



---

## **Subsystem Support in UML Essentials**

Version 1.1  
February 16, 2003

---

### *PathMATE Technical Notes*

Pathfinder Solutions LLC  
33 Commercial Drive, Suite 2  
Foxboro, MA 02035 USA  
[www.PathfinderMDA.com](http://www.PathfinderMDA.com)  
888-662-7284

# Table Of Contents

- 1. Introduction..... 1**
- 2. Analysis Conventions ..... 1**
  - Domain Membership .....1
  - Subsystems.....1
- 3. UML Essentials Features ..... 1**
  - Phase 1: Repository-Based Domain Membership .....2
  - Phase 2: Subsystem comes to Springboard .....2

## 1. Introduction

This Technical Note describes the capabilities to be deployed in support of *Subsystems* in UML Essentials products from Pathfinder Solutions. A *Subsystem* is a unit of organization within a domain to contain clusters of classes - primarily to manage complexity, and without semantic impact on its enclosing domain. Subsystems are optional, and they can be readily reorganized within a domain without changing the domain's total content or capabilities.

The term *Subsystem* has enjoyed a variety of definitions and levels of support in popular tools. In light of the varied interpretations of subsystems in the past, present, and the yet-to-be decided future (UML 2.0), UML Essentials will support very simple subsystem capabilities.

PLEASE NOTE – The field experience of Pathfinder Solutions Consultants and Analysts has shown that subsystems should be used sparingly. If a domain is sufficiently complex that it requires subdivision into subsystems, then our experience shows that most of the time the overall system would benefit from a breakup of the large domain.

## 2. Analysis Conventions

### Domain Membership

UML Essentials currently (version 4.03.003 and earlier) allows only a single class diagram for a domain, and uses the presence of a class on this diagram to determine its membership in the domain. While scoping a class to its domain package was always an MBSE modeling convention, it was not strictly required. With the recent improvement of package capabilities within our supported UML editors, this diagram-based domain membership can change.

As a first step in the support of subsystems UML Essentials extract will ignore all class diagrams when determining the domain ownership of a class, and simply use package scoping. The domain ownership of a class will be determined by the root-level package it belongs to. If a class is scoped to a nested package the nesting will be followed until the root-level package is found.

In this manner, domain membership will shift from diagram-based to repository-based.

### Subsystems

A *Subsystem* is a package that contains a subset of the classes in a single domain, and its primary manifestation is a class diagram showing these classes. A domain containing subsystems would show them as packages on the domain's class diagram. A common convention when using subsystems in a domain is to require that all classes in the domain belong to a subsystem, and show only subsystem packages on a domain's class diagram.

## 3. UML Essentials Features

UML Essentials support for subsystems will be deployed in two phases.

## Phase 1: Repository-Based Domain Membership

In this phase, the editor extract modules will determine the domain ownership of a class based on root package scoping. Any nested packages will be ignored, all classes must belong to one root package, and all root packages containing classes must be domains. A nested (non-root) package can contain classes, and these must not be domains.

A domain's class diagram can contain anything the analyst wants. Through proper naming (<domain name>.<subsystem name>), subsystem class diagrams can be accessed in report generation through the Springboard's `domain.supportDiags` field.

The subsystem itself, and any class membership in a subsystem will not be available through Springboard.

This phase will impact the UML Essentials integration elements, including PAL file open and extract.

## Phase 2: Subsystem comes to Springboard

In this phase the subsystem becomes a recognized analysis element, with UML Essentials supporting its use in report generation, and optionally its custom code generation.

Each subsystem can have their own interface class, named "services" defining a subset of the domain services. Springboard will collect all domain services from wherever they are defined and present them as a complete set in the `Domain.services` field. In addition any services defined within a subsystem will be added to that subsystem's `Subsystem.services` field. The following rules/capabilities apply:

- A "services" interface class can still be used at the domain level
- A subsystem can define 0 or one "services" interface class
- Each domain service must have a unique name throughout the domain, regardless of where they are defined.
- All domain services are still considered scoped to the domain (as opposed to being somehow constrained to any subsystem) regardless of where they are defined, and their actions may freely access all domain constituent elements.

In addition, diagram support in Springboard will be extended to support more flexible report capabilities. Two new Springboard analysis elements will be added:

**Subsystem:** supertype: *none*; subtypes: *none*  
 description (String)  
 classDiagram (Diagram): *this subsystem's class diagram*  
 domain (Domain) : *The enclosing domain.*  
 langId (String) : *name sanitized for use as a C-language identifier*  
 name (String)  
 objects (ObjectList)  
 services (DomainServiceList)  
 subsystem (SubsystemList): *any nested subsystems this contains*

**Diagram:** supertype: *none*; subtypes: *none*  
filename (String): *the fully qualified filename for diagram for use with the DIAGRAM directive*  
name (String): *The diagram's name in the host editor*

In addition, the following diagram-related fields of existing analysis elements will be added or updated:

Domain.im (Diagram): *this domain's class diagram*  
Domain.subsystems (SubsystemList): *list of subsystems in this domain*  
Domain.supportDiags (DiagramList): *supplemental diagrams, named with <domain name>.\**  
Object.std (Diagram): *this class's state transition diagram*  
Object.subsystem (Subsystem): *The subsystem this class belongs to.*  
System.domainChart (Diagram)  
System.supportDiags (DiagramList): *supplemental diagrams, named with <system name>.\**

These extensions will extend the information available through Springboard to include these currently unavailable capabilities:

- Generate subsystem membership/containment information in reports and code comments.
- Create code customizations using subsystem membership/containment information
- Allow more than one class diagram describing a domain.
- Allow clear identification of support diagrams in a generated report.
- Allow former Shlaer-Mellor users to develop their own templates to generate subsystem Access, Communication, and Relationship Reports.

This phase will impact the following UML Essentials product elements:

- editor integration elements, including extract and the XML emitter
- Springboard
- report templates
- C++, C, and Java code templates (just comments to indicate subsystem membership)