

# Achieving Academic Success Using TSP

Berin Babcock-McConnell  
Saurabh Gupta  
Jonathan Hartje  
Marsha Pomeroy-Huff  
Shigeru Sasao  
Sidharth Surana



# Agenda

- Introduction
- The Project
- Spring Semester
- Summer Semester
- Conclusion



# 1. Introduction

- A student group in the Master of Software Engineering program at Carnegie Mellon University
- Tasked to build software to autonomously control a robot for a real-world industry project
- The team was having difficulty creating a project plan which could effectively track their progress
- The team decided to try TSP, and this is the story of their success...



## 2. The Project

# What is the MSE program?

- The Master of Software Engineering (MSE) degree is a 16-month graduate program offered at Carnegie Mellon University.
- Five core courses
  - Models of Software Systems
  - Methods: Deciding What to Design
  - Managing Software Development
  - Analysis of Software Artifacts
  - Architectures of Software Systems
- Electives
- Studio project

CarnegieMellon





# What is the Studio project?

- Actual industrial software engineering project provided by corporate sponsors
- Runs continuously throughout the duration of the MSE program
- Supportive environment to practice software engineering craft
- Cornerstone of the MSE program

# Studio Project Timeline

Fall 08

Establish Project Scope/Requirements

Spring 09

Architecture

Summer 09

Implementation

# Team VdashNeg

- Berin Babcock-McConnell
- Saurabh Gupta
- Jonathan Hartje
- Shigeru Sasao
- Sidharth Surana





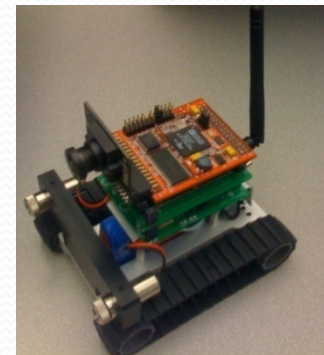
# The Mentors

- Grace Lewis
- Marsha Pomeroy-Huff
  - Certified TSP Coach



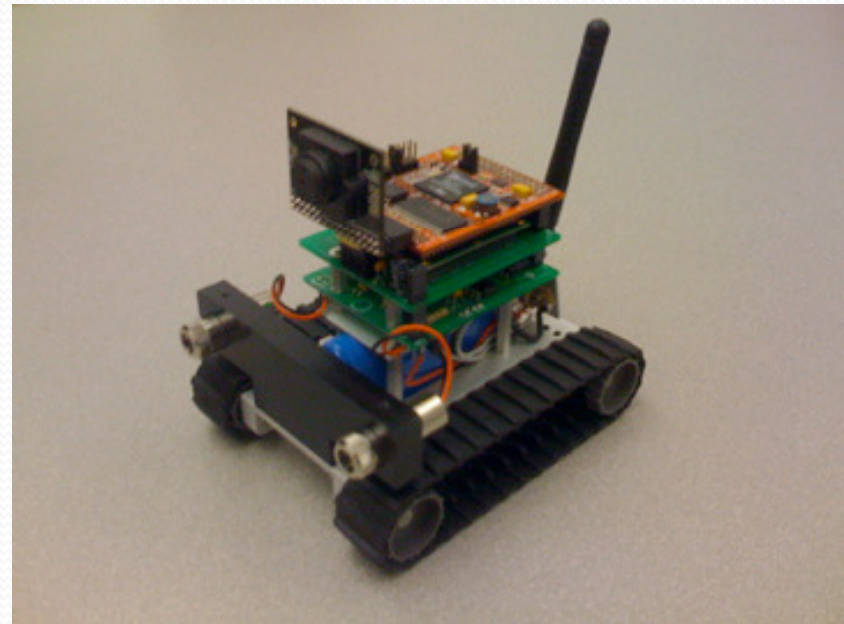
# The Project

- Use PACC Starter Kit to create software that controls an SRV-1 robot
- The mission: search and destroy while following a laid out path
- The software must be analyzable for performance and behavior
- Academic or industrial example of successful PACC utilization for system development



# SRV-1 Surveyor Robot

- 500MHz Analog Devices Blackfin processor (BF537)
- Omnivision (OV9655) 1.3 Megapixel digital camera
- 2 laser pointers for ranging
- Controlled via 802.11G wireless ethernet





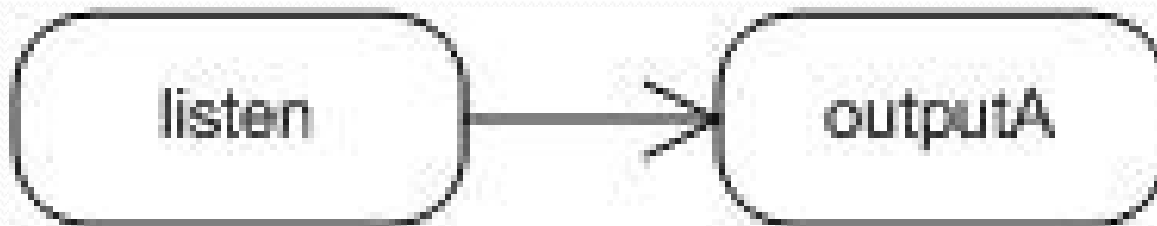
# PACC

- Predictable Assembly from Certifiable Components
- PACC Starter Kit (PSK) – developed by the SEI
- PSK is a reference implementation designed to illustrate “*predictability by construction*” (PbC)
- Power of analysis through formally defining states and architectural constructs within the software

# CCL

- Represents the software in the form of state charts

```
listen -> outputA {  
    trigger ^inA;  
    guard A_selected;  
    action {  
        ^out(myCh, inA.samples);  
    }  
}
```



# CCL cont'd

- Defines the architecture of the system in the software

```
// sending message to robot  
keyboard:keyed ~> stringToBytes:in;  
stringToBytes:out ~> bytesToNetBytes:in;  
bytesToNetBytes:out ~> gatewaySend:netBytes_in;  
gatewaySend:send_robot ~> netGateway:netWrite;
```





# Reasoning Frameworks

- CCL supports syntactic annotations for static analysis:
  - Performance analysis based on Generalized Rate Monotonic Analysis (GRMA)
    - Aperiodic tasks
    - Preemption by priority
  - Behavior analysis
    - Model checking using Linear Temporal Logic



### 3. Why we used TSP





# Problems We Encountered

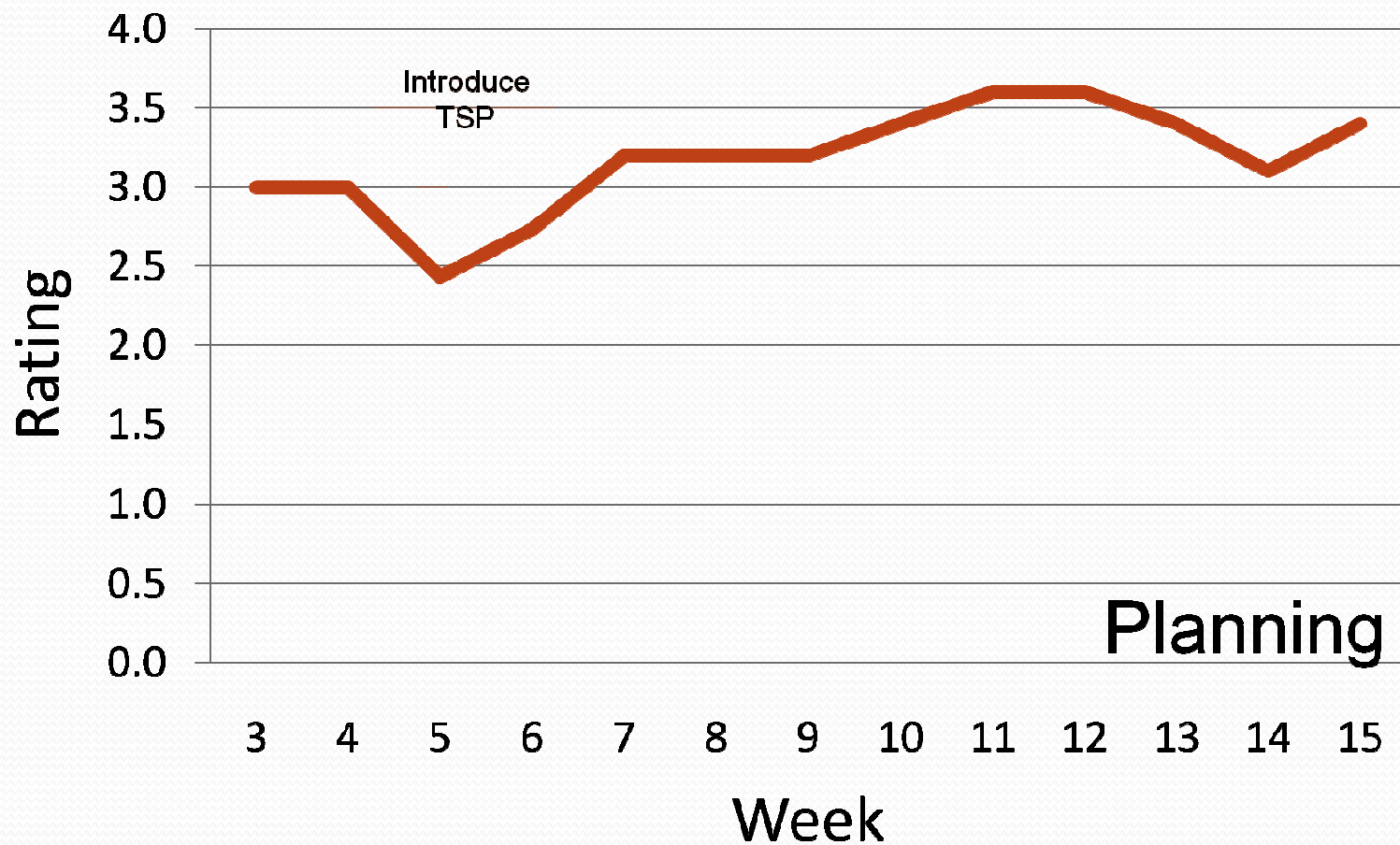
- Planning and Tracking
  - Inability to map team goals and milestones to tasks
  - Granularity of tasks
- Incomplete Software Process
  - We were using the Architecture-Centric Design Methodology (ACDM), but this is only for design
  - Team selected different techniques learned from the Management of Software Development course
  - The techniques were not cohesive
- So, we decided to try TSP.



