

# Comparing Application Frameworks

Sean A Corfield

“An Architect's View”

<http://corfield.org/>  
[sean@corfield.org](mailto:sean@corfield.org)

# Goals

- Introduce you to three frameworks
- Use a sample application to show how frameworks help with (and shape) application structure
- Cover pros and cons of each framework
- Provide you with enough information to make a decision on which one to use for a given project

# Who Am I?

- Senior Architect for Macromedia IT (since mid-2000)
- A ColdFusion developer (since late-2001)
- A Mach II developer (since mid-2003)
- A Fusebox developer (since late-2004)
- A Model-Glue developer (since early-2005)
  
- An advocate of standards and best practices (since birth?)

# Agenda

- Frameworks: What? Why?
- Overviews of:
  - Fusebox 4
  - Mach II
  - Model-Glue
- Some Similarities
- Sample Application
- Some Differences
- Some Pros & Cons
- Conclusion

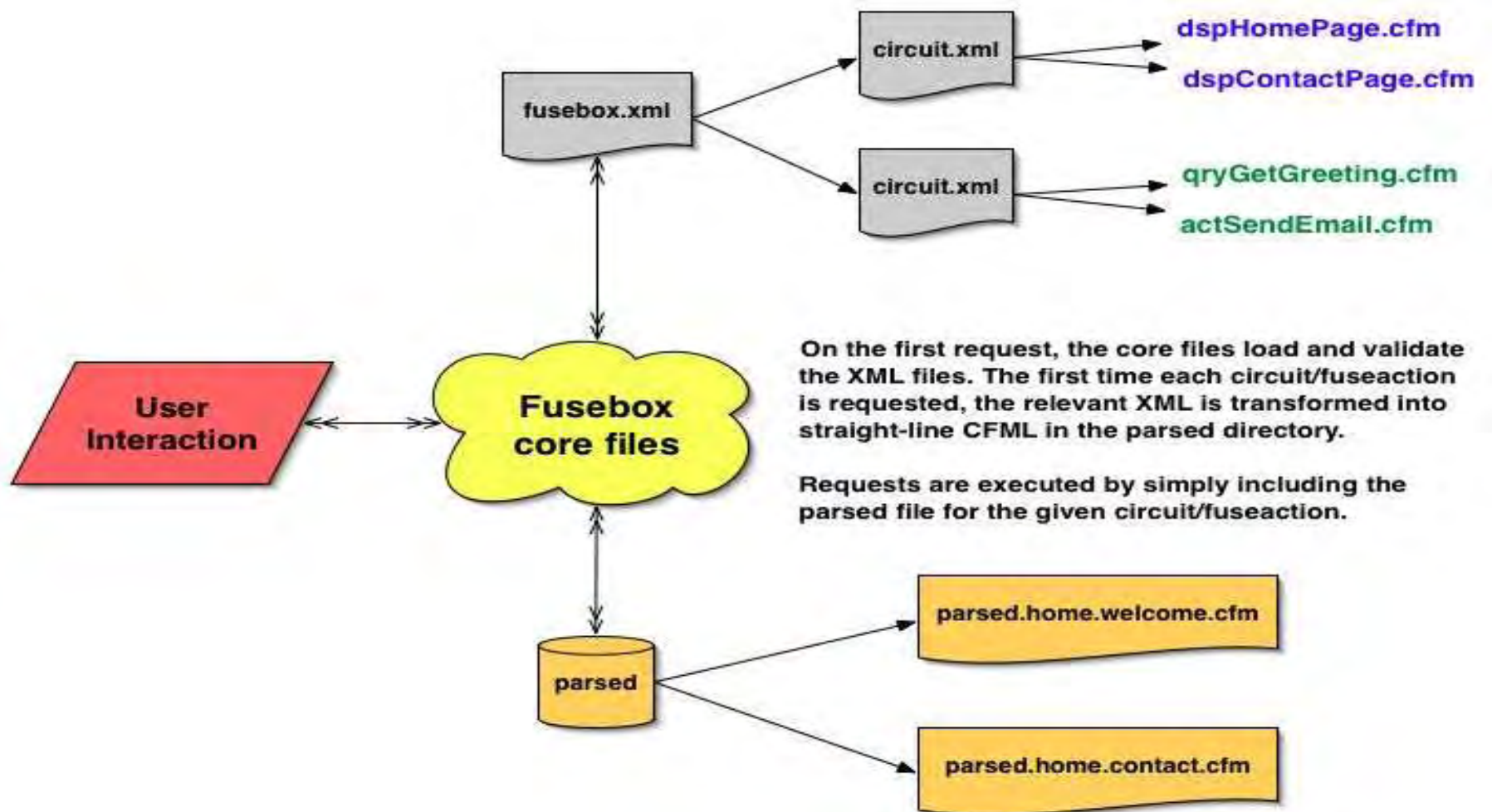
# Frameworks: What? Why?

- A framework...
  - ...is reusable code that provides the base on which to build applications in a standard way.
  - ...often implements an architecture which then shapes how the application is designed.
- A good framework...
  - ...saves development time – because it provides a lot of standard infrastructure out-of-the-box.
  - ...eases maintenance – because it provides a common structure to all applications.

