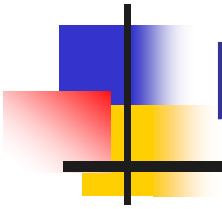
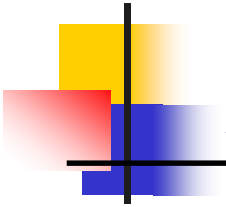


# Choice, Choices, Choices: Fat *versus* Thin Clients

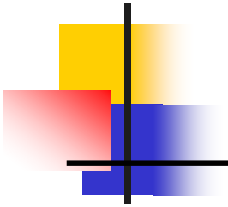
David Moskowitz

Productivity Solutions, Inc.



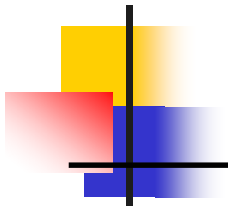
# Agenda

- Definitions
- Myths
- Choices
- Approaches
- Questions and…



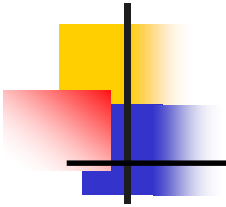
# On the Same Page…??

- A *fat* client: the application(s) reside and run on the user workstation
- A *thin* client: the application(s) reside and run on one or more server computers
  - The UI runs on the user workstation
- It's the business logic location…



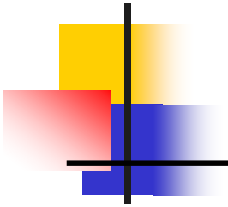
# Myths

- Thin means Web browser
  - Thin-ness is defined by where the process happens, not the implementation technology
- Web applications are thin
  - Usually, but not always. It is possible to have some "fatness" (Javascript or Applet downloaded to browser)
  - It's a "what runs where" issue, not app type



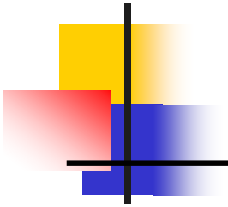
# Myths 2

- Fat clients are harder to deploy
  - Usually, but not always
    - ...if the application requires plug-ins
    - ...if a specific browser version or capability required
    - ...if there are inventory (h/w) issues
- Thin clients are easier to use
  - Thin clients are typically less interactive, less efficient, and can be harder to use than a well-designed fat client



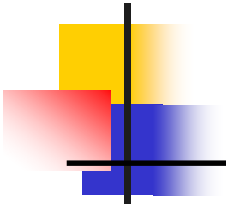
# Myths 3

- Thin clients are faster than fat clients
  - Depends…
    - upon both the application and the infrastructure
  - Most applications rarely use desktop power
    - Browser is very wasteful – thin clients
  - Overworked servers common
    - Thin clients exacerbate this by off-loading processing
    - N-tier distribution compensate for load
      - ✓ Adds its own complexities
    - Is it a client or a server problem?



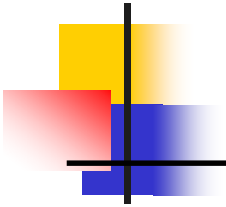
# Fat *versus* Thin

- Typical *thin* client argument: efficiency
  - Corporate efficiency
  - Controlled and efficient deployment
- Thin clients only allow specific behaviors
  - May result in lost productivity
- Fat (typically) clients offer richer UI
  - May allow broader use than original purpose
    - Examples…???



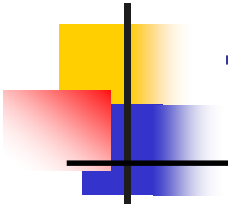
# Thin Doesn't Mean Spartan

- What about a rich-thin client?
  - User experience == fat (rich)
  - In other words: still provide a rich, expressive UI with work done on server
    - Aren't limited to just a Web browser (for the UI)
- Portal *vs.* thin *vs.* fat??
  - Where is the processing?
  - Portlet can be rich-thin
  - Tool set???



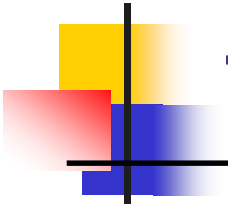
# Sometimes Thin Is Required!

- Remember, we're talking about more than laptop or desktop computers!!!
  - PDAs and mobile phones demand thin
  - Not just a matter of processing power
    - They aren't connected 100% of the time
    - Communication is slow
    - Small displays
    - Limited UI capability / mechanisms



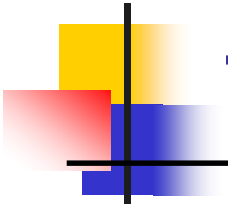
# Thin UI DOs and DON'Ts

- Don't initiate or participate in transactions
- Do accept data from users
  - Including gesture, voice, *etc*.
  - Assist entry with appropriate cues
- Do entry validation (proper range)
- Do restrict types of input (numbers, or…)
- Do require mandatory data