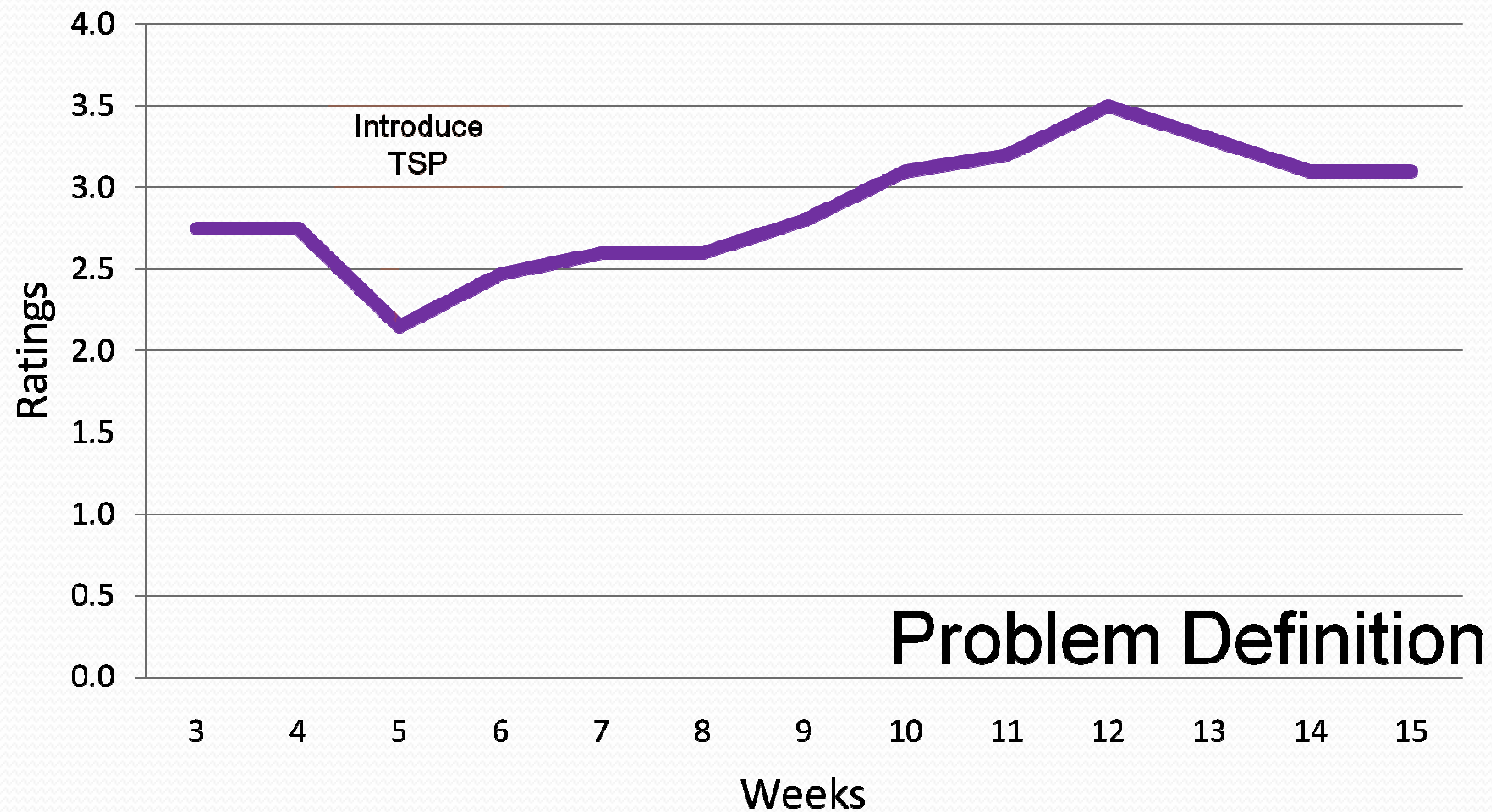
# The Benefits of Using TSP

- Risk Management
  - Organization
  - Planning and Tracking
  - Quality Control
  - Weekly Meetings
- 
- TSP provided a cohesive package, which showed how the multiple techniques fit together.

# Process Review (Planning)



# Process Review (Problem Definition)





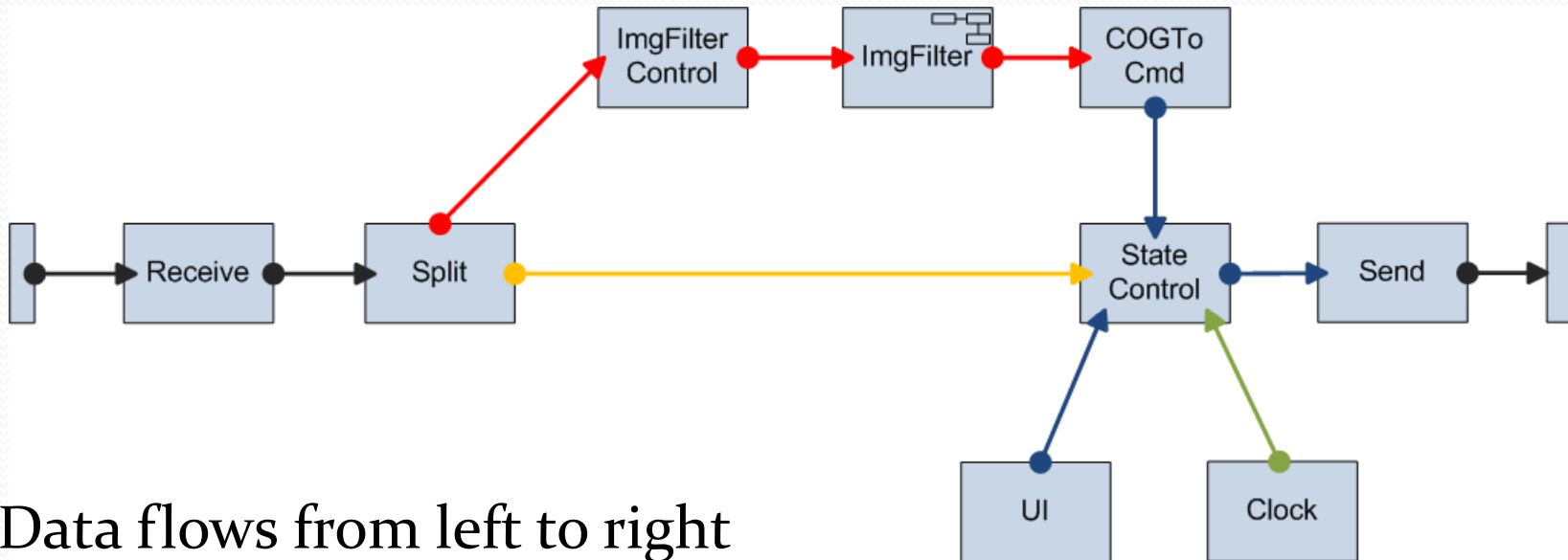
## 4. Spring Semester



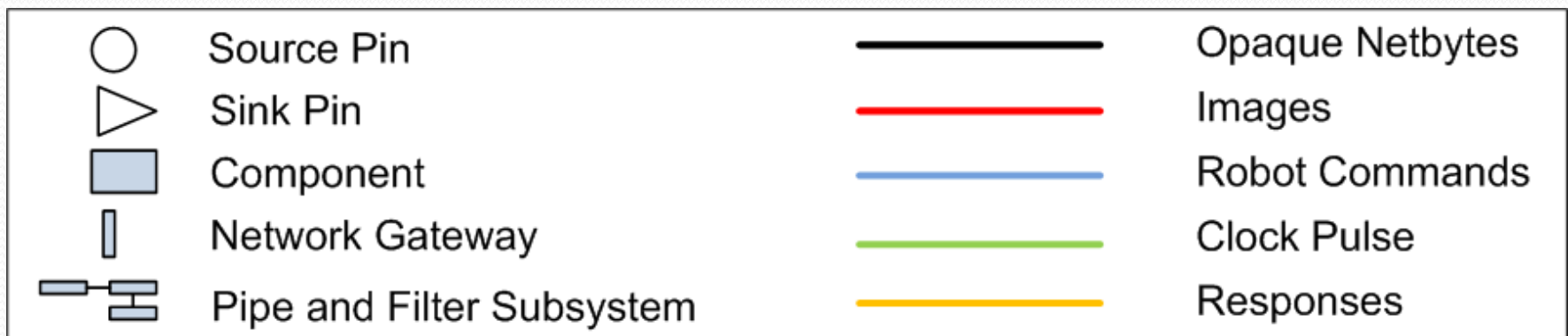
# Focus for the Spring

- System architecture
- Experimenting with the Technologies
  - Physical measurements w/ SRV-1
  - Reasoning framework annotations in CCL
  - Image processing experiments
- Predictability scenarios

