Software Structure

Jim Fawcett

Software Modeling

Introduction

What Is Software Structure?

- Partitions—classes, packages, systems
  - Separation of concerns
- Communication
  - How do the parts make requests and send notifications?
- Sharing
  - How is data shared between the parts?
- Control
  - Which parts interact with which other parts?

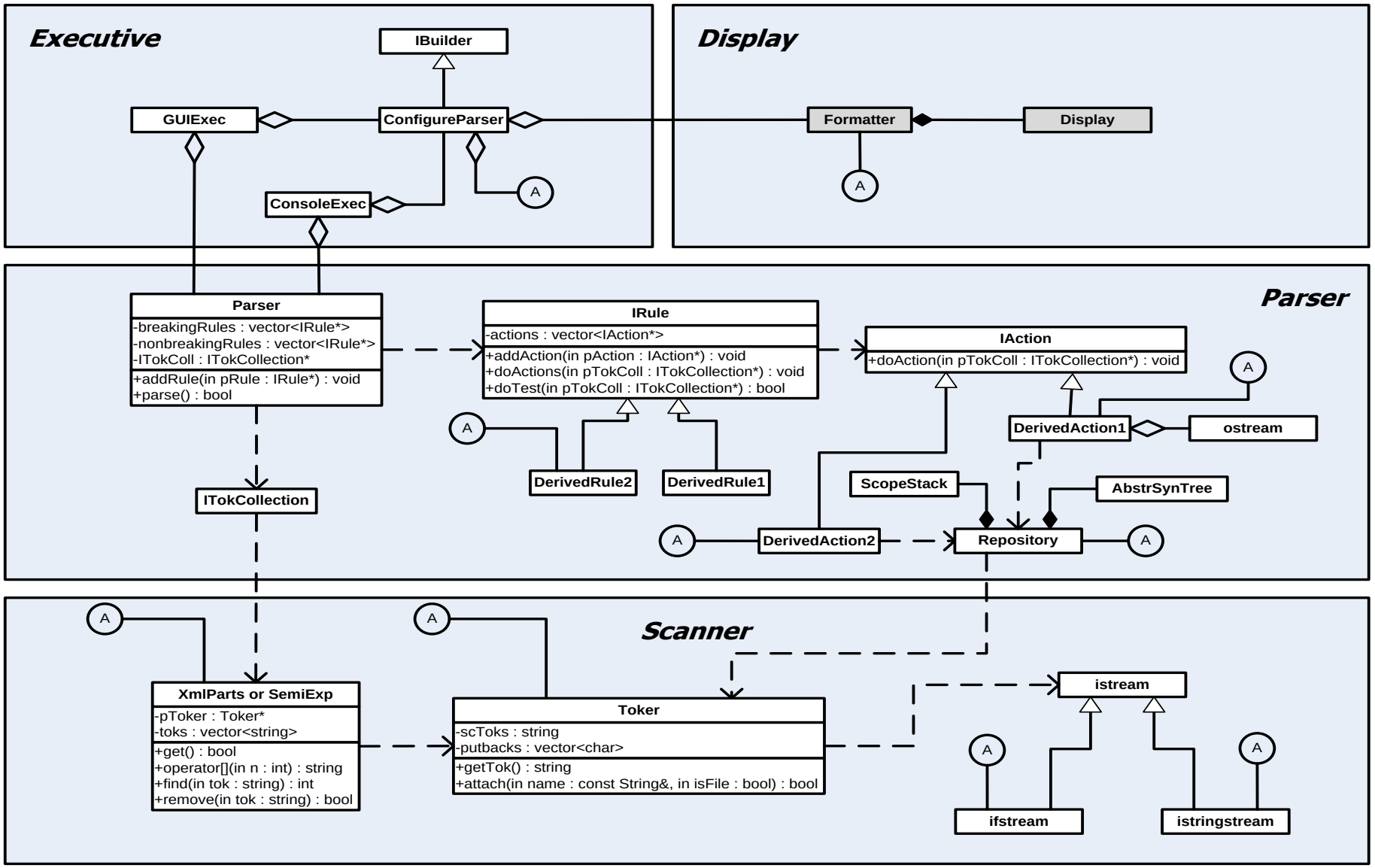
Program Structure

- Logical—class structure:
  - Interfaces, classes, and class relationships
- Package—code file structure:
  - Package dependency tree, as shown in package diagrams
  - Subsystems, e.g., collection of packages separated by interfaces with each focused on specialized processing
    - For a radar those might be signal processing, beam forming, data management, operator control, communication.
- Execution—binary structure:
  - Monolithic program, e.g., an exe
  - Program with loadable Dynamic Link Libraries (DLLs)
  - Cooperating processes, e.g., client-server, server federation.

Code Analyzer Example

- The next slide shows the logical structure of a code analyzer, focusing on the front-end analysis.
- There are four modules
  - Lexical scanner—reads token groups from stream
  - Parser with rules and actions—builds AST
  - Executive with builder—assembles all the parts
  - Display—maps AST data into information
- You will find more discussion in the Parser Blog

Parsing Facility



Software Structure Contents

- Data driven
  - Client server
  - Three tier
  - Model-View-Controller
- Layered structure driven
  - Components
  - Services
- Analysis driven
  - One pass
  - Two passes
- Communication driven
  - Client server
  - Peer-to-peer
  - Middleware
- Thread and event driven
  - Single Threaded Apartment (STA)
  - Parallel execution
  - Pipeline execution
- Enterprise computing
  - Federated systems

Data-Driven Structures

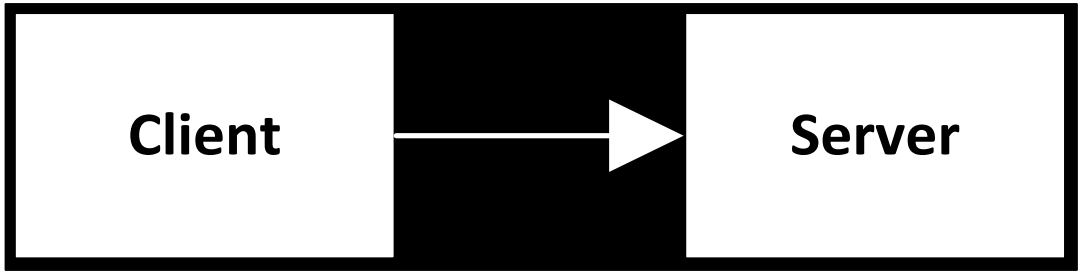
Data-Driven Structures

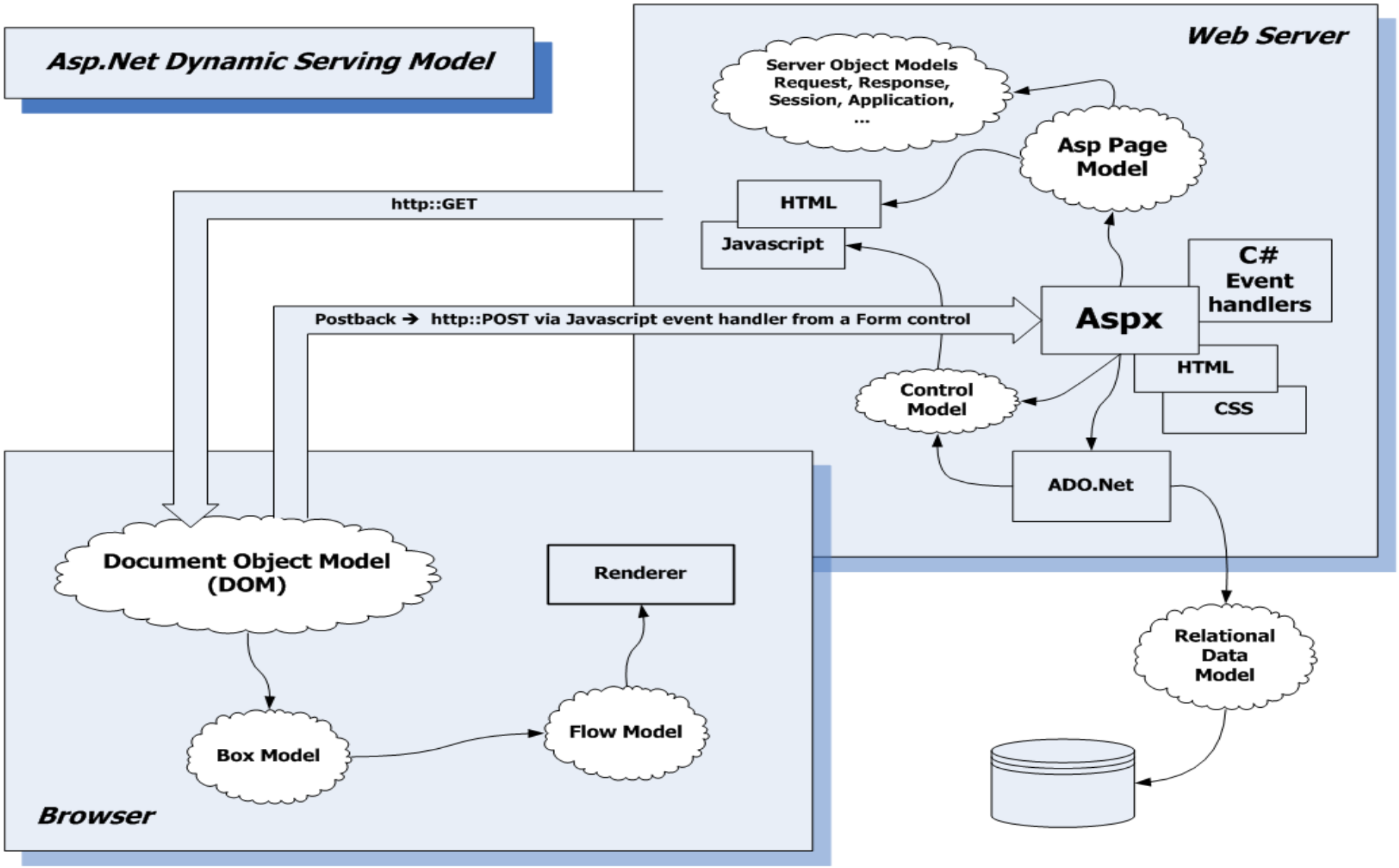
- Some program structures are driven by the presentation and management of data:
  - Client-server
  - Three-tier
  - Model-view-controller

