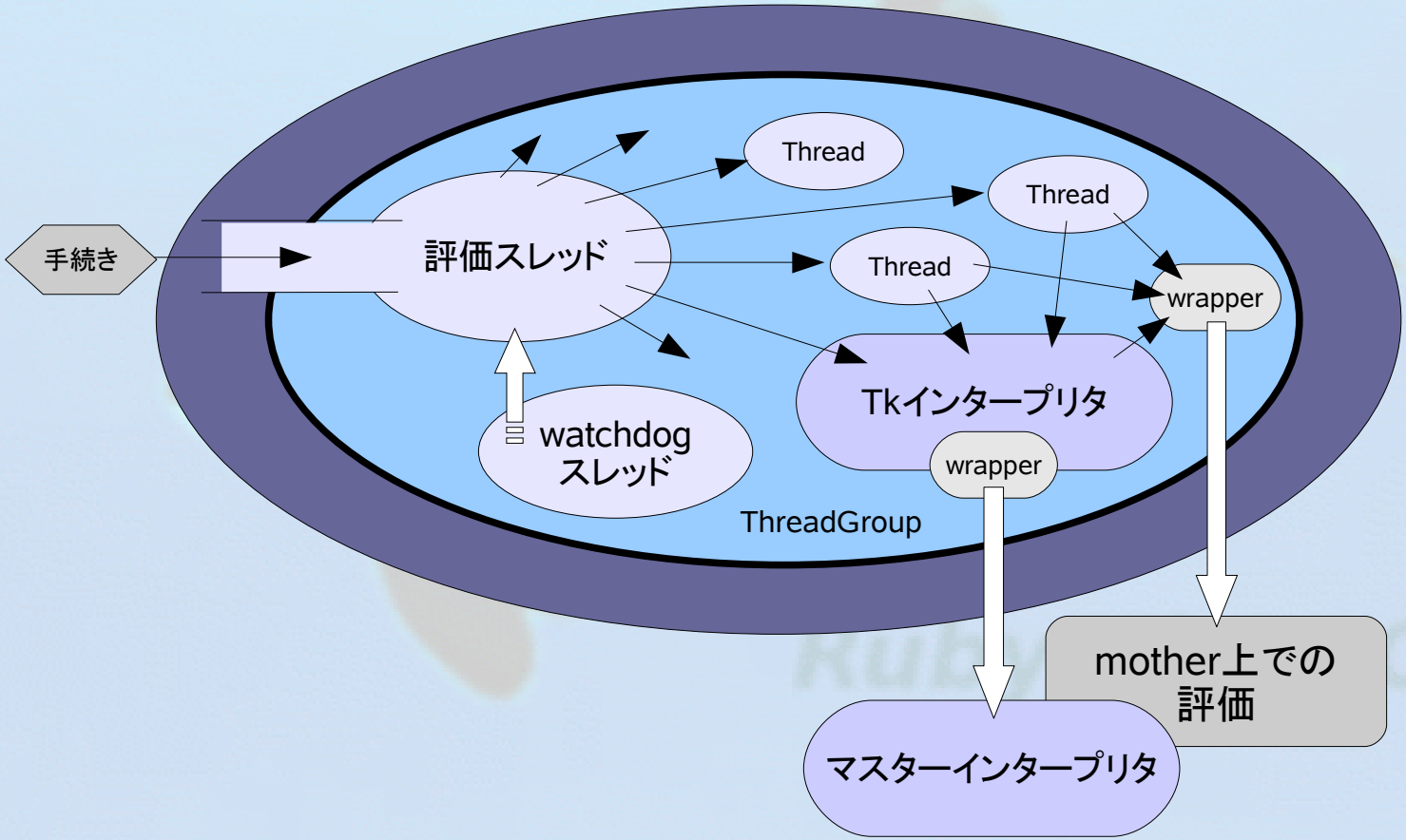
MultiTkIpクラス

- MultiTkIpクラス
  - スレッドグループとそれに対応付けたTkインタープリタとで構築
  - クラスメソッド呼び出しは現コンテキストのIPのメソッド呼び出しとなる
  - IPが現コンテキストのIPか，その直接のスレーブである場合にのみメソッド呼び出しが許可される
  - 命令は，手続きオブジェクト化して評価queueに投入される
  - 評価queueでシリアライズされた命令を評価スレッドで順に評価し，必要なら結果を返す (値の返却には例外生起を活用)



