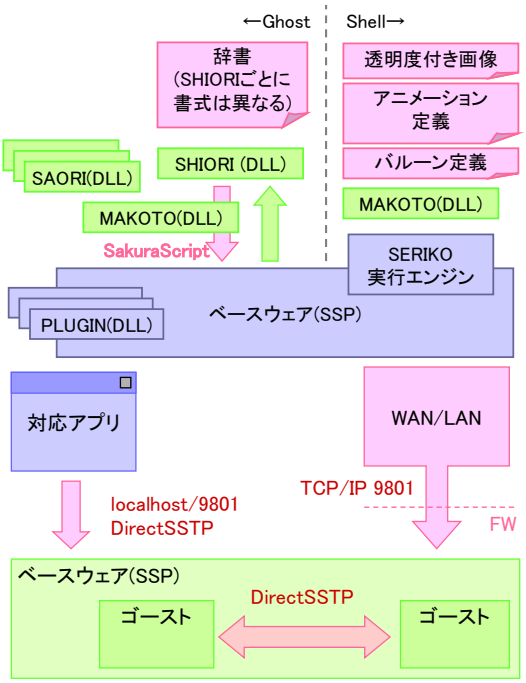
The Design and Evolution of "Ghost" [Rev.2]

ぽな@ばぐとら / SSP BUGTRAQ

はじめに:構造図

よく当勉強会の構造解説図として掲載されている右記の図、俗に「伺か」と呼ばれる一システムの内部の概略だ。この構造のそれぞれの要素のやりとりの裏側で「何が行われているのか」「その設計になった理由」「歴史的事情」をもとに、現時点の構造の問題点と利点、そして今後の拡張について考える。



1.全体の構造概観

- やたらと結合がユルい設計
  シェル(見た目)とゴースト(SHIORI・SAORI)間など
- 細かく制御できず不満もたくさん聞かれる
- なぜゆるい結合?

2.内部通信:SHIORI・SAORI・PLUGIN

- 内部通信は単なる DLL の関数呼び出し・HTTP もどきのテキストによる通信→他もみんな似てる
- DLL で問題が起きればみんなコケる
- 別プロセス分離の検討と実装(CROW)→互換性問題頻発であまり良い結果は出せなかった

3.外部通信:(Direct)SSTP・FMO

- マシン外から通信するための TCP/IP を使った(Socket)SSTP→もちろん内部でも使える
- マシン内で完結するプロセス間通信機構を使った DirectSSTP
- DirectSSTP を支える技術:hwnd・FMO・WM_COPYDATA
- 通信機構が Windows べったり過ぎる・ウインドウメッセージでの通信の設計は古い
  →パイプ(例:しなちく)など別の IPC 機構は使えないか?
- 外部連携可能なことの功罪…外部連携アプリ開発の手軽さとセキュリティ問題

4.伺か以外の例

- 「シェル:見た目定義部」を分けている構造:キャラリナ
  →直接描画もできるので密結合気味、その分作りこめばやれることはとんでもなく多い
  →SakuraScript は記述が「短縮形」すぎて難解:綾織のほうが楽な人もいるかも
- 「喋り」「画像アニメーション」「その他機能実行」を全部まとめて XML ファイルに定義:Apricot
  ややこしい技術はなく全部テキストで完結するので見通しがよく楽→やれることとのトレードオフ

5.今後の展開

- ゴースト・シェル・バルーン間の連携強化はいずれ必要になる→密結合に向かう
- ゆるい結合の利点…
  オブジェクト指向・イベント駆動・多層アーキテクチャ
- 密結合での設計にした場合、今の「制御のラクさ」「柔軟さ」はありえない
  →密結合を引き起こす機能提案はあえて拒否していることもあります