# Overview of Fusebox 4

- Procedural core framework (4 .cfm + 2 UDF libs)
- Available for CF5, CFMX, BD, PHP
- Electrical metaphor of fusebox, circuits and fuses
  - `index.cfm?fuseaction=mycircuit.myfuseaction`
  - Routes to *myfuseaction* handler in circuit *mycircuit*
  - Each fuseaction handler contains XML verbs
    - `<do action="somecircuit.somefuseaction"/>`
    - `<include template="somefuse"/>`
- Translates XML to CFML (PHP) on first use

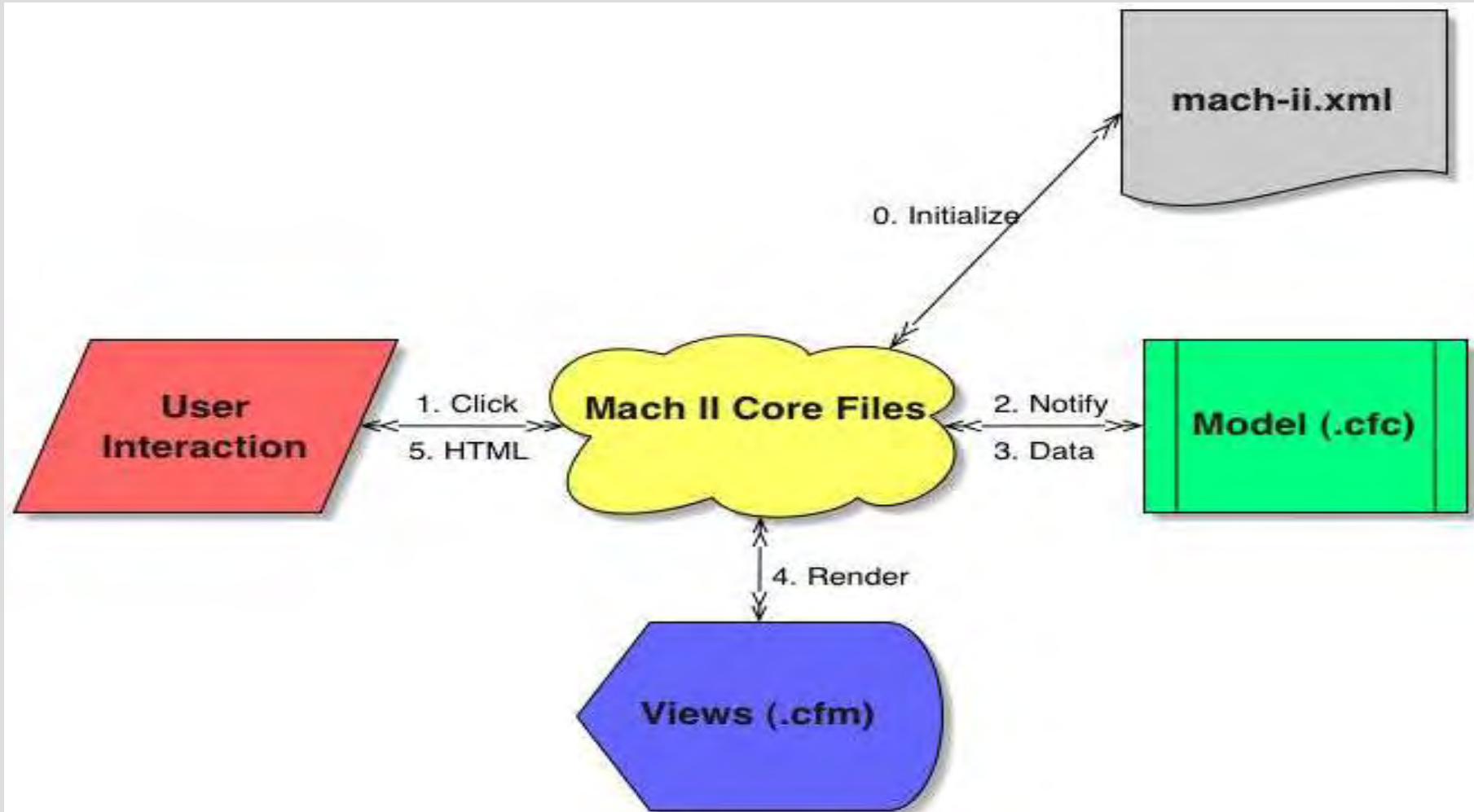
# Overview of Fusebox 4 Continued



# Overview of Mach II

- Object-oriented core framework (29 CFCs)
- Available for CFMX, also BD, limited PHP beta
- Event-based, implicit invocation architecture
  - `index.cfm?event=myevent`
  - Routes request to *myevent* event handler
  - Each event handler contains XML verbs
    - `<notify listener="somecfc" method="somemethod"/>`
    - `<announce event="anotherevent"/>`
- Dynamically processes the queue of events (semi-interpreted)