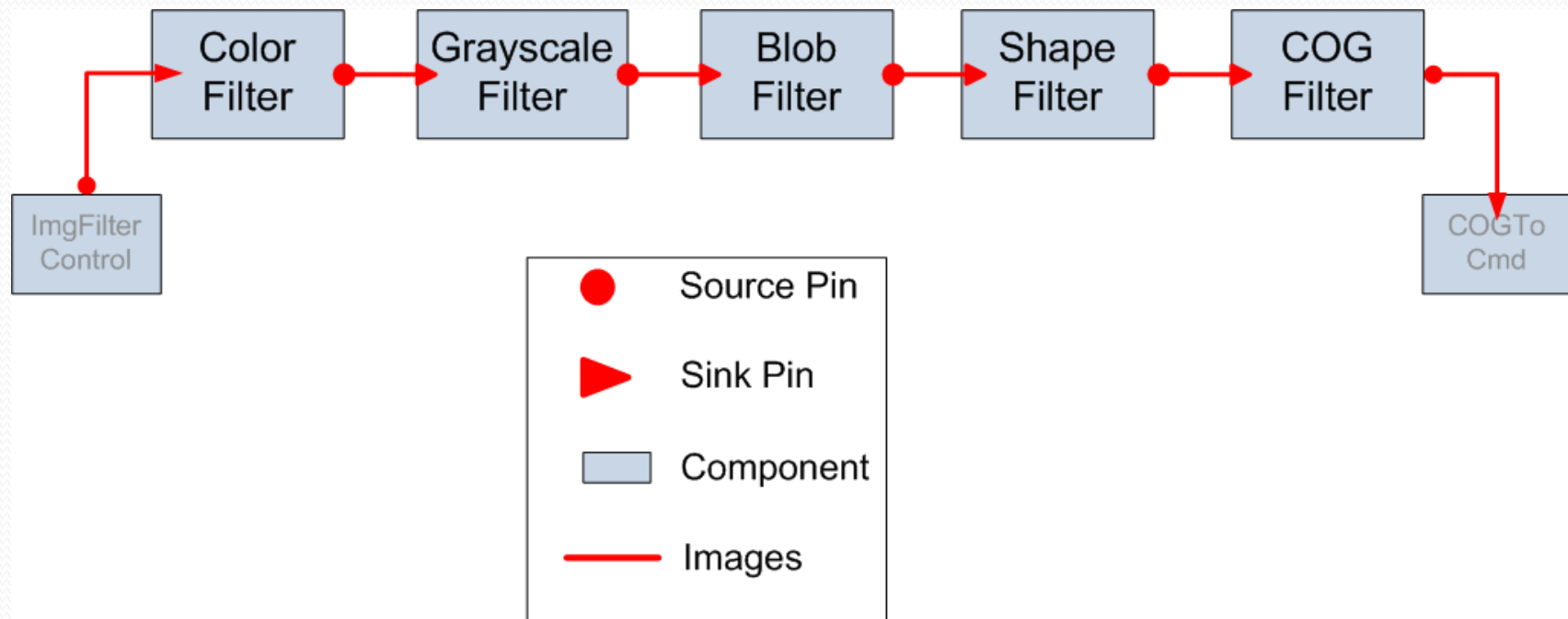
# Architecture (Dynamic View)



Data flows from left to right



# Image Filter Expanded



Data flows from left to right



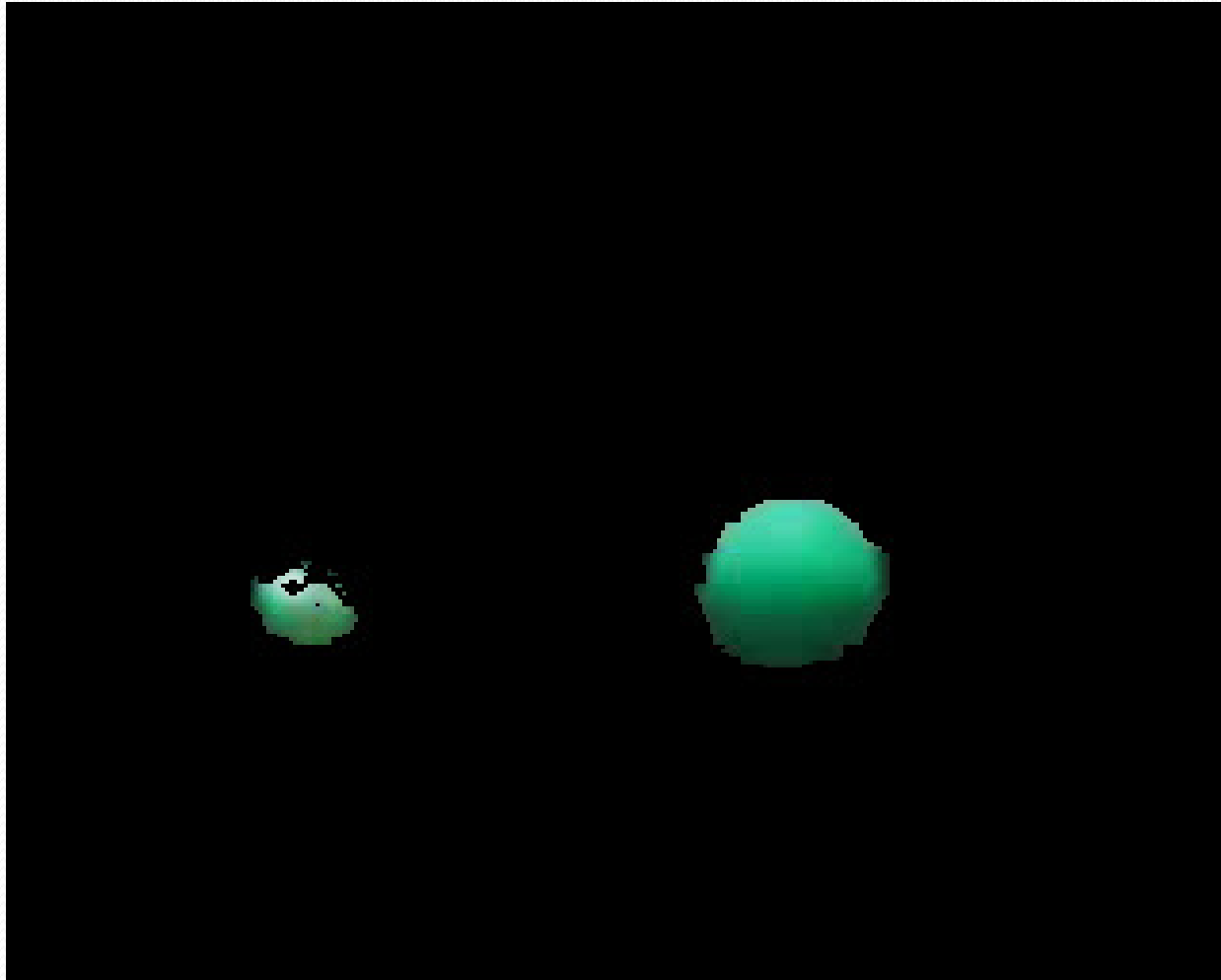
# Image Filters in Action...

World from the SRV-1 eye



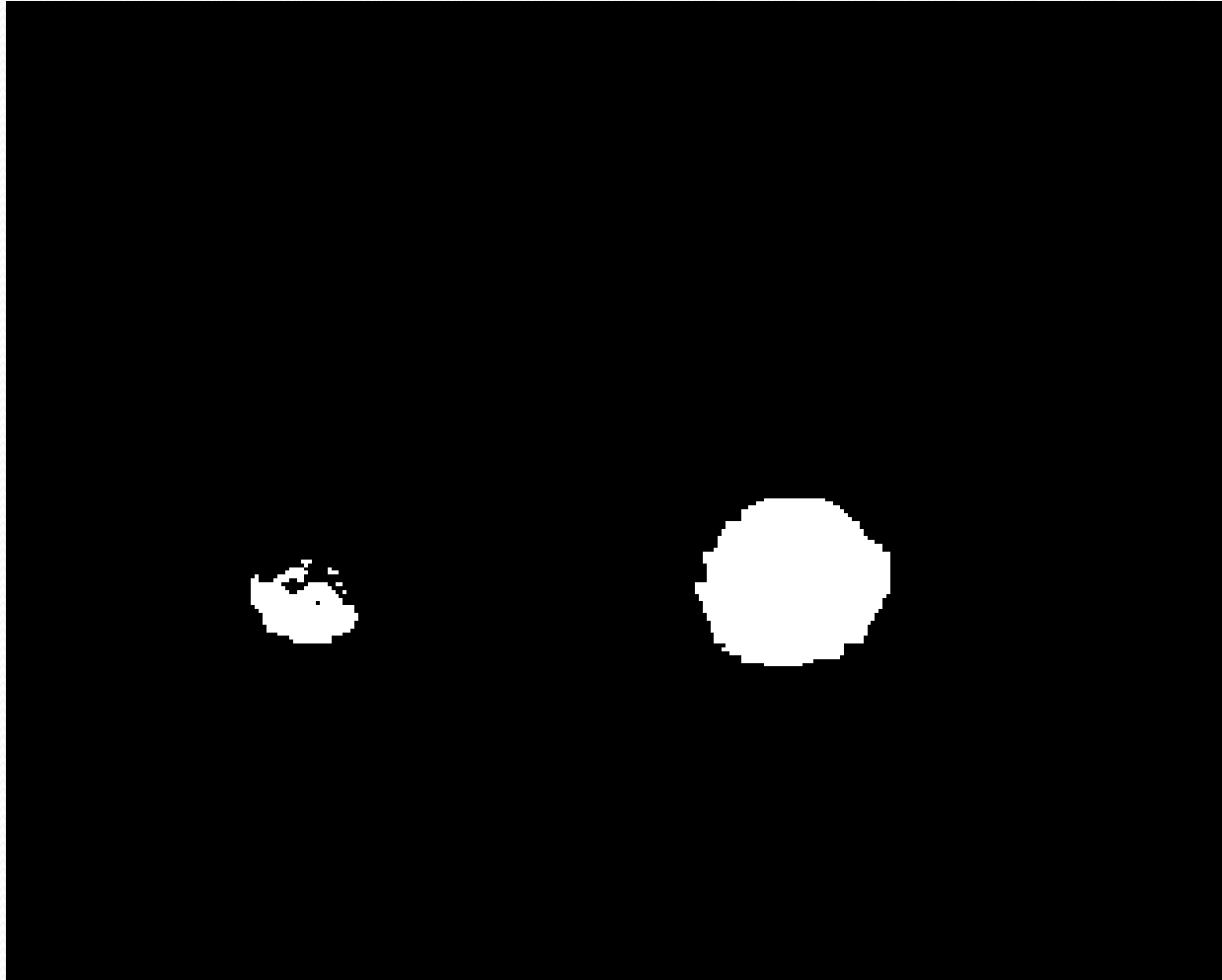
# Image Filters in Action...

