Napa GUI Renewal

Legacy Transformation of 2M sloc Napabasic Motif GUI to Ruby WPF GUI

We all love legacy! ☺

Napa GUI Renewal

1. Motivation for renewal
2. Scope or work
3. Business environment
4. Alternatives for implementation
5. Technical challenges and Solutions
6. Project challenges and solutions

1. Motivation for GUI renewal

- Motif = Old
  - Emerged in the 1980s
  - No significant development since 1994
  - Presumed “dead” since around 2000
  - Low in features compared to modern toolkits
  - Ugly look-n-feel
  - No native support on Windows
  - Native API in C/C++ (Not our choice)
  - Limited 3rd party components
  - Difficult to find developers

2. Scope of Motif + NapaBasic GUI

- Developed between 1996...today
  - 2...30 developers, ~150 man-years so far
- GUI sourcecode
  - 2200 widget-definitions
  - Widget layouts with custom tree files
  - NapaBasic event handlers and logic
  - 40 MB source code (~2.0 msloc)

3. Business Environment

- Continous pressure to keep adding features to the product
- Maintenance development for existing customers
- Custom projects of customised features for special customers
- No possible of ”freezing” the feature development during major technology overhaul

4. Alternatives for implementation

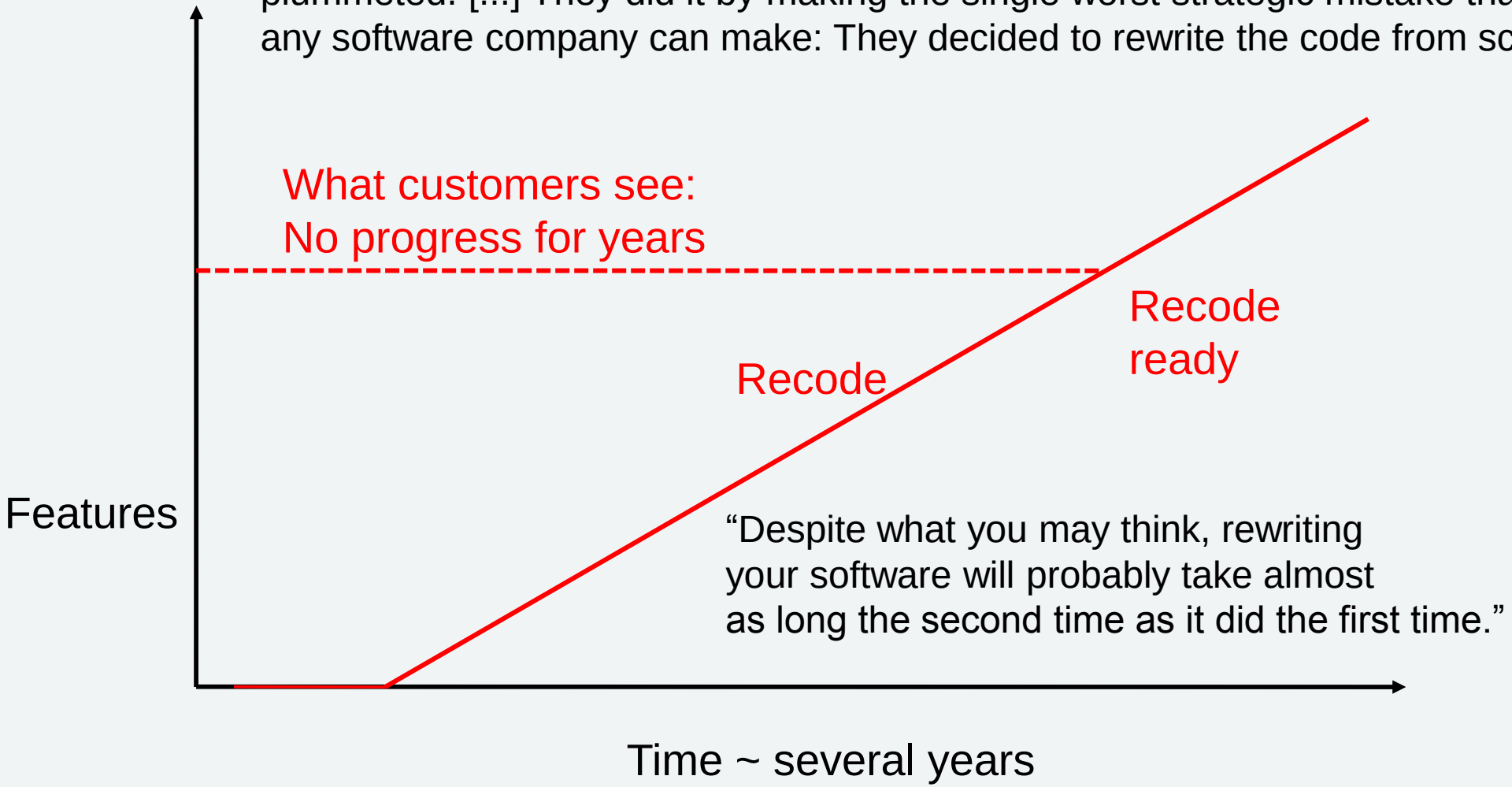
- Q1: Recode vs. Reuse codebase ?
- Q2: Target platform?
- Q3: Target language(s)?