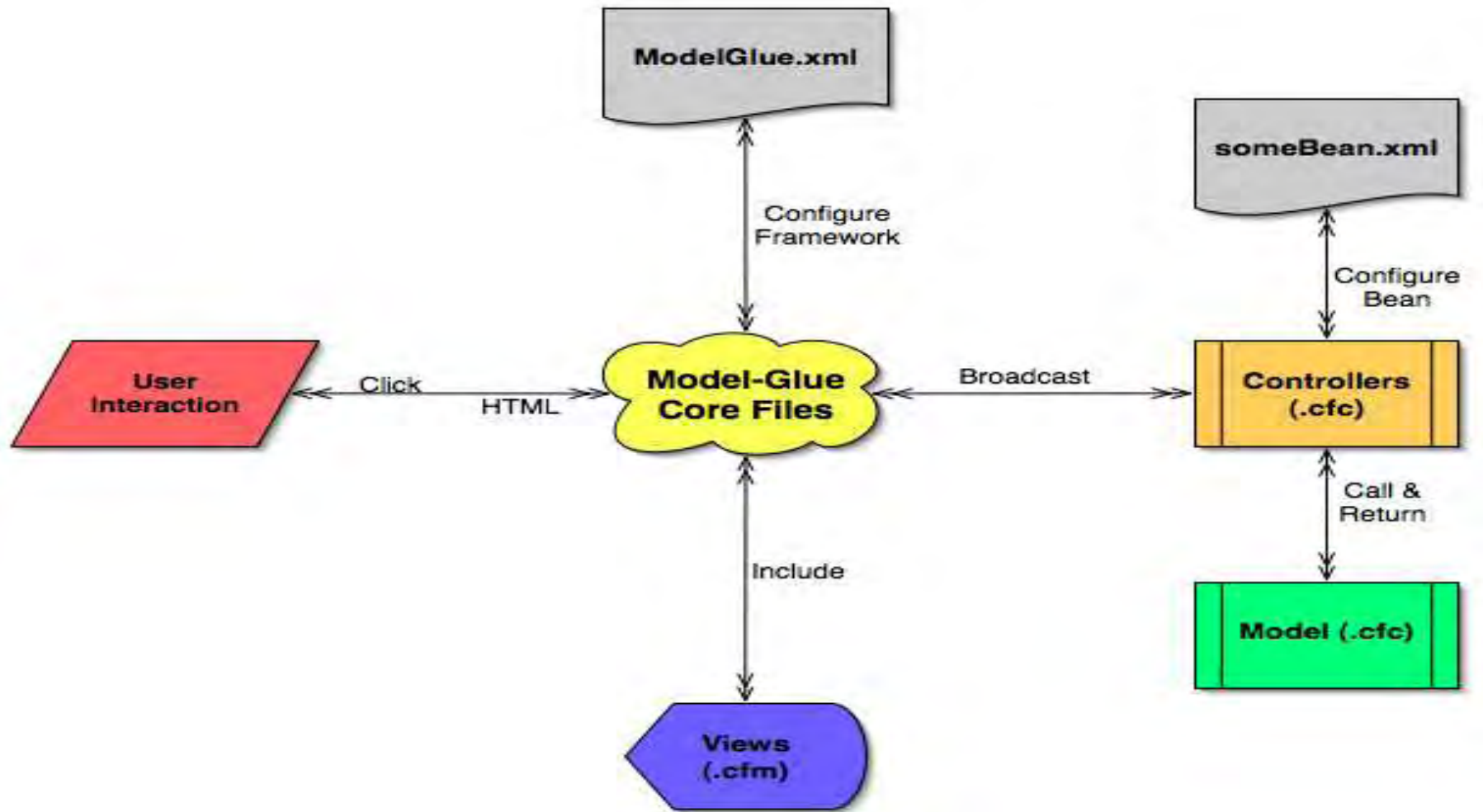
# Overview of Mach II Continued



# Overview of Model-Glue

- Object-oriented core framework (22 CFCs)
  - Includes a number of utilities
- Available for CFMX, also BD
- Event-based, implicit invocation architecture
  - `index.cfm?event=myevent`
  - Routes request to *myevent* event handler
  - Each event handler does one or more of:
    - Broadcasts messages
    - Renders views
    - Maps results to new events

# Overview of Model-Glue Continued



# Some Similarities

- Focused on separation of logic from presentation
- index.cfm & framework-as-controller
- XML-based configuration
- One-stop core files
- Public / private “handlers”
- Plugin architecture (Fusebox / Mach II)