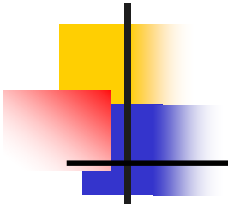
# Thin UI DOs and DON'Ts

- Don't massage data
  - Do connect to mapping / transformation objects
- Don't format/create SQL
  - This should be done by a service layer
- Do render data from business components
- Do provide user with status information
  - Working, connected, offline, busy…



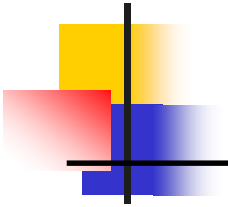
# Thin UI DOs and DON'Ts

- Perform appropriate format transformation
  - Including I18N or L10N
  - According to user customization preferences
  - To fit a particular device
- Bottom line: If it isn't part of data entry, rendering, related preparation/validation, communications, or related state(s) it doesn't belong in the thin client



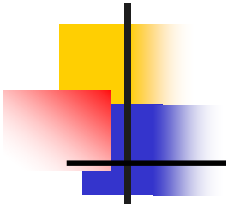
# Considerations

- Version tracking a fat client
- Version tracking thin client libraries
  - Infrastructure issues
- Cannot deploy a fat client everywhere
  - "thin" clients (mobile devices, *etc*.)
  - Multiple development efforts
- Easier to lock down behaviors for thin client
- ???



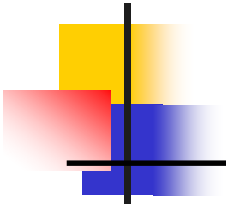
# Considerations

- Java client – write once…
  - Server-side and… ???
- .NET development can play very nicely
  - Doesn't have the market penetration of Java
  - Limits client side development to supported devices
- Internet-capable small devices
- TCO
  - Deployment
  - Maintenance
  - Upgrades
  - …???



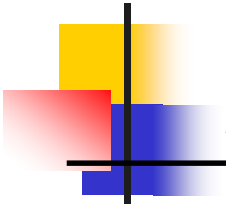
# TCO?

- How do you determine the cost of ownership associated with a User Interface?
  - …or is it like pornography… you know it when you see it?
- What about the cost of support?
  - Thin client potentially provides more control over what users can do
    - Is this an issue in your organization?
- Most user questions or problems have roots in UI design (Don Norman)
  - Increases TCO



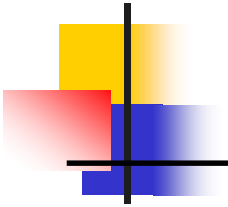
# While We're Talking about TCO

- There is also the cost of hardware
- Balance the cost of client power *versus* server power (or number of servers)
  - Upgrade cycle for workstations *vs*. upgrade cycle for servers
  - Server farms or clusters
- Browser wastes cycles, so….
  - …how much computing power do you need?
  - …what else is the h/w intended to do?



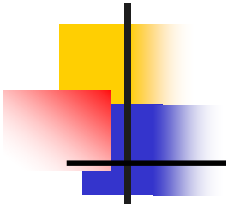
# Approaches

- Data-centric versus user-centric
- Service-centric – as part of UI design?
  - Why not?
  - Deliver **service** to the user…
- Thin GUI layer delegates many user interface responsibilities
  - Or does it?
  - Think patterns?



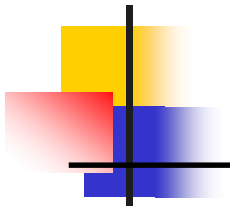
# Implementing the Client

- How do you craft a UI that works?
- Try an agile approach
  - ...with active and constant user involvement
  - Different agile: UI folks and end-users, not just programmers
- Prototyping is critical
  - It's more than stories (or use cases)
- General rule: team composition based upon delivered functionality (duh???)



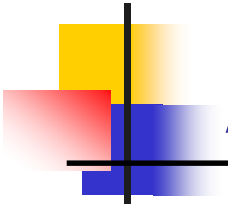
# Testing the Client

- What do you test?
  - How do you test a cell phone UI?
- Thin client
  - HtmlUnit
  - JWebUnit
    - Built on top of HttpUnit, provides different abstraction
    - Abstracts User/Browser rather than Browser/Server
  - HttpUnit
- Rich-thin and fat: add JUnit



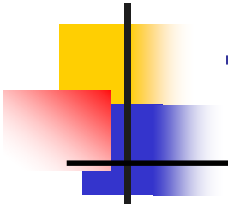
# User Friendly… Huh?

- Customizations?
- Personalization?
- User interaction
  - Does it get in the way?
  - Does it facilitate productivity?
  - The user should never have to adapt to the UI
    - How do you achieve this?
    - Does fat or thin make a difference in this regard?
    - What about a rich-thin client?
    - When is it (is it?) bad to require the user learn…?