Structure: Client-Server

- Behavior:
  - Server is passive, waits for client requests.
  - Server contains data shared among its clients.
  - Server handles multiple concurrent clients.
  - Without additional structure system may become tightly coupled and difficult to change.

- Example:
  - Web server and browser clients




Asp.Net Dynamic Serving Model
Web Server
Server Object Models
Request, Response,
Session, Application,
...
Asp Page
Model
HTML
Javascript
http::GET
C#
Event
handlers
Aspx
Postback → http::POST via Javascript event handler from a Form control
HTML
CSS
Control
Model
ADO.Net
Document Object Model
(DOM)
Renderer
Relational
Data
Model
Box Model
Flow Model
Browser

Sharing Data

- Relational databases—SQL Server, mySql, …
  - ACID—Atomicity, Consistency, Isolation, Durability
  - ACID => transactional
- NoSQL databases—MongoDB, CouchDB
  - Key-value, document, hierarchal
  - Very flexible data structure
  - Consistency is pushed onto the application
- File systems
- Ad hoc in-memory repositories
- Extensible record stores—Google's Big Table
  - Distributed partitioned tables
- Document stores—CouchDB
  - Multi-indexed objects aggregated into domains

Separation of Concerns

- Except for the simplest of applications it's not a good idea to bind presentation, control, and data together.
  - There often are many views, more than one application mode, many sources of data.
  - If we bind these all together, we get spaghetti code.
    - Very hard to test, hard to maintain, hard to document

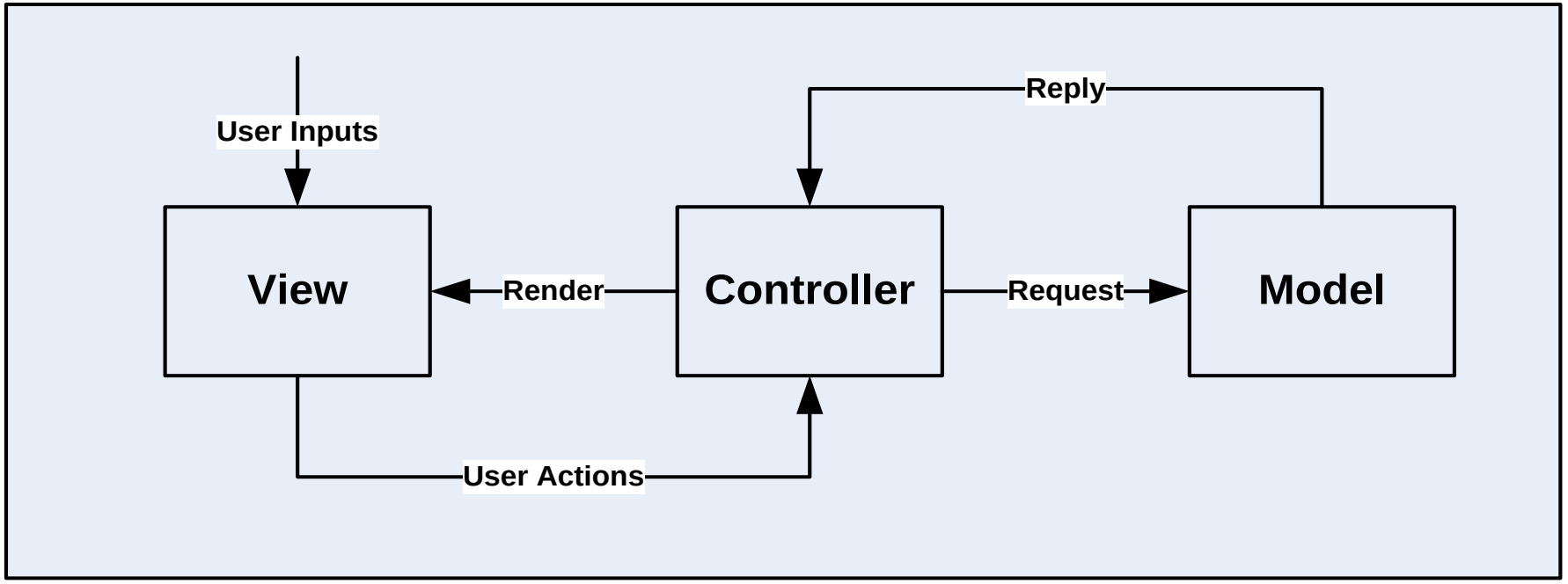
Structure: Three Tier

- Structure:
  - Partitioned into presentation, application logic, and data management.
  - Intent is to loosely couple these three aspects of an application to make it resilient to change.
- Examples:
  - Most well-designed applications

Model-View-Controller

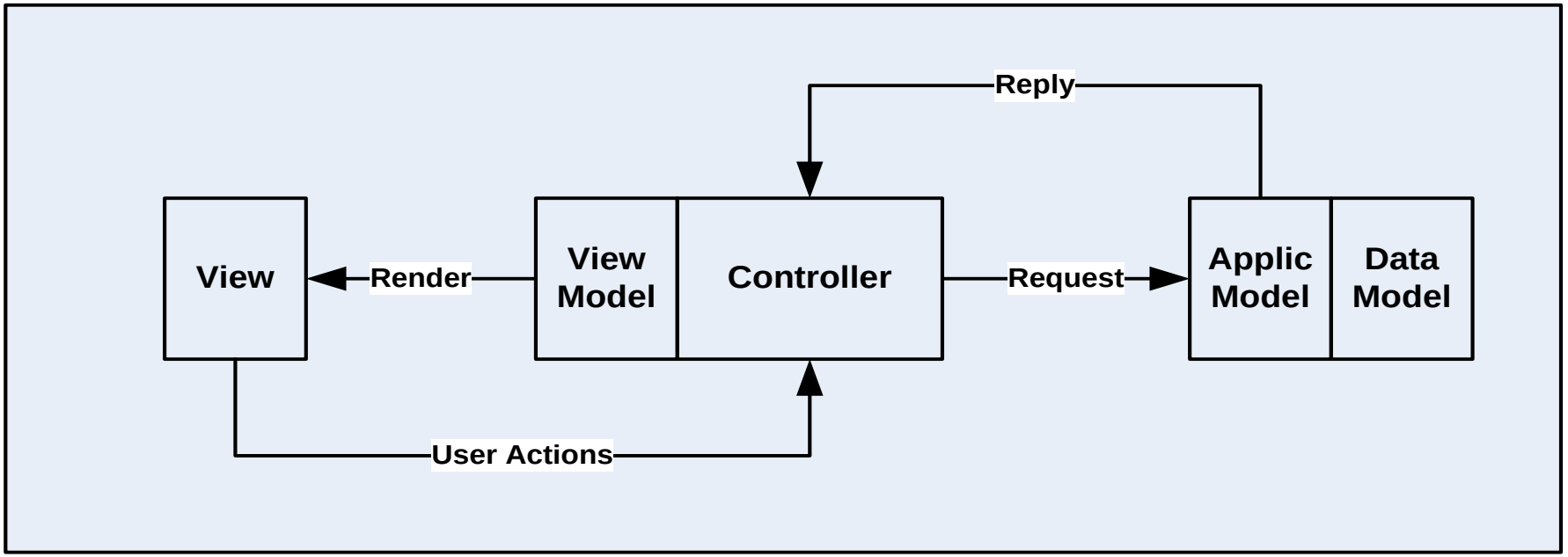
- Structure:
  - MVC is a refined version of the three-tier structure, intended to support multiple views and data models.
  - Models do all data storage management.
  - Views present information to user, format output, but do no other transformations on data.
  - Controllers accept inputs, implement application processing, and use models and views to provide the application's behavior.
  - Application phases often have one controller each.
  - Models may be shared between controllers.
- Example: Asp.Net MVC