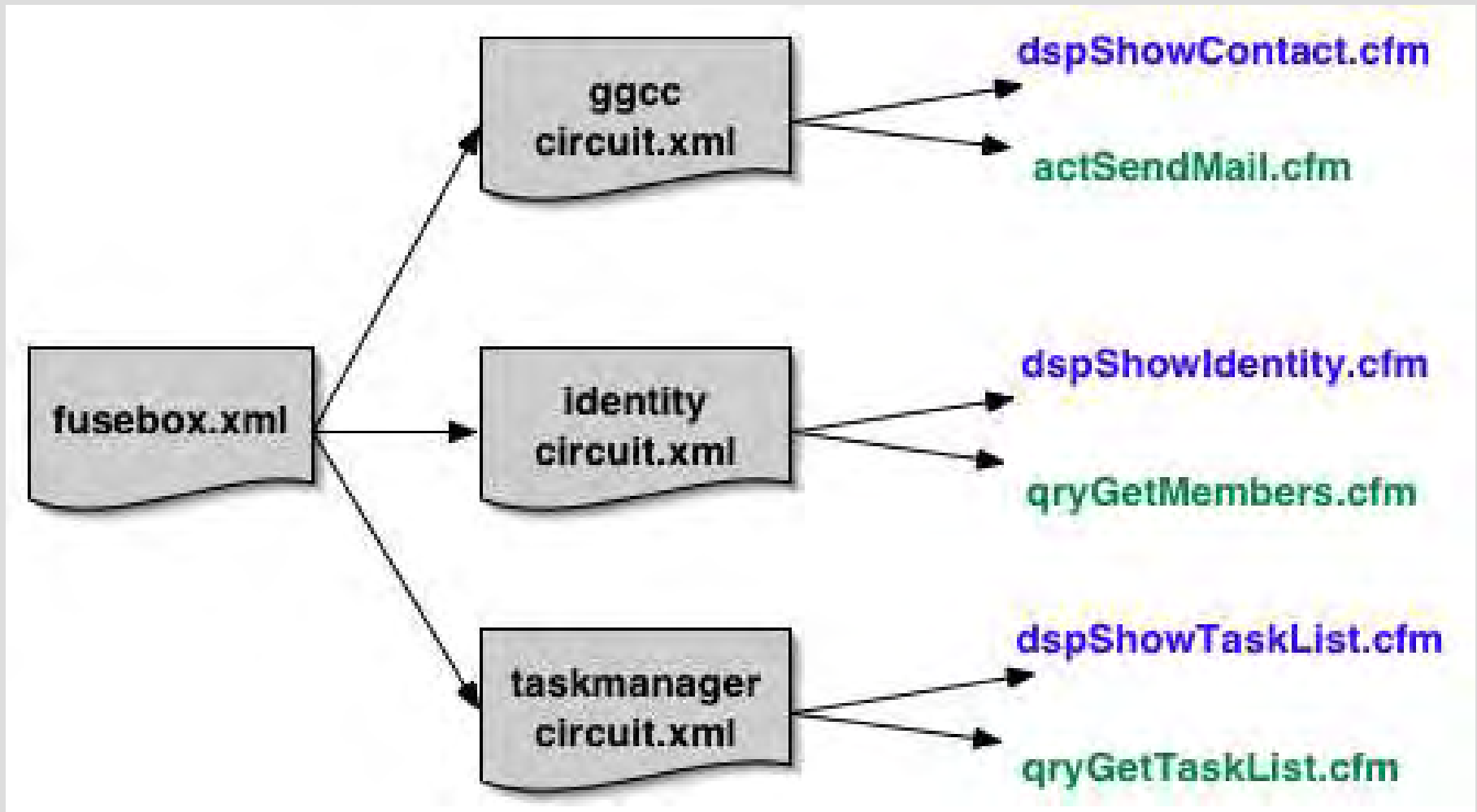
# Sample Application

- A task manager for a cat club
  - Simple user identification
  - List tasks
  - Add / edit tasks
  - Assign tasks to users
- Intended to be a demo not necessarily best practice!
- [demo]

# Old-School Fusebox

- Created with Adalon using FLiP (Fusebox Lifecycle Process)
- Design focuses on pages, exits (links / submit buttons)
- Circuits used for related functionality: user identity, task management, general site stuff
- Traditional fuse structure: **action**, **query**, **display**, **layout**.
- [look at source code]

