Software Structure

Jim Fawcett

CSE681 – Software Modeling and Analysis

Fall 2017

Introduction

What is Program Structure?

An Example

What is Software Structure?

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

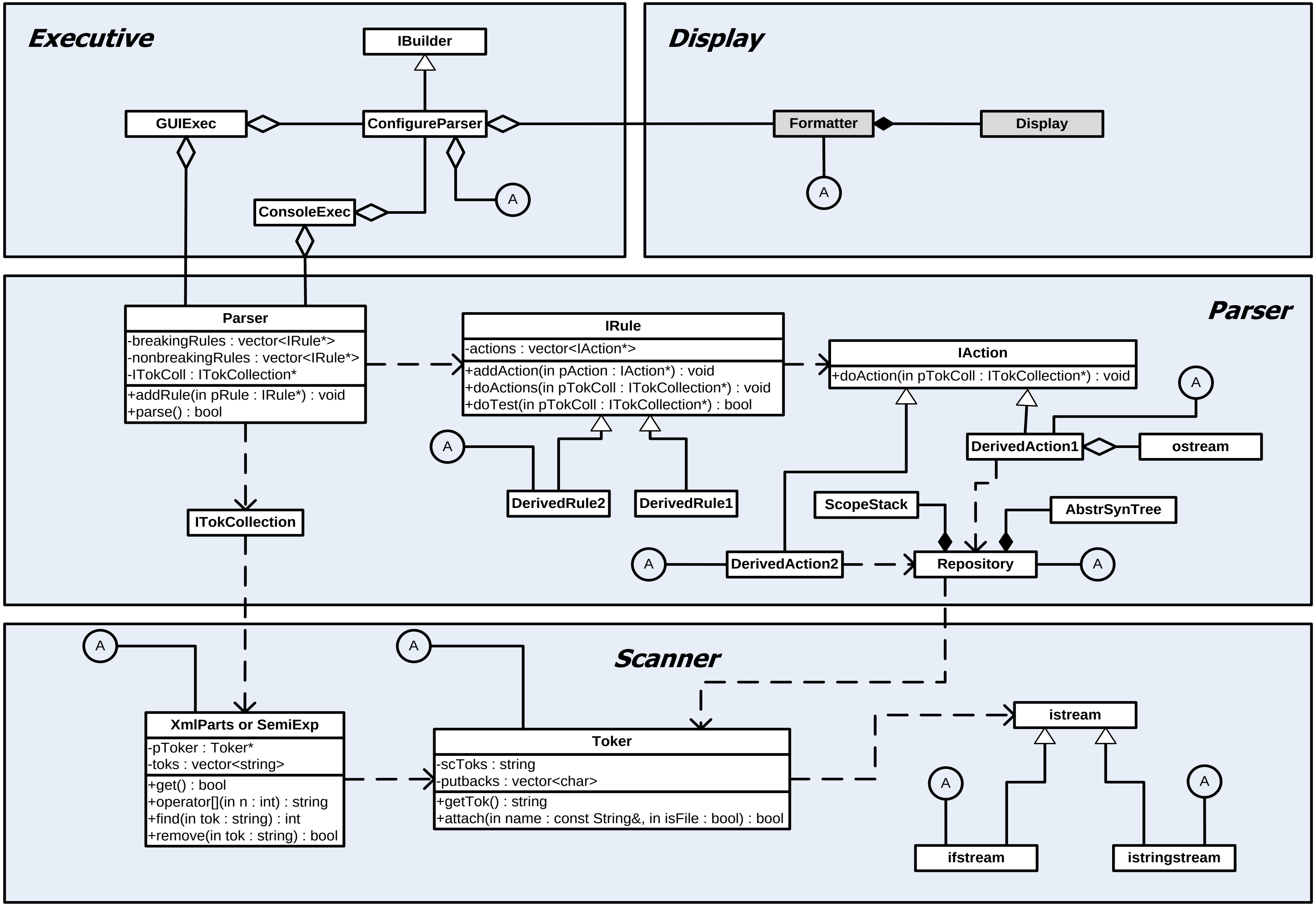
Program Structure

- 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, etc.

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



Executive

IBuilder

GUIExec

ConfigureParser

ConsoleExec

A

Display

Formatter

Display

A

Parser

Parser

-breakingRules : vector<IRule*>
-nonbreakingRules : vector<IRule*>
-ITokColl : ITokCollection*

+addRule(in pRule : IRule*) : void
+parse() : bool

IRule

-actions : vector<IAction*>

+addAction(in pAction : IAction*) : void
+doActions(in pTokColl : ITokCollection*) : void
+doTest(in pTokColl : ITokCollection*) : bool

IAction

+doAction(in pTokColl : ITokCollection*) : void

ITokCollection

DerivedRule2

DerivedRule1

DerivedAction1

ostream

ScopeStack

AbstrSynTree

DerivedAction2

Repository

A

Scanner

XmlParts or SemiExp

-pToker : Toker*
-toks : vector<string>

+get() : bool
+operator[](in n : int) : string
+find(in tok : string) : int
+remove(in tok : string) : bool

Toker

-scToks : string
-putbacks : vector<char>

+getTok() : string
+attach(in name : const String&, in isFile : bool) : bool

istream

ifstream

istringstream

A

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 & Event Driven**
  - Single Threaded Apartment (STA)
  - Parallel execution
  - Pipeline execution
- **Enterprise Computing**
  - Federated systems

Data Driven Structures

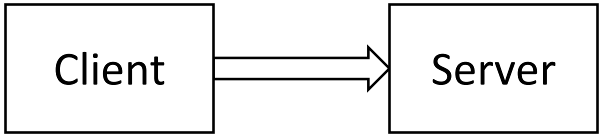
Programs that are dominated by management of data

Web applications are often Data Driven

Data Driven Structures

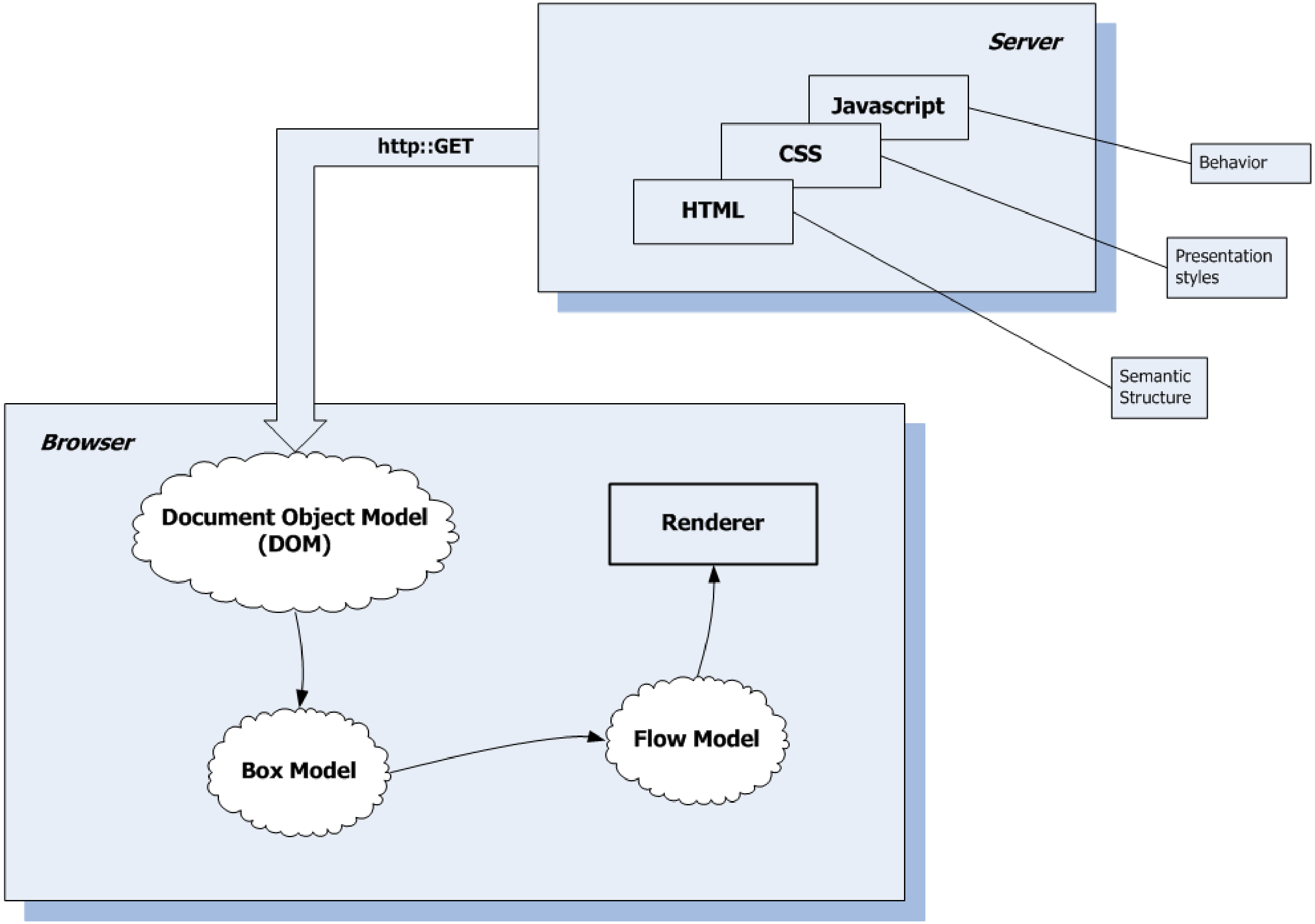
- Some program structures are driven by the presentation and management of data:
  - Client-Server
  - Three-Tier
  - Model-View-Controller