Basic MVC Structure



MVC—With Views and Application Models

- Views and models often have some substructure, e.g.:



N-Tier Structure

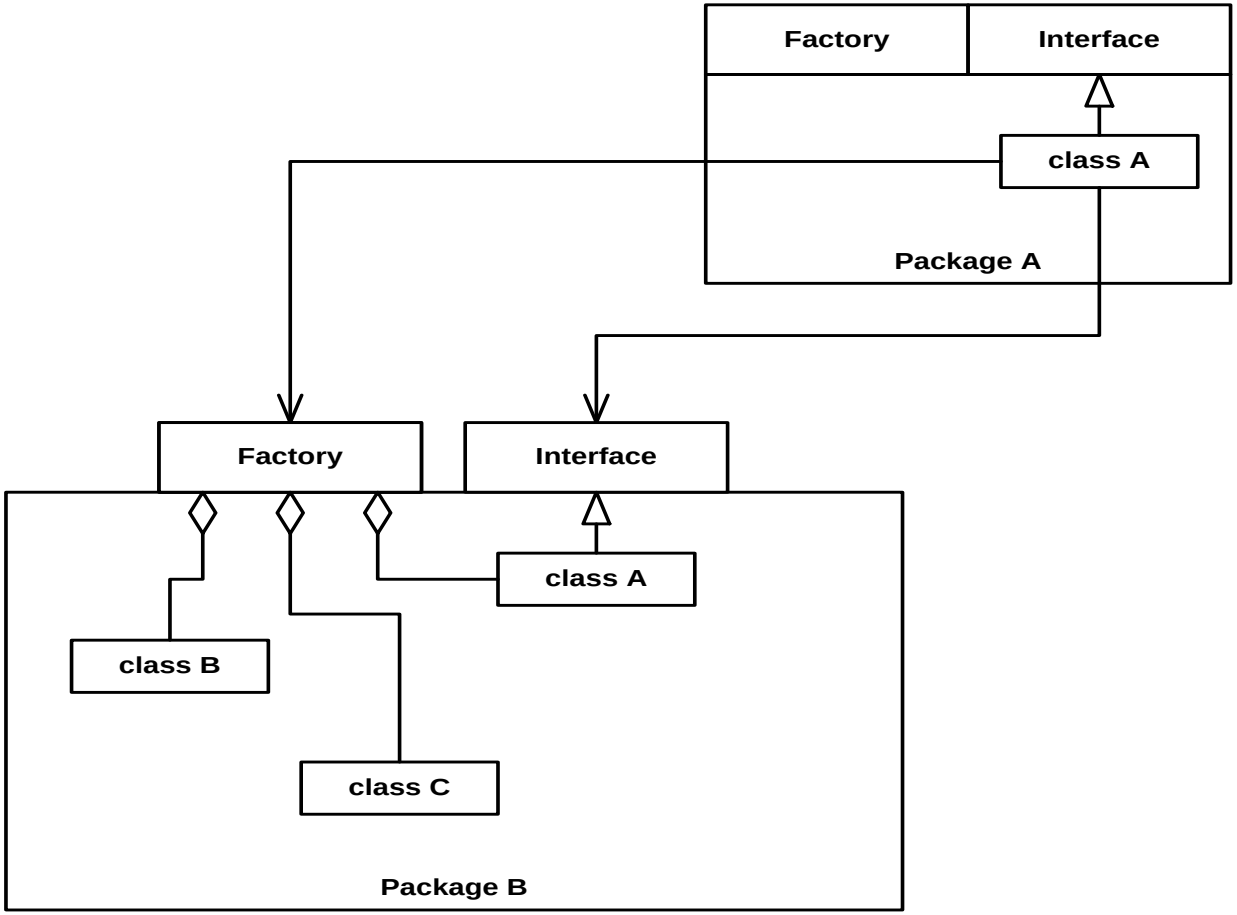
- So, the three-tier MVC has morphed into a five-tier V-VM-C-AM-DM
  - View—what gets rendered
  - View model—an abstraction of the view
  - Controller—routes View events to handlers in the Application model
  - Application model—classes that model the “business” logic
  - Data model—models data storage tables
    - Database, XML file, custom data structures

Layer-Driven Structures

Component-Layered Structures

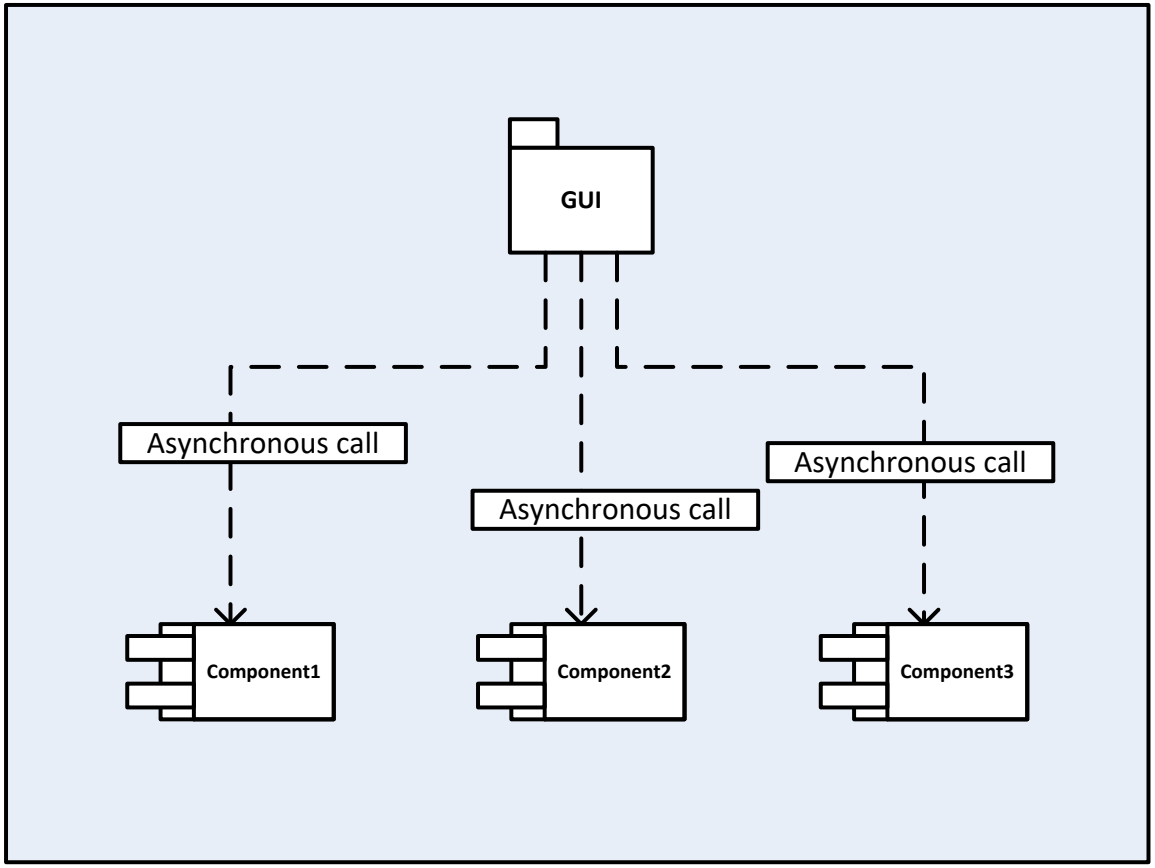
- Structure:
  - A componentized system is composed of an application with many pluggable component parts.
  - A component is pluggable if it implements a plug-in interface, published by the application, provides an object factory for activating its internal objects, and is packaged as a Dynamic Link Library (DLL).
- Example:
  - http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Parser/ almost implements

