

Infomercial

Prism:

"microarray technology" for clinical neonatology

powered by

Mindwrapper:

software wrappers for the way you think

Donnal Walter, M.D., Ph.D.

Neonatology Section Meeting

October 13, 2004

- I. An infomercial
 - a. Promoting a product/service, but...
 - b. Also weighing potential benefits and risks
 - c. Consider this a research project
- II. Two "deliverables"
 - a. **Prism**: a set of custom clinical applications
 - b. **Mindwrapper**: the underlying framework
- III. Prism introduction
 - a. Metaphor: *microarray technology*
 - i. "A tool used to sift through and analyze the information contained within a genome."
 - ii. "Tool for studying how large numbers of genes interact with each other and how a cell's regulatory networks control vast batteries of genes simultaneously."
 - iii. By analogy, Prism is a tool for analyzing how items of clinical information interact with each other.
 - iv. Looking at details *and* getting the big picture
 - b. The name is *not* an acronym
 - i. Not "Patient Record", or "Information System", not "Model" or "Matrix" etc.
 - ii. Revealing the rainbow of vivid colors in white light
 - iii. (And potentially rejoining the colors together again)
 - c. Characteristics
 - i. Customized and customizable
 - ii. Evolutionary by design
 - iii. Modular
- IV. Mindwrapper introduction
 - a. The name is a pun
 - i. wrapping one's mind around a complex domain
 - ii. wrappers make complex software more accessible
 - iii. wrapping also means encapsulation
 - 1. hiding details
 - 2. exposing essentials
 - b. Mindwrapper is a framework
 - i. one-tier, two-layer architecture
 - ii. hierarchical information model
 - iii. declarative notation
 - iv. built-in object-oriented database management

- V. Prism demonstration
 - a. This demo slightly premature, but only slightly
 - i. Mindwrapper is 90-95 % complete (> 3 yr)
 - ii. Prism is 30-40 % complete (<< 3 mo)
 - iii. Prism development generally much more rapid
 - b. Authentication process
 - i. perhaps not strictly necessary
 - ii. but will make HIPAA compliance easier
 - iii. preliminary to e-signing
 - iv. tracking provider-specific data
 - c. "Splash screen"
 - i. attractive entry to application
 - ii. display progress on loading application and data
 - d. Provider window
 - i. overview
 - 1. menubar
 - 2. identification panel
 - 3. notebook of functions
 - ii. patient lists
 - 1. "My List"(s) maintained by user
 - 2. team-oriented lists
 - 3. NICU list
 - 4. complete list
 - iii. sorting patient lists
 - 1. by last name (or even by first name)
 - 2. by bed location
 - 3. by date of birth (or age in days)
 - 4. by hospital of origin
 - 5. other information
 - a. gestational age
 - b. birth weight
 - iv. other provider functions
 - 1. "todo" list
 - 2. resources
 - a. referring doctor contact information
 - b. commonly used phone numbers, pagers
 - c. but not try to duplicate other sources

- e. Patient focus window
 - i. overview
 - 1. screen resolution
 - a. [800 x 600] vs [1024 x 768] or higher
 - b. maximized vs "restore down"
 - 2. menubar
 - 3. no tool bar at present
 - 4. identification panel
 - 5. notebook of functions
 - ii. example functions
 - 1. history (maternal and birth)
 - a. amount of information on screen
 - b. layout of information
 - c. drop-down lists, etc
 - d. narrative text areas
 - 2. physical exam
 - a. amount of information on screen
 - b. radio buttons and check boxes
 - c. "not assessed" status
 - d. auto-normal, auto-update ??
 - e. more narrative style, "phrase building" ??
 - 3. pharmacokinetics
 - a. best current demo of interaction
 - b. easy to incorporate calculations
 - 4. nutrition
 - a. replacement of "FeN Planner"
 - b. with additional features
 - 5. fluids
 - 6. laboratory
 - a. table/list format
 - b. automatic downloading from Meditech
 - c. manual entering of data if necessary
 - 7. tracking
 - 8. growth
 - 9. billing

VI. A peek at the Prism code in Mindwrapper syntax

a. dependent cells:

```
class KElim(mw.Number):
    scale = 0
    digits = 3
    def forward(pk, tr, hr):
        return (log(pk) - log(tr)) / hr

class VolDist(mw.Number):
    scale = 0
    digits = 2
    def forward(kE, dose, cPk, tDose, tInf, tPost):
        d = kE * tInf
        a = exp(-d)
        b = exp(-kE * tPost)
        c = exp(-kE * tDose)
        x = ((1 - a) * b) / ((1 - c) * d) # adjustment
        vol = dose / cPk # volume of distr
        return vol * x # adjusted v distr
```