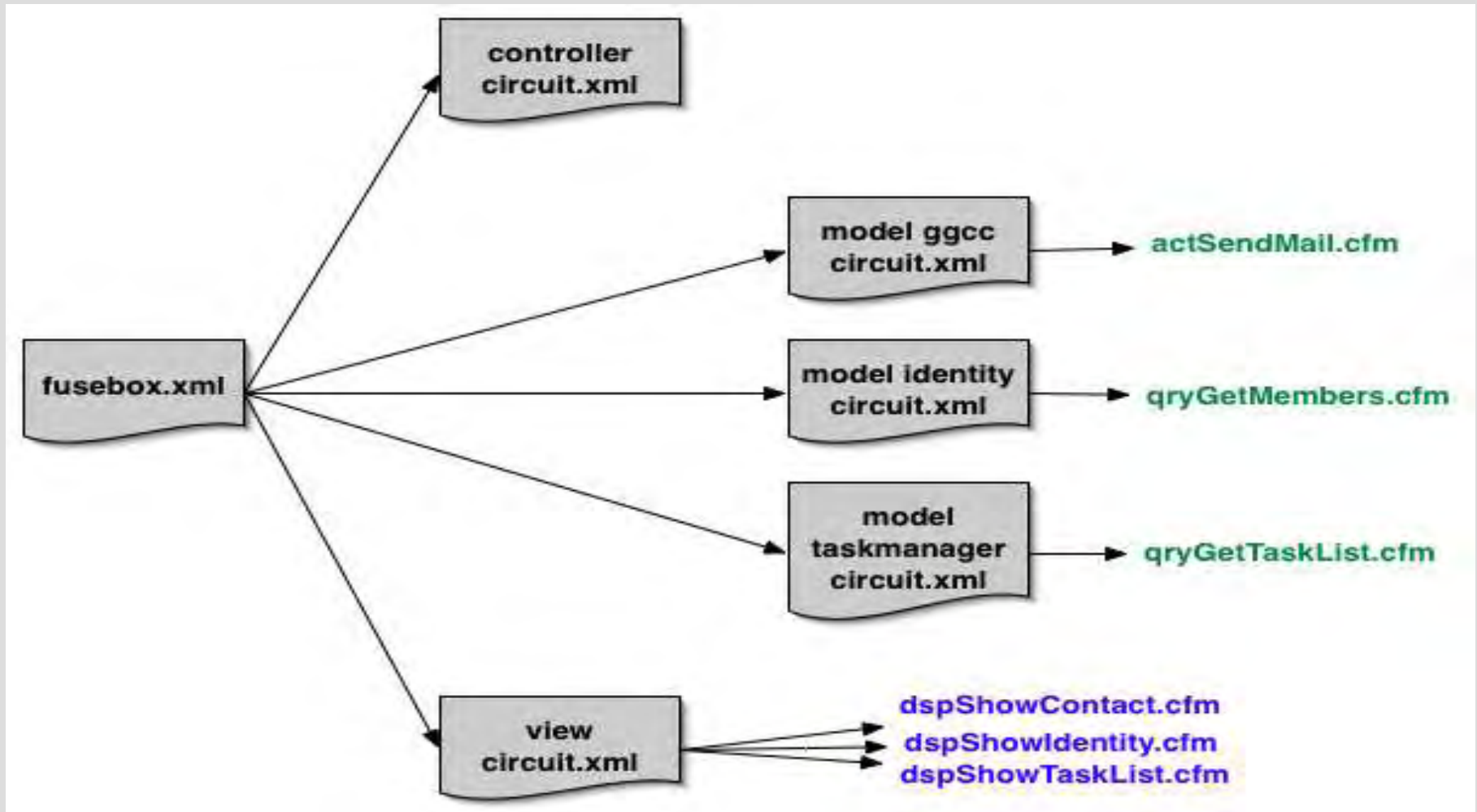
# Old-School Fusebox



# MVC Fusebox

- Separation of fuses into three primary circuits: **M**odel, **V**iew, **C**ontroller (each of which may have sub-circuits)
- Only Controller has public fuseactions (Model and View are all internal)
- Model contains all the **action** and **query** fuses, View contains the **display** and **layout** fuses
- [look at source code]

