

Korea
Software
Technology
Association



UML2.x 기초 다루기 실습

훈련기간: 2010.01.25 ~ 02.05

강사명: 손재현 -넥스트트리소프트
-jhsohn@nexttree.co.kr

□ 교육 목표 & 특징

- UML2.x의 이해
- 유스케이스 작성
- 객체모델링 이해
- UML2.x의 다양한 다이어그램 이해 및 활용
- 모델링 도구 사용법 습득

- 본 강의는 아래 기술에 대한 이해를 필요로 합니다.
 - 객체지향 언어(Java) 기초
 - 개발프로세스 이해

□ 교육은 매 회 4 시간씩 총 5회에 걸쳐 진행합니다.

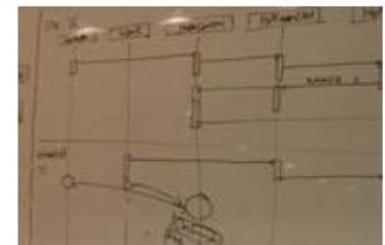
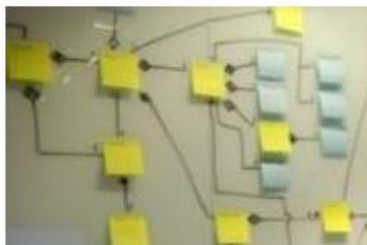
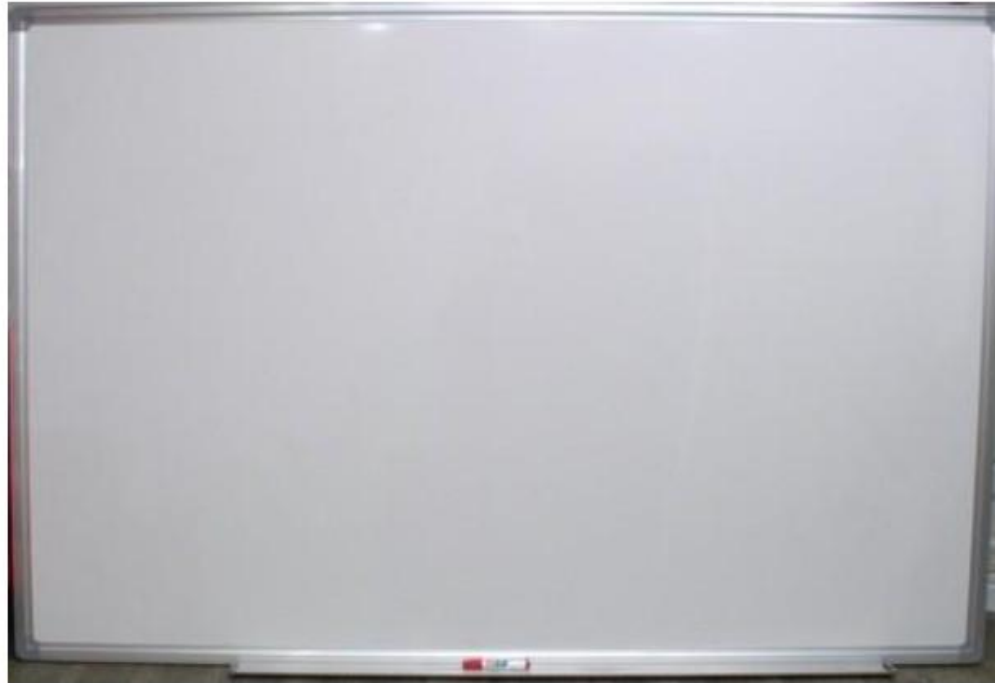
1 일차	2 일차	3 일차	4 일차	5 일차
<ul style="list-style-type: none"> - UML 개요 - UML 소개 - UML 역사 - UML 다이어그램분류 	<ul style="list-style-type: none"> - 구조 다이어그램 - 클래스 - 객체 - 컴포넌트 - 배치 	<ul style="list-style-type: none"> - 행위 다이어그램 - 유스케이스 - 액티비티 - 상태기계 	<ul style="list-style-type: none"> - 상호작용 다이어그램 - 상호작용 Overview - 시퀀스 - 커뮤니케이션 - 타이밍 	<ul style="list-style-type: none"> - 유스케이스 I - 유스케이스 개요 - 유스케이스 내용 - 유스케이스 다이어그램
6 일차	7 일차	8 일차	9 일차	10 일차
<ul style="list-style-type: none"> - 유스케이스 II - 유스케이스 목표수준 - 유스케이스 명세 - 유스케이스 패턴 	<ul style="list-style-type: none"> - 유스케이스 III - 유스케이스 분석기법 - 분석클래스 - 제어클래스 - 실체클래스 	<ul style="list-style-type: none"> - 요구사항 모델실습 I - 유스케이스 - 사용자 시나리오 - 핵심개념 모델 	<ul style="list-style-type: none"> - 요구사항 모델실습 II - 인터페이스 추출 - 유스케이스 분석 - 컴포넌트 식별 	<ul style="list-style-type: none"> - 설계모델 실습 - 컴포넌트 설계 - 유스케이스 설계 - 도메인 모델