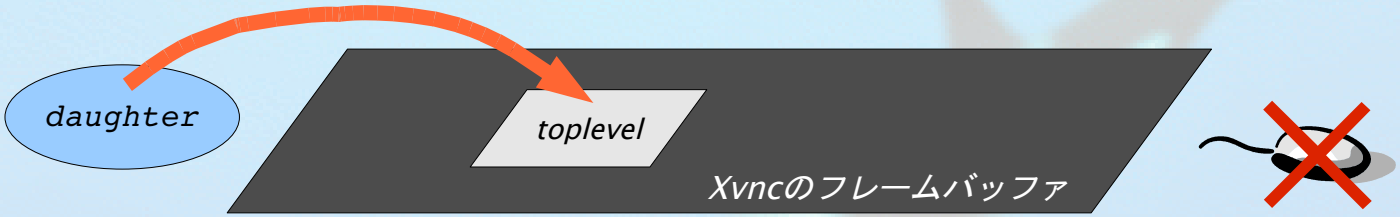
ウィジェット集合の封じ込め

- MultiTkIpオブジェクトを中心としてdaughter (sandbox)を構築
  - daughterごとのウィジェット集合の封じ込めを実現
  - 包み込むことでMultiTkIpオブジェクトへのアクセスを制約
  - Tcl/Tkコマンドやメソッドのwrapによる監視と処理依頼の実装