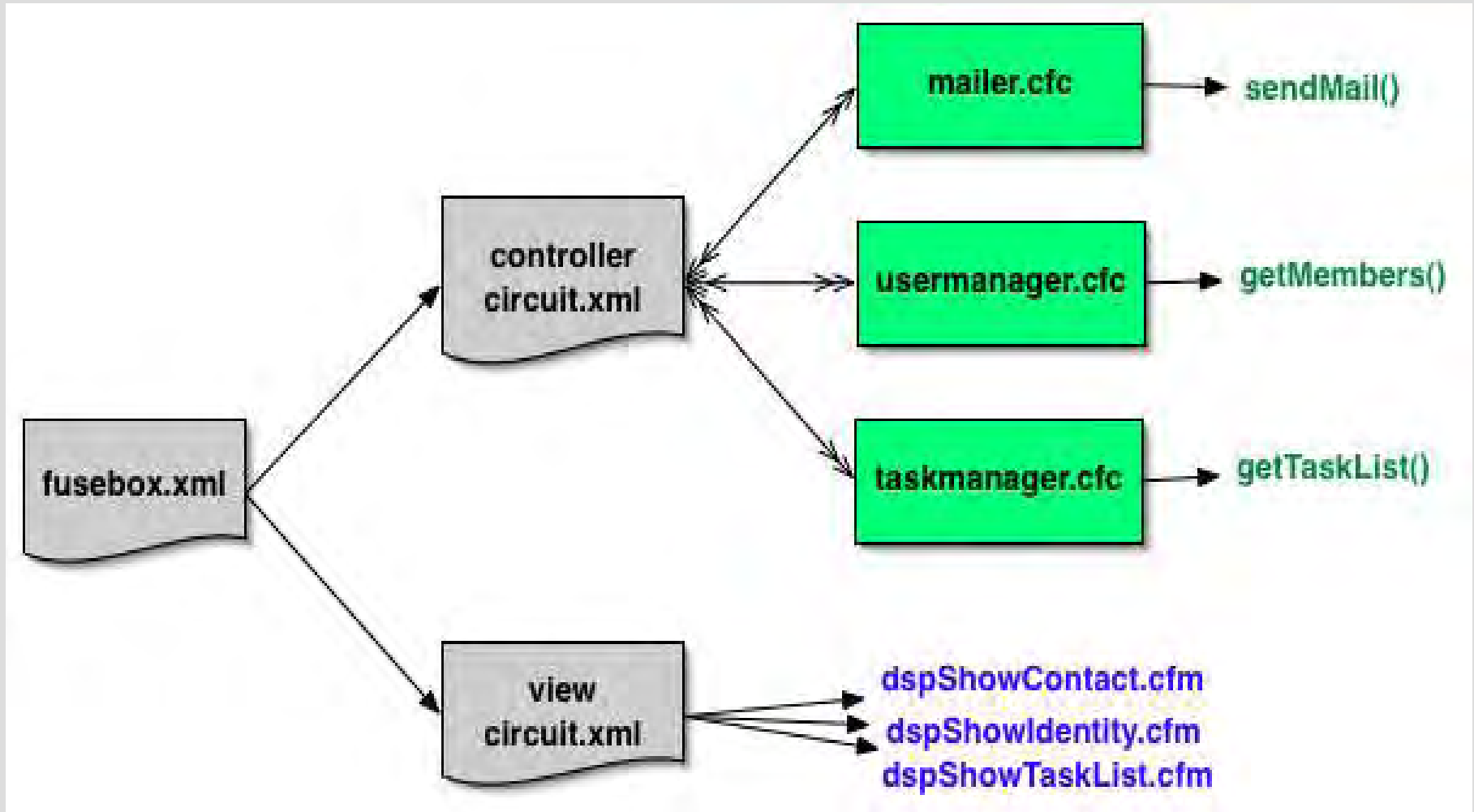
# MVC Fusebox



# OO Fusebox

- Model is replaced by CFCs
  - **action** and **query** fuses become one or more methods – each circuit might become one CFC
  - Controller circuit uses `<invoke>` verb to call methods
- Some fuses must become multiple methods or must return complex results
- Initialization of service CFCs happens in `fusebox.init.cfm` (for example)
- [look at source code]