1일차 – UML 툴 소개 및 설치

1. UML 툴 소개
2. EA 설치
3. UML Spec 다운로드

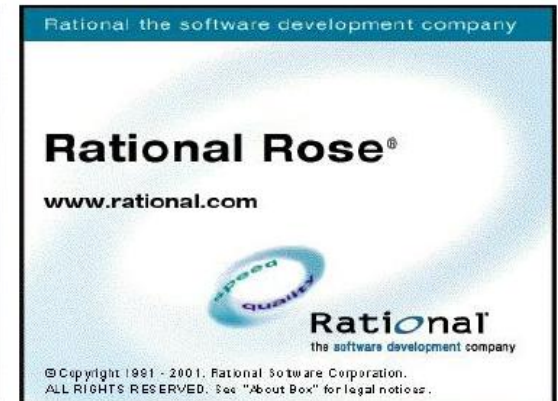
ONE STEP AHEAD

□ 칠판 – sketch



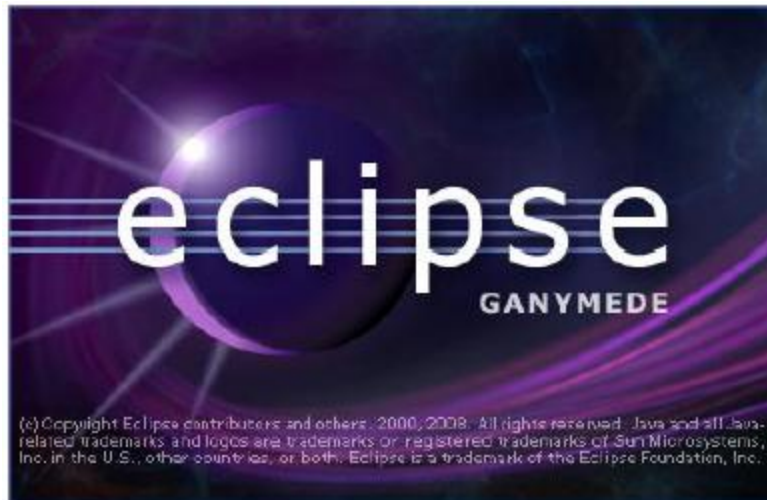
□ 상용툴

- EA (Enterprise Architect)
- Rational Rose
- Together
- XDE ...



□ 오픈소스 툴

- AndroMDA
- Eclipse UML
- Start UML
- Umbrello ...

