



Patterns and Systems Architecture

INCOSE, Orlando Chapter

February 2009

Dr. Robert Cloutier
Associate Professor
School of Systems and Enterprises
Stevens Institute of Technology
Hoboken, NJ





“The human eye has a readiness for patterns. Much is not seen simply because the mind is blind, not eyes. The eyes see in lines, curves, and patterns. Man himself works in patterns simple or complex, and such things are often evidence of man’s previous presence.”



William Tell Sackett, Treasure Mountain
By Louis L'Amour, 1972



Patterns in Prehistoric Art, Society, and Communications

Patterned Body Anthropomorphs



Petroglyphs, Coso Range, California



Source: A Field Guide to Rock Art Symbols Of the Greater Southwest, Alex Patterson, Johnson Books, Boulder, 1992

Rock Art at Petroglyph National Monument, Albuquerque, NM

Petroglyphs are powerful cultural symbols that reflect the complex societies and religions of the surrounding tribes.

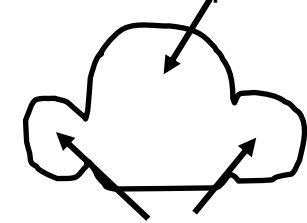


Brief Pattern History

- 1964 - Christopher Alexander
 - Books on Architecture, building and urban planning
 - Notes on the Synthesis of Form
 - A Pattern Language
 - A Timeless Way of Building
- 1987 - Ward Cunningham & Kent Beck
 - Decided to use some of Alexander's ideas
 - Developed five patterns for guiding novice Smalltalk programmers
 - Presented paper at OOPSLA'87 in Orlando
 - *"Using Pattern Languages for Object-Oriented Programs"*.
- 1995 - Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides
 - Software Design
 - Design Patterns: Elements of Reusable Object-Oriented Software

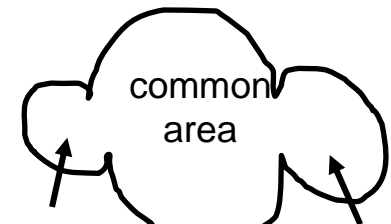


Shared couple's realm



private realms

House for Couple



Parent's realm

Children's realm

House for Couple w/ Small Children



Abstraction - Why Patterns Work



Photograph of a White Rose

- Abstraction hides detail
- Does not obfuscate the item of interest



Abstraction White Rose
Georgia O'Keefe, 1927



Implicit & Explicit Knowledge

- **Implicit Knowledge**
 - Internal, unshared knowledge acquired via formal or informal mentoring, and work experience in the work environment
 - The holder of that implicit knowledge may become a bottleneck in applying systems experience on current or future projects
- **Explicit Knowledge**
 - Implicit Knowledge made known through documentation and sharing

Describing architectures in the context of known and understood patterns should foster better and more consistent understanding across the many stakeholder communities



Architecture Patterns

System architecture patterns constitute high-level structures appropriate to the design of the major elements and subsystems of the **system or enterprise** of interest. They express the relation between the context, a problem, and a solution, documenting attributes and usage guidance. Architecture patterns are **time-proven in solving problems** which are **similar in nature** to the problem under consideration.

Value of Architecture Patterns

- Knowledge Management
 - Enables reuse of good concepts and implementations, and to preserve them for future projects
- Control Complexity
 - Architectural patterns may help control the complexity of an architecture by standardizing it on a well known and practiced pattern
- Common Understanding
 - Describing parts of the architecture in the context of known and understood architecture patterns results in a common understanding of the architecture
- Mitigate Risks
 - Using and applying known architecture patterns in architecture design introduce less risk than a new architecture design



Patterns Are Not Created - They are Mined

- Over time, similar designs are arrived at independently by different designers on various projects
- Where the same elements exist in multiple designs, the design can be studied and documented to encourage reuse
 - This reuse prevents engineers from having to reinvent, or rediscover the same concept
 - And, it provides a common vocabulary for the design concepts that a project can share

Important Pattern Concepts

- **Separation & Abstraction**
 - Separating the idea from the reality, capturing more generalized concepts when documenting the pattern
 - Removing detail from something complex to make it simpler to understand
 - can be thought of as a continuum with a multiple levels, and what is detail to one level of architecture is a requirement at another level
- **Rule of three**
 - a pattern is not a pattern unless there are at least three independent, observable applications utilizing the proposed pattern as part of the solution



Enterprise Architecture Pattern Drivers

- Corporate systems engineering departments attempt to capture knowledge explicitly through artifacts
- A significant component of this corporate knowledge, however, is implicit
- This implicit knowledge is useful to others only if it is shared
- If a pattern exists only in the form of implicit knowledge, it is not accessible by others
- Developing increasingly complex systems drives a need for a common lexicon between systems architects
- In communities that use patterns, the patterns often become standardized through multiple implementations, presentations at research and professional conferences, and publication in research journals
- This standardization fosters reuse of designs and even code that might be generated from the architecture patterns.
 - Such reuse can in turn improve development efficiency and productivity

Expressing Patterns

- Was no set manner to describe patterns - even where used extensively
- Appears to be some common attributes used to describe software patterns
- Example to the right are some software pattern description template examples

Pattern Template	Patterns for Effective Use Cases	Architecture Patterns	U.S. Treasury Architecture Guidance	A Pattern Language for Pattern Writing
[Ris03]	[Ado03]	[TOG02]	[TOG02]	[Mes]
Pattern Name	Pattern Name	Name	Name	Pattern Name*
Aliases				<i>Aliases</i>
Problem	Problem Statement	Problem	Problem	Problem*
	Metaphoric Story		Implementation	
Context	Context	Context	Structure	Context*
Forces	Forces affecting the Problem	Forces	Interactions	Forces*
			Consequences	<i>Indications (symptoms)</i>
Solution	Solution	Solution		Solution*
Resulting Context		Resulting Context	Assumptions	<i>Resulting Context</i>
Rationale		Rationale	Rationale	Rationale*
Known Uses				
Related Patterns		Related Patterns		<i>Related Patterns</i>
Sketch	Picture			
Author(s)				<i>Acknowledgements</i>
Date				
Email				
References				
Keywords				
Example	Examples	Examples		Examples*
		Known Uses		<i>Code Samples</i>
				* - required, <i>italics</i> - optional