Hiding Implementation Details



Example Componentized System

Separate presentation from application logic

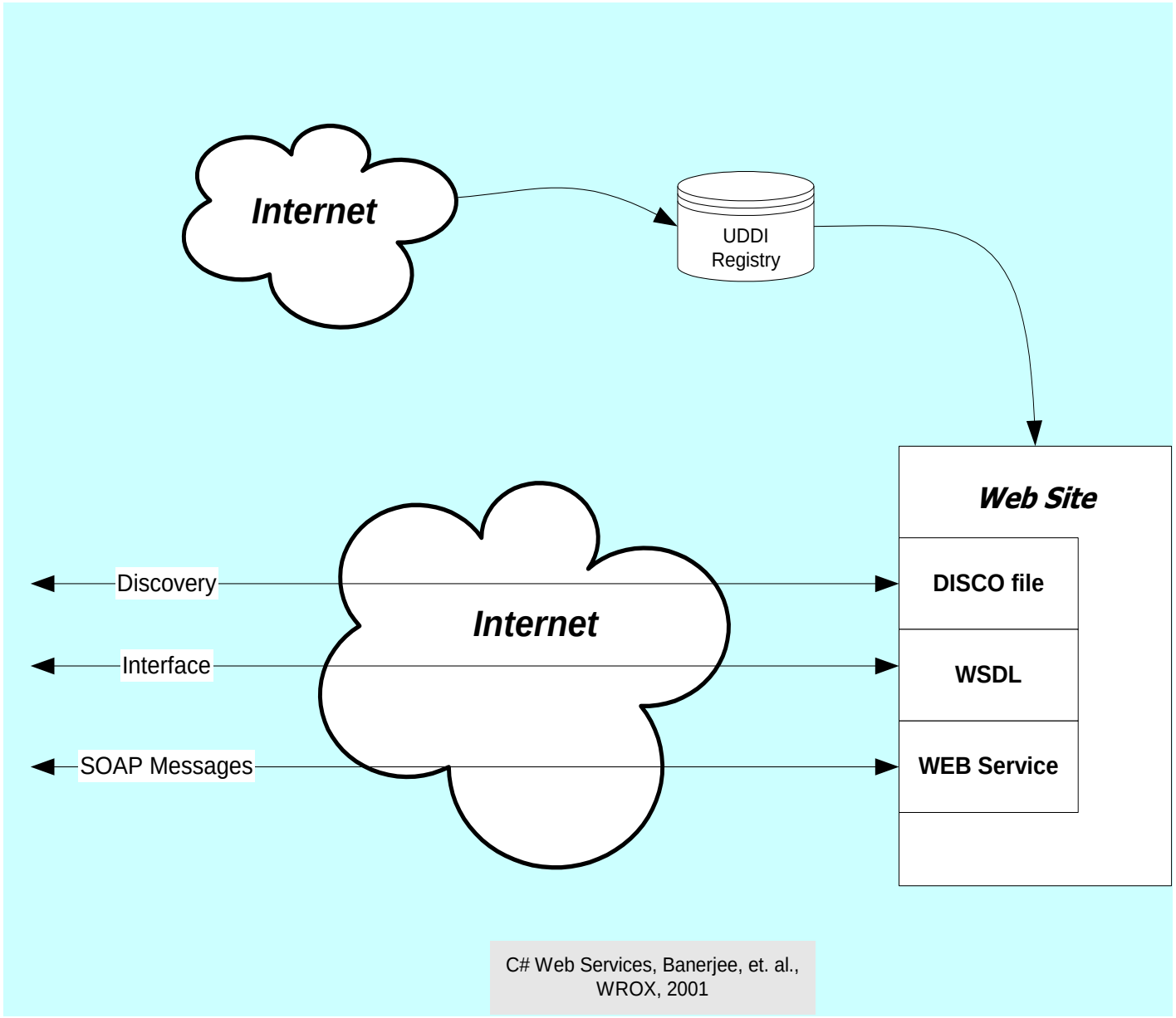


Service Layered Structure

- Provides a structure based on:
  - System services—things the user doesn't think about
    - Communication, storage, security, file caching, …
  - User services—things the user manipulates as part of the use of the system
    - Input, Display, Check-in/Check-out, …
  - Ancillary—things that are not part of the system mission but are necessary
    - Logging, extension hooks, test hooks, …

Distributed Services

- Structure:
  - Service-oriented systems are simply client server.
  - Usually the server is implemented with a web service or operating system service.
    - Web service is a web application that provides an interface for client software to access.
    - OS service is a system application that provides an interface for requests and an administration interface for setting service startup and shutdown policies.
  - Windows Communication Foundation (WCF) has extended that model to support hosting in:
    - Desktop application
    - Windows service hosted with Windows Service Control Manager (SCM)
    - Web service hosted by Internet Information Server (IIS).



C# Web Services, Banerjee, et. al., WROX, 2001

WCF Protocols

- WCF supports:
  - Http—SOAP over Http in clear text—BasicHttp
  - Http—SOAP with security extensions—WsHttp
  - NetTcp, SOAP over TCP
- SOAP—Simple Object Access Protocol
  - An XML body for HTTP or TCP messages
  - Usually contains a message body in XML defined by a data contract
- WCF is a very flexible, relatively easy to use, but heavyweight communication mechanism

REpresentational State Transfer

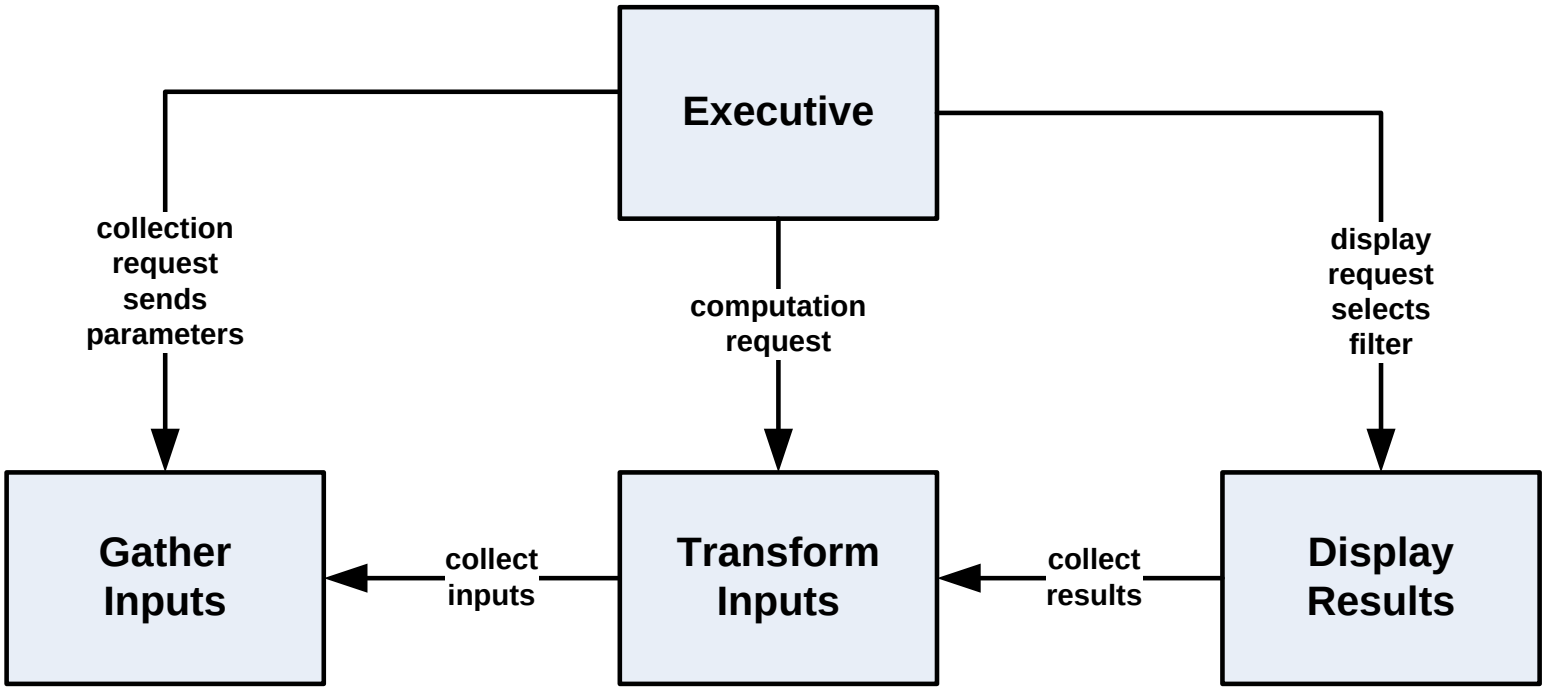
- REST is a message-passing communication system built on the HTTP protocol, using the web verbs:
  - Get—retrieve a resource without changing the state of the server
  - Post—send information to the server that may change its state
  - Put—place a resource on the server
  - Delete—remove a resource from the server
- Its encoding is UTF text, not SOAP or some other complex messaging format, but may use encryption, as in HTTPS.