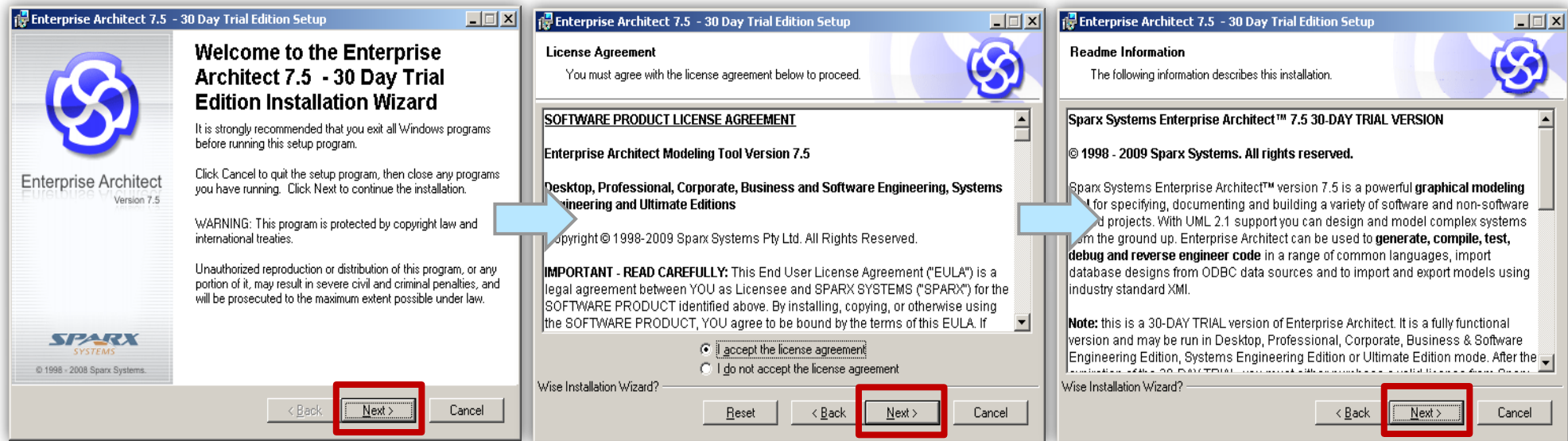


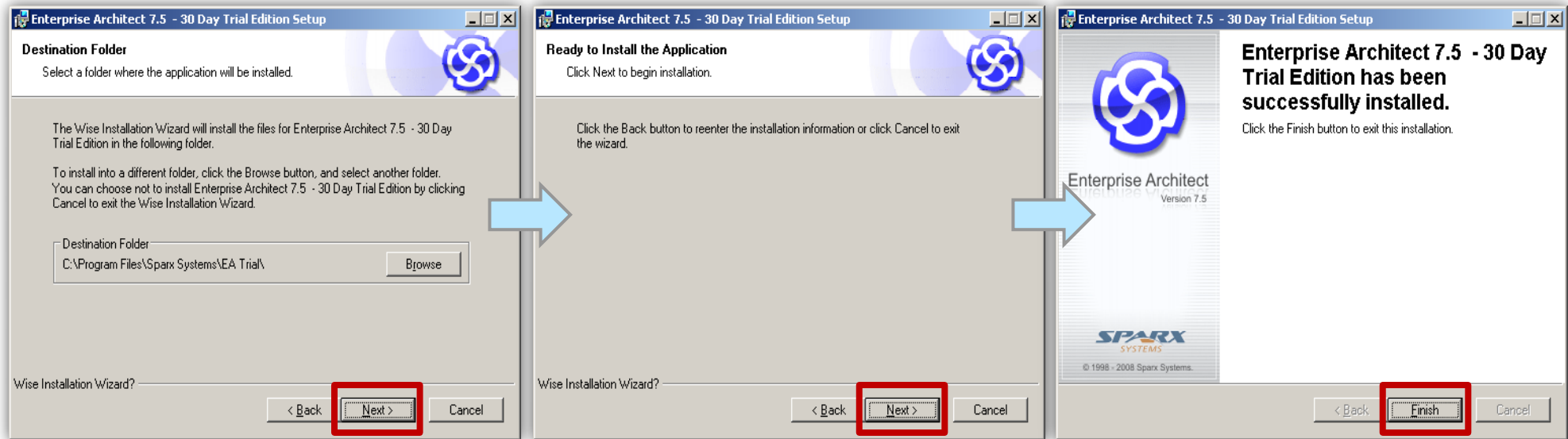
□ http://www.objectsbydesign.com/tools/umltools_byCompany.html