Q1: Recode vs. Reuse codebase?

- Recode
  - Recode piecewise
    - Motif/Napabasic ←→ Wpf/Ruby: monstrous interoperability challenges
  - Recode in one go
- Reuse
  - Reuse Napabasic ”as is”
    - Reuse old parser and redirect GUI calls
    - New parser and redirect GUI calls
  - Transform to code that creates new GUI ★

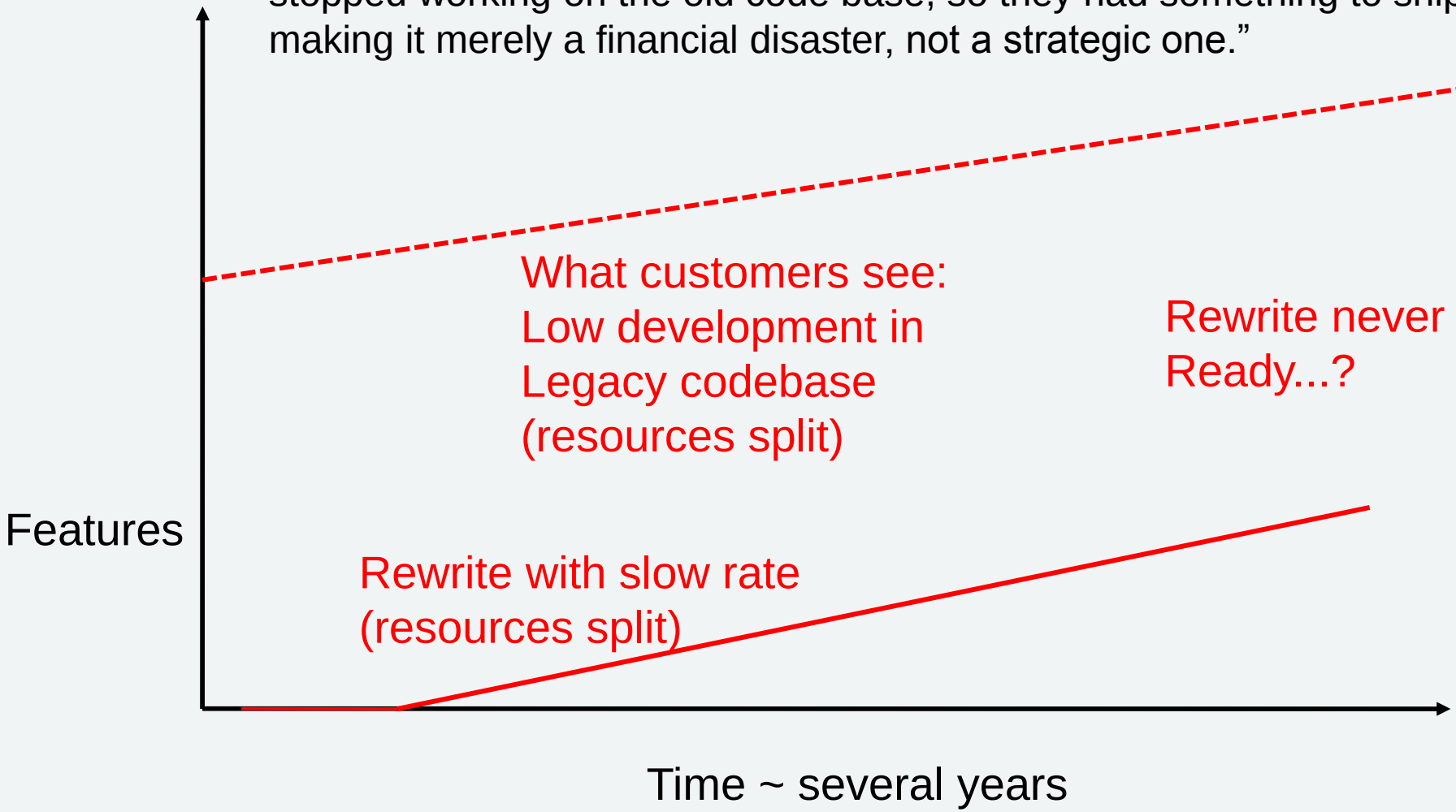
Full Recode scenario ☹

Joel on Software: “Netscape 6.0 finally went to public beta in April 2000. The last major release was almost three years ago. Three years is an awfully long time. During this time, Netscape sat by, helplessly, as their market share plummeted. [...] They did it by making the single worst strategic mistake that any software company can make: They decided to rewrite the code from scratch."