Analysis-Driven Structure

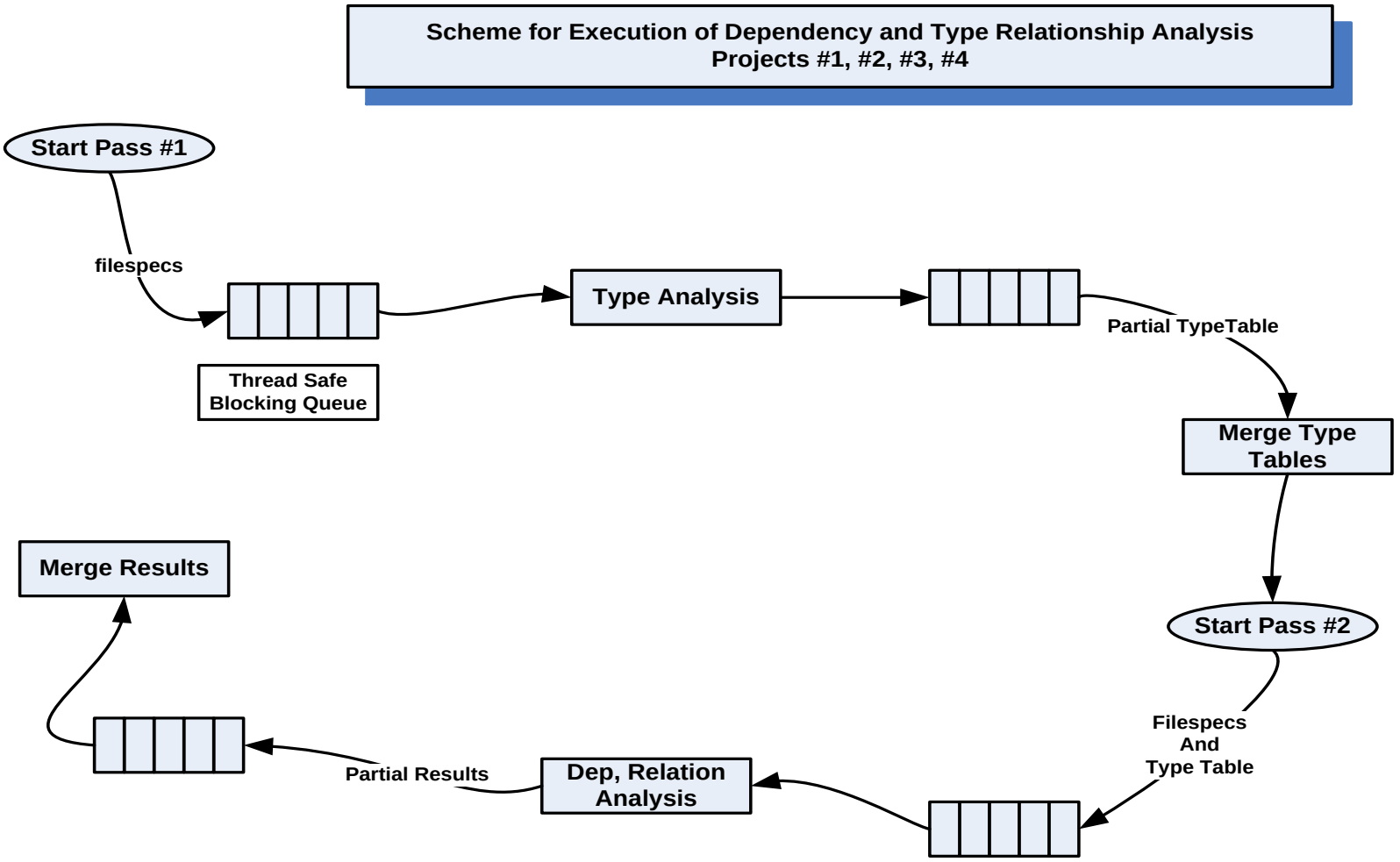
Analysis-Driven Structure

- Packages
  - Gather working set (inputs needed for analysis)
  - Execute one or more phases of analysis
  - Filter and interpret resulting data to provide information
  - Present the analysis information

Package Structure—Analysis Driven



Pipelined Dependency Analysis



Communication-Driven Structure

Communication-Driven Structure

- When users, data, and application logic are distributed across processes and machines communication becomes important:
  - Client-server
  - Peer-to-peer
  - Communication middleware
    - RPC (RMI)
    - Message-Passing

Performance

- Suppose that processing a request takes T units of time if requester and provider are in the same process.
- Executing the same request across processes takes about 10 T units of time.
- Executing the same request across a network takes about 100 T units of time.
- Executing the same request across the Internet takes about 1,000 T units of time.

Structure: Client-Server

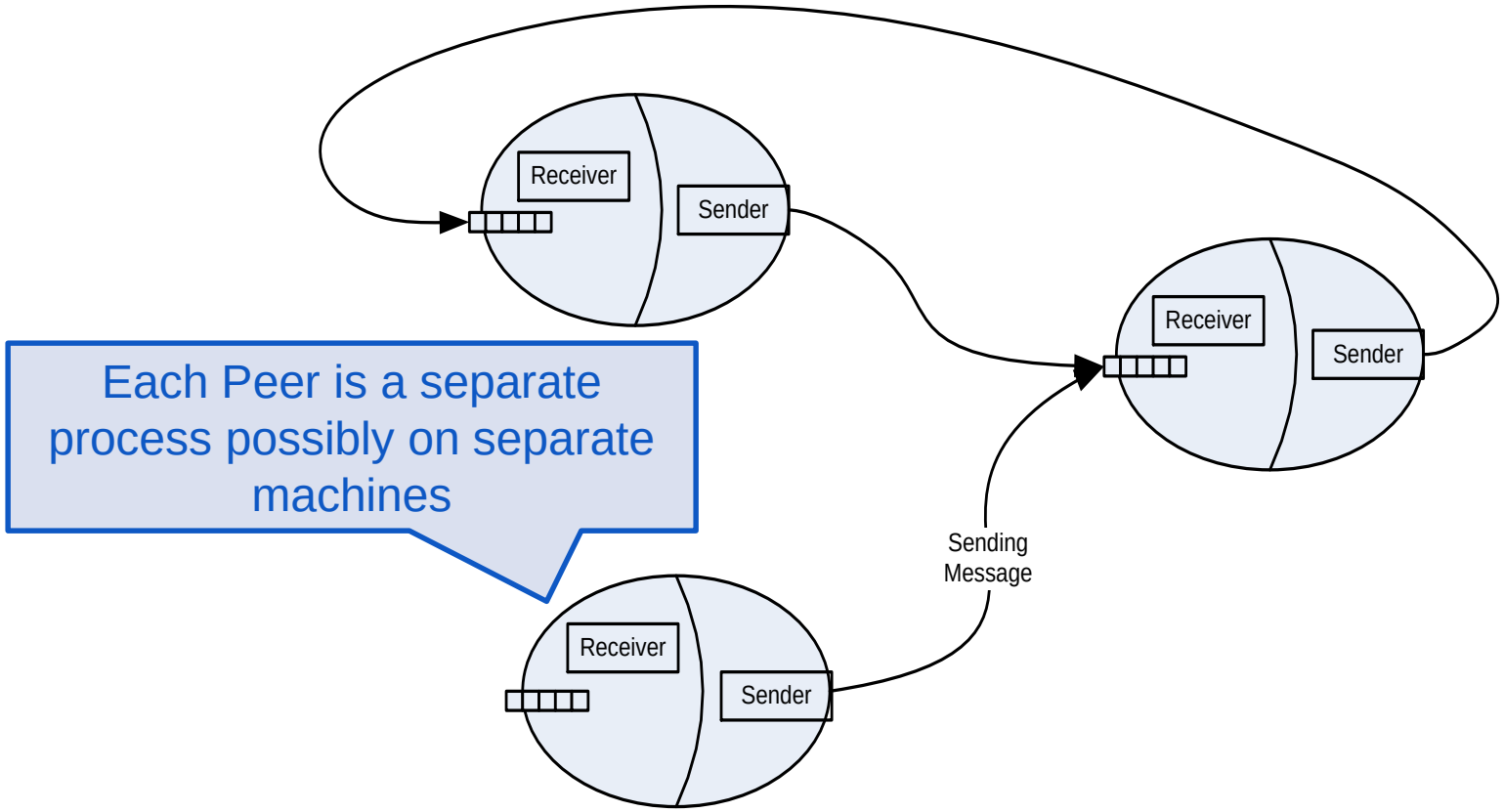
- Behavior:
  - Server is passive, waits for client requests
  - Server handles multiple concurrent clients
  - Without additional structure system may become tightly coupled and difficult to change
- Example:
  - Web server and browser clients