Robot Eye → ColorFilter



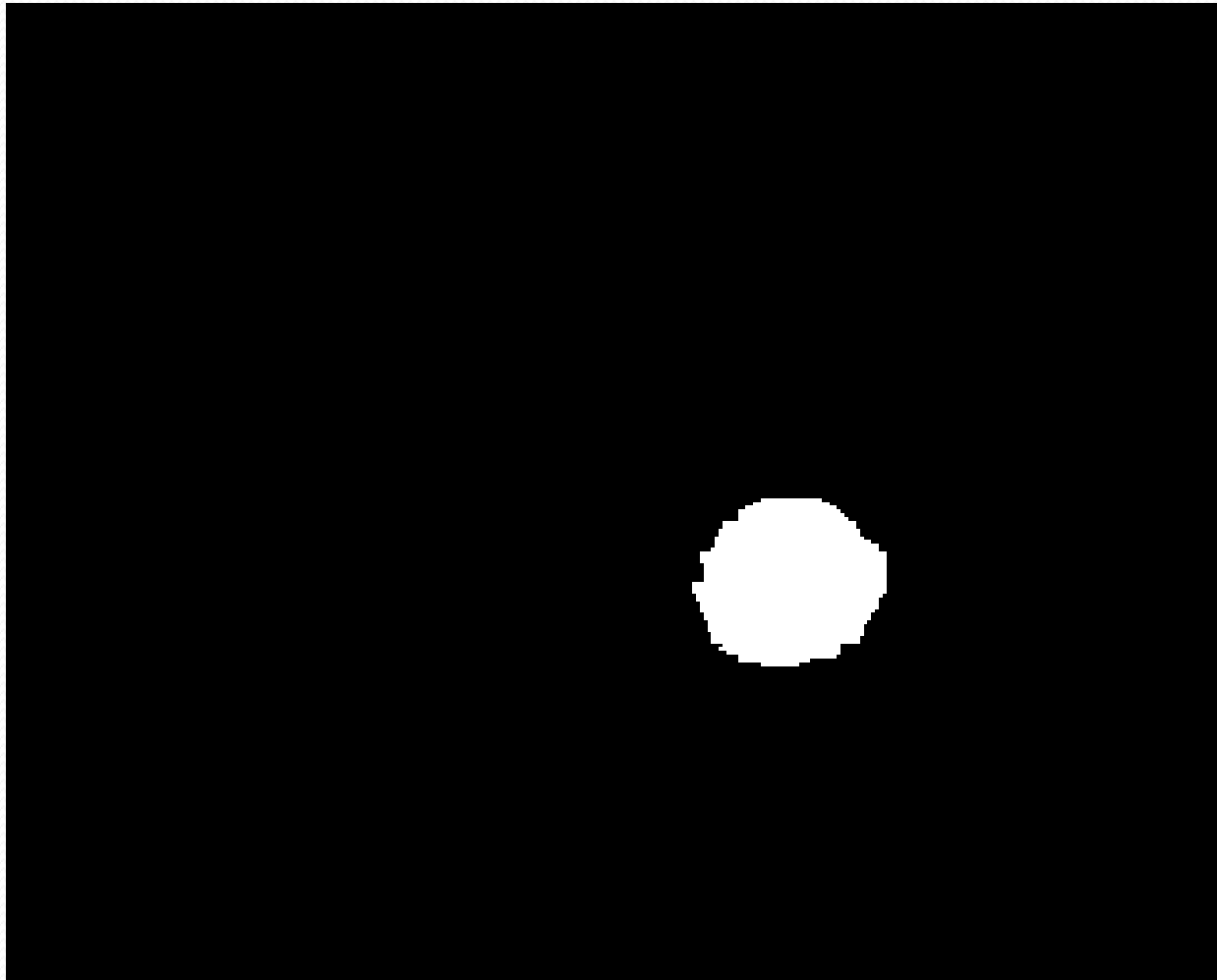
# Image Filters in Action...

Robot Eye → ColorFilter → GrayscaleFilter



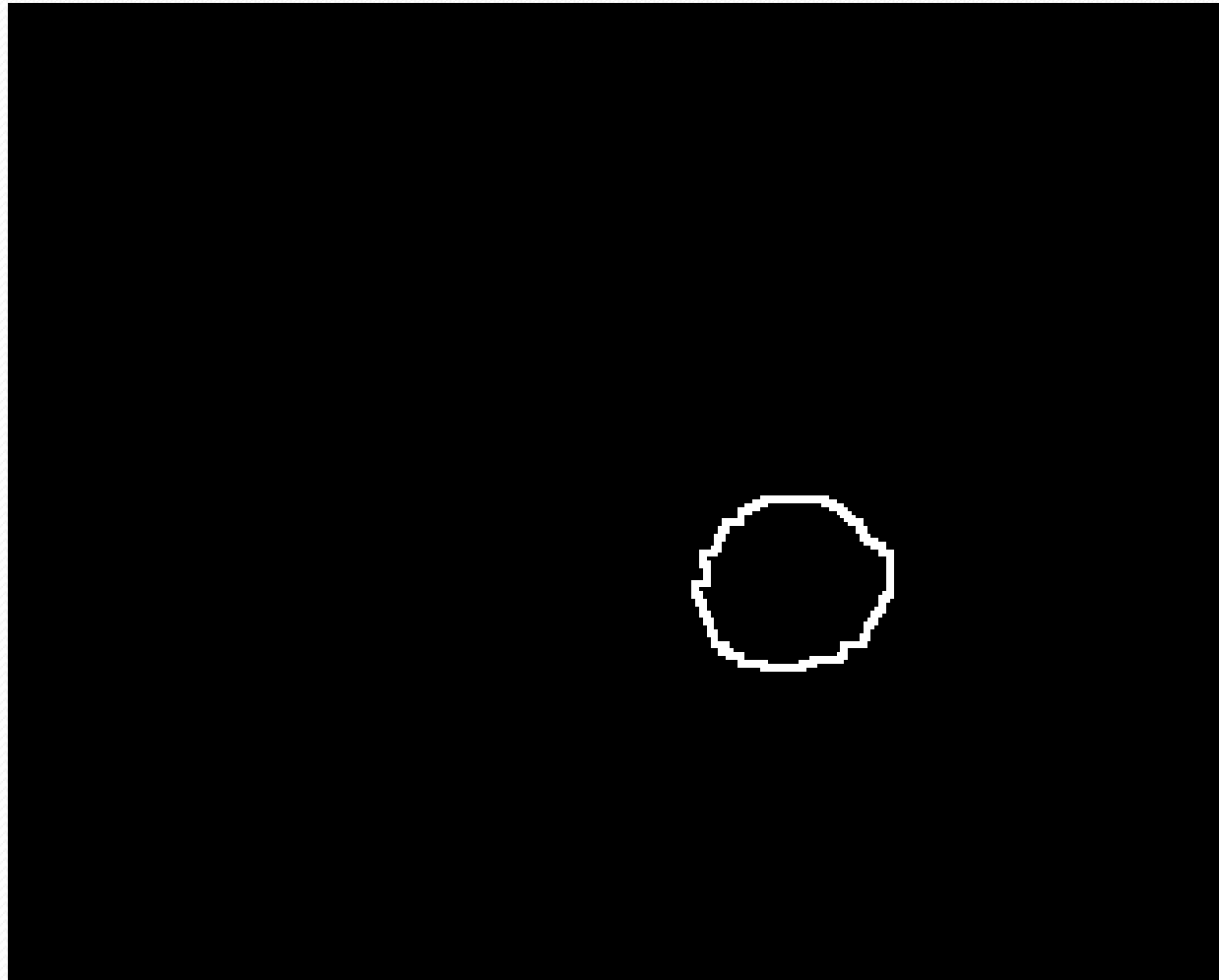
# Image Filters in Action...