A Systems Engineering Pattern Form

Pattern Name	The name of the pattern should be descriptive to enable the pattern user to understand the usage
Aliases	Other names by which the pattern may be known
Keywords	Keywords which assist in locating appropriate patterns when needed
Problem Context	Brief discussion of the constraints the pattern may impose
Problem Description	What is the problem this pattern can be used to solve
Forces	Challenges that exist in the problem being addressed by the pattern, and the problems in applying the pattern
Pattern Solution	Discussion on how the pattern solves the problem being addressed
Sketch	This can be one or more diagrams necessary to represent the pattern
Interfaces	Discussion of the critical interfaces or information flows necessary in implementing the pattern
Resulting Context	What are the unaddressed issues remaining when the pattern is applied/used
Example	An example of how the pattern may be applied
Pattern Rationale	Why the pattern works
Known Uses	Where else is the pattern being used in other places or applications
Related Patterns	Other patterns what may work in conjunction or in association with this pattern
References	Other information that may be useful in understanding or applying the pattern
Authors	Who documented the pattern



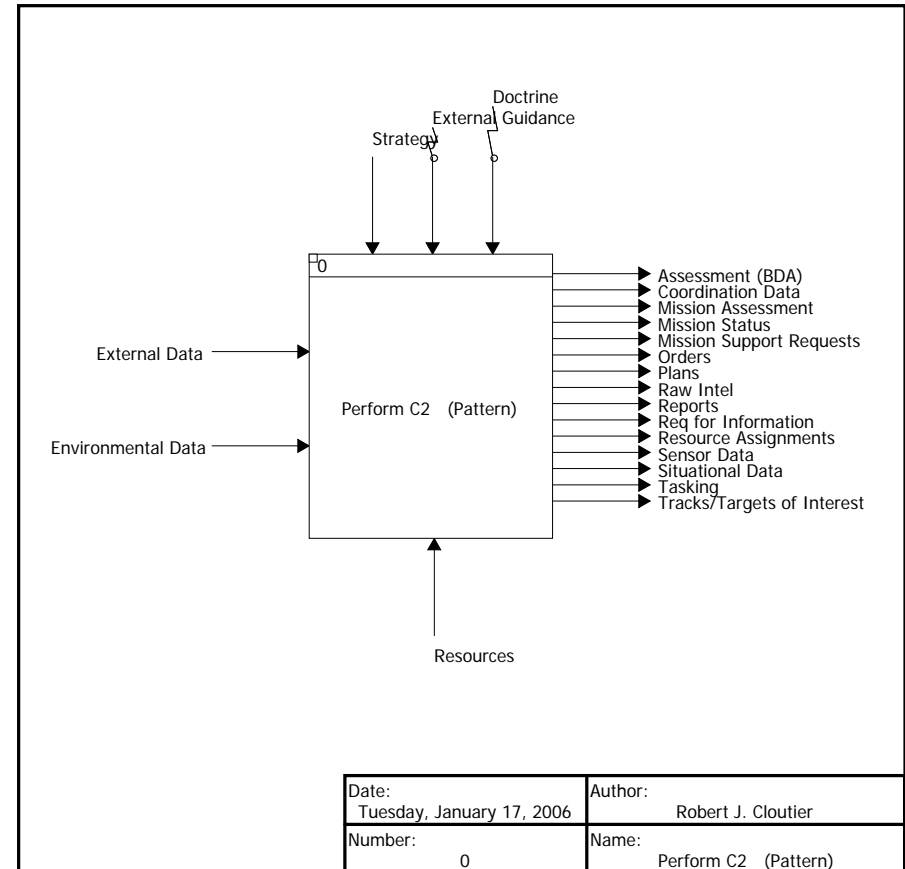
A Pattern Form for System Architects

Pattern Name:	The name of the pattern should be descriptive to enable the pattern user to understand the usage
Aliases:	Other names by which the pattern may be known
Keywords:	Keywords which assist in locating appropriate patterns when needed
Problem Context:	Brief discussion of the constraints the pattern may impose
Problem Description:	What is the problem this pattern can be used to solve
Forces:	Challenges that exist in the problem being addressed
Pattern Solution:	Discussion on how the pattern solves the problem being addressed
Sketch:	This can be one or more diagrams necessary to represent the pattern
Interfaces:	Describe critical interfaces/information flows for the pattern
Resulting Context:	What are the unaddressed issues remaining when the pattern is applied/used
Example:	An example of how the pattern may be applied
Known Uses:	Where is the pattern being used
Related patterns:	Patterns that work in conjunction or in association with this pattern
References:	Additional info that may be useful
Pattern Rationale:	Why the pattern works
Author(s):	Who documented the pattern