Structure: Peer-to-Peer

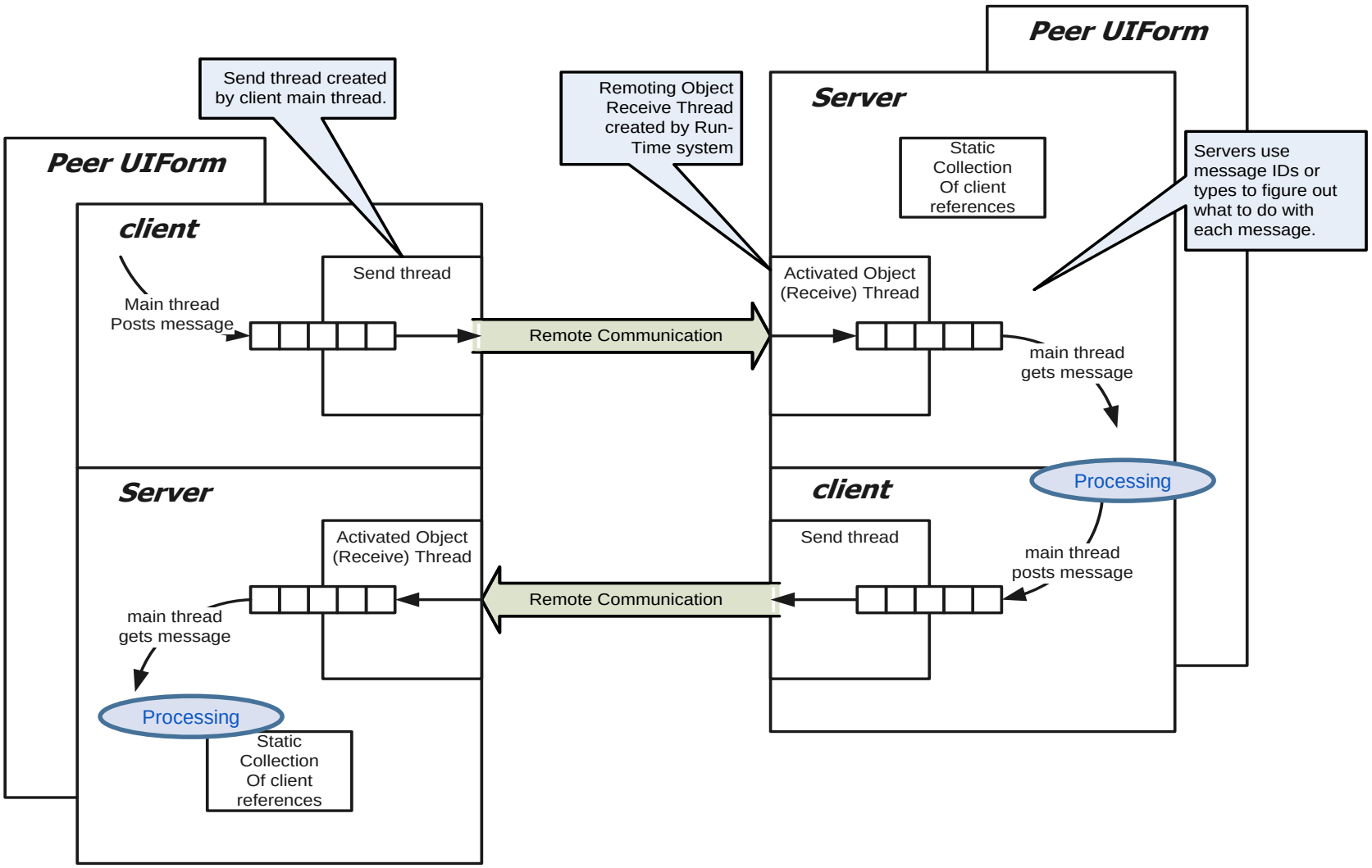
- Behavior:
  - Peers interact, sending and receiving messages from each other.
  - Peers are sometimes identical.
  - Many peer-to-peer models support central or distributed locater services.

- Examples:
  - http://www.ecs.syr.edu/faculty/fawcett/handouts/CoreTechnologies/SocketsAndRemoting/code/WCF_Fawcett_Examples/WCF_Peer_Comm/
  - BitTorrent
  - Napster

Peer-to-Peer Asynchronous Message-Passing Structure



A Reusable Communication Structure



Communication Types

- Remote Procedure Call (RPC):
  - Supports function call semantics between processes and machines.
  - Sends messages over wire but provides stack frames for client and server to support the function call model.
  - Examples: COM, CORBA, WCF
- Message passing:
  - Sends message with encoded request and/or data
  - Message contains endpoint information for routing
  - Directly supports asynchronous processing
  - Examples: Internet, web, SMA and OOD projects

Communication Patterns

- TwoWay:
  Synchronous request, wait for reply
- Duplex:
  Asynchronous request, reply sent as callback
- OneWay:
  Send message and forget
  - Receiver may send result back to requester as a subsequent OneWay message
- Examples:
  - All of the above are supported by WCF

Communication Style

- Push model
  - Send information to a remote endpoint via a service call, perhaps via a message:

```
void PostMessage(Message msg);
```

- Pull model
  - Retrieve information from a remote endpoint via a service call, perhaps by a streaming download:

```
Stream downLoad(string filename);
```

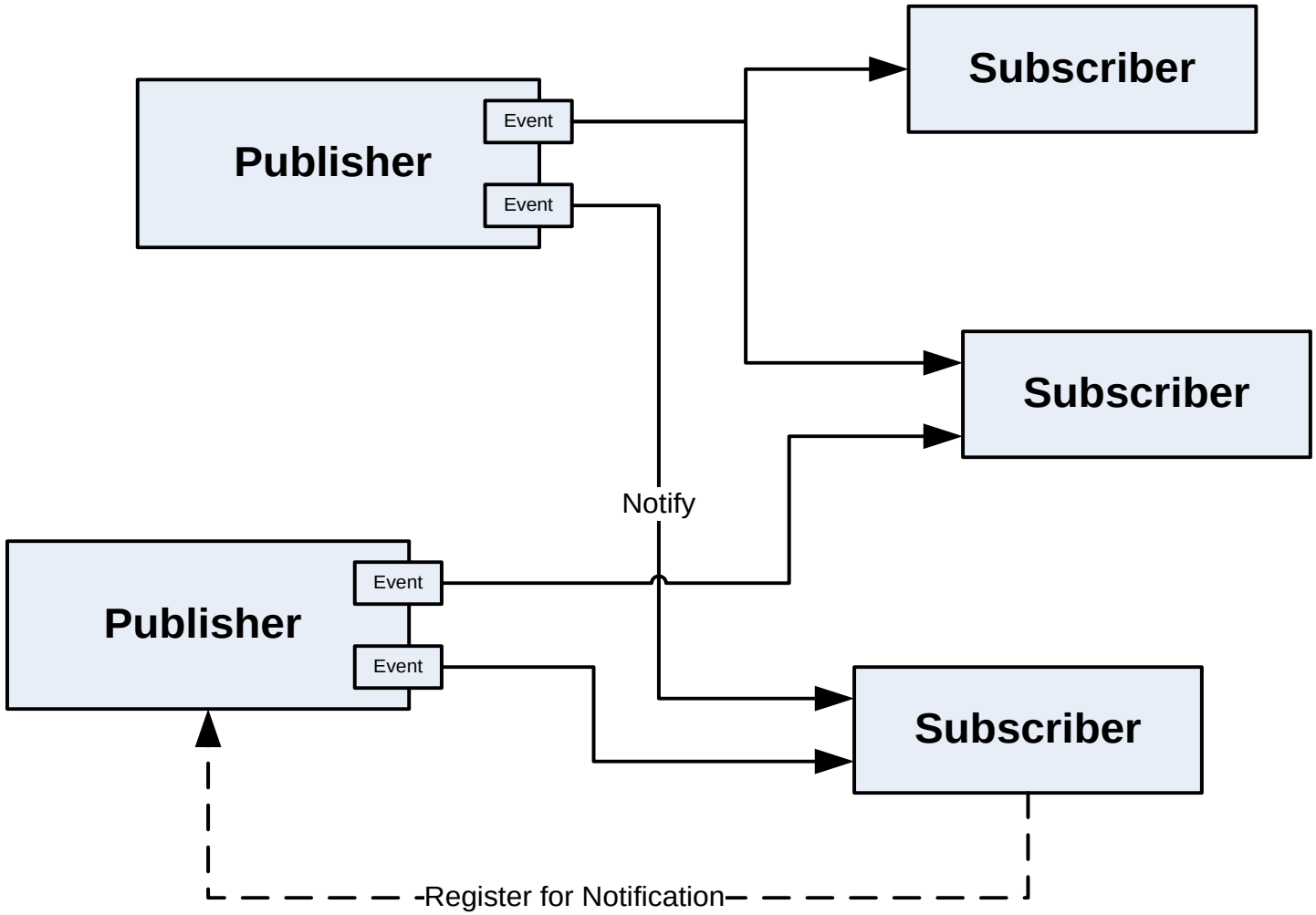
Communication Style

- Pull service and caching
  - A software repository could expose a WCF service that provides information about its package contents including dependencies.
  - That allows a client, for example, to pull from the repository all files in a package dependency list that are not already in its file cache.

Thread and Event-Driven Structure

Structure: Publish and Subscribe

- Structure:
  - Many-to-many connection of publishers and subscribers.
  - Each subscriber registers for notifications with a specific interface.
  - Publishers send notifications to all enrolled subscribers when a publisher event occurs.
  - Publishers can support multiple events.
  - Publishers don’t need to know anything about the subscriber.