Robot Eye → ColorFilter → GrayscaleFilter → BlobFilter



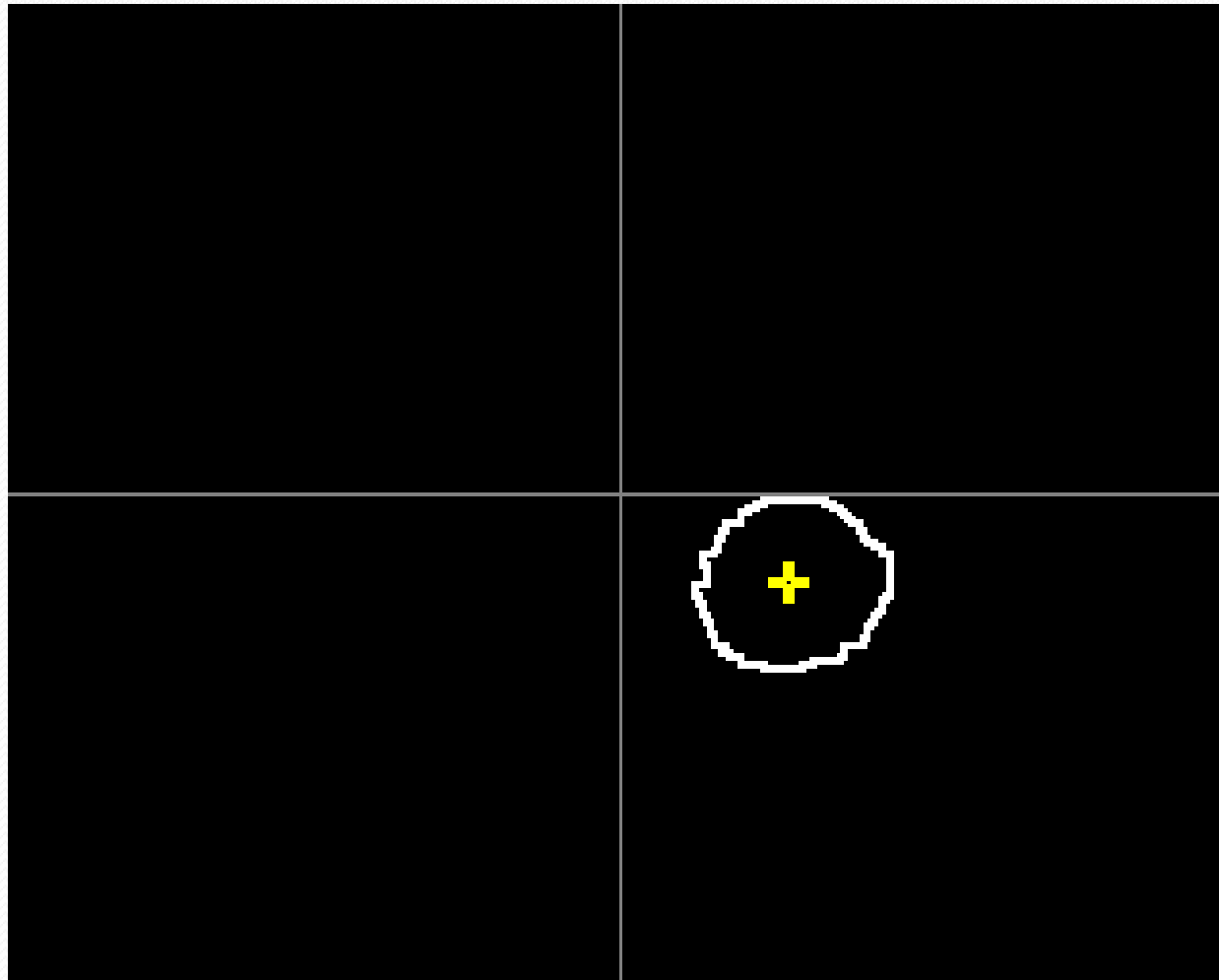
# Image Filters in Action...

Robot Eye → ColorFilter → GrayscaleFilter → BlobFilter → ShapeFilter



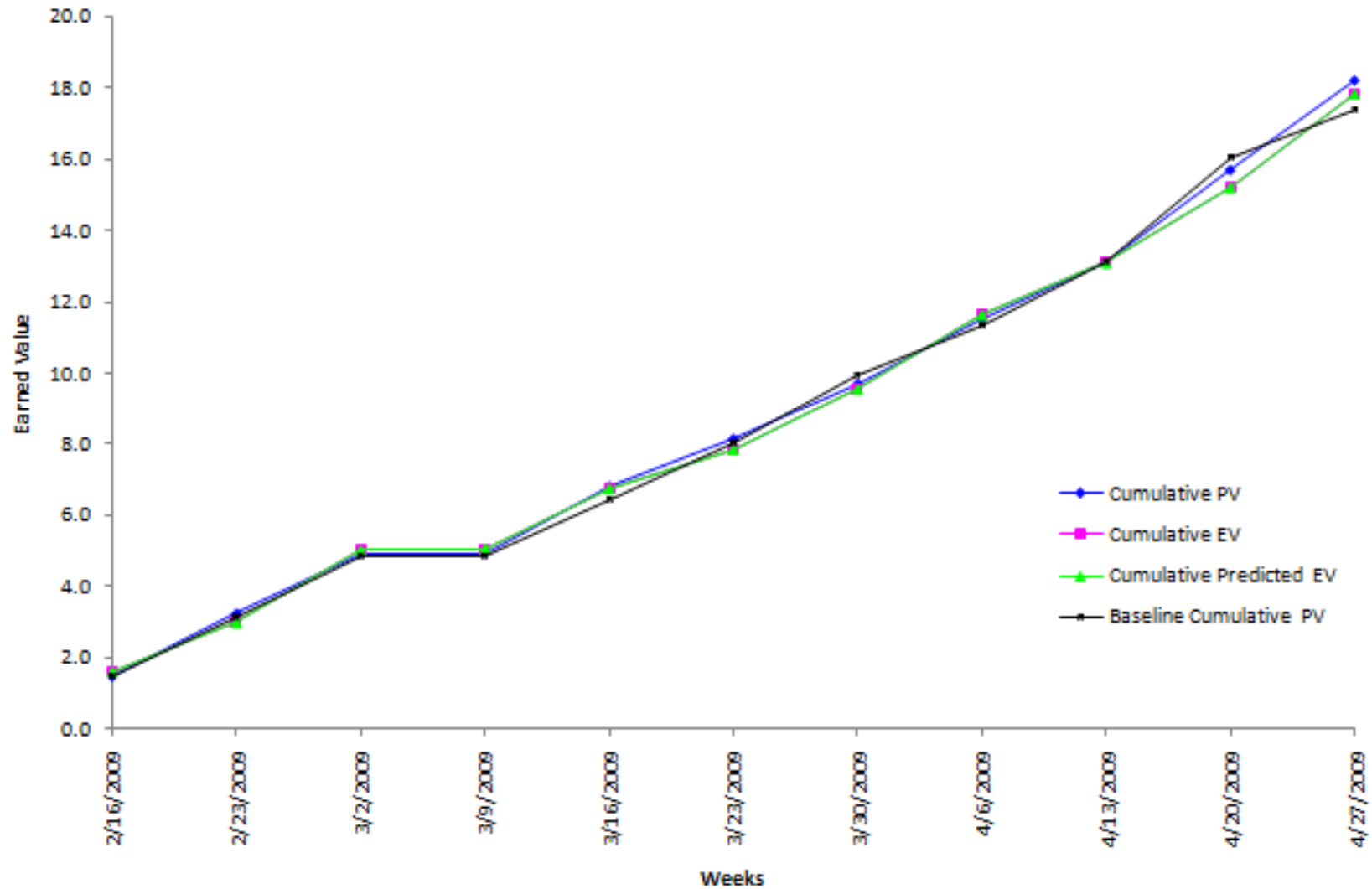
# Image Filters in Action...