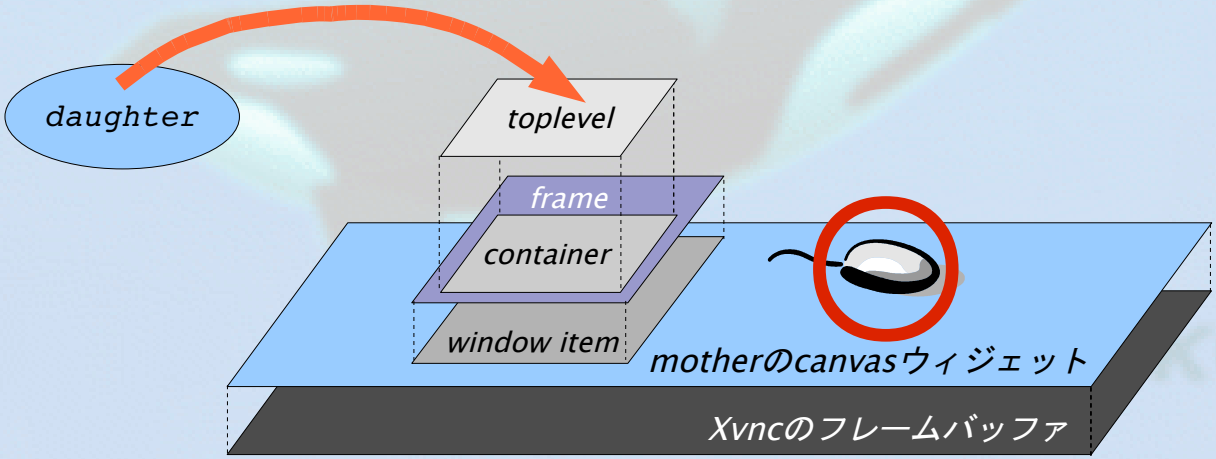


ウィンドウマネージャ機能の構築

- ウィジェット集合を封じ込めただけではダメ
  - ウィンドウマネージャが存在しないため，ウィンドウ操作不能



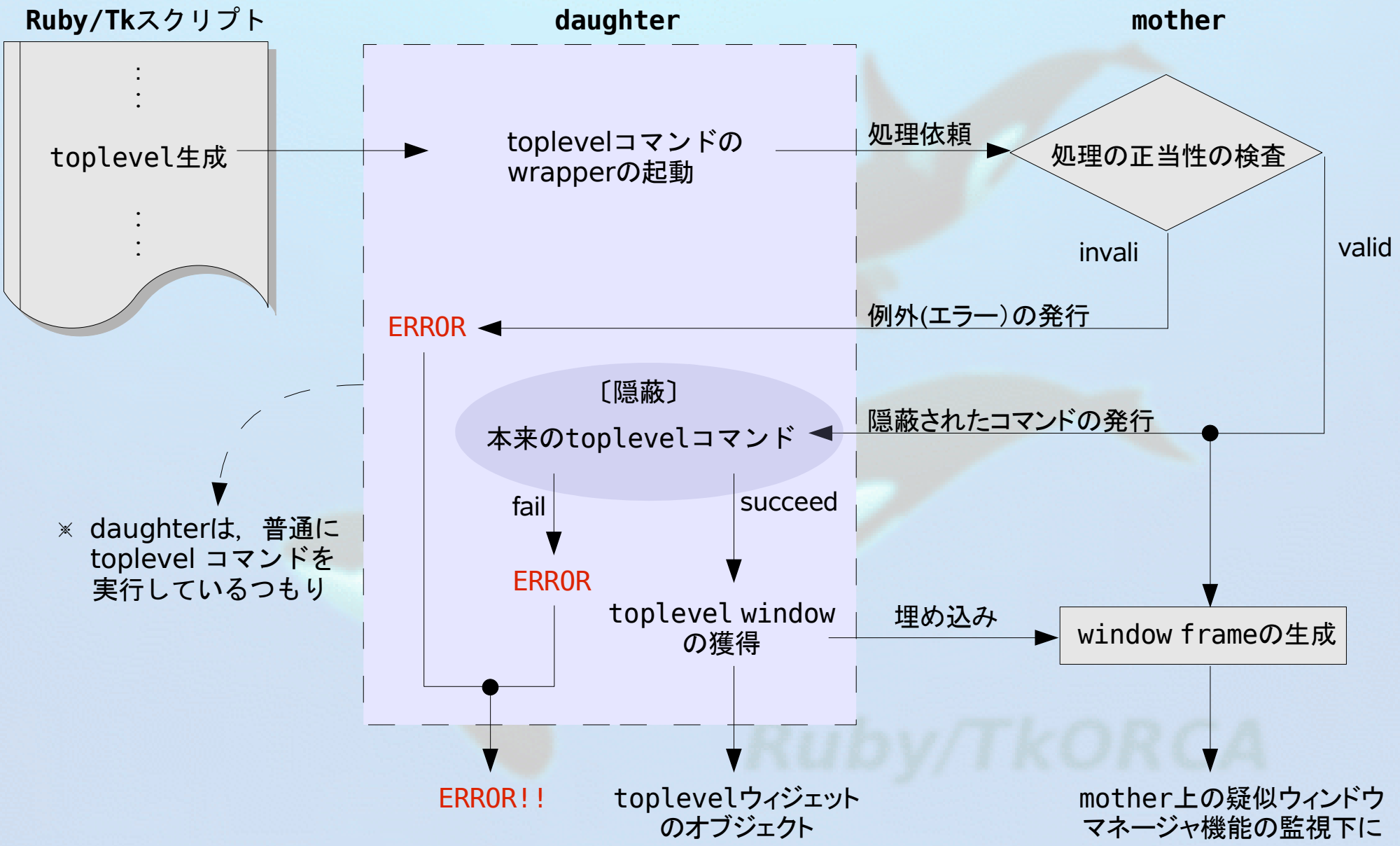
- motherに属するマスターTkインタープリタのキャンバスウィジェット上にウィンドウマネージャ機能を構築
  - マスターIPのコンテナにスレーブIPのトップレベルウィジェットを埋め込む



Tcl/Tkコマンドのwrap

- 一般的なウィンドウマネージャは稼働していない
  - ⇒ daughter上でのウィンドウマネージャ関係コマンドは，ほぼすべてが機能しない
  - ⇒ motherのウィンドウマネージャ機能を呼び出す必要あり
- daughterのTkインタープリタ上のウィンドウマネージャに係わるコマンドのすべてをwrapする
  - daughterのIPはスレーブインタープリタであるので，マスターインタープリタであるmotherのIPによる支配的操作が可能
  - toplevel, wm, winfo, grab等のコマンドをwrap（ motherへの処理依頼や情報問い合わせ ）
  - daughterからは，普通にコマンドを実行しているだけに見える
  - ウィンドウ操作がすべてmotherの処理を経由する
    ⇒ ウィンドウ操作のすべてをmotherの監視・制御の下に置ける
  - ⇒ daughter上のGUIアプリの全体 (アプリ内部からUIまで) をきめ細やかに監視・制御することも可能

例：toplevelコマンドのwrap



デモンストレーション

- C言語で作られた外部ライブラリを使用し，コンソール出力もするようなアプリケーション
- OpenGL (Mesa) を用いているアプリケーション
- 複数アプリケーションの同時サービス
- 5分で作るネットワークGUIアプリ

・・・ などから，時間が許す範囲で

まとめ

- ネットワークGUIアプリ用フレームワークRuby/TkORCAにおいて，一つのプロセス上で，ウィジェット集合の操作権限を分離しつつ，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を実装するために用いた方法を示した
- Ruby/TkORCAを用いれば，小物のGUIツールを手間をかけずにリモートアプリ化したり，特殊なライブラリや機密情報を一切見せることなく外部公開したりといったことも簡単にできる
- 具体的な性能数値計測が優先課題
  - 操作による画面更新がなければ通信量はわずかなので，接続のみなら最大の99クライアント(Xvncの仕様による制限)でも十分に対応可能
  - 一般のVNCサーバは通信帯域制限の機構を持たないため，高レートのアニメーションなどで画面更新が非常に頻繁なものを最高クラスの品質で無条件に提供すると，数クライアントでも通信帯域を使い切ってしまう場合がある (⇒　大規模サービスでは通信帯域制限が必須)
  - サービスの品質と (OSによる) 通信帯域制限との組み合わせに対し，どの程度の数のクライアントの実用に耐えうるかの計測が必要