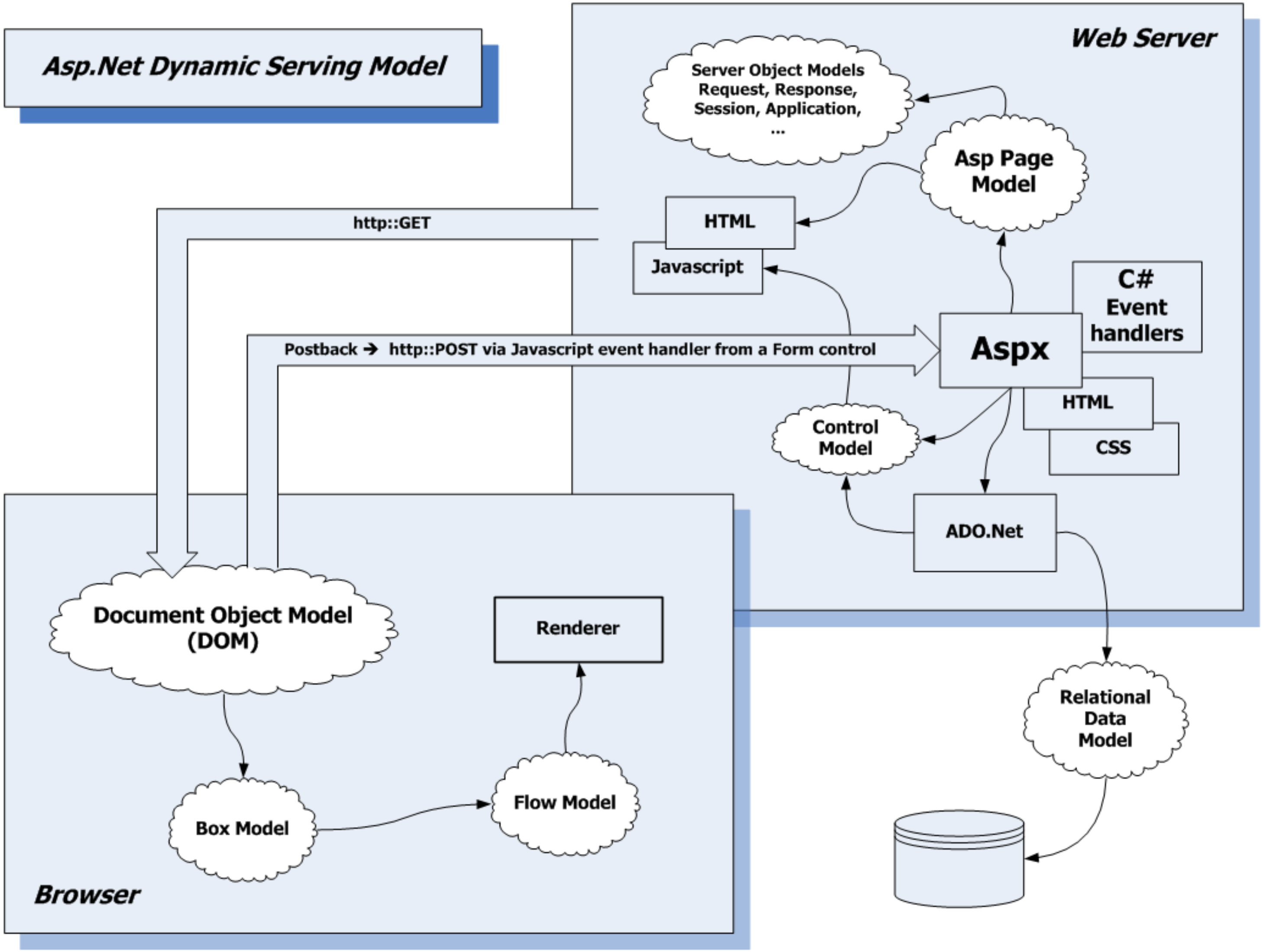
Structure: Client-Server



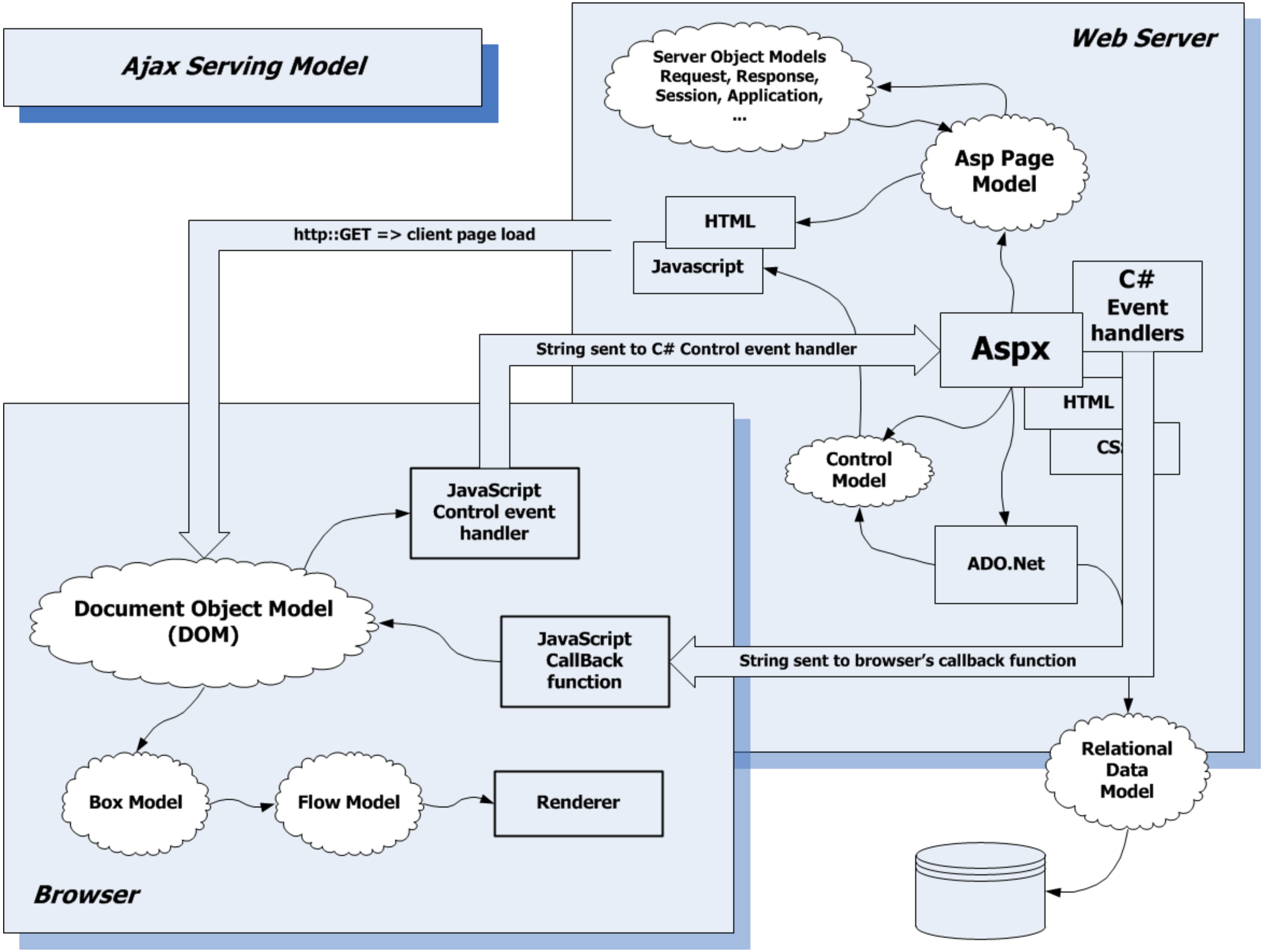
- Behavior:
  - Server is passive, waits for client requests
  - Client requests are synchronous – after sending request client waits for reply
  - Server contains data shared among its clients
  - Server handles multiple concurrent clients
  - Without additional structure system may become tightly coupled and difficult to change
- Example:
  - Web server and browser clients

Static Webpage Model




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

Ajax Serving Model



Web Server

Server Object Models Request, Response, Session, Application, ...

Asp Page Model

HTML

Javascript

http::GET => client page load

C# Event handlers

Aspx

String sent to C# Control event handler

HTML

CS

Control Model

ADO.Net

Relational Data Model

String sent to browser's callback function

JavaScript Control event handler

JavaScript CallBack function

Document Object Model (DOM)

Box Model

Flow Model

Renderer

Browser

Sharing Data

- Relational Databases – SQL Server, mySql, …
  - ACID – Atomicity, Consistency, Isolation, Durability
  - ACID => Transactional
- No SQL 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

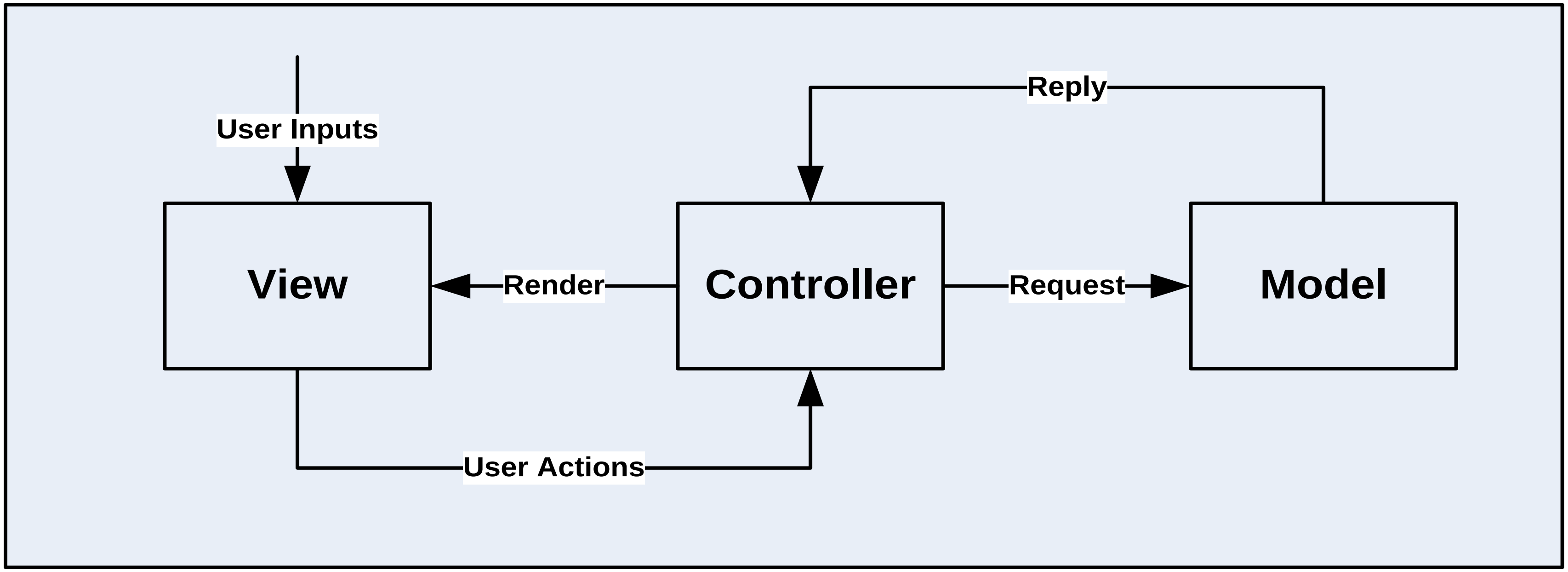
Separation of Concerns

- 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.