To sort by a different column, click on the column heading [Updates?](#)

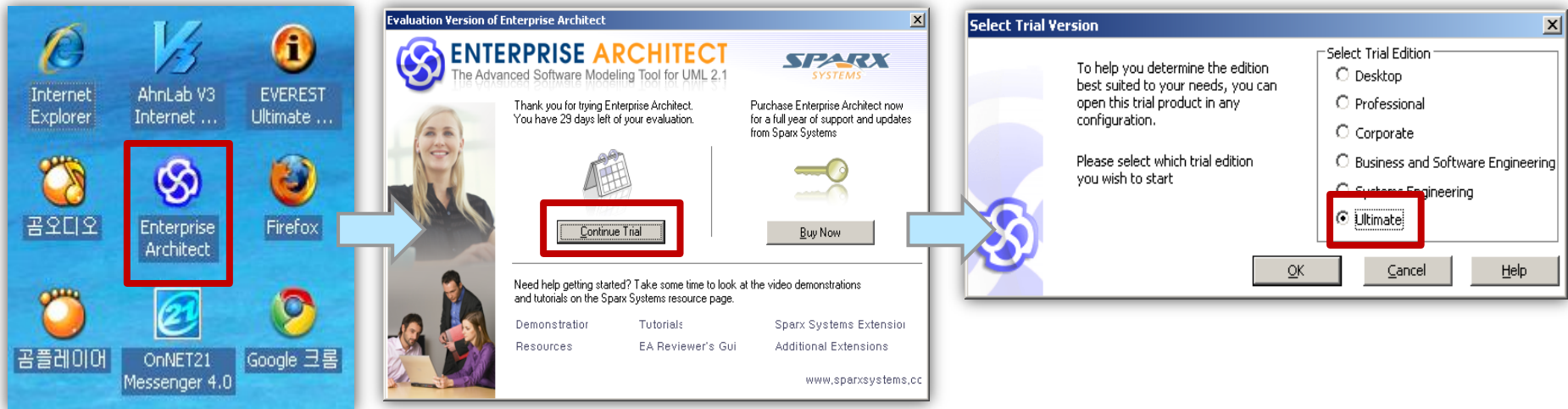
Company	Product	Version	Date	Platform	Price
Altova	UModel UML 2.0, Java round-trip engineering	2005	05/2005	Windows	\$129
AndroMDA	AndroMDA Open-source MDA tool generates J2EE/EJB code Spring, Hibernate, Struts, Web Services, reads XMI	3	05/2005	Java VM	\$0
Aonix	Ameos Supports MDA, UML 2.0 Profiles, generates C, C++, Ada, Java	1	09/2003	Windows, Linux	n/a
Aonix	Software through Pictures UML Supports MDA, generates C, C++, Ada, Java	8.3.1	03/2003	Windows, Linux	n/a
Arion Software	UML2COM integration tool for VC++/C++, add in for Rational Rose, COM+ code generator	1	02/2001	Windows	\$990
Artisan	Real-time Modeler real-time modeling, multi-user object repository, UML 2.0 template packages	4.3	06/2003	Windows	\$2495
Artisan	Real-time Studio Professional adds round-trip engineering for C, C++, Java, ADA, Spark DOORS integration, state machine animation	4.3	06/2003	Windows	n/a
Artiso	Artiso Visual Case CASE tool provides both UML and database design, SQL editing	2.6	01/2004	Java VM	\$449
Atos Origin	Delphia Object Modeler (D.OM) auto-generation of functional prototypes from UML models, XMI, class and state diagrams, report generation	3.2.6	02/2001	Windows	\$0
BOUML	BOUML open-source, UML 2.0, Java, C++, IDL, HTML documentation generation import Rational Rose, high performance	2.5.5	09/2005	Windows, Unix	\$0
Beto Software	IntelliUML Teresa UML class and sequence diagrams, integrated with IntelliJ, XMI 1.2, PNG	2	10/2005	Java VM	\$499
Borland	Together Edition for Visual Studio .NET integration of Together with Visual Studio .NET	2005	05/2005	VS .NET	\$1500
Borland	Together Designer BPMN, UML 2.0, only supports a designer role (no code view)	2006	09/2005	Java VM	\$3495
Borland	Together Architect BPMN, MDA, QVT, OCL, UML 2.0, metrics, audits, code-model synchronization	2006	09/2005	Java VM	\$11500
CanyonBlue	Koneso Modeler web-based, collaborative, multi-user tool	1	06/2001	Java VM	n/a
Codagen	Codagen Architect OMG MDA-compliant tool generates code from UML models	3.2	05/2003	Windows	n/a