Subscriber
Publisher
Event
Event
Subscriber
Notify
Publisher
Event
Event
Subscriber
Register for Notification

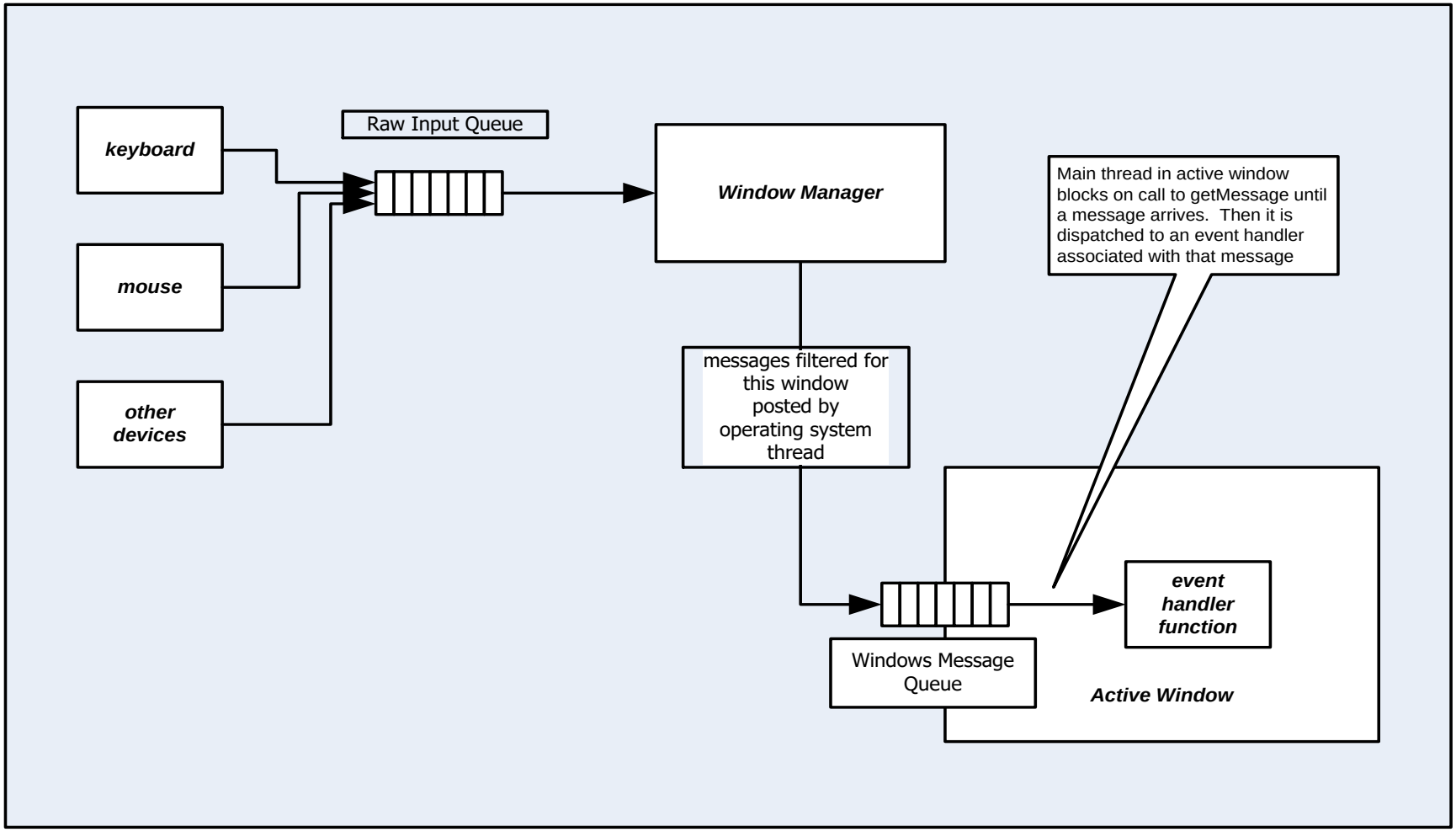
Threading-Driven Structure

- Some program structures are a consequence of specific threading models.
  - Event-driven and Single-Threaded Apartment (STA)
  - Parallel execution
  - Pipelined execution

Structure: Event Driven

- Structure:
  - Events from multiple concurrent sources generate messages which are enqueued, and typically are processed by a single handling thread.
  - Messages are dispatched to event-handlers for processing.
- Example:
  - Windows processing

Event Driven

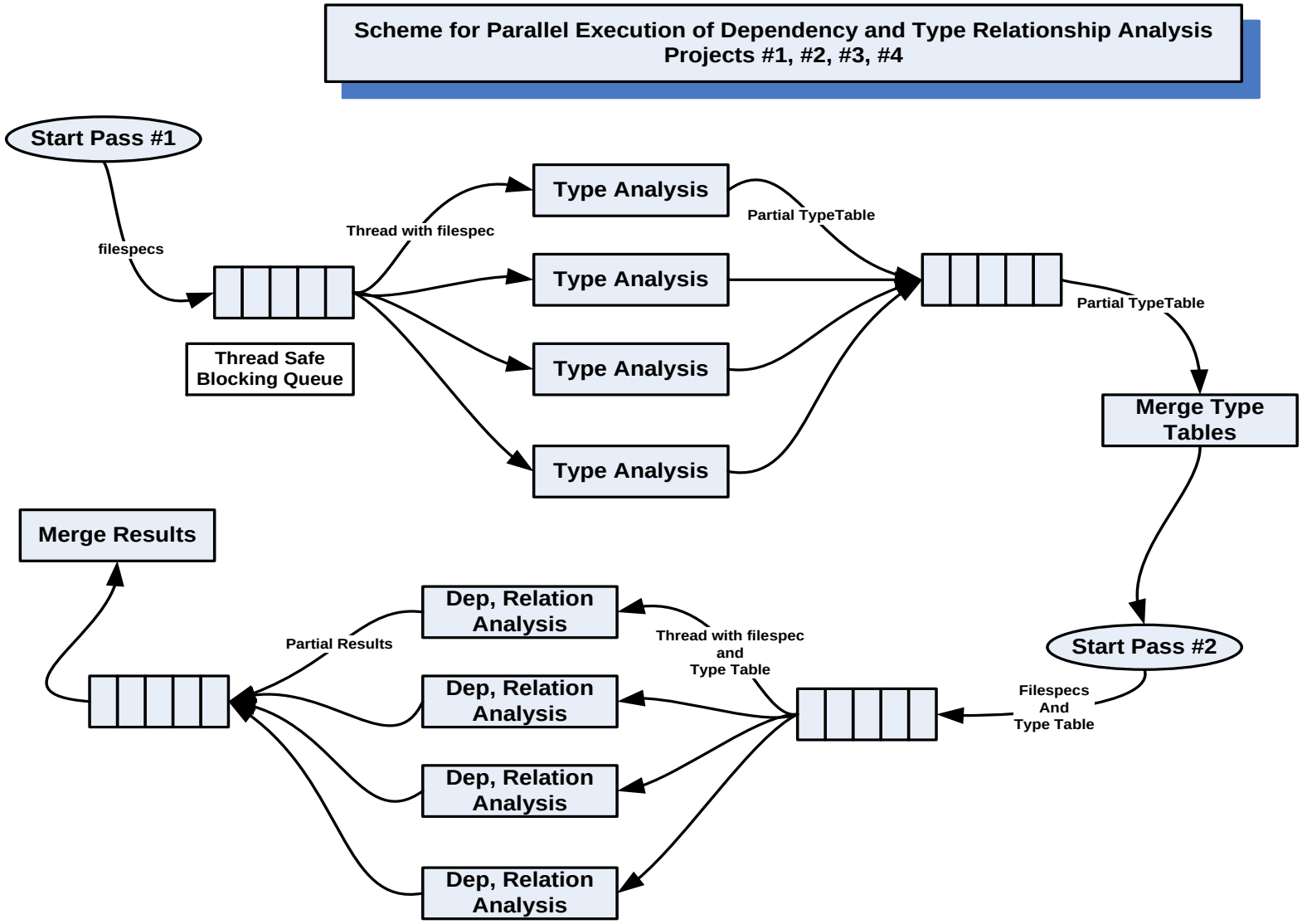


Single-Threaded Apartment

- Graphical user interfaces all use the STA model.
  - Possibly concurrent clients send messages to the GUI's message queue.
  - All messages are retrieved by a single thread, the one that created the window.
  - Child threads, often used to execute tasks for the GUI, are not allowed to directly interact with the window.
  - Instead they must send or post messages to the window's message queue.
  - This is often done with Form.Invoke or Dispatcher.Invoke.

Parallel Execution

- Structure:
  - Often concurrent programs provide enqueued task requests.
  - Threads, perhaps from a thread pool, are dispatched to handle each task.
  - Tasks must be independent in order to fully realize the benefits of concurrency.
- Example:
  - Concurrent execution of dependency analysis tasks.