Basic MVC Structure

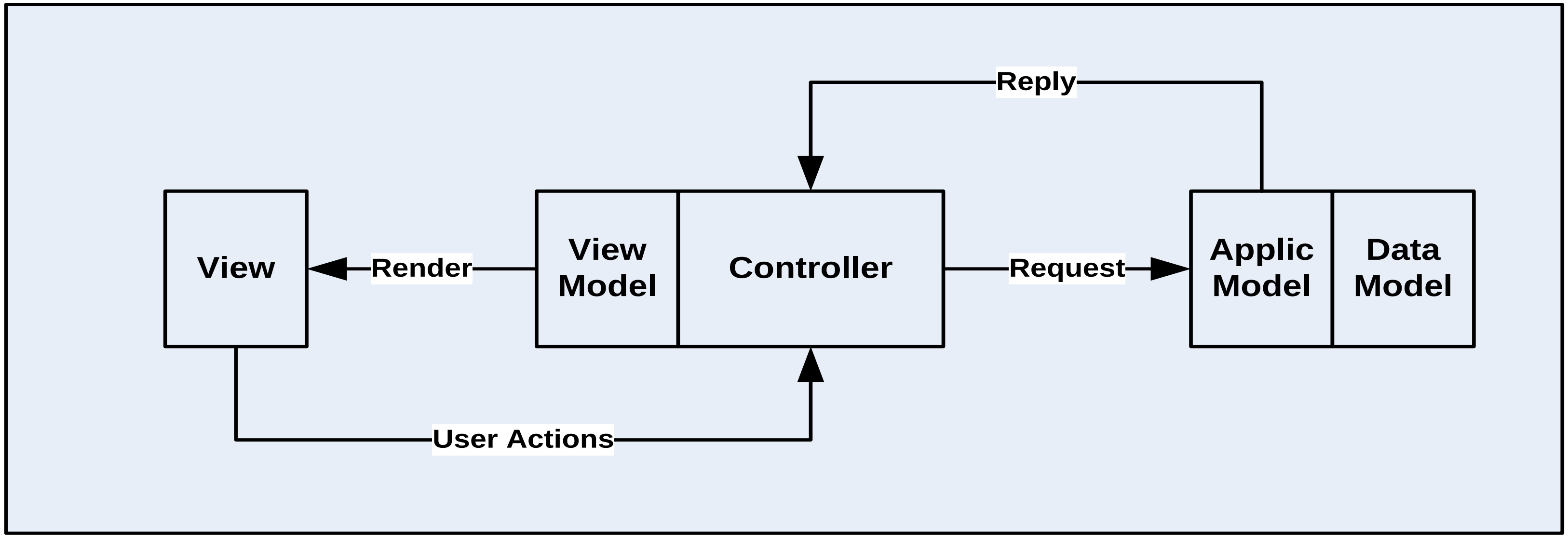


Model-View-Controller

- 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

MVC – With View & Application Models

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



View – View Model

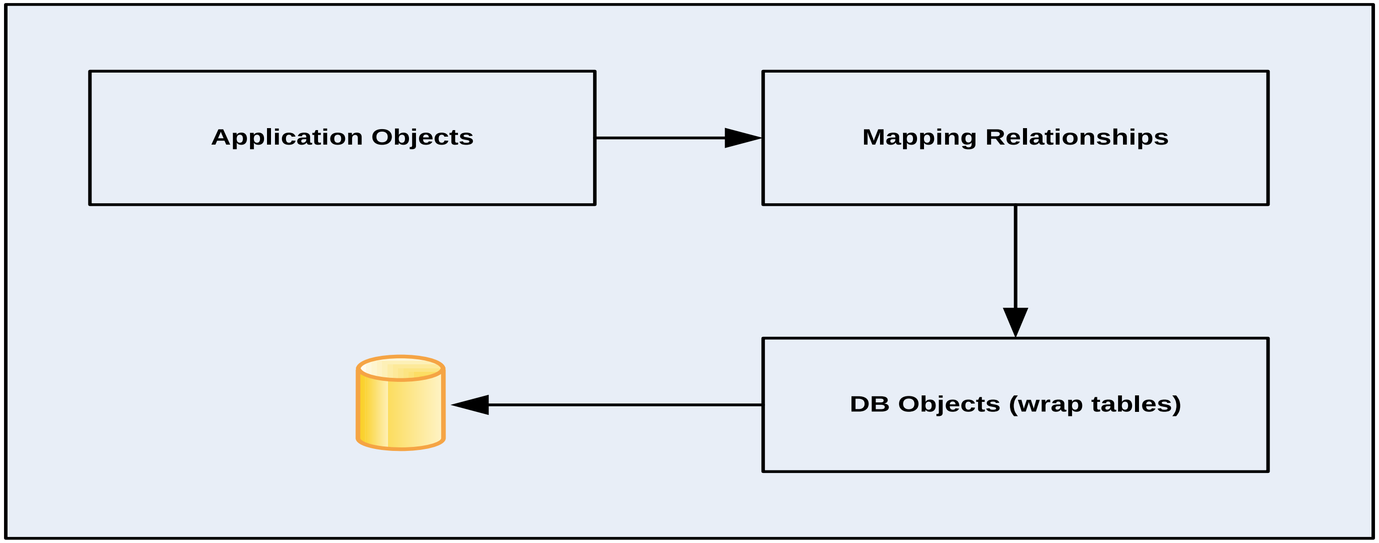
- A view is what gets rendered
- A view model is an abstraction that:
  - Defines resources that many be used in several places.
  - Defines styles that may be used in several places
  - Defines an object model for the application to manipulate

Application vs. Data Models

- Application model
  - Defines classes for all the entities a user knows and cares about, e.g., orders, customers, products, etc.
- Data model
  - Defines wrapper classes for tables and stored procedures
  - Manages connections
- Object to Relational Mapping
  - Relationships between application objects and data objects.