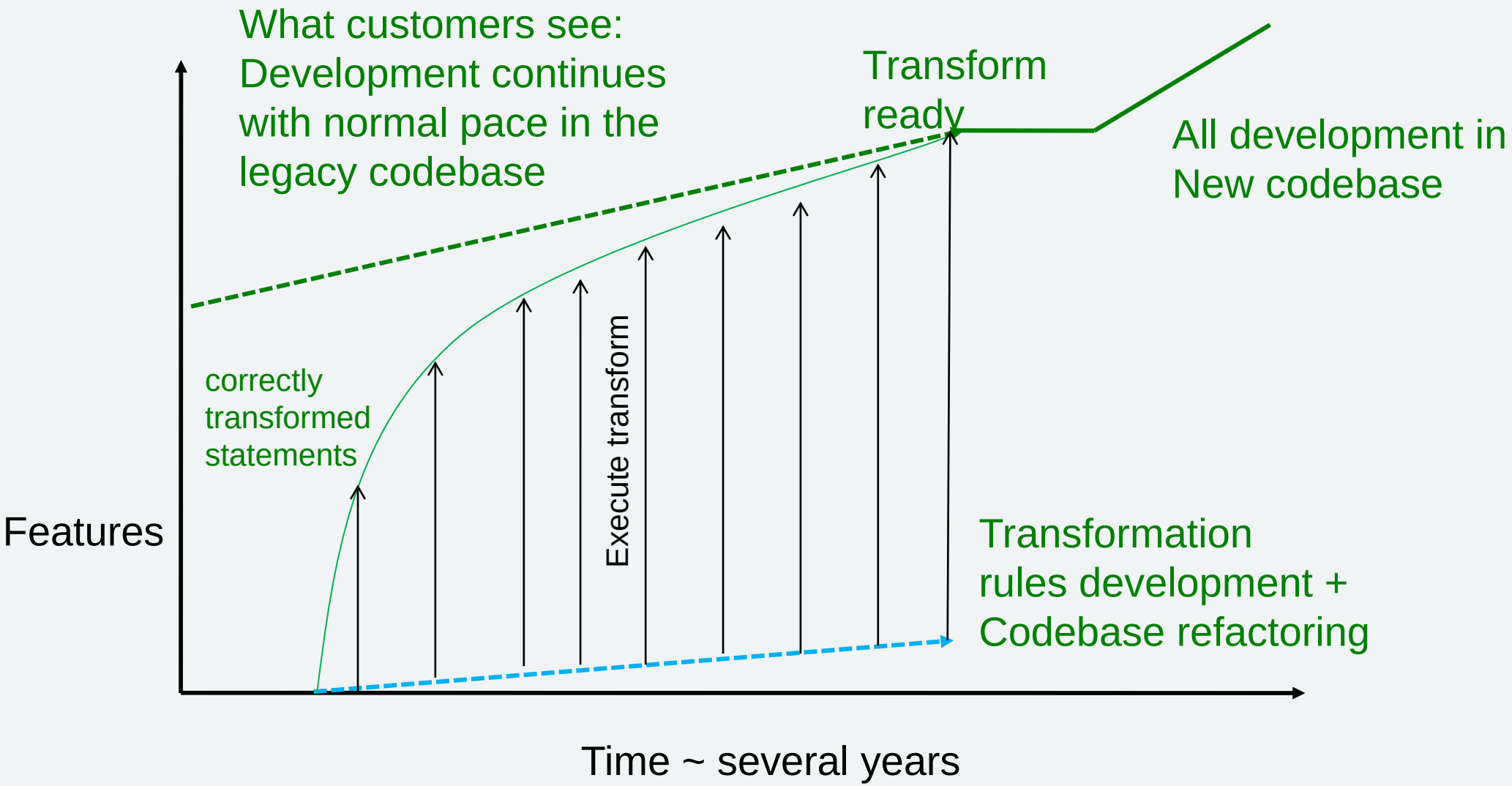


”Recode on the side” scenario ☹

“Microsoft almost made the same mistake, trying to rewrite Word for Windows from scratch in a doomed project called Pyramid which was shut down, thrown away, and swept under the rug. Lucky for Microsoft, they had never stopped working on the old code base, so they had something to ship, making it merely a financial disaster, not a strategic one.”