Robot Eye → ColorFilter → GrayscaleFilter → BlobFilter → ShapeFilter → COGFilter



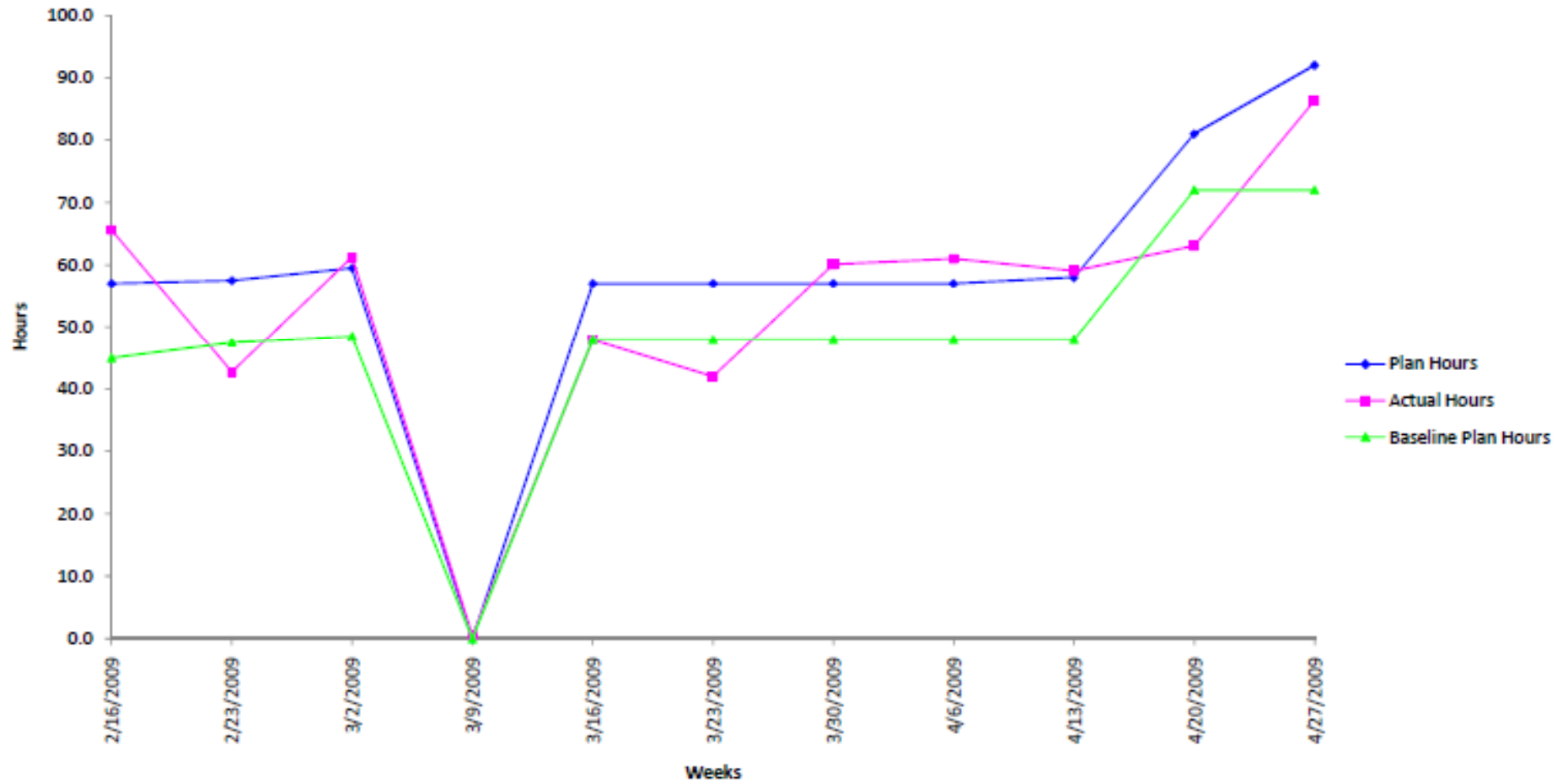
Spring 2009

### Cumulative Earned Value



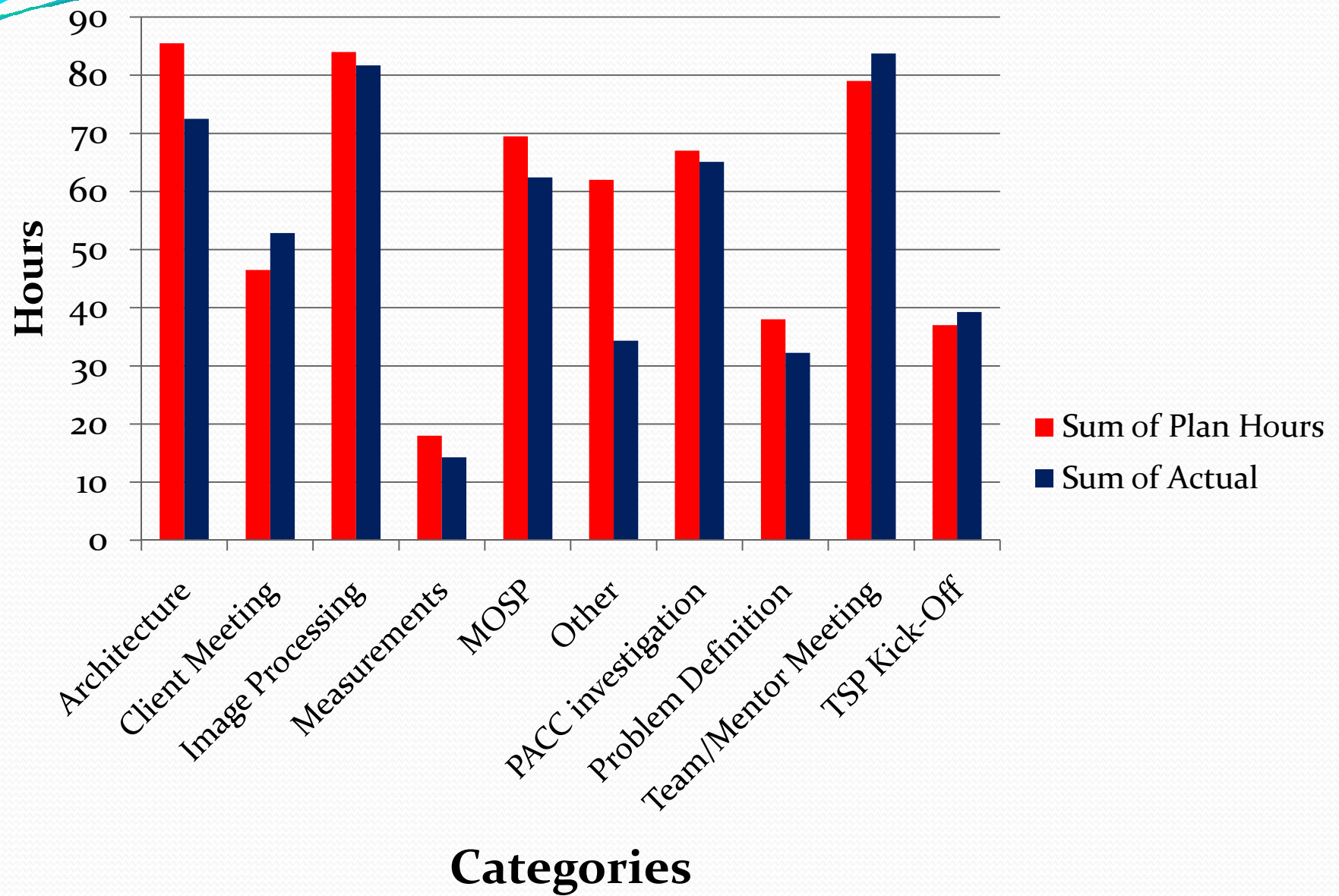
Spring 2009

Planned and Actual Hours per Week





## Actual vs Planned Hours





# 5. Summer Semester



# Focus for the Summer Semester

- **Iteration 1 (5/18 - 6/7)**
  - Support libraries
  - Finalize predictability scenarios and artifact updates
- **Iteration 2 (6/8 - 6/28)**
  - Image filter components
  - Complete base system with basic state control
- **Iteration 3 (6/29 - 7/19)**
  - Complete final state control implementation
  - Finalize test cases for system verification
- **Iteration 4 (7/20-8/7)**
  - Final code freeze. Focus remaining efforts on critical fixes
  - Deliver final system to clients and execute D-Day test plan

# The Matrix

	Component	DLD	DR	DINSP	CODE	CR	CINSP	UT
	Initial Main	sid			sid			bb
	NetBytes ToBytes	shig	jh		shig	bb		sid
	Bytes ToString	shig	sid		shig	bb		jh
	Send	sid	sg		sid	bb		jh

# The Matrix

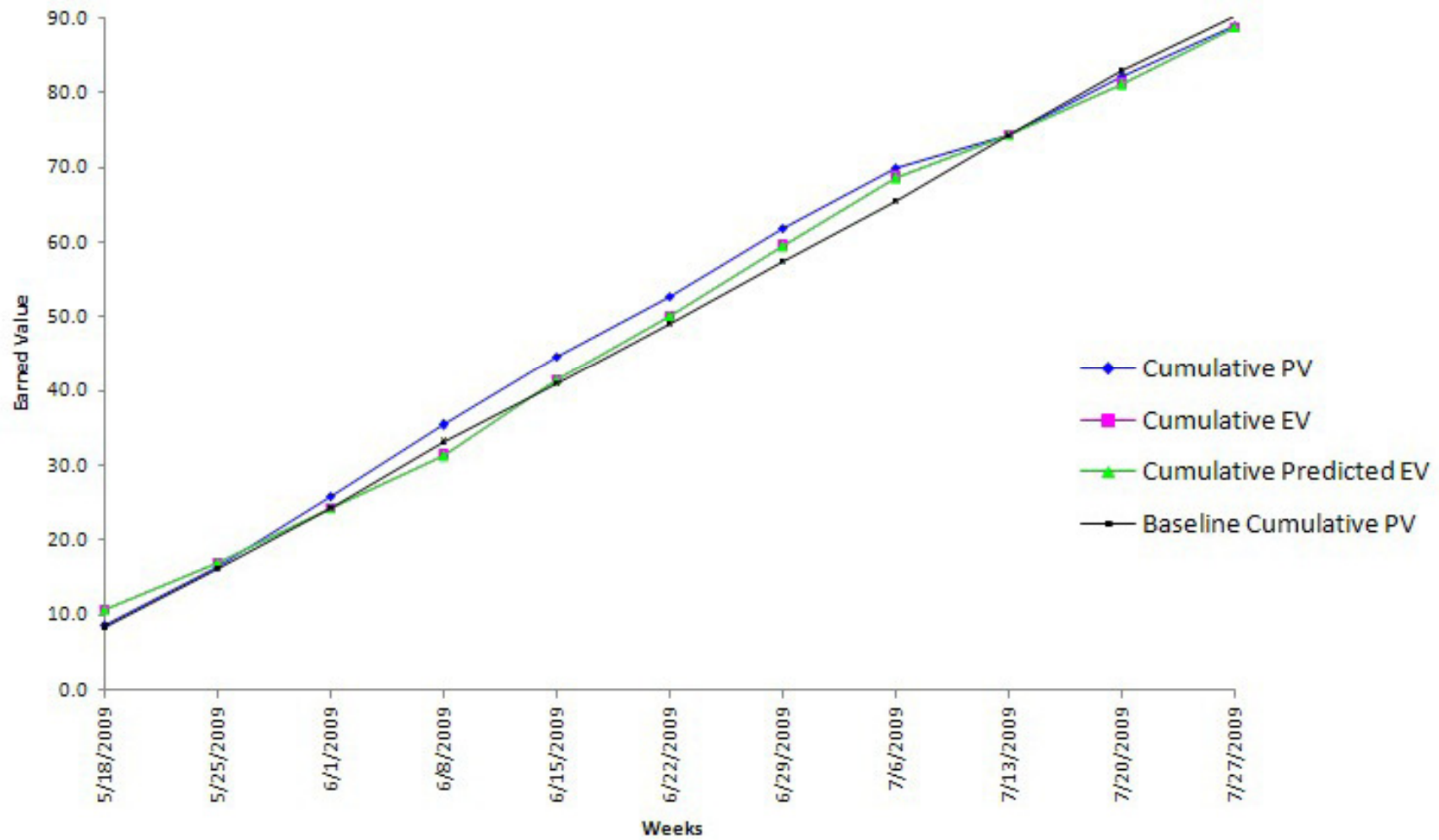
	Component	DLD	DR	DINSP	CODE	CR	CINSP	UT
	Initial Main	sid			sid			bb
	NetBytes ToBytes	shig	jh		shig	bb		sid
	Bytes ToString	shig	sid		shig	bb		jh
	Send	sid	sg		sid	bb		jh

# The Matrix

	Component	DLD	DR	DINSP	CODE	CR	CINSP	UT
	Initial Main	sid			sid			bb
	NetBytes ToBytes	shig	jh		shig	bb		sid
	Bytes ToString	shig	sid		shig	bb		jh
	Send	sid	sg		sid	bb		jh