Object Relational Mapping

- Data Layers often have an ORM substructure

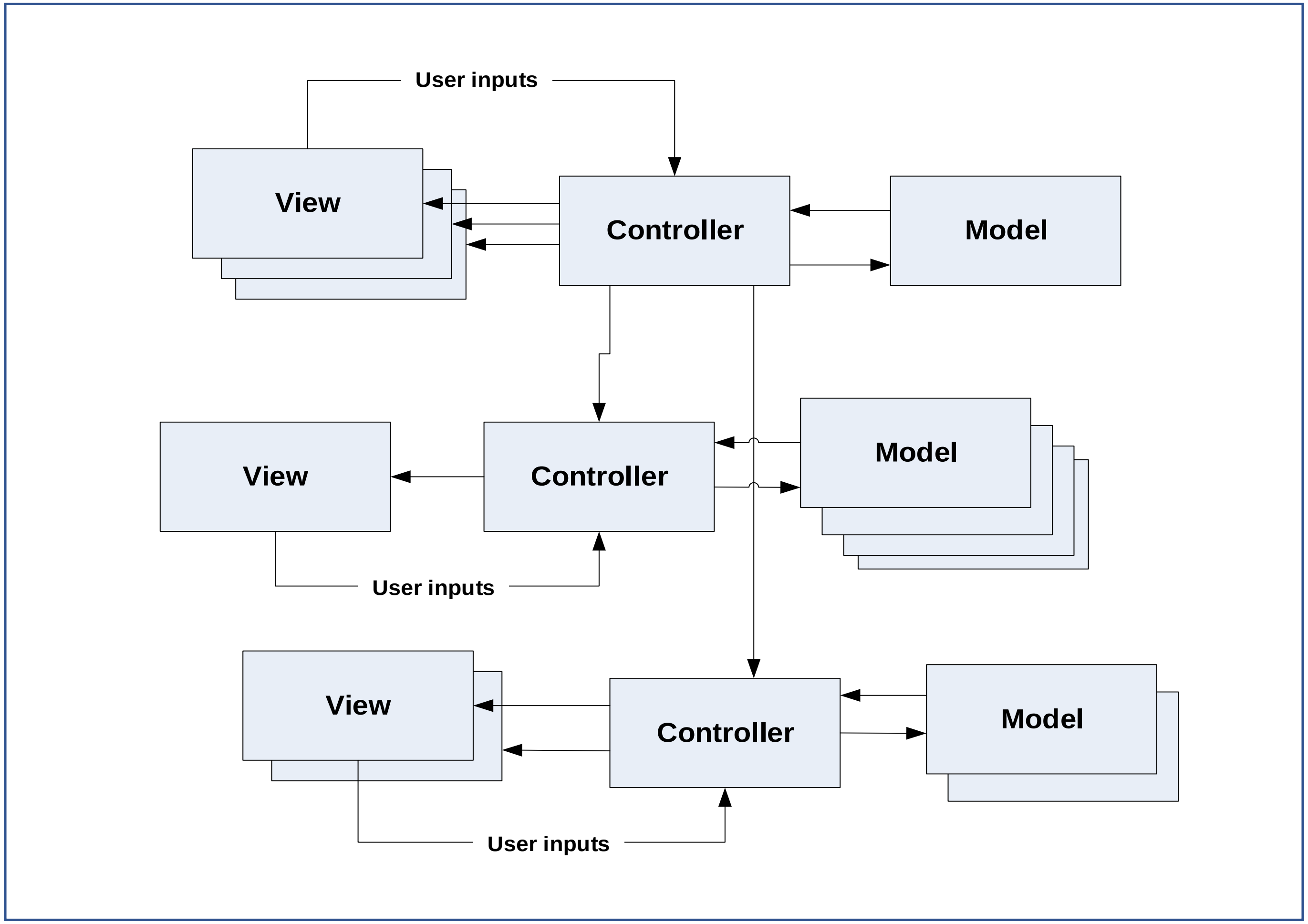


- Examples: Hibernate, Microsoft Entity Framework

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

MVC – Multiple Controllers



Layer-Driven Structures

Components

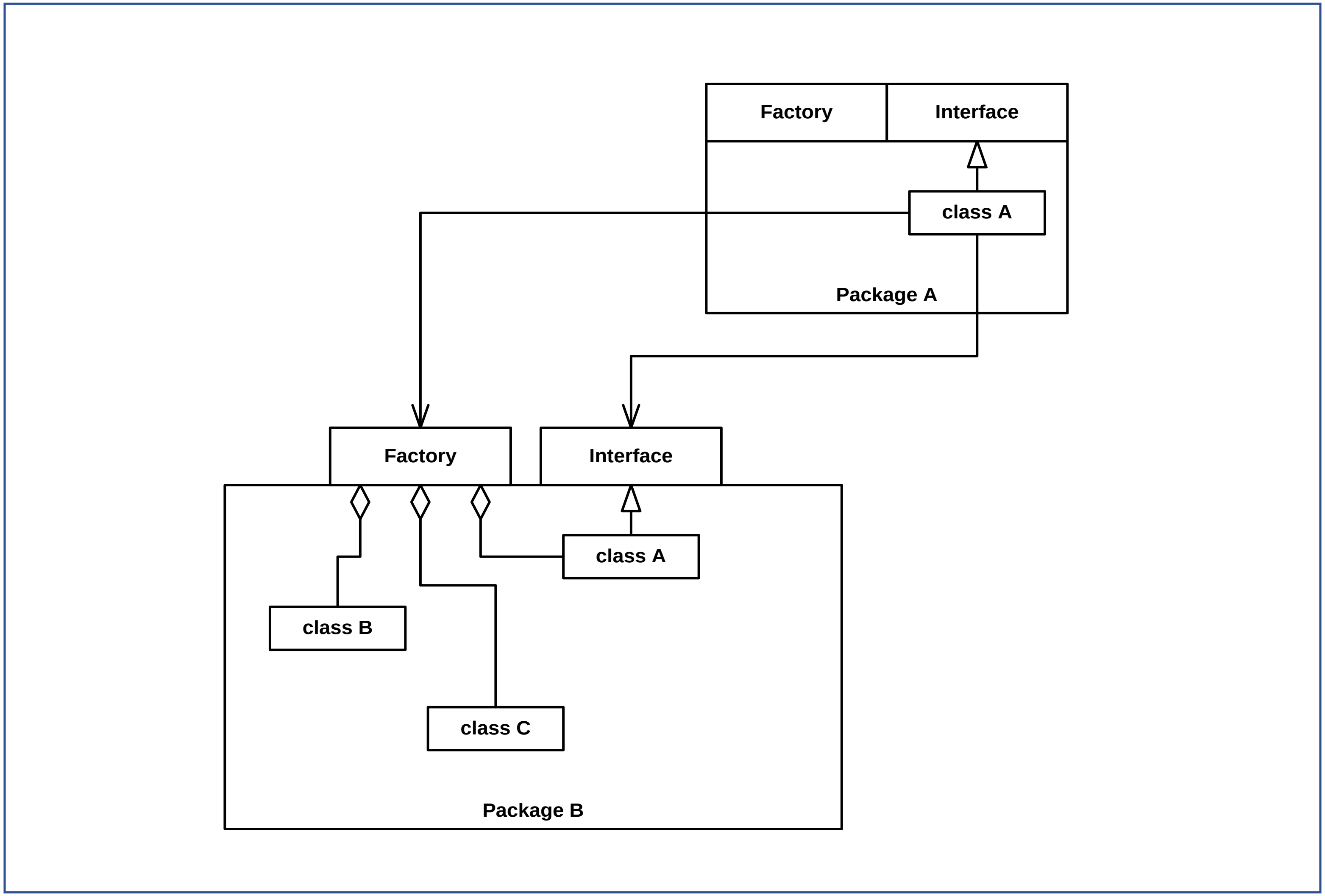
Services

REST

Component Layered Structure

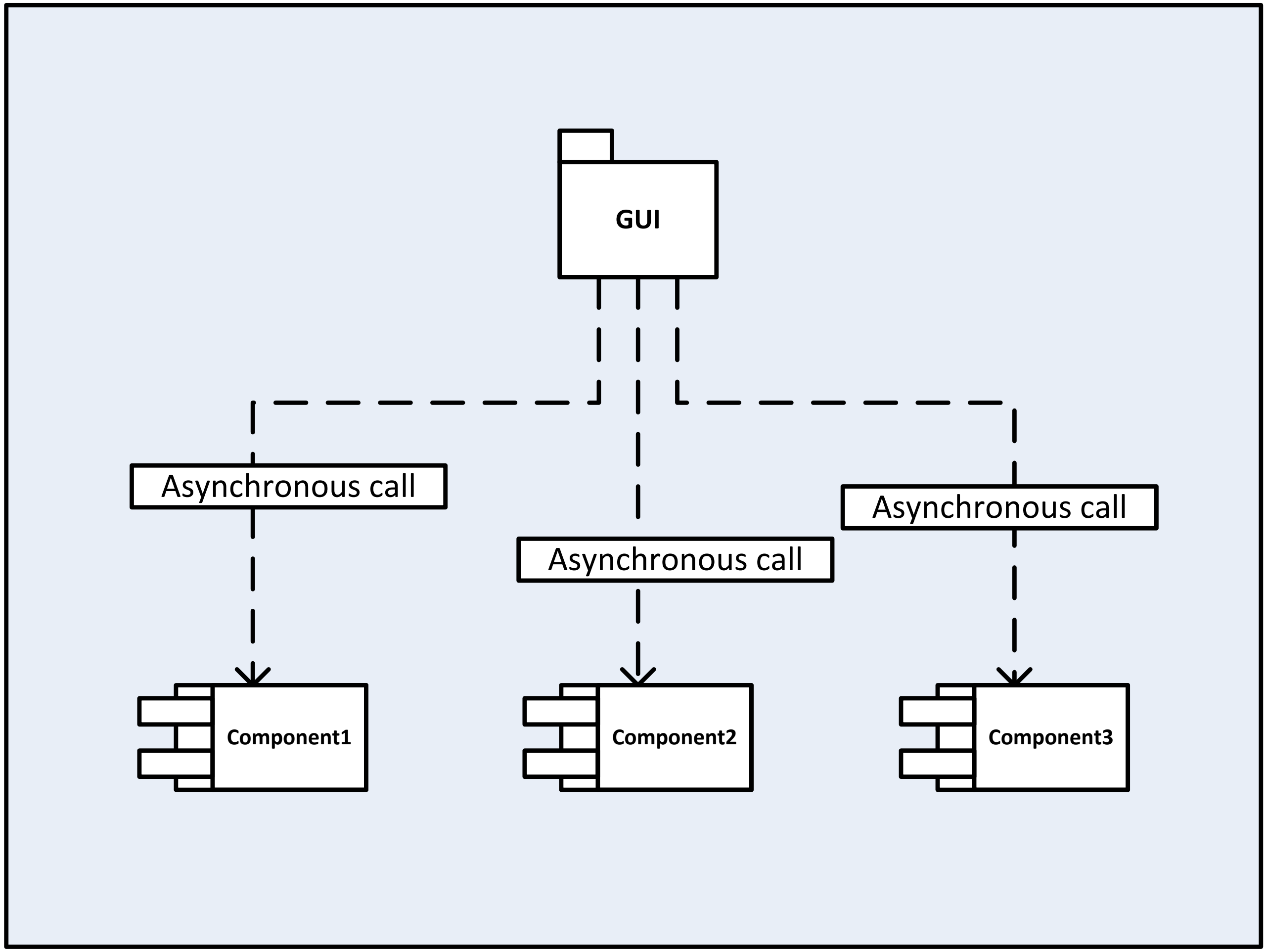
- Structure:
  - A componentized system is composed of an application with many pluggable component parts.
  - A component is pluggable if it implements a plug-in interface, published by the application, provides an object factory for activating its internal objects, and is packaged as a dynamic link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

Hiding Implementation Details



Example Componentized System

Separate presentation from application logic

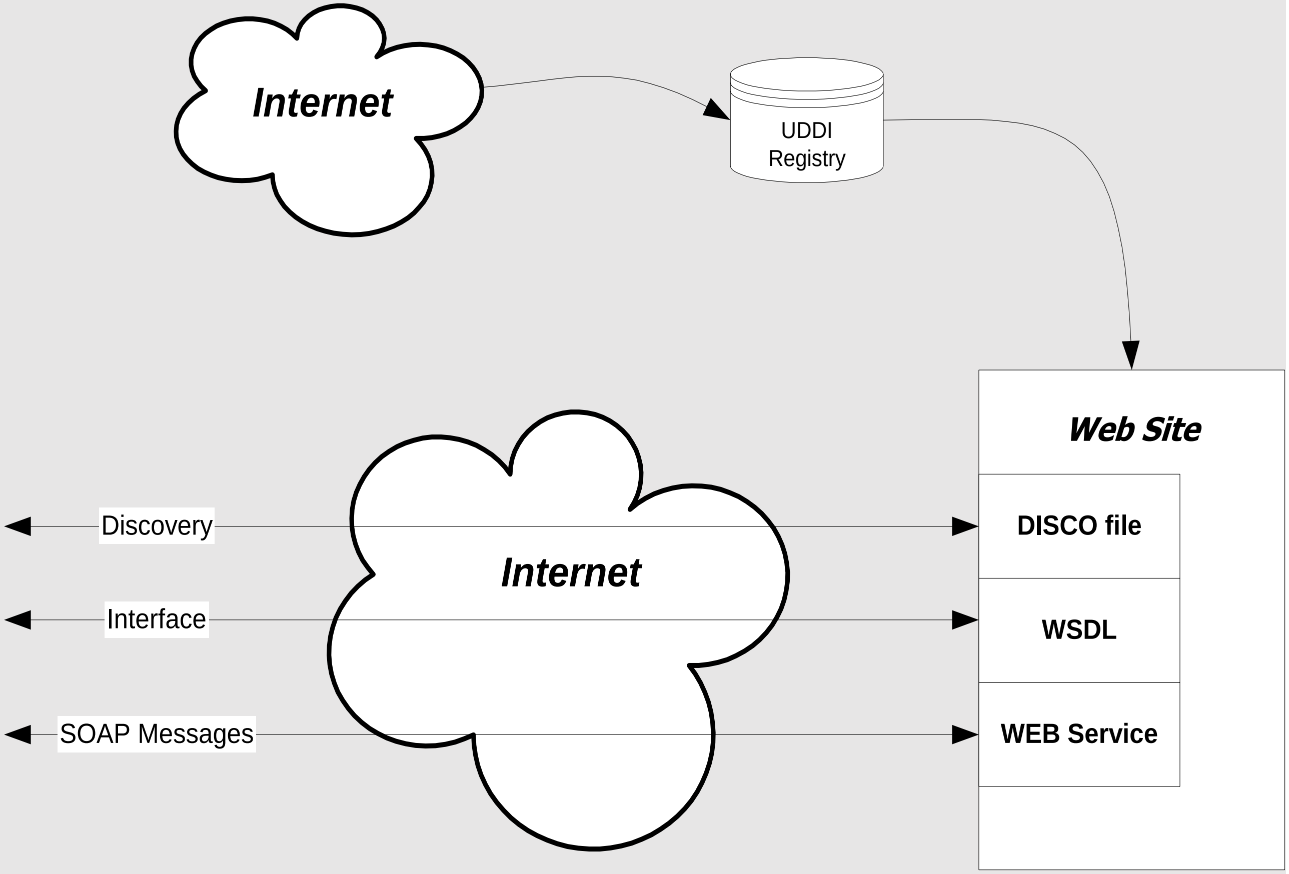


Service Layered Structure

- 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 heavy weight 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.
- Nouns – the resources exposed by the system – are identified by URIs – Uniform Resource Identifiers
- Its encoding is UTF text, not SOAP or some other complex messaging format, but may use encryption, as in HTTPS.

Published REST APIs

- Amazon REST API
  - https://docs.aws.amazon.com/apigateway/api-reference/
- Azure REST API
  - https://docs.microsoft.com/en-us/rest/api/azure/
- Google Drive APIs
  - https://developers.google.com/drive/api/v2/reference/
- Dell Open Automation Guide
  - https://www.dell.com/support/manuals/us/en/04/force10-open-automation/oa_9.8.2.0_cli_config_pub/rest-api?guid=guid-3b60f154-bfd4-4da3-aa11-8e97c7018d4a&lang=en-us
- IBM REST APIs
  - https://www.ibm.com/support/knowledgecenter/en/SSMKHH_10.0.0/com.ibm.etools.mft.doc/bi12017_.htm