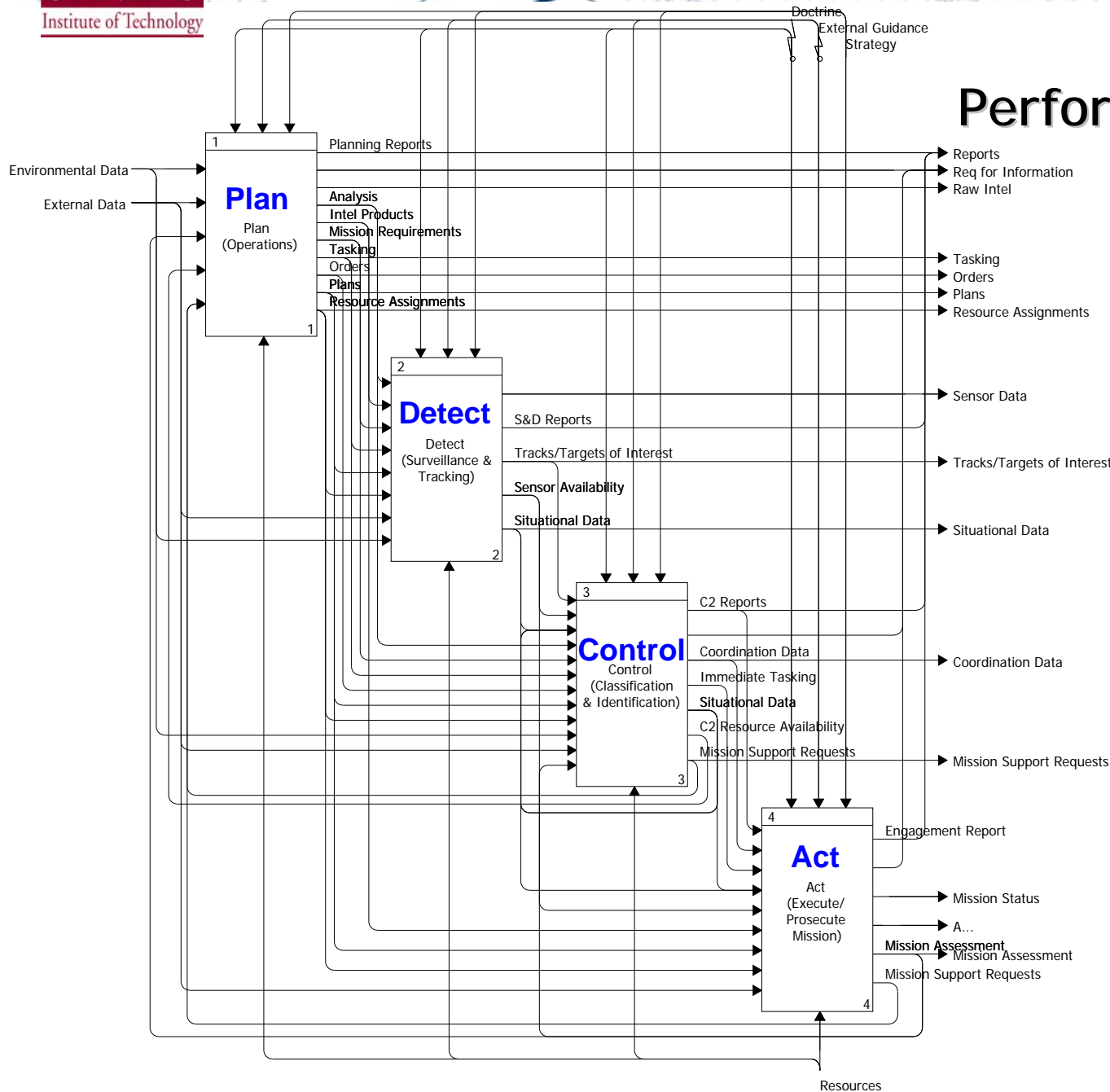
System Architecture Pattern Example

Perform C2 A-0 Diagram

- Pattern began as a successfully implemented logical architecture, on the first system.
 - Next project for that same architect had similar requirements - able to reuse most of the logical architecture.
 - A year later, a mid-grade architect assigned to a similar project - his mentor told him about previous work.
 - The two architects went back to the earlier project models, and used them to guide the future logical architecture.
-
- Later, they found out about my research.
-
- The Perform C2 pattern was documented using those guidelines.



While the pattern is documented in IDEF0 (instead of UML/SysML), it does demonstrate the magnitude of a system architecture pattern

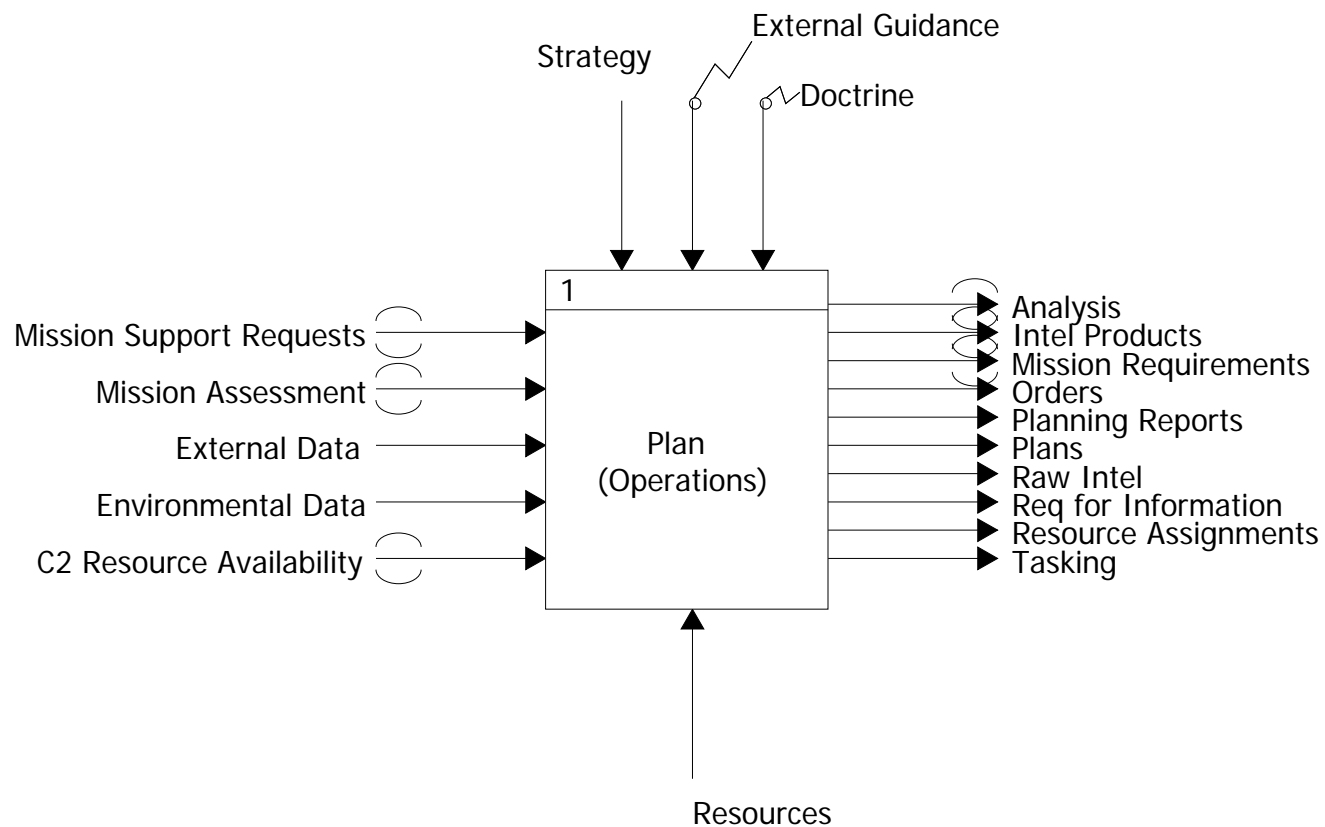


Perform C2 A0 Diagram

- This pattern was mined from 4 different systems
 - Including a Coast Guard system and a DHS proposal)
- Inputs and Outputs are generalized for reuse
- When using a pattern like this,
 - Unnecessary IERs are deleted
 - New and unique IERs are added
- Has captured implicit and explicit knowledge from C2 experts and can be used for:
 - A Stakeout
 - Planning a seizure
 - Tracking a potential threat
 - Prosecuting a threat
 - Controlling assets in a coordinated operation (the extended enterprise)