Legacy Transform



Q2: Target platform

- Java / Swing / Java3D
  - + Existing investement in Java at Napa
  - + Multiple vendors for Java
  - + Good multiplatform support
- DotNet / WPF / Hoops 3D ★
  - + Technical features of the platform
  - + Accelerated 3D graphics support
  - + Integration of Napa with other SW

Q3: Target Language for transform

- C#
  - + Most standard, Most supported on Dotnet
  - + High performance
  - + Best integration with WPF
- (Iron)Ruby ★
  - + Power and flexibility of dynamic language
  - + Better match for weakly typed NapaBasic
  - + Allows tweaks and tricks to help in the more difficult challenges of transformation

5. Technical challenges and solutions

a) Complex multi-part grammar
b) Mismatch of widget classes
c) Mismatch of widget attributes / events
d) Lacking language features in target
e) Lacking type information
f) Mismatch of types
g) Layering violation in the platform
h) Graphic areas

Complex multi-part grammar

- NapaBasic has evolved during 25 years
- Originally commands given interactively
- Later lists of commands to ”macros”
- Later variable references @x
- Later @goto and @label ... @if... Then
- Later syntax for GUI callbacks ${id}
- The grammar layers are implemented as a series of preprocessors – no single parser
- ANTLR Grammar + Parser developed

Mismatch of widget classes

- PUSHBUTTON -> Button (ok)
- FORM -> ?
- MENUPANE + MENUBUTTON -> Menuitem
- COMBOBOX multipart

Mismatch of widget attributes / events

- @Mtf.SetResource(acheckbox,'selected','True')
  - acheckbox.is_checked = true
- @Mtf.SetResource(btn,'background',c(i))
  - btn.background = Wpf::SolidColorBrush.new(c[i-1].to_color)
  - btn.background = c[i-1].to_color.to_brush
- @ui.getarr(${id},'children[n]',c)
  - c = self.children_names

Lacking language features in target

- GOTO
- SUBROUTINE
- OUT/VAR parameters

Lacking type information

- @if (a < b) @goto x
  - a and b can be strings in Napabasic
- @if x = 0 then
  - X can be widget (nil in Ruby)

- Solutions:
  1. type inference (for eg. Widgets)
  2. Refactoring sources

Mismatch of types

- @x='True'
  - Lack of booleans in Napabasic
- @wid=ui.wid('Par*MyWindow')
- @s=os.str('w#',wid)
  - Widget id used as integer, but object in Wpf

Layering violation in the platform

- GUI events calling fortran core...
- ...and fortran core calling GUI
- Impossible to support in WPF

Graphic areas

- Important areas of NAPA GUI (3D CAD)
- Simple from widget structure point of view
- Complex custom Fortran/C implementation
- Solution: Recoding replacement (stable)

Most powerful enabling factors

1. IronRuby flexibility / monkey-patching
2. Refactoring legacy sources to transform
   - Complete removal of many features
3. Extract method + map

6. Project challenges and solutions

- Getting people testing and using new GUI
- Re-testing all parts of GUI
- Resources partly tied to maintenance of the legacy SW
- Effort/Schedule Estimation difficult
- Schedule slip... Problem?

Estimation

- Development of transformation rules: 2008/03 – 2011/06 (1-4 people at a time)
- Debugging WPF GUI and fixing transform: 2011/06 - current
- Estimation difficult in the debug phase:
  - Like walking in forest: seeing the nearby trees
  - More focus on refactoring in fixing badly transforming code
  - Method A: Extrapolate error frequency
  - Method B: Extrapolate from fixed codebase

Risks

- Transformation technologically feasible?
  - 10% risk
- Final post-transform fixes take too long?
  - Minimize by taking transform far enough
- Estimation fails
- GUI usability development ignored
- Lack of faith -> People leave company

Plan ahead

- More recruitment to GUI renewal, especially in Romania
- 3 current developers from India
- Help from other teams, 1 per team allocated...? (How works with Scrum?)
- Subcontracting in Finland...?
- Lessening feature-development pressure

Conclusions

- Weird source language makes conversion tricky – but also more valuable
- Significant problems encountered but powerful solutions found.
- During the project, feature development has been proceeding without freeze