Analysis Driven Structure

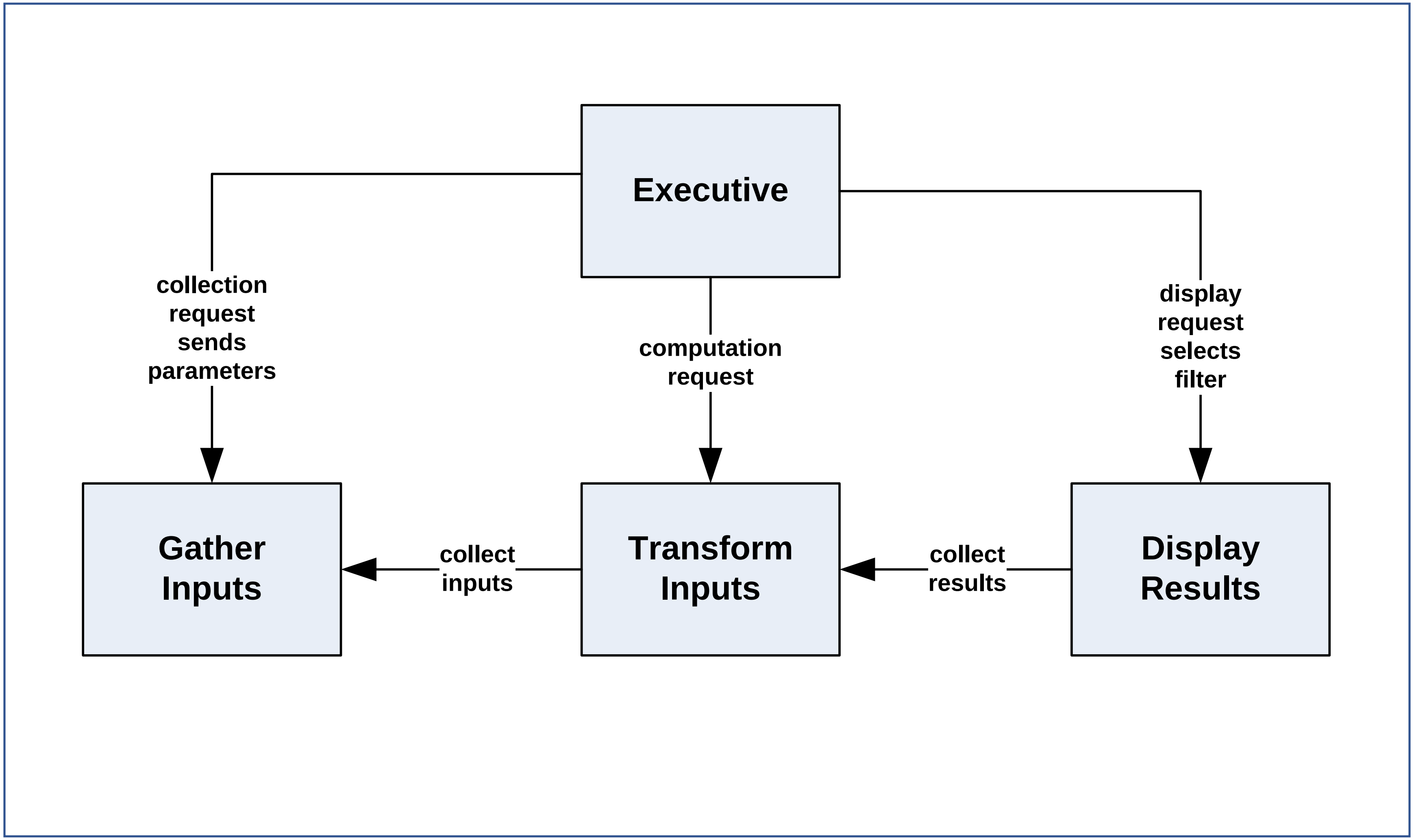
Packages

Pipelines

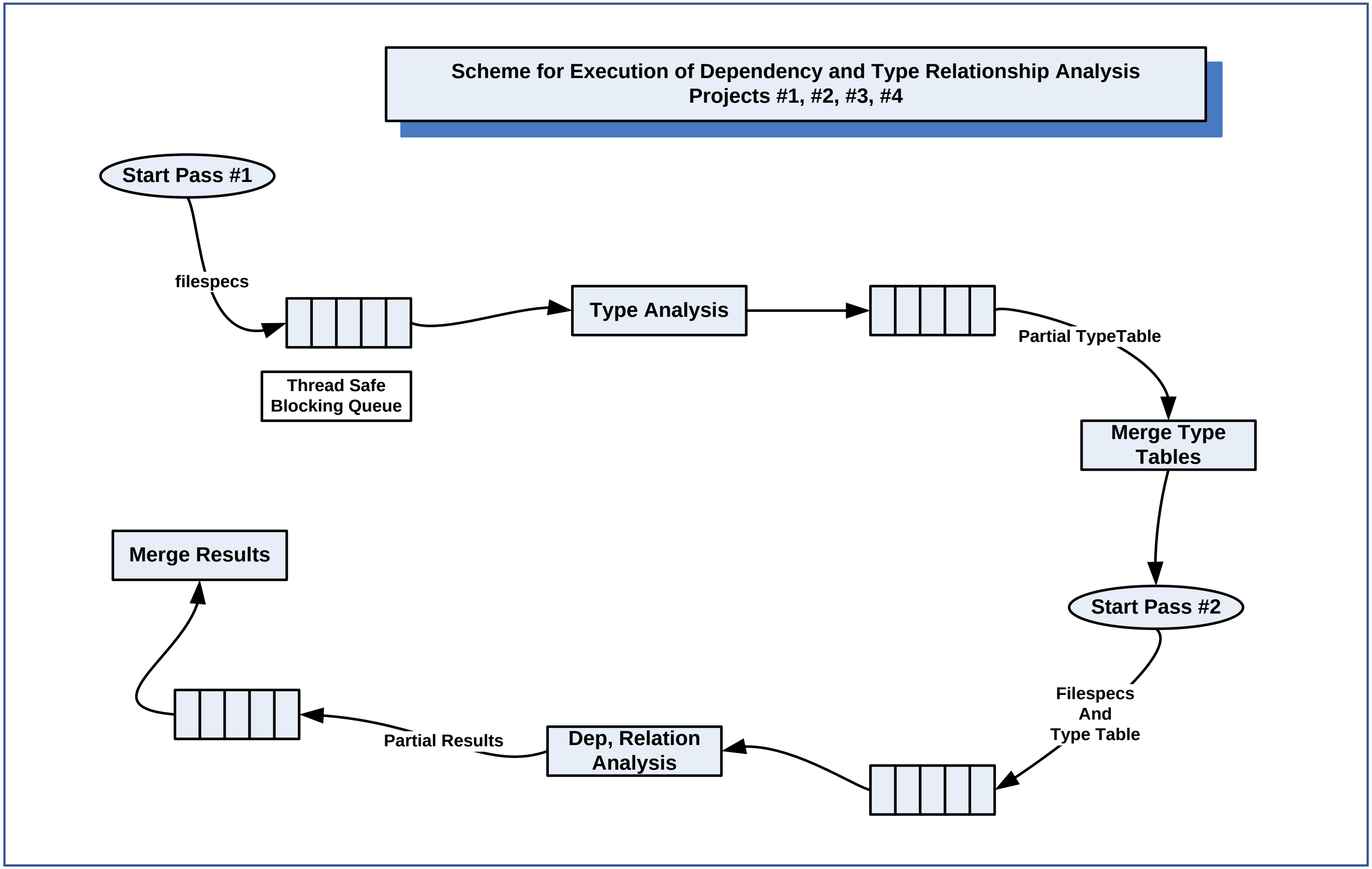
Analysis Driven Structure

- 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

Client-Server

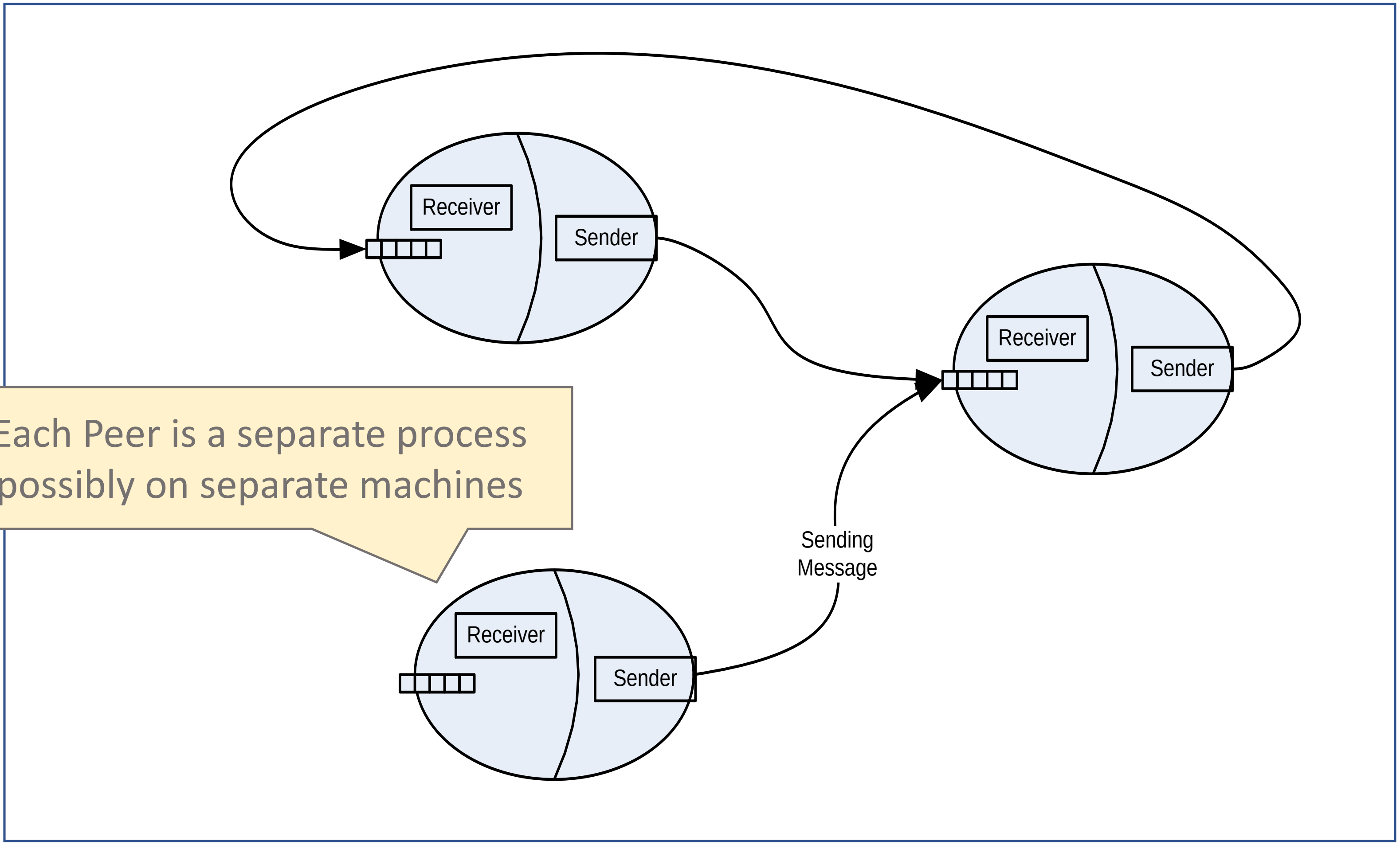
Peer-to-Peer

MiddleWare

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

Peer-To-Peer Asynchronous Message-Passing Structure



Each Peer is a separate process possibly on separate machines

Communication 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 1000 T units of time.

Structure: Client-Server

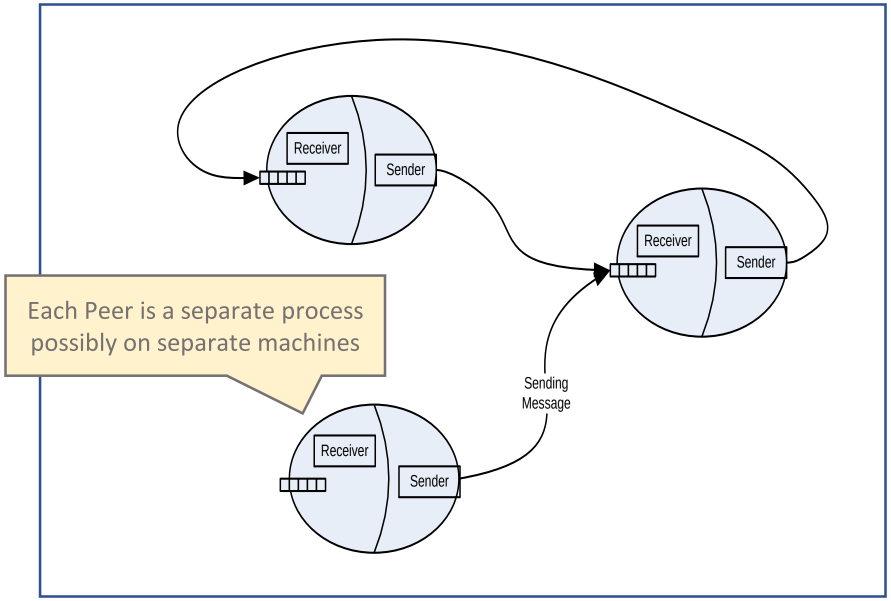
- Behavior:
  - Server is passive, waits for client requests
  - Server handles multiple concurrent clients
  - Without additional structure system may become tightly coupled and difficult to change