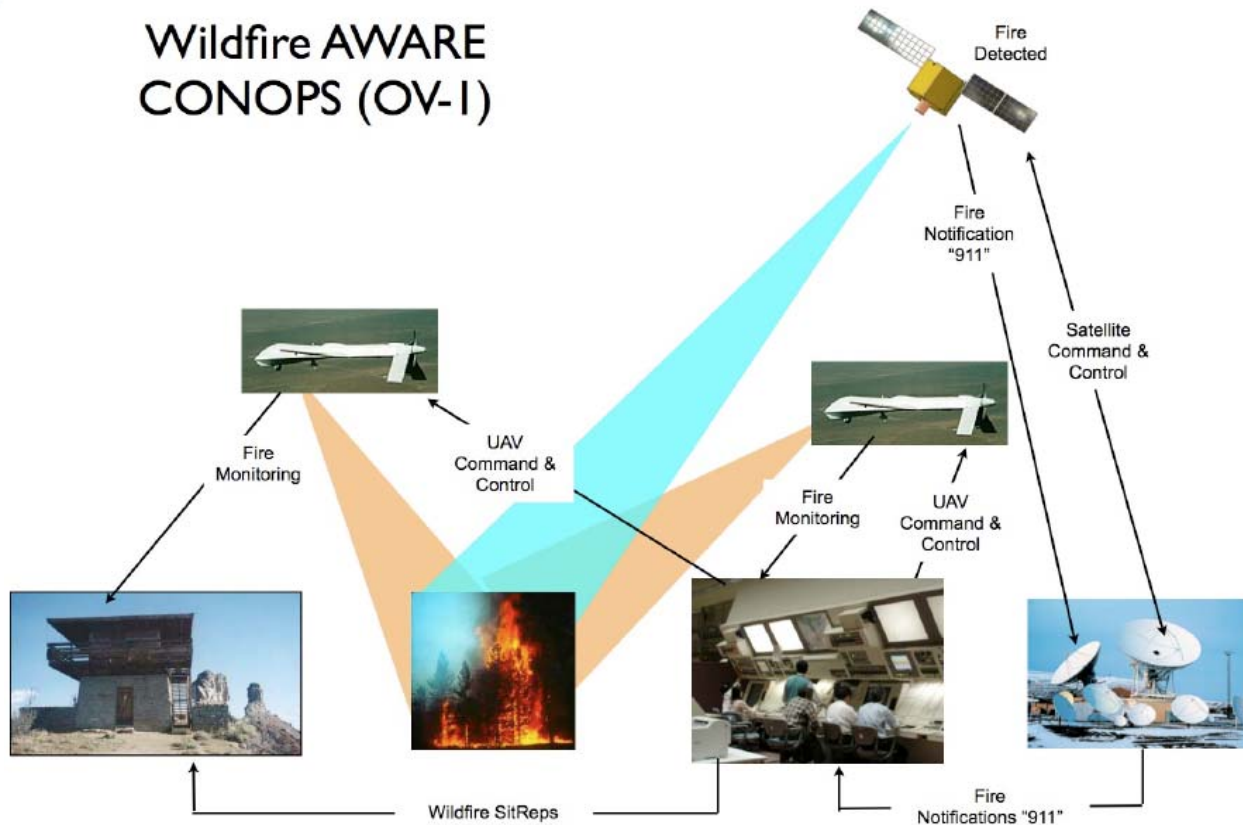


The Smaller "Plan Operation" Pattern



The FireSAT Mission Case Study

Wildfire AWARE CONOPS (OV-I)



- Introduced in 1998 by Larson & Wertz
- Intended to be an exercise for space engineering students
- Based on the fictitious NASA and the US Forest Service requesting a study
- Investigate feasibility of developing the mission concept for an on orbit fire detection system
- The stated system need:

The United States needs a more effective means to detect and monitor potentially dangerous wildfires