b. data maps:

```
class Pharmacokinetics(mw.Map):
    def assemble(self):
        self.add('time', Timing)
        self.add('dose', mw.Number,
                scale = 0.1, digits = 1)
        self.add('peak', mw.Number,
                scale = 0.1, digits = 1)
        self.add('trough', mw.Number,
                scale = 0.1, digits = 1)
        self.add('weight', mw.Number,
                scale = 0.001, digits = 3)
        self.add('kElim', KElim,
                args = [self.peak,
                       self.trough,
                       self.time.hrPkTr])
        self.add('volDist', VolDist,
                scale = 0.0001,
                digits = 4,
                args = [self.kElim,
                       self.dose,
                       self.peak,
                       self.time.hrDosing,
                       self.time.hrInf,
                       self.time.hrPostInf])
        ...
```

c. a presenter

```
class PatientPanel(mw.Subpanel):
    cols = 6
    growRows = [1]
    growCols = [0, 2, 3, 4, 5]
    def assemble(self, ref):
        self.add(node = mw.StaticText,
                 align = mw.CENTER,
                 text = 'Weight')
        self.add(node = mw.Spacer)
        self.add(node = mw.StaticText,
                 align = mw.CENTER,
                 text = 'kElim')
        self.add(node = mw.StaticText,
                 align = mw.CENTER,
                 text = 'vDist')
        self.add(node = mw.StaticText,
                 align = mw.CENTER,
                 text = 'vD/kg')
        self.add(node = mw.StaticText,
                 align = mw.CENTER,
                 text = 'clearance')

        self.add(node = mw.NumberField,
                 size = (50, -1),
                 align = mw.RIGHT,
                 margin = (0, 4),
                 args = ref.weight)
        self.add(node = mw.SpinButton,
                 margin = (0, 4),
                 align = mw.LEFT,
                 args = ref.weight)
        self.add(node = mw.TextReader,
                 size = (50, -1),
                 margin = (0, 4),
                 args = ref.kElim)
        self.add(node = mw.TextReader,
                 size = (50, -1),
                 margin = (0, 4),
                 args = ref.volDist)
        self.add(node = mw.TextReader,
                 size = (50, -1),
                 margin = (0, 4),
                 args = ref.volDistPerKg)
        self.add(node = mw.TextReader,
                 size = (50, -1),
                 margin = (0, 4),
                 args = ref.clearance)
```

d. the containing presenter

```
class WorkArea(mw.Panel):
    args = pk.Pharmacokinetics
    def assemble(self, pkref):
        self.add(None, KineticsPanel, grow = True,
                 args = pkref)
        self.add(None, PatientPanel, text = 'Patient',
                 margin = 4, grow = True,
                 args = pkref)
        self.add(None, TimingPanel, grow = True,
                 args = [pkref.time, pk.TimeCalculation])
```

- VII. Advantages (of Prism/Mindwrapper)
 - a. The software is "Free"
 - i. Not in the sense of "free beer"
 - ii. But in the sense of "free speech"
 - iii. We are in control
 - b. Highly mapped data: "microarray technology"
 - c. Flexibility
 - d. Maintainability
 - i. importance of Python
 - ii. importance of Mindwrapper
 - iii. importance of unit testing
- VIII. Potential risks
 - a. Scalability?
 - i. Small test cases work well
 - ii. But will it bog down or become unmanageable with increase in size of several orders of magnitude?
 - 1. application programmer interface (API)
 - 2. implementation (optimization)
 - b. "Open-source" limitations or drawbacks
 - i. Potential lack of support and maintenance
 - ii. Requires critical mass for sustainability
 - c. Potential integration problems with hospital infra-structure
 - i. Downloading information from the HIS
 - ii. Uploading information to a *potential* EMR
 - iii. But Prism may actually be easier to integrate

- IX. Philosophical considerations
- a. The *centering* discovery process



- b. The *fractal* structure of information
- c. The importance of *test-driven* development
- d. The importance of *encapsulation*
- e. The importance of *inheritance*

```
Node
  Model
    Cell (Text, Number, Timepoint)
    Map
      Folder (Unit, Cabinet)
  Presenter
    Subpanel (Panel, Frame)
    Notebook
    Adjuster (Slider, Spinbutton)
    Chooser (Choice, ...)
  ...
```

- X. Timeframe
 - a. Mindwrapper will surely never be "finished" per se,
 - i. but it is now 90% "complete" (or more).
 - ii. The most significant deficiency at present is the printing framework. If this turns out to be a bigger hurdle than expected, I have a fall-back plan that will take less time.
 - b. Prism will evolve even more dramatically than Mindwrapper and over a longer time, but the essentials are now 30-40% "complete", and development is considerably more rapid.
 - c. We need to schedule enough time for adequate testing.
 - d. We also need a robust "roll-out" strategy, or "migration path".