- Example:
  - Web server and browser clients

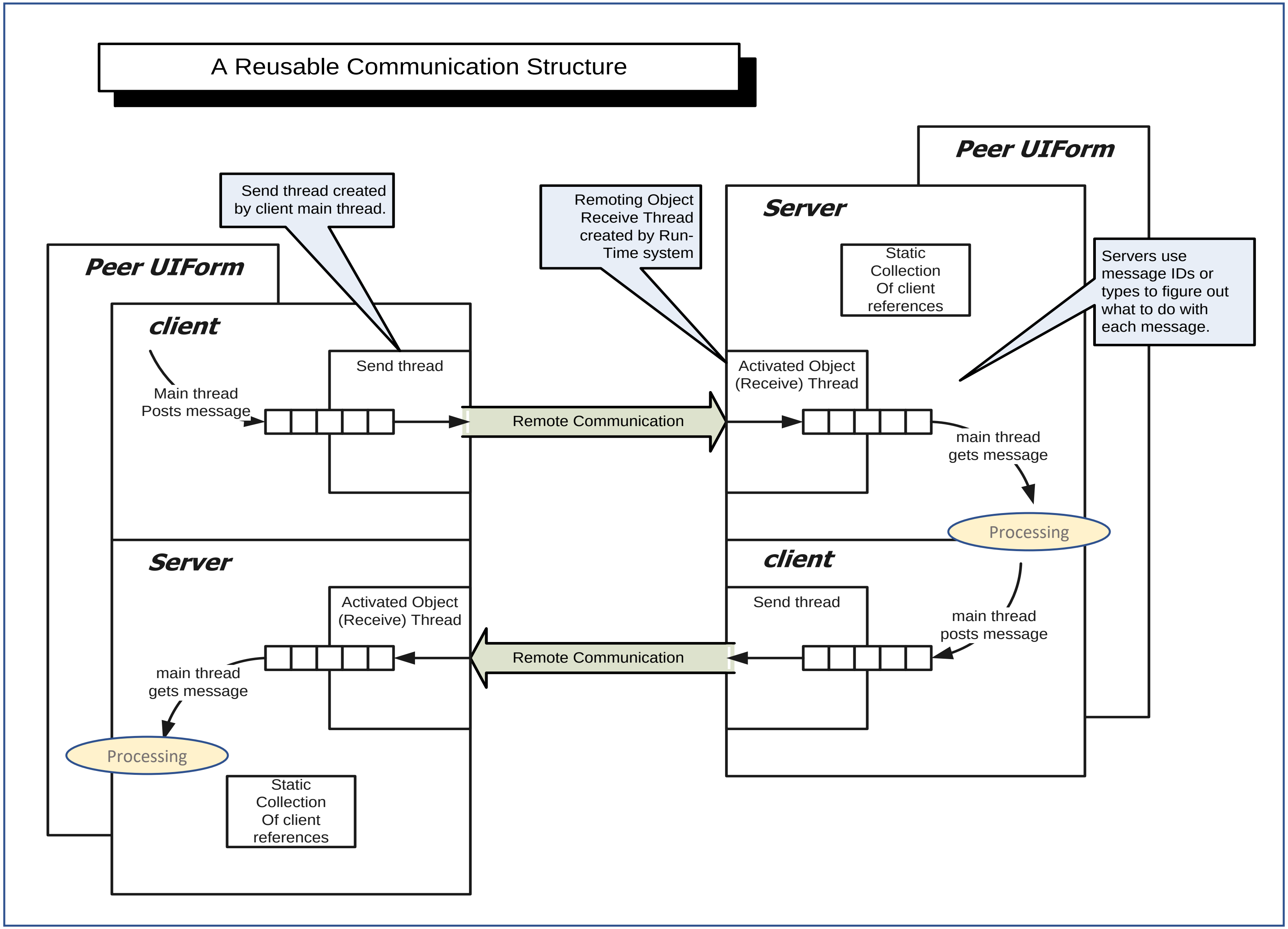
Structure: Peer-To-Peer

- 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:
  - Project #4
  - Bit-Torrent
  - Napster



Each Peer is a separate process possibly on separate machines


A Reusable Communication Structure
Peer UIForm
Server
Static Collection Of client references
Servers use message IDs or types to figure out what to do with each message.
Remoting Object Receive Thread created by Run-Time system
Send thread created by client main thread.
Peer UIForm
client
Main thread Posts message
Send thread
Remote Communication
Activated Object (Receive) Thread
main thread gets message
Processing
client
Send thread
main thread posts message
Server
Activated Object (Receive) Thread
Remote Communication
main thread gets message
Processing
Static Collection Of client references

Communication-Driven Program 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.
  - That makes sense only if the packages are versioned, so we can distinguish between copies versus updates.

Thread & Event Driven Structure

Publish and Subscribe

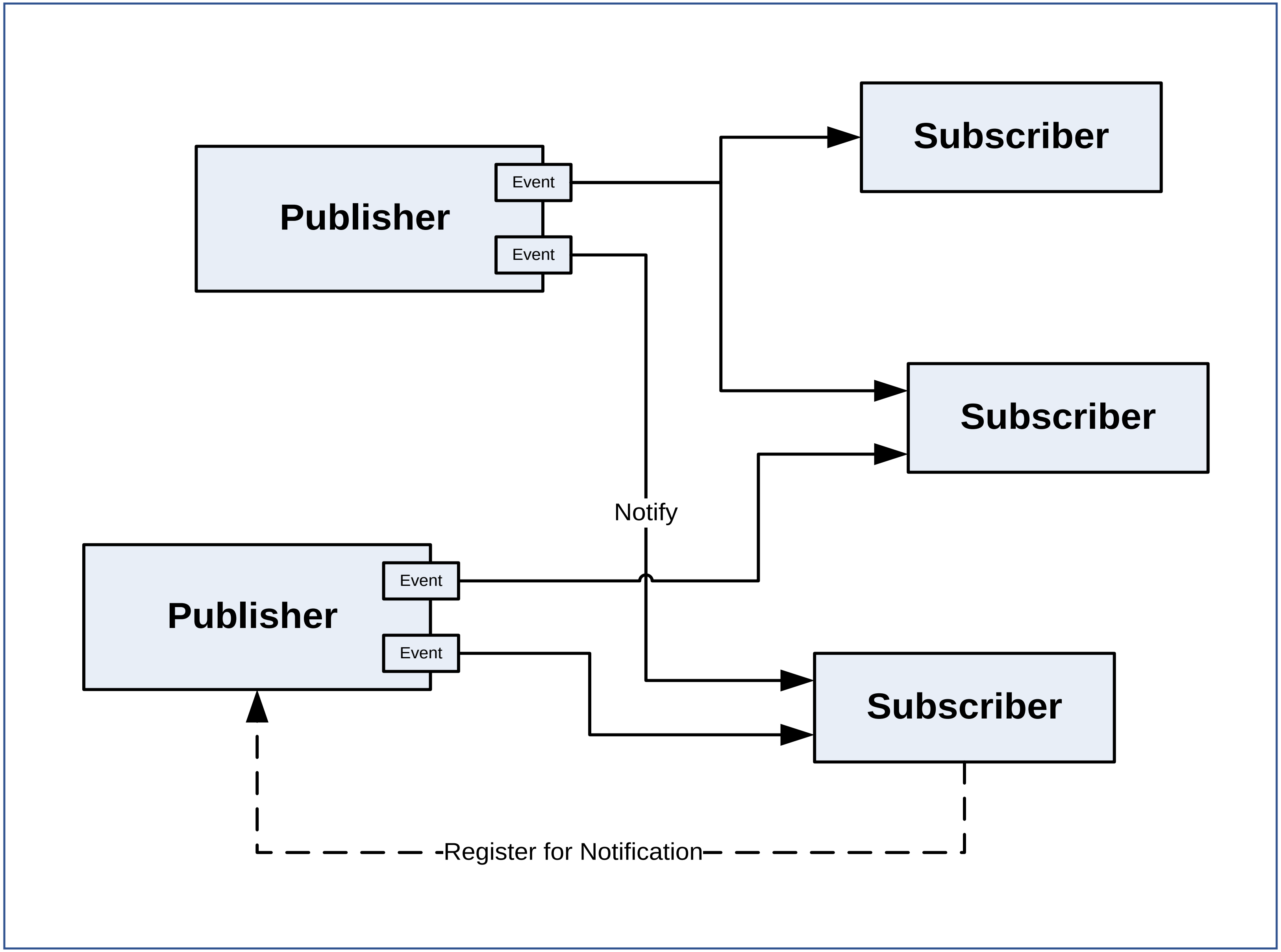
Event Driven

Parallel Processing

Pipelines

Structure: Publish & 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.



Event-Driven Program Structure

Threading Driven Structure

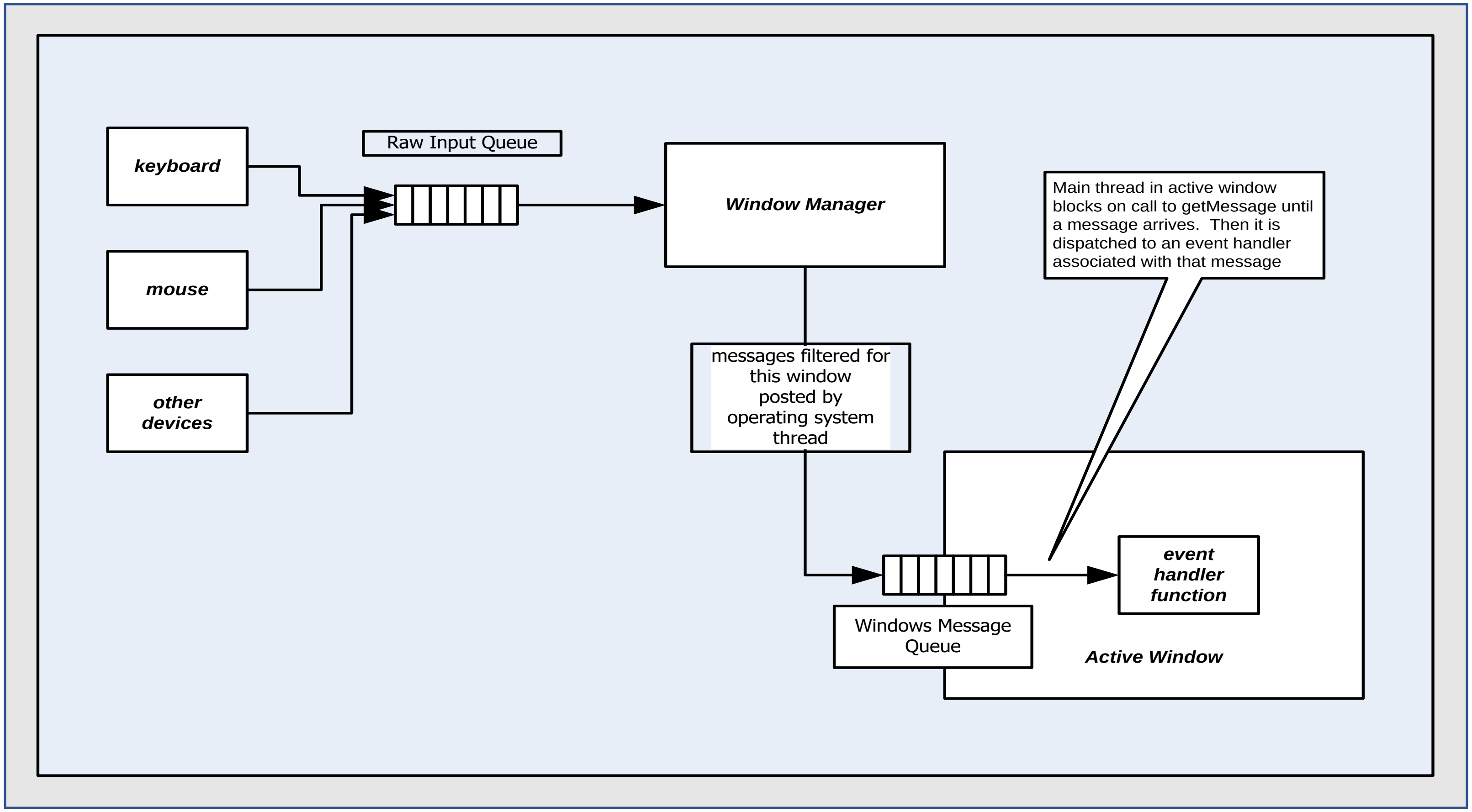
- Some program structures are a consequence of specific threading models
  - Event-driven and Single Threaded Apartment (STA)
  - Parallel execution
  - Pipelined execution