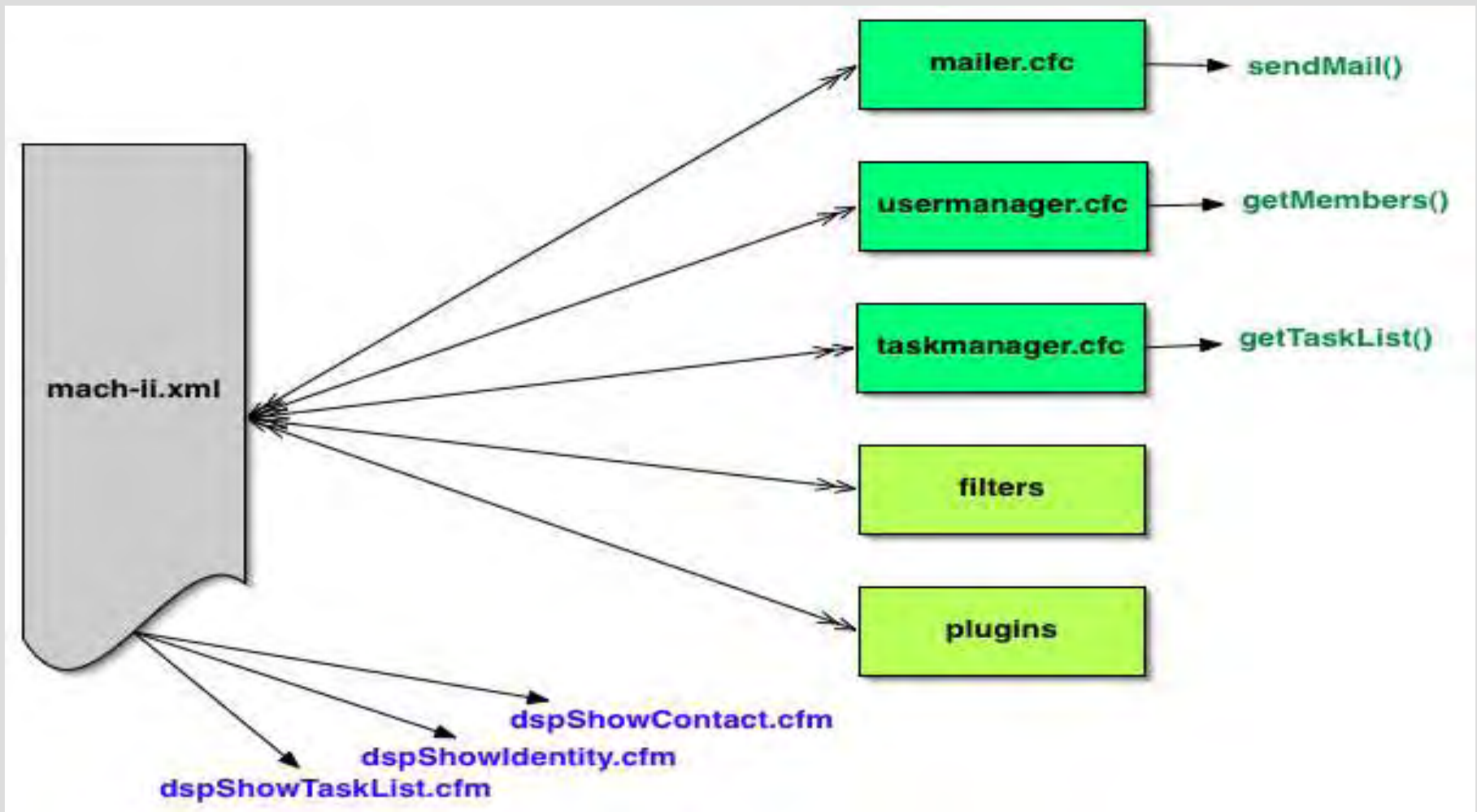
# OO Fusebox



# Mach II

- Model is CFCs (very similar to OO Fusebox)
  - Best practice would split CFCs into listeners, business objects, data access objects etc
- Controller (mach-ii.xml) uses <notify> verb to call methods
- Controller also invokes views directly (there's only one XML file)
- Initialization of service CFCs is handled automatically by the framework
- [look at source code]

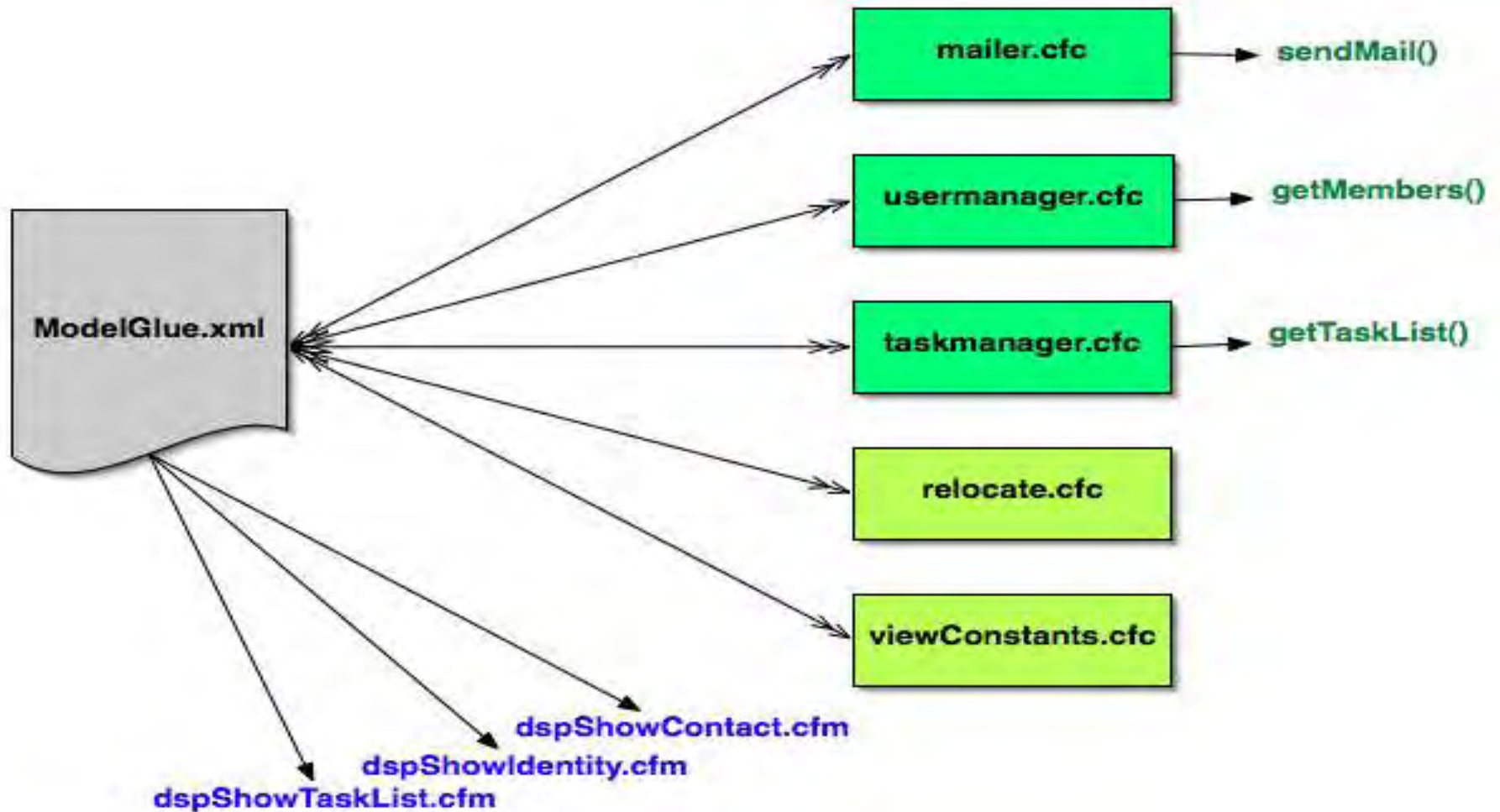
# Mach II



# Model-Glue

- Model is CFCs (very similar to Mach II)
  - Best practice would separate controllers from model
- Each event handler (ModelGlue.xml):
  - Broadcasts messages
  - Framework invokes registered “listener” methods on controllers (CFCs)
  - Includes views
  - Maps result states to new events
- Initialization of service CFCs is handled automatically by the framework (bean factory)
- [look at source code]