- ❑ Enterprise Architect 7.5.847 (30-day trial)
- ❑ 다운로드 URI: <http://www.sparxsystems.com/>





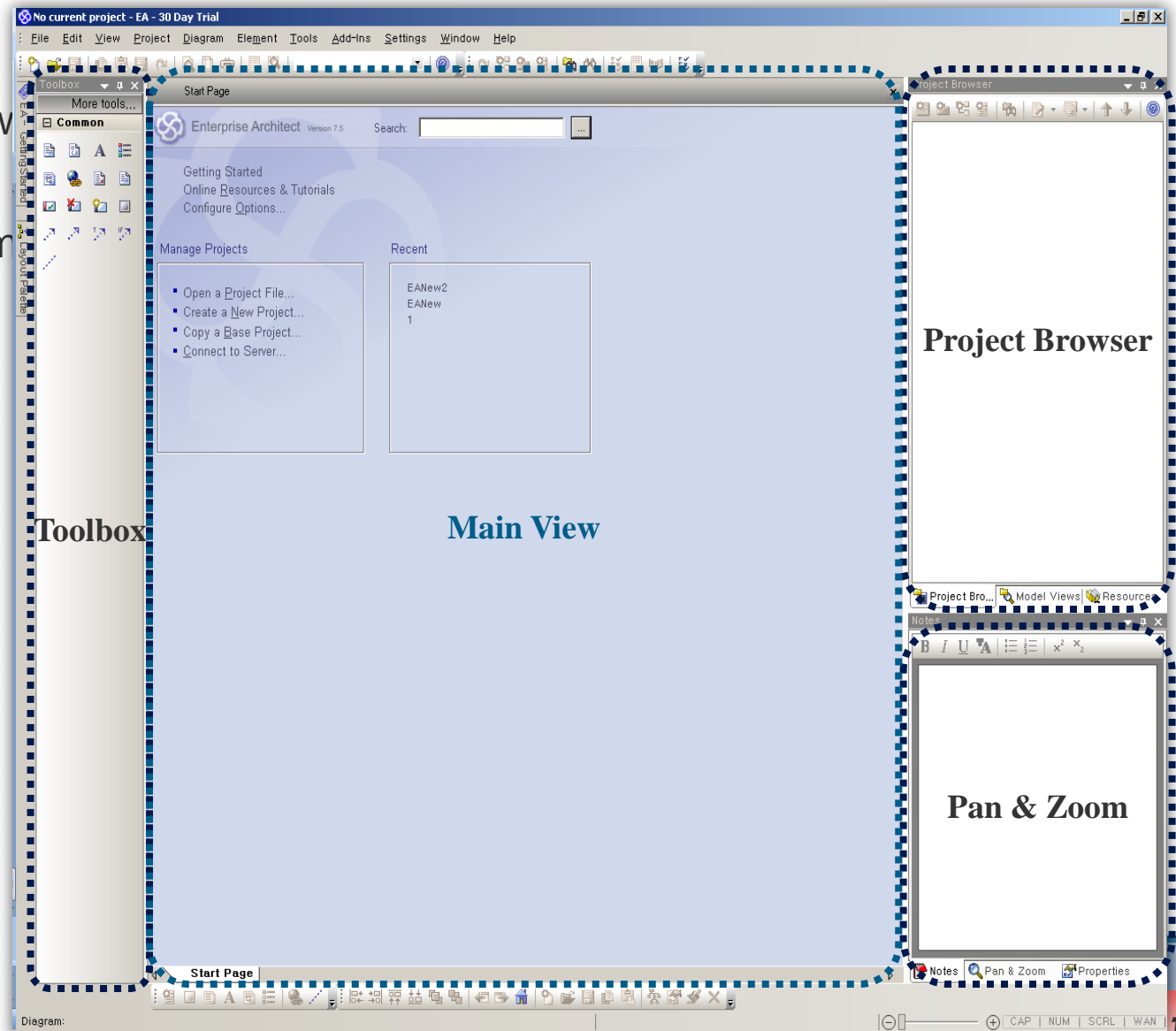
□ 바탕화면 >> Enterprise Architect 실행 >> Continue Trial >> Ultimate