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

Windows Processing

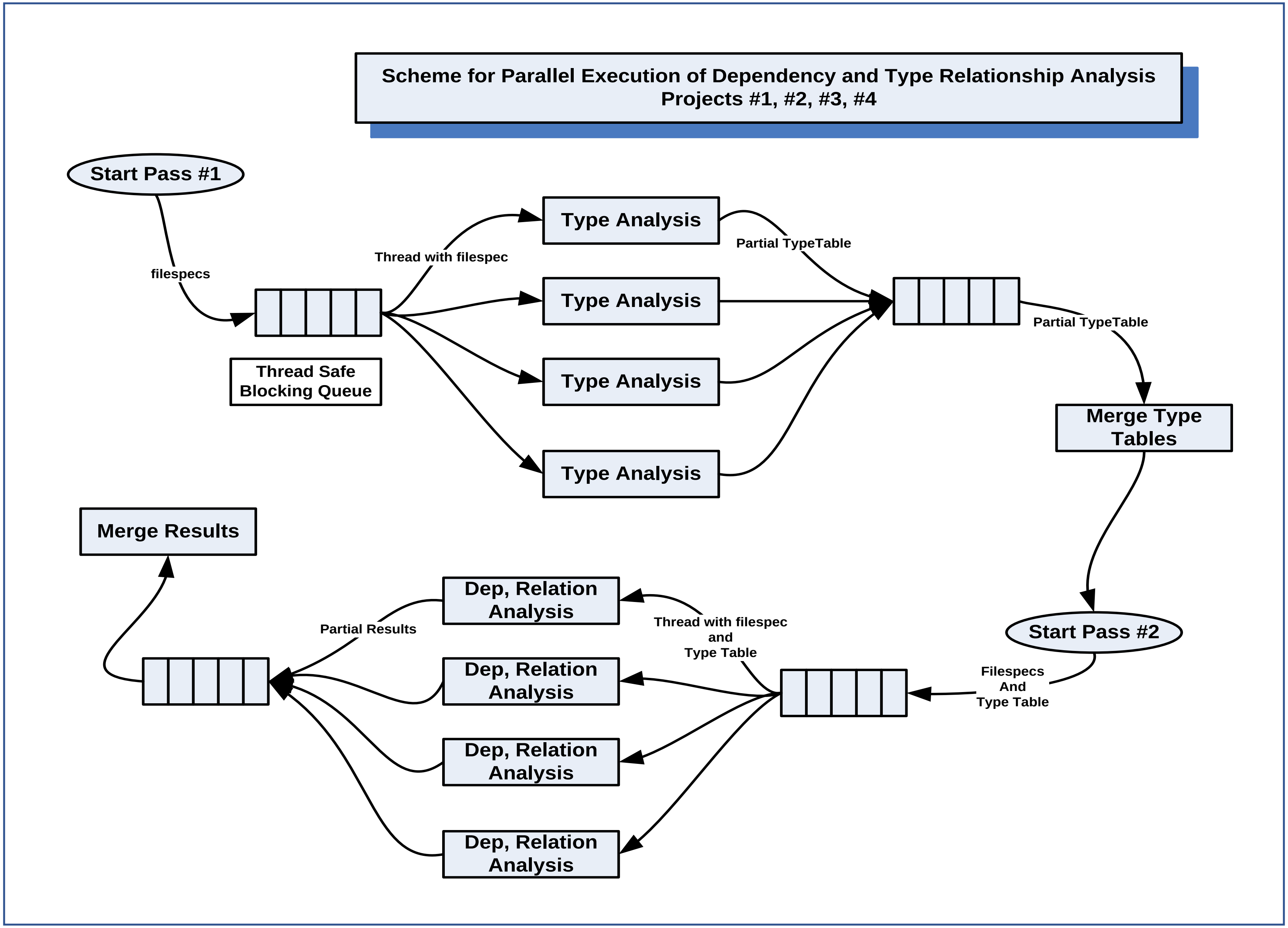

keyboard
mouse
other devices
Raw Input Queue
Window Manager
messages filtered for this window posted by operating system thread
Main thread in active window blocks on call to getMessage until a message arrives. Then it is dispatched to an event handler associated with that message
event handler function
Windows Message Queue
Active Window

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.

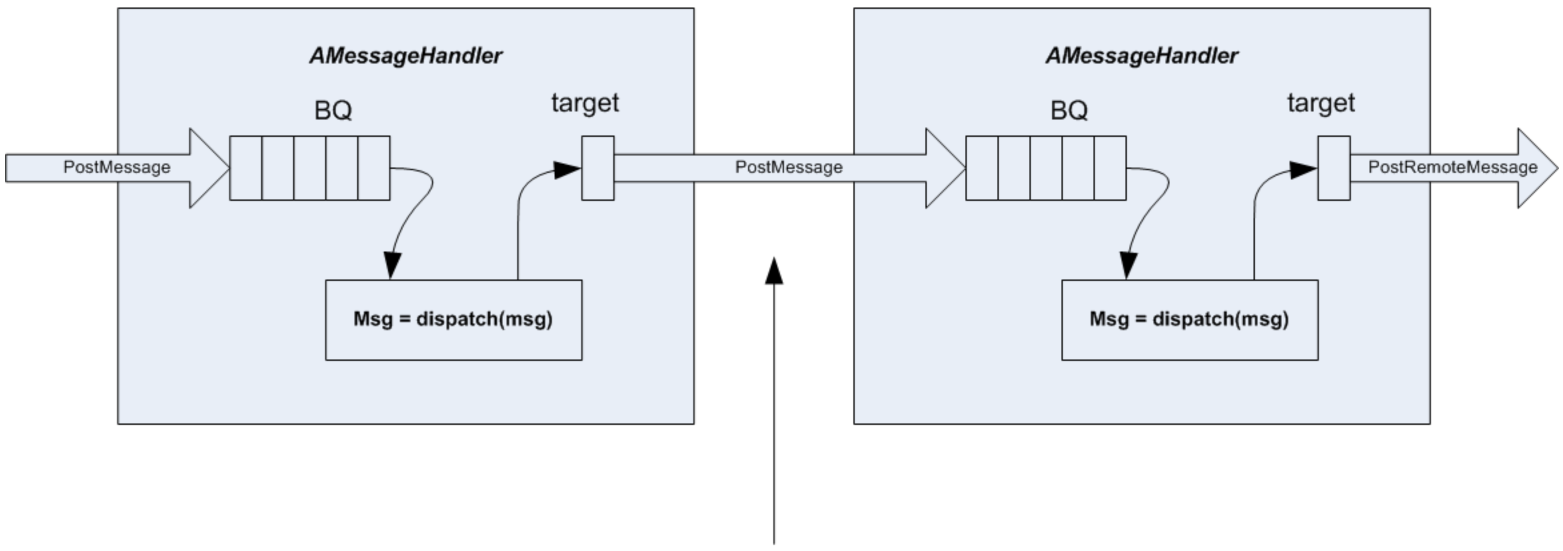


Thread-Driven Program Structure

Pipeline Execution

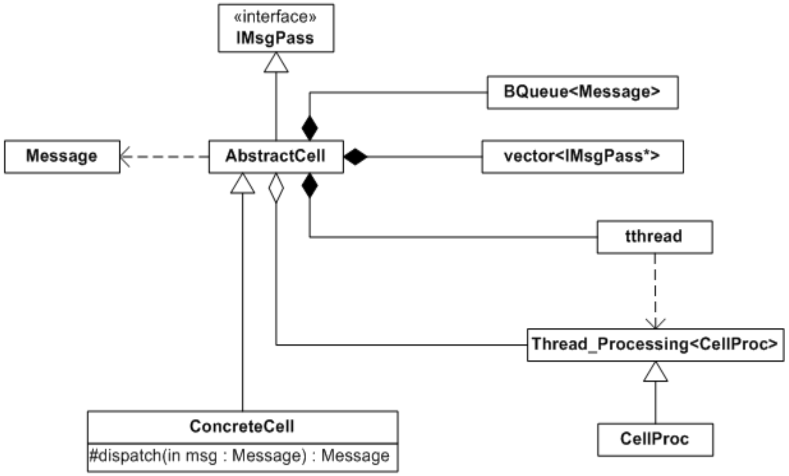
- Structure:
  - Composed of cells.
  - Each cell has a message queue and a child thread that processes messages.
  - Result messages may be sent on to another cell.
  - Each cell type is defined by the way it overrides a virtual message processing function.
- Example:
  - Project #4, CSE687 – OOD, Spring 2010

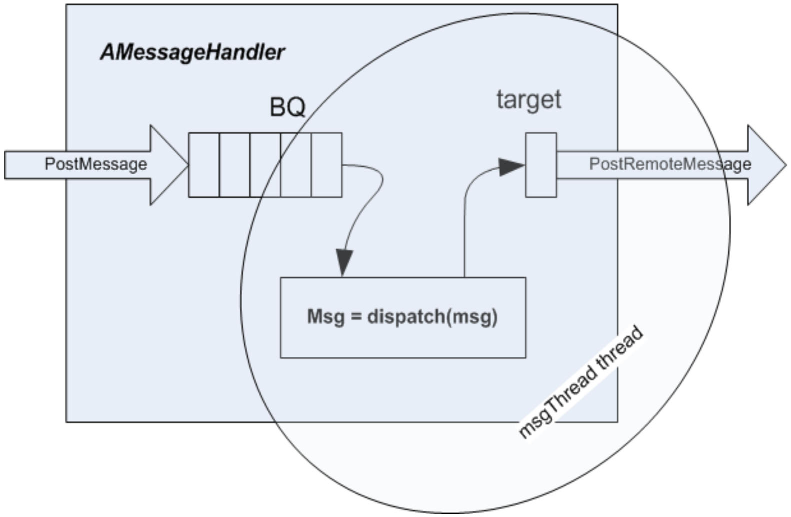
Cell Processing



Executive makes these connections

Cell Structure

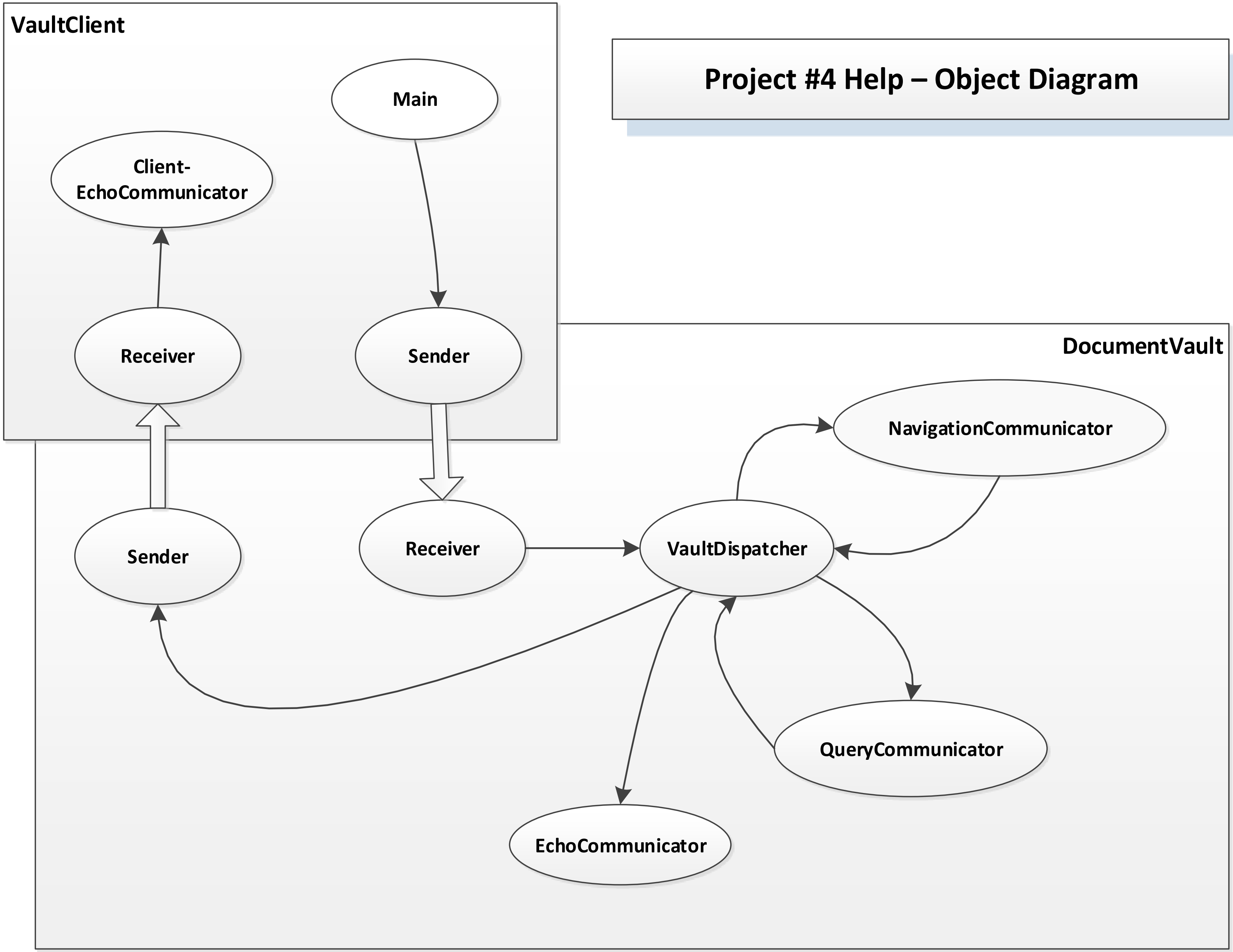




Pipe-lined Cell Communicators

- Document Vault (Project #4 – Fall 2013)
  - Uses pipe-lined cells as communicators
  - Mediator (dispatcher) controls routing of messages
  - Each cell has capability to send and receive messages
  - Makes very flexible configuration of client and server capabilities