# Model-Glue



# Some Small Differences

- **Fusebox**
  - Per-circuit XML files
  - CFCs and MVC are optional in Fusebox
  - Circuits allow for modular applications
  - Basic permissions model built in
- **Mach II / Model-Glue**
  - Single XML file
  - Requires CFCs and enforces MVC
  - Monolithic applications (but can share session)
  - Roll your own permissions/security

# Some Big Differences

- **Fusebox/Model-Glue**
- **Static event handling**
  - circuit.xml/ModelGlue.xml is the entire logic path
  - All execution paths are spelled out
  - Dynamic events mean redirects
- **Fusebox**
- **Explicit invocation**
- **Mach II**
- **Dynamic event handling**
  - mach-ii.xml has no logic paths
  - Execution paths are implied by code
  - Dynamic events happen internally
- **Mach II/Model-Glue**
- **Implicit invocation**

# Some Fusebox Pros & Cons

- **Pros**

- Easier to learn
- Supports multiple programming styles
- Fine-grained control over framework behavior
- Better modularity
- XML grammar is expressive & extensible

- **Cons**

- Still easy to write spaghetti code
- Lots of framework options to learn
- Procedural code isn't "cool"

# Some Mach II Pros & Cons

- **Pros**

- Enforces / supports best practices more fully
- Fairly simple, consistent framework structure
- CFCs are building blocks that extend the framework (filters / listeners)

- **Cons**

- OO, MVC, CFCs & **implicit event queue** have a steep learning curve
- Monolithic XML file can be hard to manage
- Less granular control of framework
- Views require XML declarations

# Some Model-Glue Pros & Cons

- **Pros**

- Enforces / supports best practices more fully
- Very simple, consistent framework structure
- CFCs are building blocks that extend the framework (controllers)

- **Cons**

- OO, MVC and CFCs have a steep learning curve
- Monolithic XML file can be hard to manage
- Less granular control of framework

# Other Factors

- All support an OO style but Mach II & Model-Glue require it – your comfort level with OO design may influence your choice
- Fusebox has great tool support: Eclipse plugin, Dreamweaver, FuseBuilder, Adalon and others
- Framework architecture complexity:
  - Fusebox <-----> Model-Glue <-----> Mach II

# Other Factors Continued

- Documentation is skimpy for all three (but several books exist for Fusebox)
- User community is strong for all three (larger and better established for Fusebox – because Fusebox has been around longer)
- All sets of core files are essentially the product of one person but all three frameworks are opening up somewhat

# Other Frameworks

- Other HTML framework choices:
  - onTap – Isaac Dealey – <http://fusiontap.com/>
    - Procedural, implicit invocation, massive HTML library
  - Reaction – Murat Demirci
    - Page Controller, ASP.NET-style code behind concept
- There are also some model-only frameworks
  - Tartan – Paul Kenney – <http://tartanframework.org/>
    - Service / Command / Data Access / Value Object, OO, works well with Fusebox, Mach II and Model-Glue
  - ColdSpring – Dave Ross – <http://cfopen.org/projects/coldspring/>
    - Inversion of Control, CFC Container (like Java's Spring)

# Conclusion?

- You tell me...
- ...after seeing this presentation,
  - Who would use Fusebox?
  - Who would use Mach II?
  - Who would use Model-Glue?
  - Who would make a choice on a project-by-project basis?

# What Do I Use?

- I helped introduce Mach II to Macromedia so I use Mach II for projects at work – it's a Web Team standard
- I'm a contributor to Mach II (lead developer for 1.0.10)
- I recently converted a key Mach II application at Macromedia to use Model-Glue instead
- I use Fusebox 4.1 for all my projects outside work
- I'm a member of Team Fusebox

# Conclusion

- If you have a very complex application – that has a lot of internal state transitions which can occur dynamically – then Mach II may be more appropriate
- If you want a simpler OO framework, Model-Glue is a very good choice
- Otherwise Fusebox is easier to learn, supports more programming styles (including OO) and has a stronger support community with several books available

# Resources

- Fusebox <http://fusebox.org/>
- Mach II <http://mach-ii.com/>
- Model-Glue <http://model-glue.com/>
- Also <http://corfield.org/fusebox/>  
<http://corfield.org/machii/>
- Fusebox tools:
  - Fusebox Plugin for Eclipse  
<http://cfopen.org/projects/fusebox3cfe/>
  - Fusebox Explorer for Dreamweaver  
<http://cfopen.org/projects/fuseboxexplorer/>
  - FuseBuilder <http://fusebuilder.net/>
  - Adalon <http://adalon.net/>

# Frameworks: Questions and Answers?

Sean A Corfield  
“An Architect's View”  
<http://corfield.org/>  
[sean@corfield.org](mailto:sean@corfield.org)