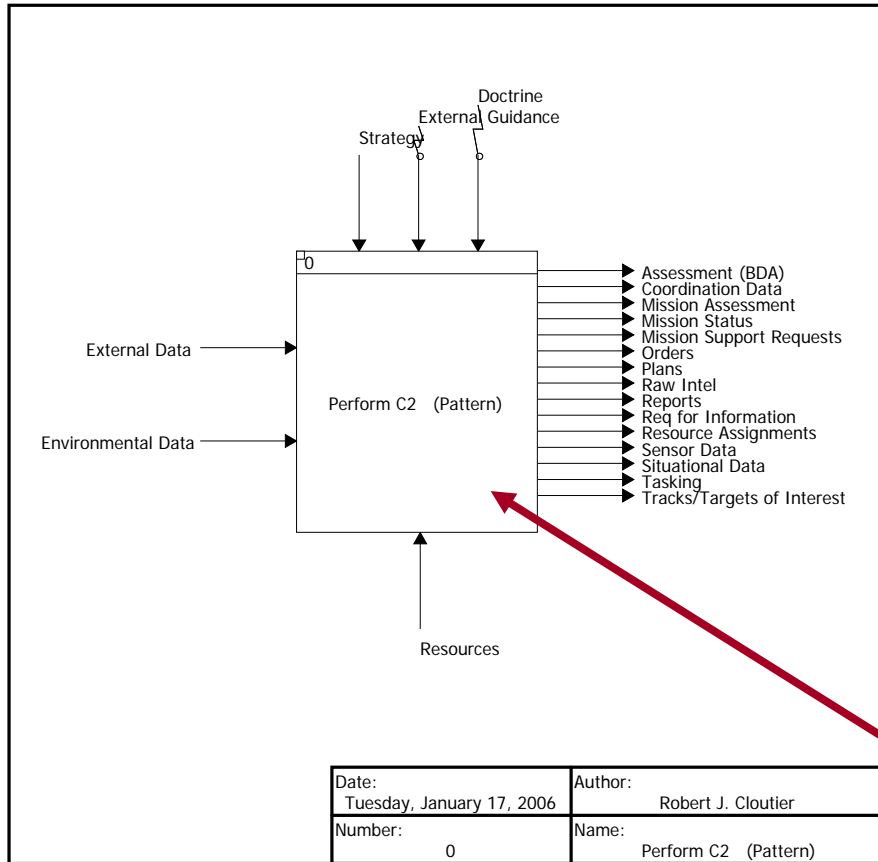


FireSAT Goals, From the Case Study

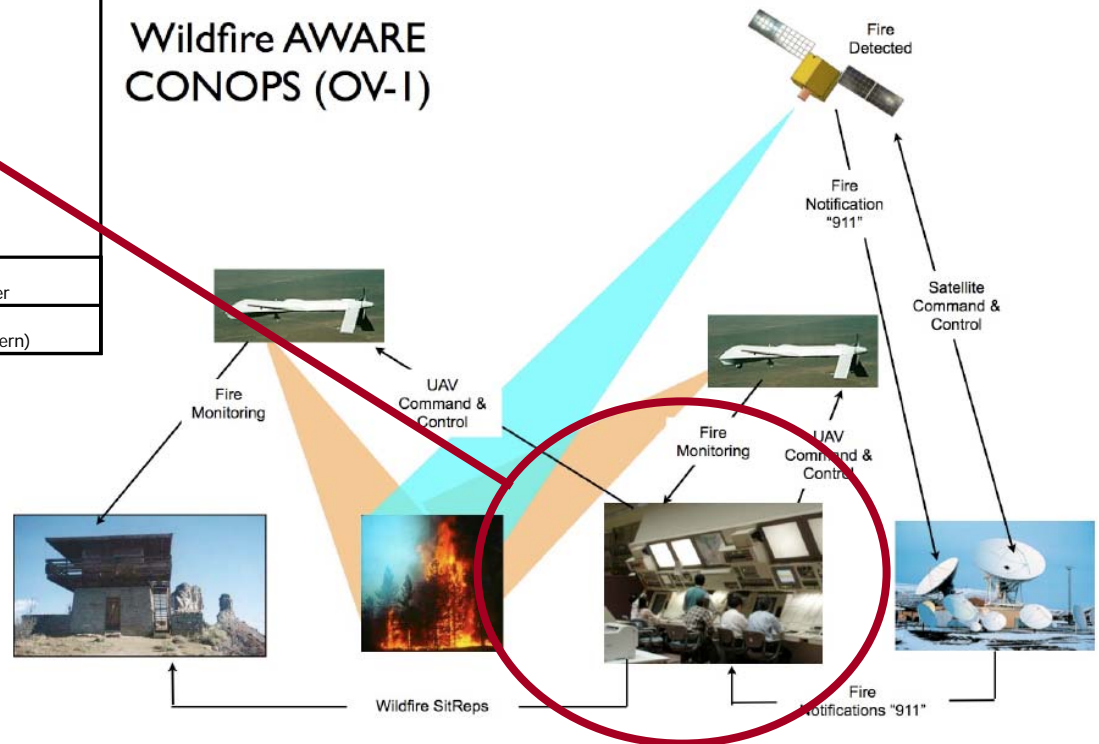
1. Provide timely detection of potentially dangerous wildfires
2. Provide continuous monitoring of high priority and potentially dangerous wildfires
3. Reduce the economic impact of wildfires
4. Reduce the risk to firefighting personnel
5. Develop an integrated ground, air and space architecture to detect and monitor wildfires in the U.S.
6. Collect statistical data on the outbreak, spread, speed, and duration of wildfires
7. Detect and monitor wildfires in other countries
8. Collect other forest management data
9. Demonstrate to the public that positive action is underway to contain wildfires



FireSAT Perform C2 A-0 Diagram



Wildfire AWARE CONOPS (OV-1)

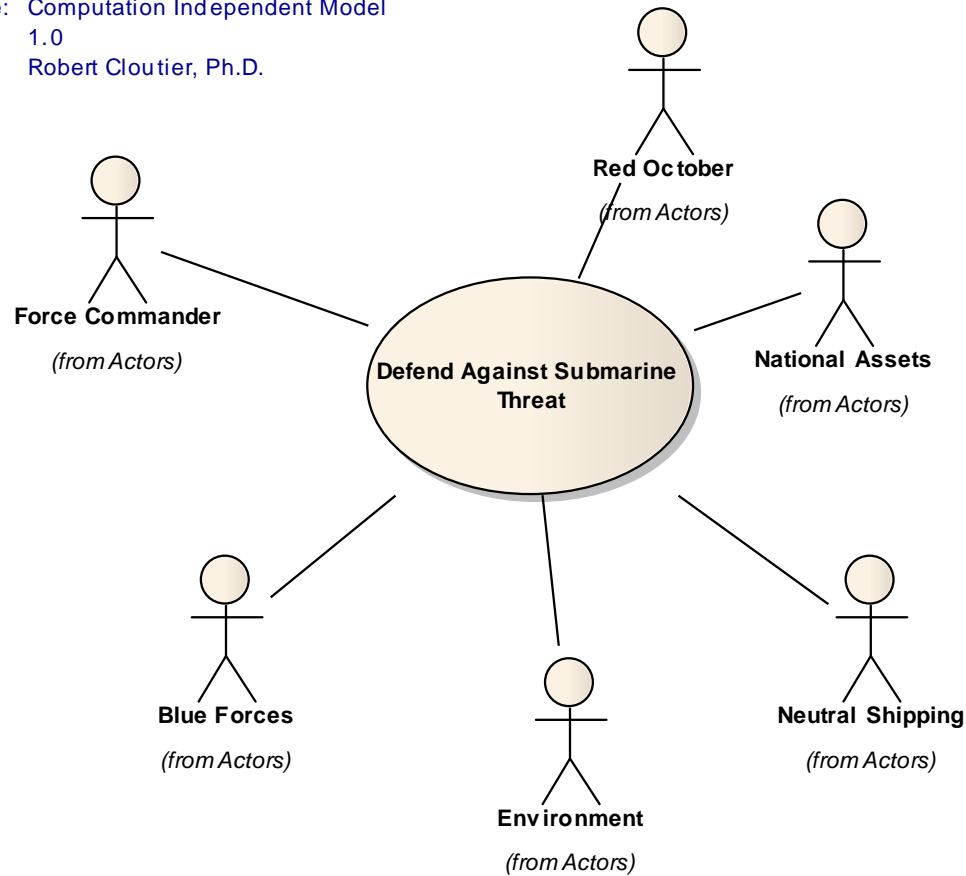




An Overly Simplified Example Hunt for Red October uc Part of the CIM

uc Defend Against Submarine Threat

Name: Defend Against Submarine Threat
Package: Computation Independent Model
Version: 1.0
Author: Robert Cloutier, Ph.D.