Project #4 Help – Object Diagram



Enterprise Computing

Federations

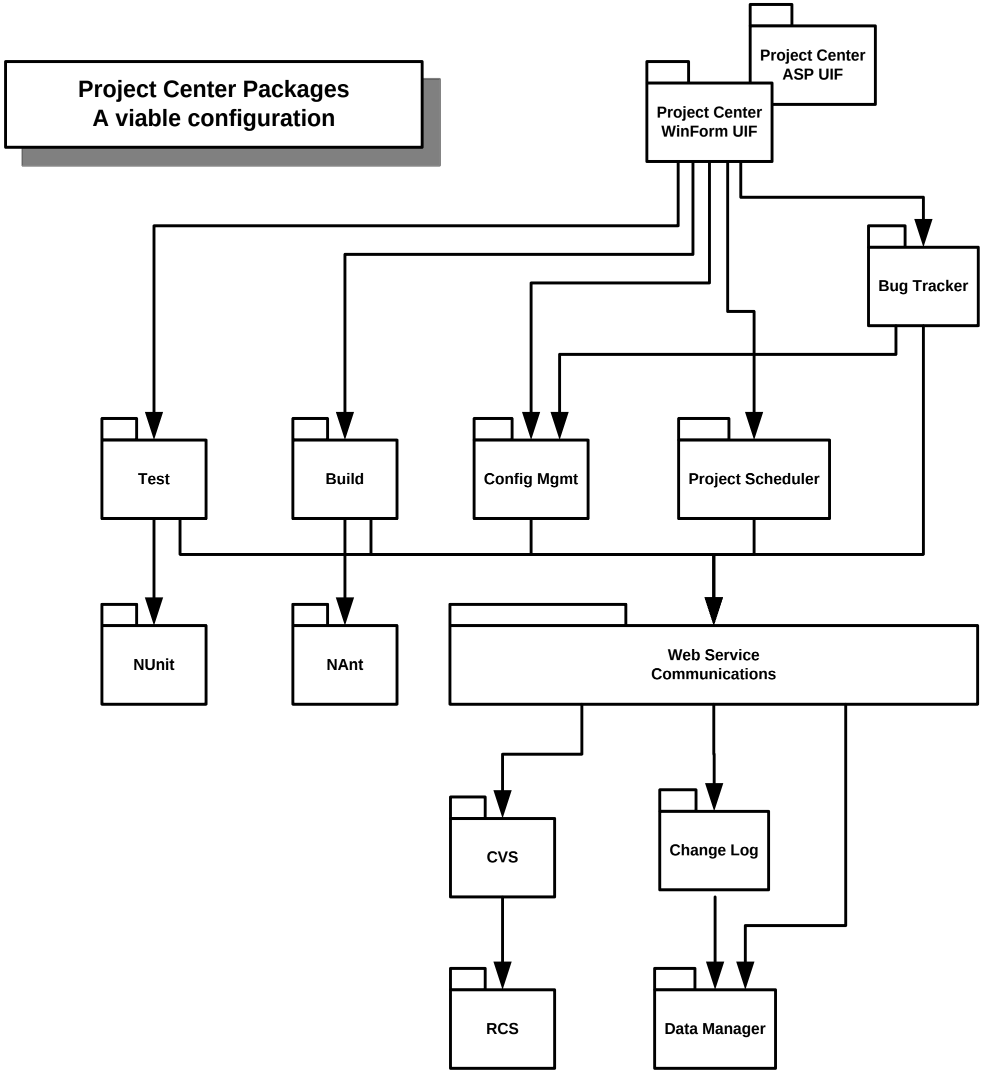
Collaboration Systems

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 S.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 managment
    - Nant – sofware 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

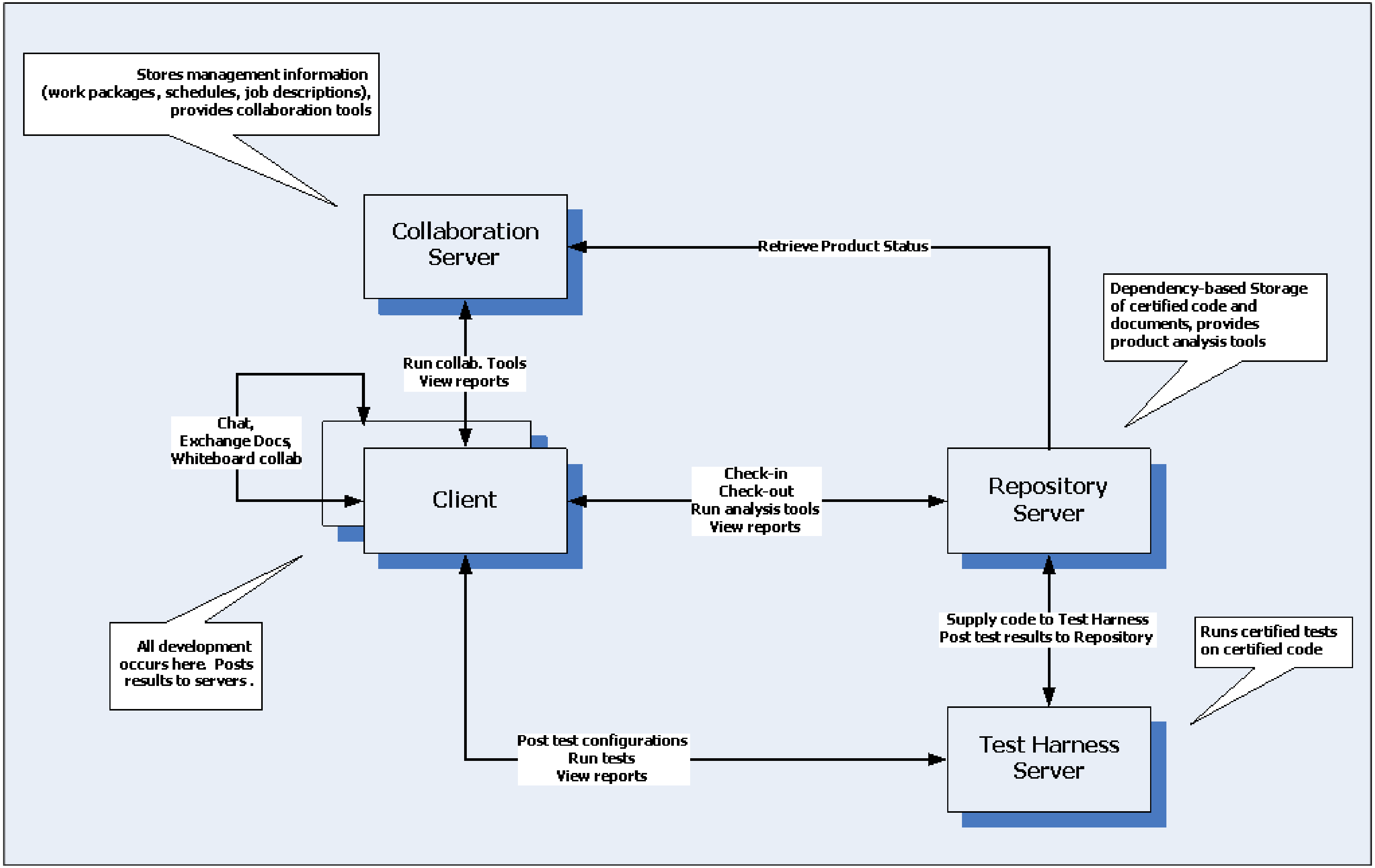
Federation Structure

- 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


Stores management information (work packages, schedules, job descriptions), provides collaboration tools
Collaboration Server
Retrieve Product Status
Dependency-based Storage of certified code and documents, provides product analysis tools
Run collab. Tools
View reports
Chat, Exchange Docs, Whiteboard collab
Client
Check-in
Check-out
Run analysis tools
View reports
Repository Server
All development occurs here. Posts results to servers.
Supply code to Test Harness
Post test results to Repository
Runs certified tests on certified code
Post test configurations
Run tests
View reports
Test Harness Server

Other System Structures

Agents

Cloud Computing

Agent-Based

- System uses Software Agents
  - Semi-autonomous, mobile, task oriented software entities. Crawl web, or network, or data structure
  - May be scheduled
  - Provide scriptable user specific services
    - Collect information from a large set of data
    - Perform analyses on changing baseline and report
    - Conduct specific tests
    - Make narrowly specified modifications to baseline
- Example:
  - CSE681 Project #5, summer 2009, http://www.ecs.syr.edu/faculty/fawcett/handouts/CSE681/Projects/Pr5Su09.doc

Master’s Thesis Research Examples

- The following are all based on Software Matrix structure – Autonomous cells often used with mediator
  - Software Matrix – Gosh, 2004
  - Self Healing Systems – Anirudha, 2005
  - Cross Platform Development – Appadurai, 2007
  - Model-Driven Development – Patel, 2007
- http://www.ecs.syr.edu/faculty/fawcett/handouts/webpages/research.htm

Other Structures

- TeraScale computing:
  - Term defined by Intel to describe parallel execution on a many core processor.
    - Expectations are chips with scores of processors
- Cloud Computing
  - Term adopted by many to describe remote execution and storage of applications. The cloud provides a stable endpoint that may map onto any one of a large set of computing resources.
  - Example:
    - Microsoft’s Azure platform
    - Amazon Web Services
    - Google Cloud

Other Structures we won’t discuss

- GPU computing
- Neural Networks
- Baysian Networks
- Deep Learning Networks
- Adversarial Networks

SMA Projects - 2015

- Project #2 – Fall 2015
  - NoSql Database
    - Key/Value store
    - Provides cloning, persistence, querying, views
- Project #4 – Fall 2015
  - Client-Server
    - Focus on NoSqlDb performance testing
    - May have multiple concurrent clients
    - Both client and server may use DLLs for significant processing
- Project #5 – Fall 2015
  - Federation of clients and servers
    - Focuses on data service layer
    - May have a dedicated virtual server with child services on each of the Federation servers

SMA Projects – Before 2015

- Project #2 – Fall 2013
    - Cooperating monolithic processes
        - Composite Text analyzer
        - Metadata generator
- Project #4 – Fall 2014
    - Client-Server
        - May have multiple concurrent clients
        - Both client and server use DLLs for significant processing
- Project #5 – Fall 2013
    - Federation of clients and servers
        - Focuses on Software Repository server
        - May wish to use virtual servers

Virtual Server

- Clonable Server
  - Create an instance of some running server on my desktop
  - Clone some part of it’s data store
- Examples – Originals hosted by development project
  - Repository – holds my team’s code resources
  - TestHarness – used to test locally before checking in to Project
  - Collab – holds my team’s work plans, status information
    - Provides whiteboard with webcam and document views to collaborate with remote team.
  - Clients – Enables access and use of the other parts

The End