Scheme for Parallel Execution of Dependency and Type Relationship Analysis
Projects #1, #2, #3, #4
Start Pass #1
filespecs
Thread Safe
Blocking Queue
Thread with filespec
Type Analysis
Type Analysis
Type Analysis
Type Analysis
Partial TypeTable
Partial TypeTable
Merge Type
Tables
Start Pass #2
Filespecs
And
Type Table
Thread with filespec
and
Type Table
Dep, Relation
Analysis
Dep, Relation
Analysis
Dep, Relation
Analysis
Dep, Relation
Analysis
Partial Results
Merge Results

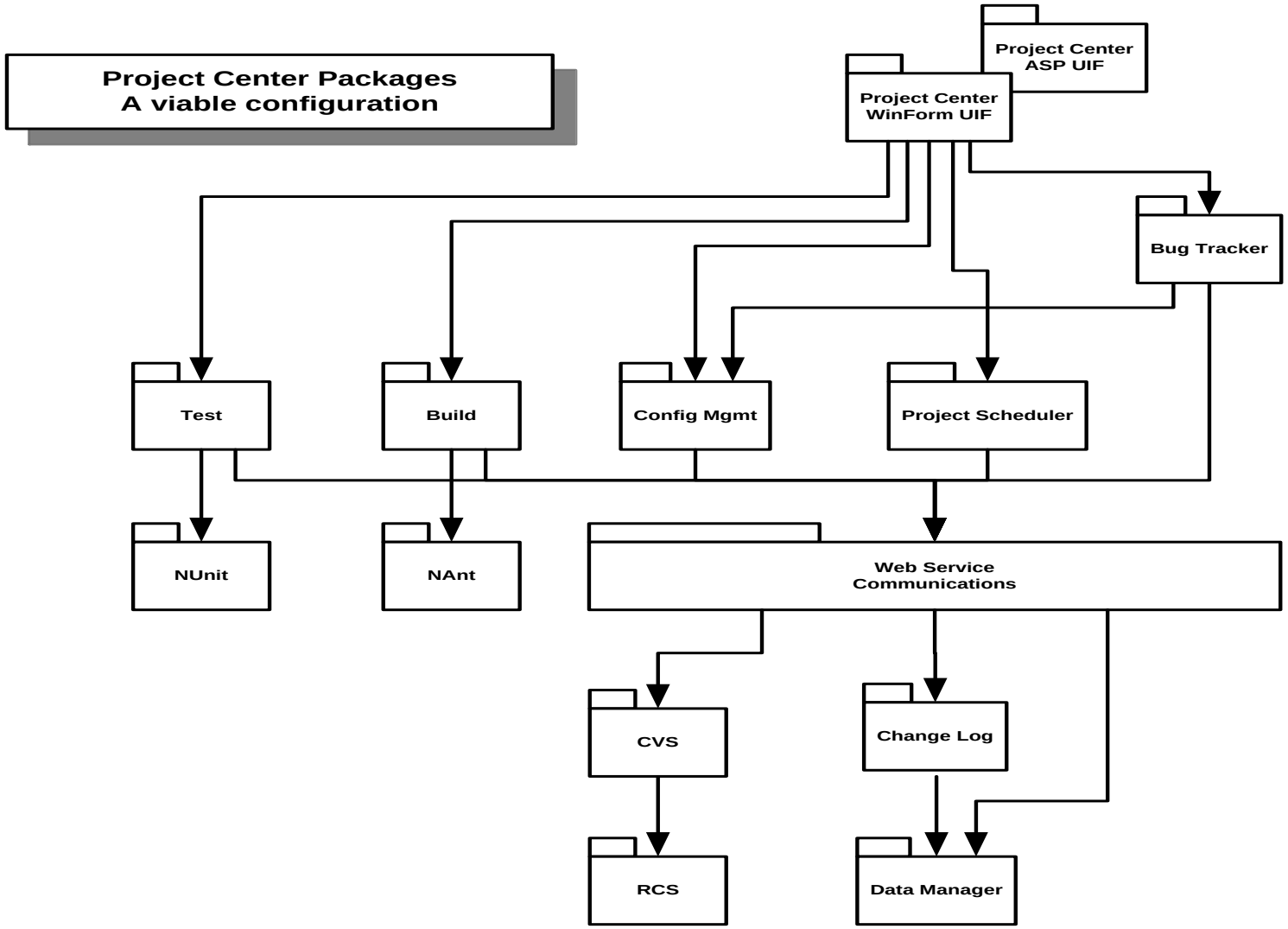
Enterprise Computing

Enterprise Computing

- Large enterprise applications are usually constructed as a federation of lower-level systems and subsystems.
  - The federation is glued together with network-based middleware, or more commonly now, with web services.
- Example: PeopleSoft, used by Syracuse U
  - Payroll and accounting
  - Academic planning and record keeping
  - Employee services
  - A variety of web applications, like mySlice

Enterprise App: Project Center

- Federation of tools supporting software development
  - Open source tools with integrating wrappers:
    - CVS—configuration management
    - Nant—software builds
    - Nunit—software testing
  - Newly developed and legacy tools:
    - Bug tracker, change tracker, project scheduler
- http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/ProjectCenter.htm


Project Center Packages
A viable configuration
Project Center ASP UIF
Project Center WinForm UIF
Bug Tracker
Test
Build
Config Mgmt
Project Scheduler
NUnit
NAnt
Web Service Communications
CVS
Change Log
RCS
Data Manager

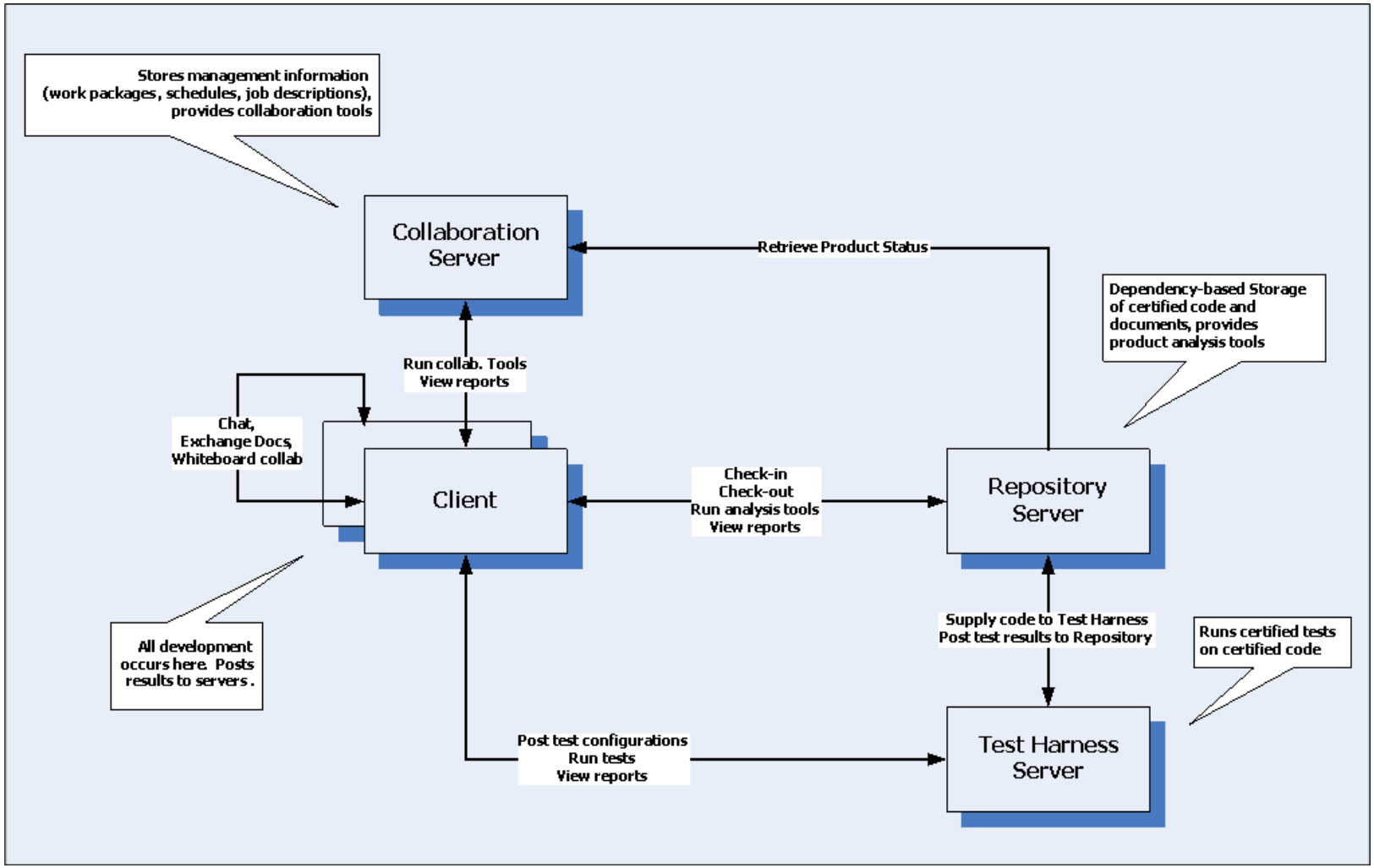
Federation Structure

- Federated systems often are based on one of two design patterns:
  - ***Façade*** provides an integrating interface that consolidates a, possibly large, set of system interfaces into a single application interface in an attempt to make the system easier to use than working directly with its individual parts.
  - ***Mediator*** serves as a communication hub so that all the various subsystems need know only one interface, that of the mediator.

Collaboration System

- System that focuses on sharing of processes and products among peers with a common set of goals.
  - Primary focus is organizing and maintaining some complex, usually evolving, state:
    - Software development baseline
    - Set of work plans and schedules
    - Documentation and model of obligations
    - Communication of events
- Example:
  - Collab – CSE784, Fall 2007, http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/CServ.htm

Example Collaboration System



ENGINEERING@SYRACUSE