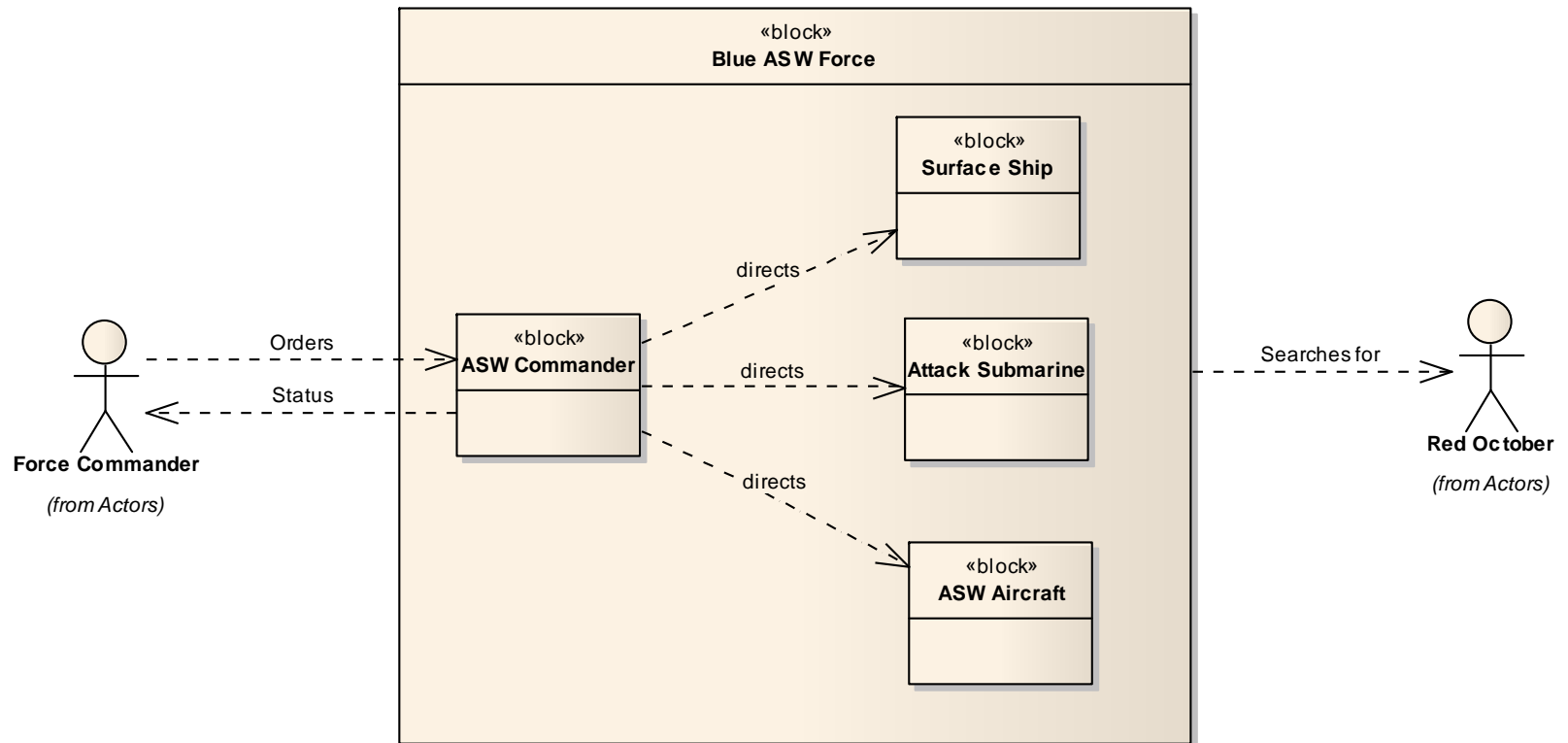




Good Guy bdd Part of the CIM

bdd Computation Independent Model [ASW Blue Forces]

Name: ASW Blue Forces
Package: Computation Independent Model
Version: 1.0
Author: R. Cloutier, Ph.D.



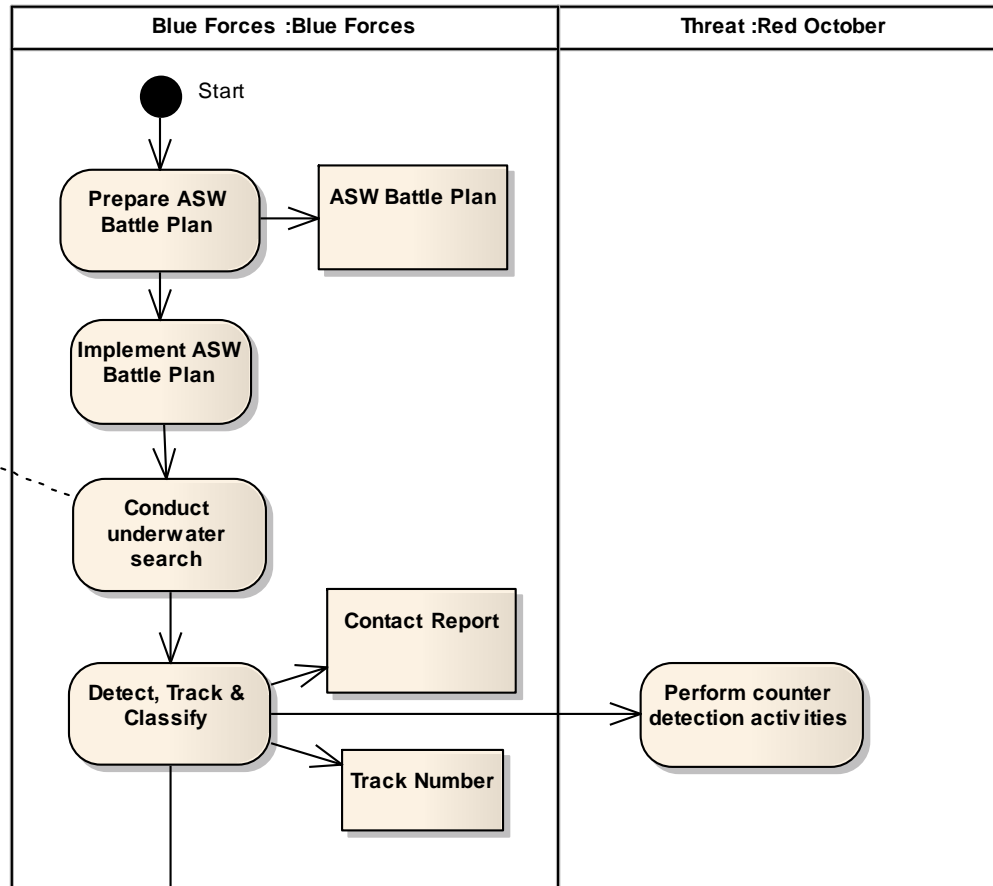


Defend Against Submarine Threat Pattern Tag?