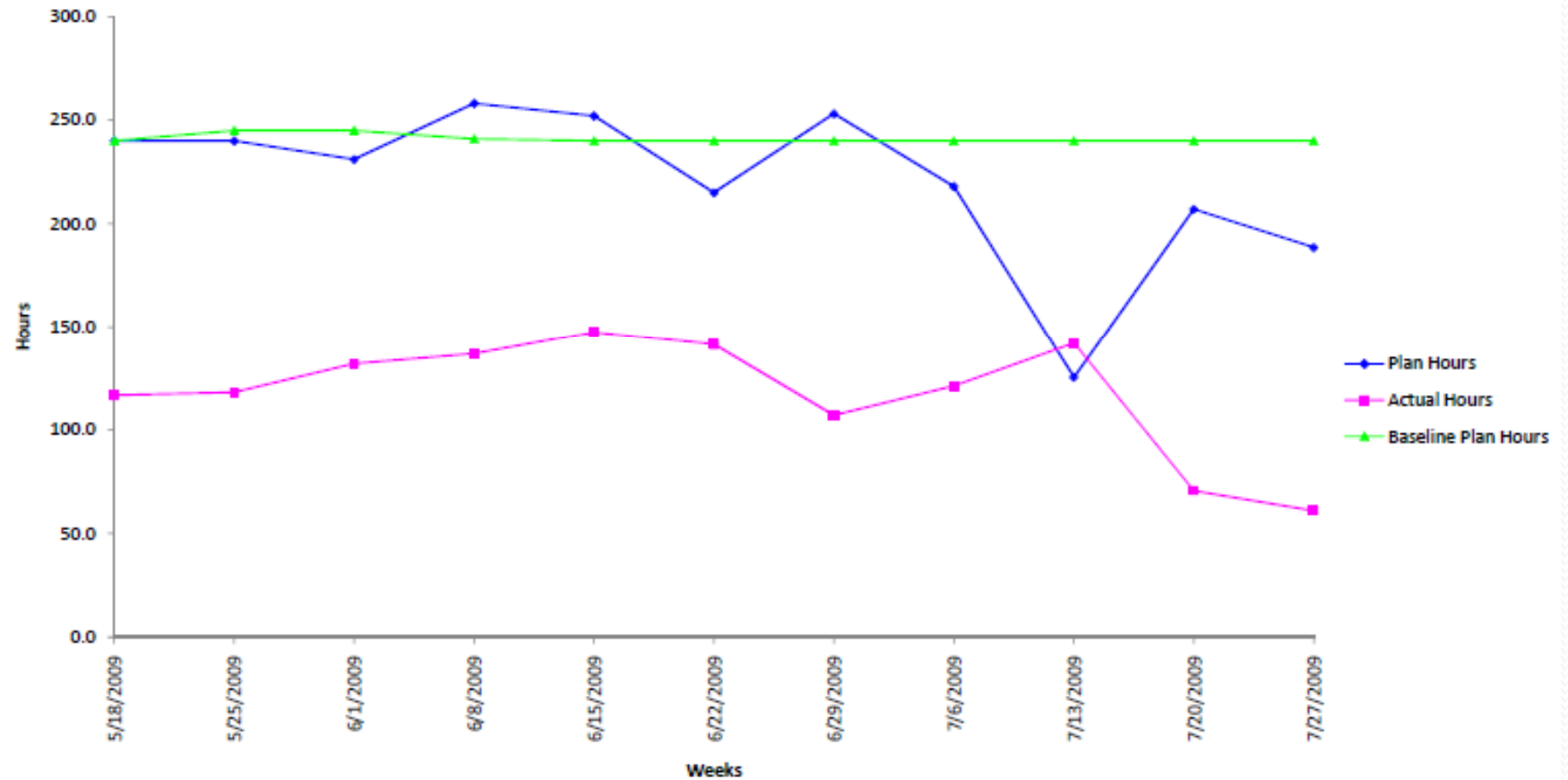
Summer 2009

Cumulative Earned Value



## Summer 2009

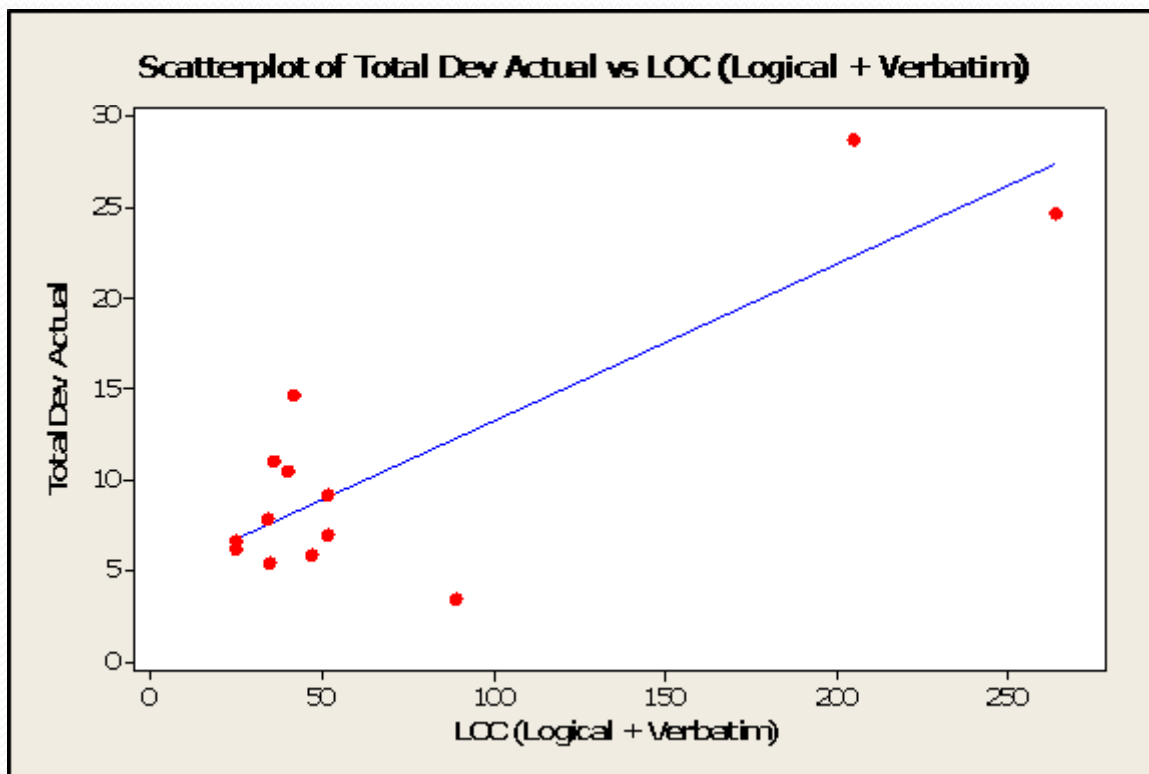
Planned and Actual Hours per Week





# Improving Our Estimates

- In iteration 1 & 2, the team overestimated by over 110%
- Used data from iteration 1 & 2 to construct a parametric model



$$F(y) = 3.49 + 0.0387x$$
$$R^2 = 80\%$$

# Improving Our Estimates

Actual

COG To Cmd	34.8
UI	13
State Control	83.9
Main	13.6

Ad-hoc

MRE		
COG To Cmd	51.15	0.469828
UI	20	0.538462
State Control	135	0.609058
Main	20	0.470588

**MMRE**

**52.20%**

PROBE

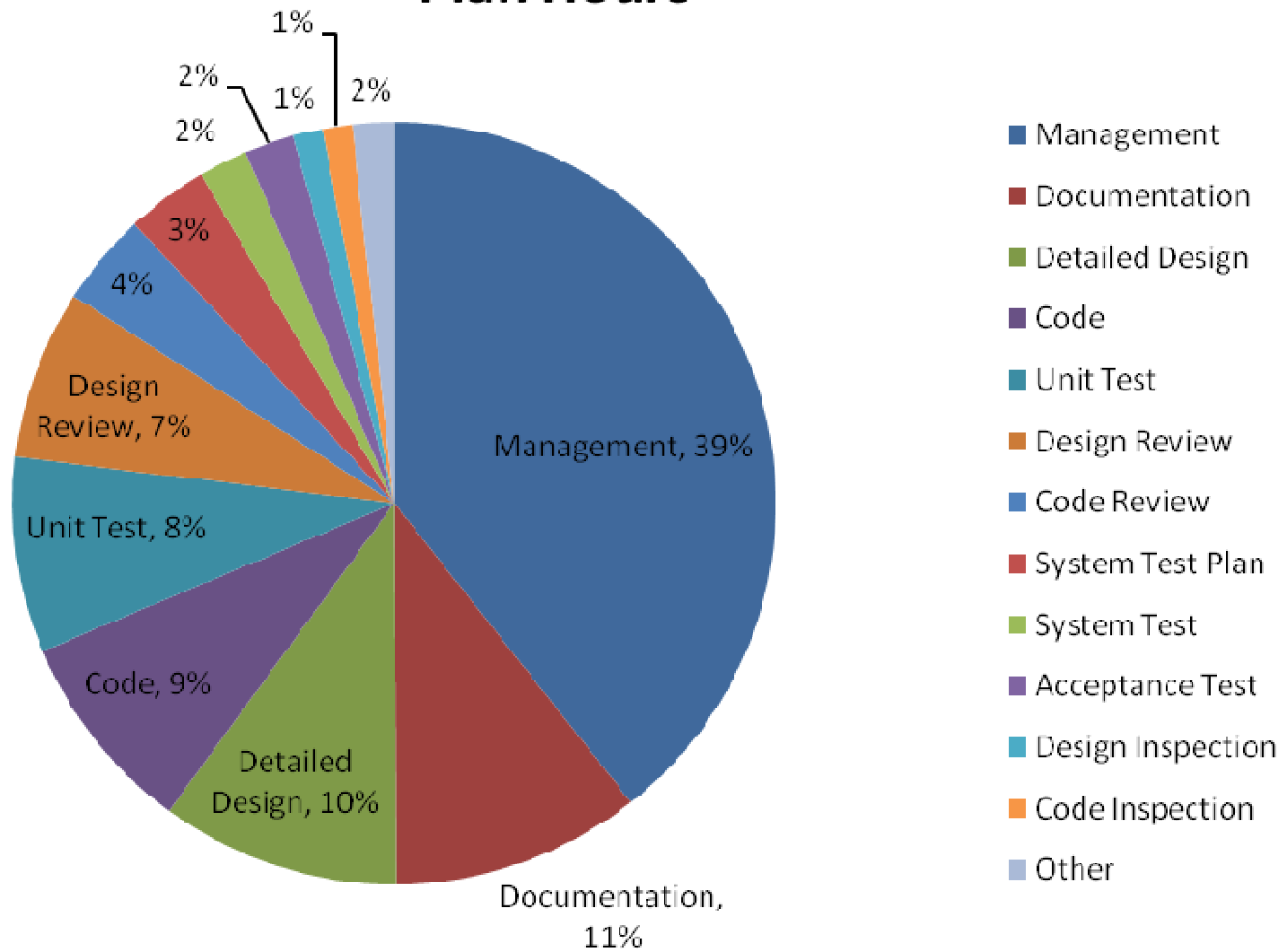
MRE		
COG To Cmd	18.97	0.454885
UI	15.1	0.161538
State Control	80.89	0.035876
Main	15.1	0.110294