EA 레이아웃 구성 (1/4)

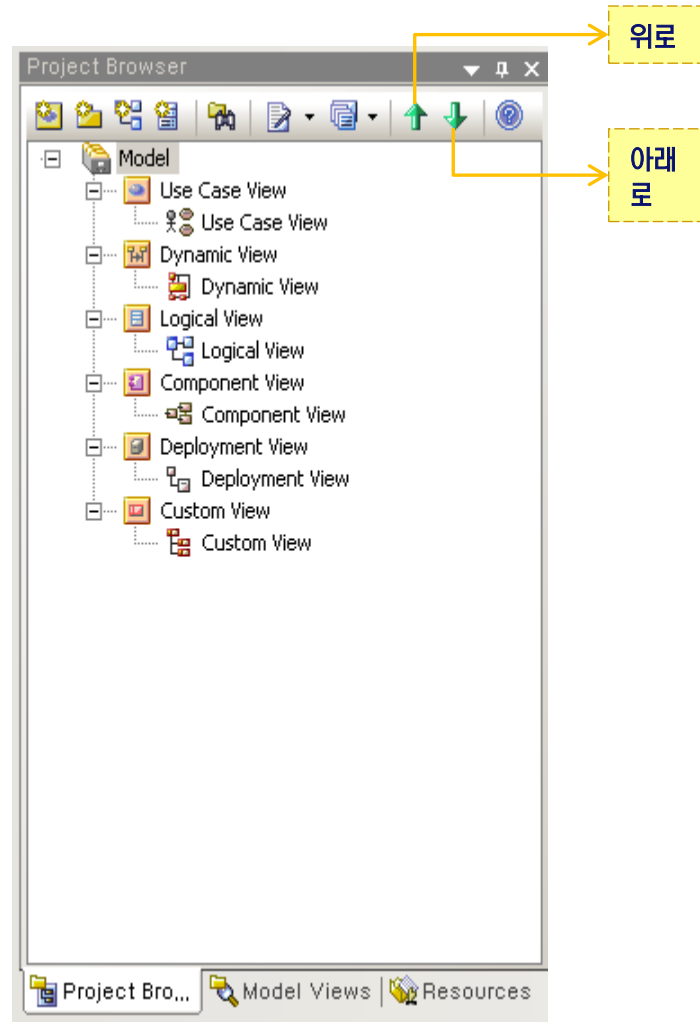
□ 레이아웃 구성

- Project Browser
- Toolbox