act Defend Against ASW Threat

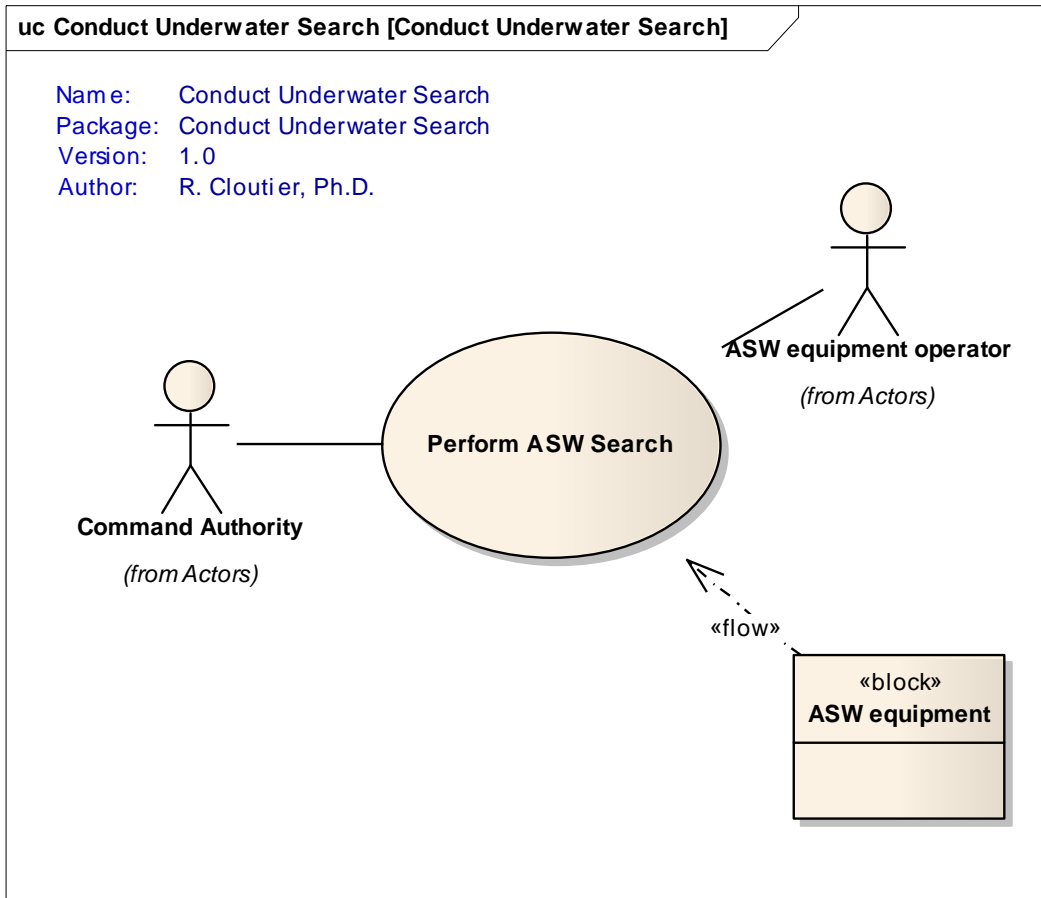
Name: Defend Against ASW Threat
 Package: Computation Independent Model
 Version: 1.0
 Author: Robert Cloutier, Ph.D.