**MMRE**

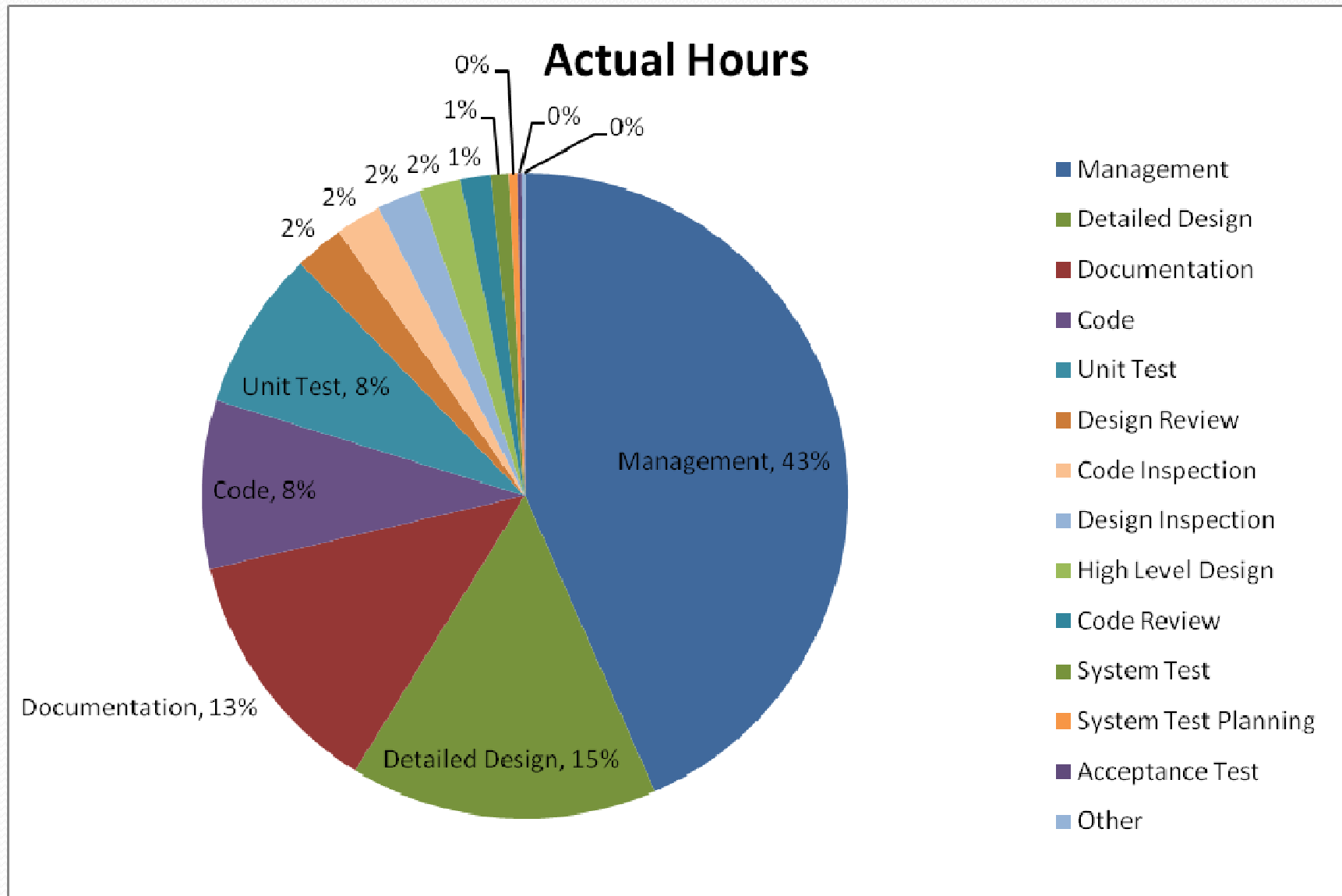
**19.06%**

Summer 2009

## Plan Hours



Summer 2009



# Quality Metrics

Development Time Ratios	Plan	Actual
REQ Inspection / Requirements	0.00	0.00
HLD Inspection / High-Level Design	0.00	0.00
Detailed Design / Code	1.44	2.17
DLD Review / Detailed Design	0.58	0.16
Code Review / Code	0.46	0.27

# Quality Metrics

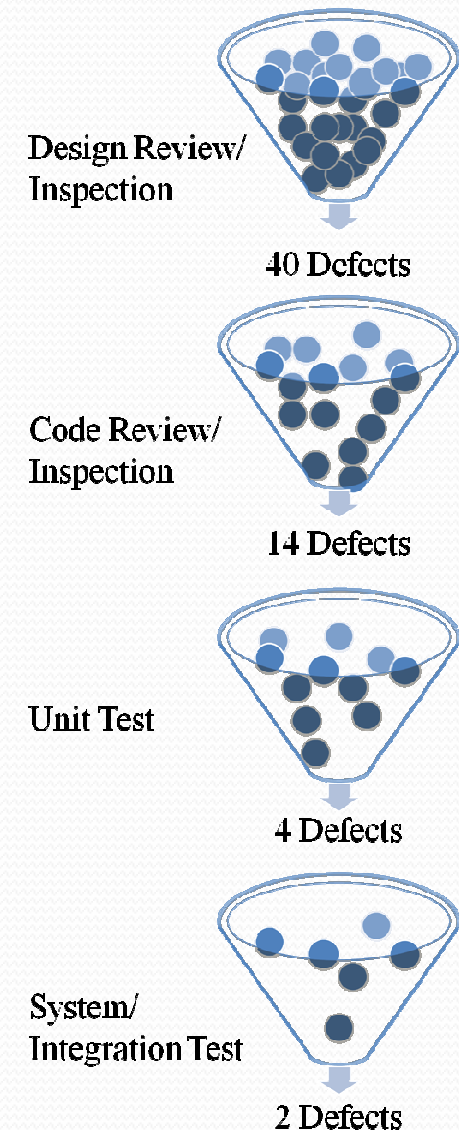
	Defects Injected		Defects Removed		Phase Yields
	Actual	Actual%	Actual	Actual%	Actual
Planning	0	0.0%	0	0.0%	0%
Requirements	0	0.0%	0	0.0%	0%
System Test Plan	0	0.0%	0	0.0%	0%
REQ Inspection	0	0.0%	0	0.0%	0%
High-Level Design	0	0.0%	0	0.0%	0%
Integration Test Plan	0	0.0%	0	0.0%	0%
HLD Inspection	0	0.0%	0	0.0%	0%
Detailed Design	52	65.0%	0	0.0%	0%
DLD Review	0	0.0%	29	36.3%	56%
Test Development	0	0.0%	0	0.0%	0%
DLD Inspection	0	0.0%	11	13.8%	48%
Code	28	35.0%	3	3.8%	8%
Code Review	0	0.0%	11	13.8%	30%
Compile	0	0.0%	0	0.0%	0%
Code Inspection	0	0.0%	12	15.0%	46%
Unit Test	0	0.0%	10	12.5%	71%
Build and Integration Test	0	0.0%	2	2.5%	50%
System Test	0	0.0%	2	2.5%	100%
Total Development Defects	80	100.0%	80	100.0%	

20K LOC

80 Defects

# Quality Metrics

	Defects Removed		Phase Yields
	Actual	Actual%	
Planning	0	0.0%	0%
Requirements	0	0.0%	0%
System Test Plan	0	0.0%	0%
REQ Inspection	0	0.0%	0%
High-Level Design	0	0.0%	0%
Integration Test Plan	0	0.0%	0%
HLD Inspection	0	0.0%	0%
Detailed Design	0	0.0%	0%
DLD Review	29	36.3%	56%
Test Development	0	0.0%	0%
DLD Inspection	11	13.8%	48%
Code	3	3.8%	8%
Code Review	11	13.8%	30%
Compile	0	0.0%	0%
Code Inspection	12	15.0%	46%
Unit Test	10	12.5%	71%
Build and Integration Test	2	2.5%	50%
System Test	2	2.5%	100%
Total Development Defects	80	100.0%	





# Conclusion

- Team delivered to their clients one week ahead of schedule
- Only two defects found in system test, and no defects reported by clients after delivery
- By contrast, other MSE teams spent an additional two months in the fall 2009 semester on bug fixes and enhancements
- We became better engineers