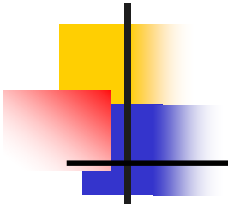
# Approaches and Patterns

- UI needs to work with different "backends" use the Separated Interface pattern (Fowler)
- UIs, particularly in small/mobile devices, need to track state
  - Use State pattern (Gamma *et al*)
    - Is there room?
  - UI state, not the transaction or the process state
- Don't forget the classics for decoupling: Decorator, Mediator, and Observer (Gamma)



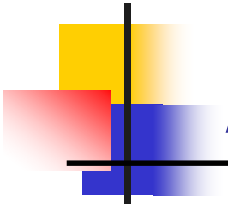
# The Blind Archer Trap

- One of the biggest problems building a UI
  - Maybe the entire application…???
- Build it without dedicated, committed, consistent end-user involvement
  - Joined at the hip!!
- Constant prototyping (driven by requirements) with end-users
  - Removes communications barriers
    - Totally avoids the difference between geek- and normal-speak
- Ticket to ultimate disaster



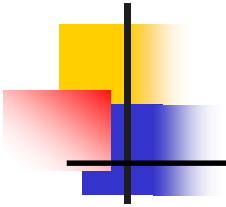
# Feature Creep

- Too easy to start thin and make it fat
  - Review the DOs & DON'Ts
- Increases testing required
  - Can be prevented by test first IFF tests are based upon current requirements and stories (use cases, whatever)
- Increases support costs…
  - …particularly if there isn't complete, constant, and continuous communications between development community, support groups, and end-users



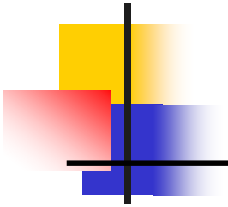
# Architecture Issues

- To the thin client: architecture looks 2-tier
  - It doesn't make any difference how many tiers there might be, now or in the future
- Architectural goal: the thin client is independent of the back end
  - Except for consistent and published interfaces and messages
  - Effort required to define the initial interfaces and messages
    - It won't happen overnight or in a day
    - Allow them to evolve during phase 1 development



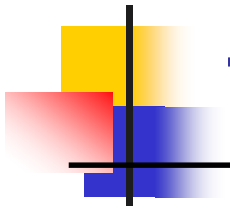
# Critical Architecture Issues

- For both fat and thin clients...
  - ...the overall application architecture is some form of distributed computing system.
  - ...the distribution of functionality should be transparent to the end-user.
  - ...information flow between the client and the server(s) for both fat and thin clients should occur only through messages.



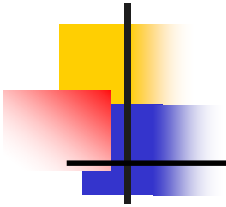
# Critical Architecture Issues

- No matter what it's called, we're talking about distributed computing and some form of service-oriented architecture
- Fat or thin, the server provides a service to the client
  - Keep this in mind as you craft the overall architecture for the application
- SOA demands well defined messages & interfaces
  - Used to talk about crafting the application API



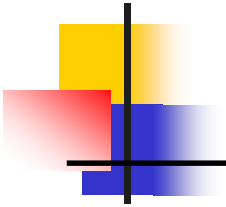
# Thin Client Checklist

- Development (partial list)
  - Language and API selection
  - Tools and library selection
- Services (partial list)
  - Connection logic
  - Message interfaces
  - Wrappers



# Thin Client Checklist

- Control (minimal)
  - Communications tools
  - System-level Monitors
  - Load-balancing (server-side)
  - Scheduling (mobile devices and server-side)
- Storage (full)
  - Persistency
  - Persistent queues and buffers
  - Replication (distributed?)



# Summary

- Thin doesn't mean Spartan or browser only
  - Think rich-thin client
- Review the DOs and DON'Ts
  - Actually do more than "review"
- SOA:  get used to it!
- Get (and keep) users involved in the development of the client (fat or thin)
  - User-driven or risk failure

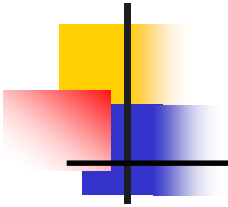
# Questions ? ? ?

If you don't ask, who will?

There aren't any dumb questions.

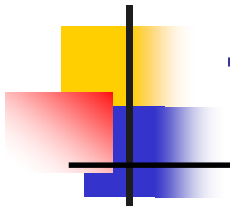
If not now, when?

The only dumb question is the one not asked!



# Readings

- *Patterns of Enterprise Application Architecture*, Martin Fowler, A&W, 2003
- *Design Patterns: Elements of Reusable Object-Oriented Software*, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, A&W, 1995
- *Design of Everyday Things*, Donald A. Norman, Basic Books, 2002 (there are older editions – originally: *Psychology of Everyday Things*)



# Thank You

For more information:

David Moskowitz

Productivity Solutions, Inc.

147 Ashland Avenue

Bala Cynwyd, PA 19004

+1-610-664-7726

davidm2@usa.net