Pattern: Apply Perform ASW Search uc pattern





Perform ASW Pattern

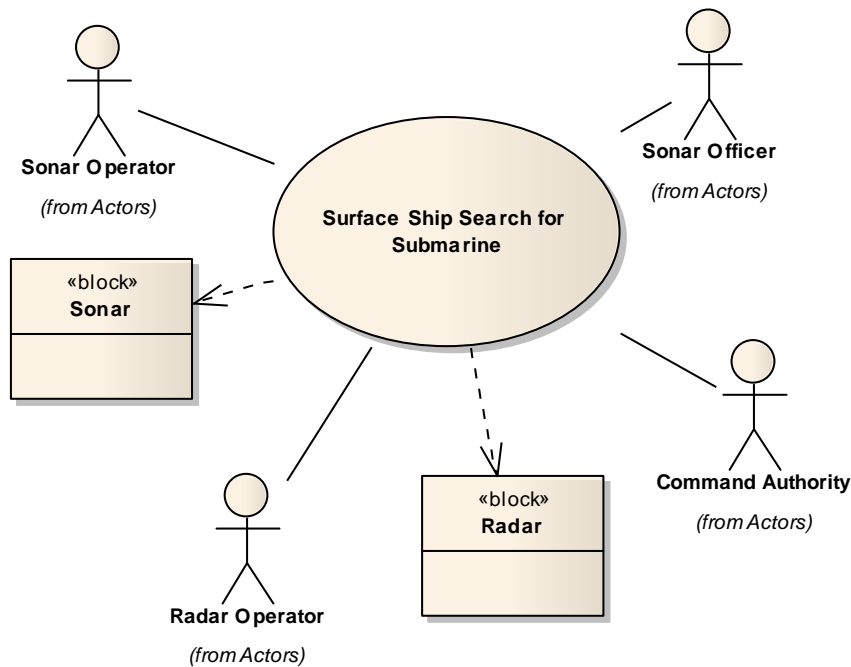




Perform ASW Search Pattern Applied to CIM

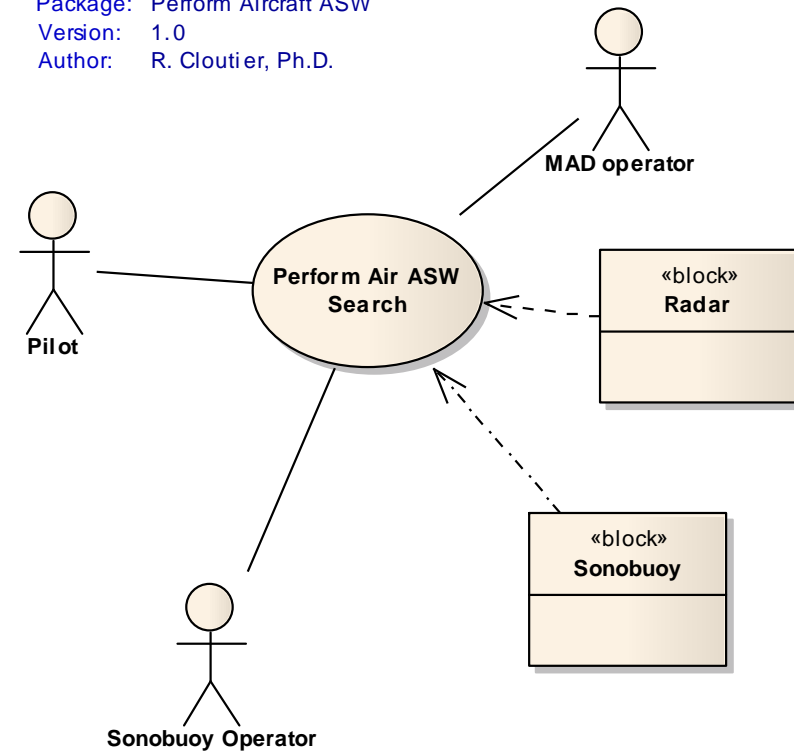
uc Perform Surface ASW [Perform Surface Ship ASW Search]

Name: Perform Surface Ship ASW Search
 Package: Perform Surface ASW
 Version: 1.0
 Author: R. Cloutier, Ph.D.

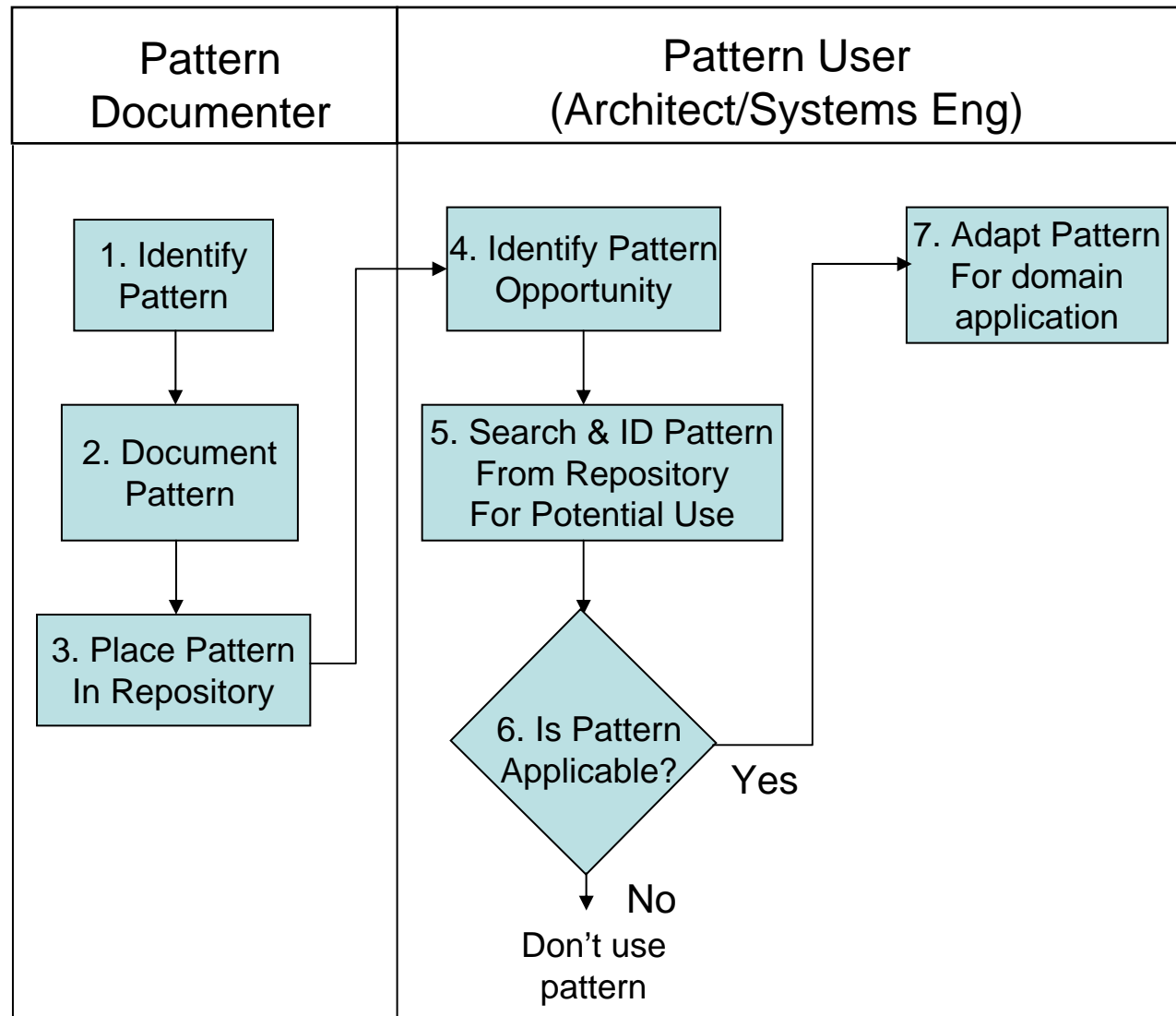


uc Perform Aircraft ASW [Perform Aircraft ASW]

Name: Perform Aircraft ASW
 Package: Perform Aircraft ASW
 Version: 1.0
 Author: R. Cloutier, Ph.D.



Thoughts on a Pattern Utilization Process





Challenges in Using Patterns

- Mining patterns
 - As one moves up the complexity spectrum of systems, there are fewer that can be mined for patterns
 - Usually represent systems from different companies - IP issues
 - Acquisition timeline so long, technology changes by the time a system is fielded
 - System Documentation differs, if one can even locate it...
 - System Engineering tool of choice - PowerPoint
 - IDEF0 -predominately Telelogic (Popkin) SA and Vitech CORE
 - UML - predominately IBM (Rational) Rose, Telelogic TAU, Telelogic (iLogix) Rhapsody
 - SysML emerging (from a host of vendors)
- Methodology Wars
 - Some systems developed using Functional decomposition, some using Object Oriented decomposition
 - Will result in different “looking” solution at high levels
- Correct level of documentation to satisfy the intent of patterns
- Recommend you not develop your own pattern lexicon
 - will make it difficult to talk about patterns in other parts of your companies
 - Will make it difficult to share patterns across broader patterns community



Contact Info:

Dr. Robert (Rob) Cloutier

robert.cloutier@stevens.edu