- Pan & Zoom

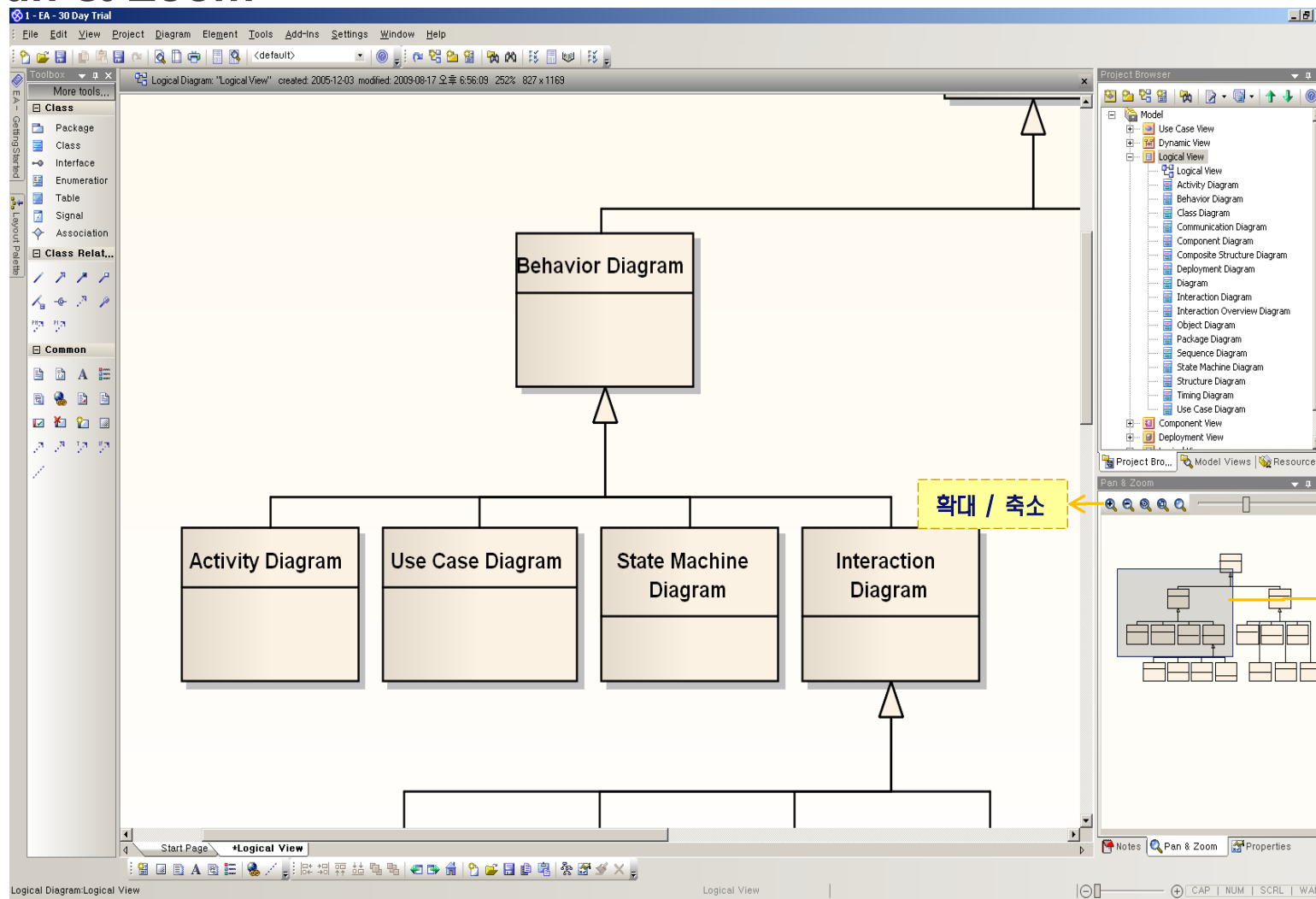


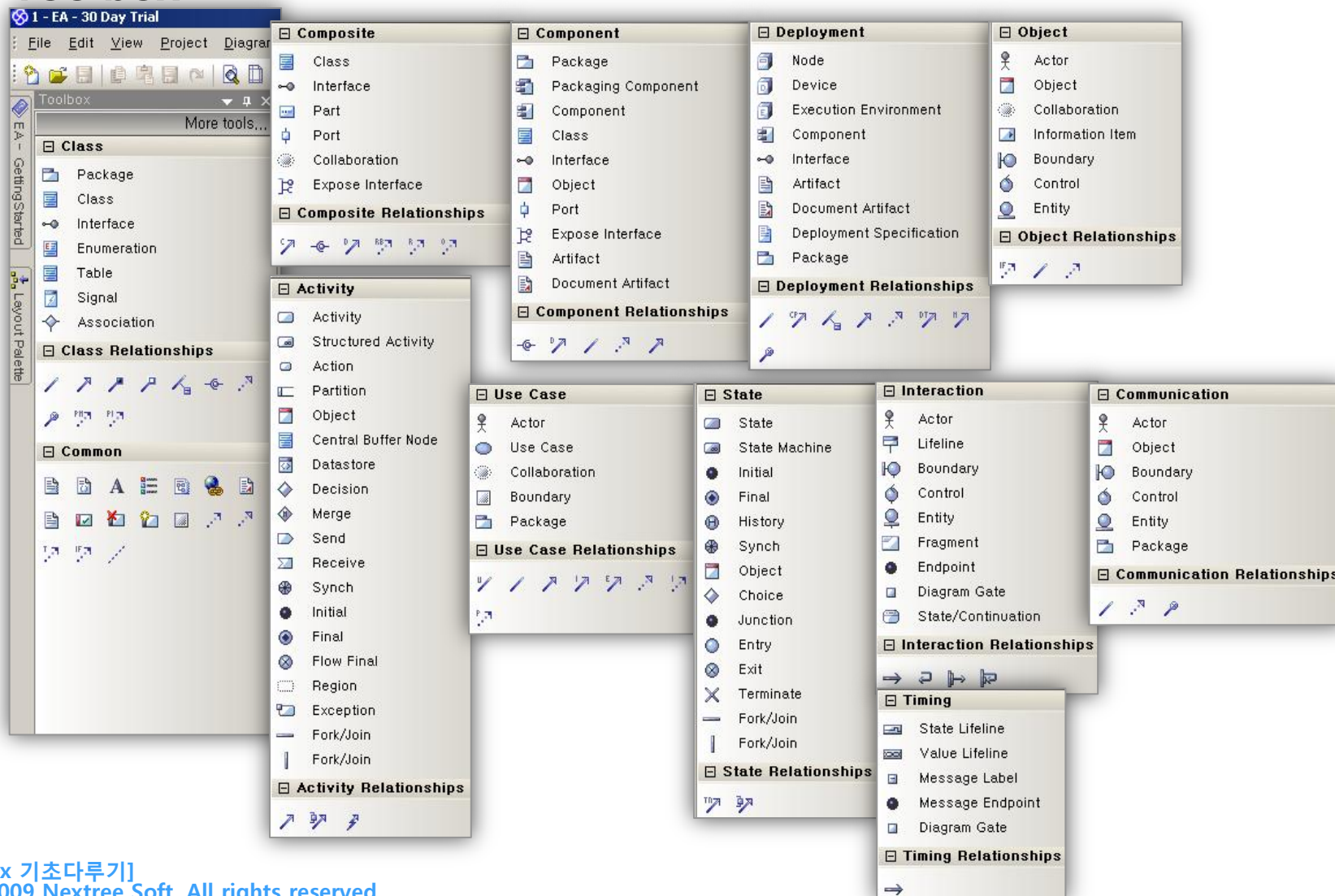
□ Project Browser

- Use Case View
- Dynamic View
- Logical View
- Component View
- Deployment View
- Custom View

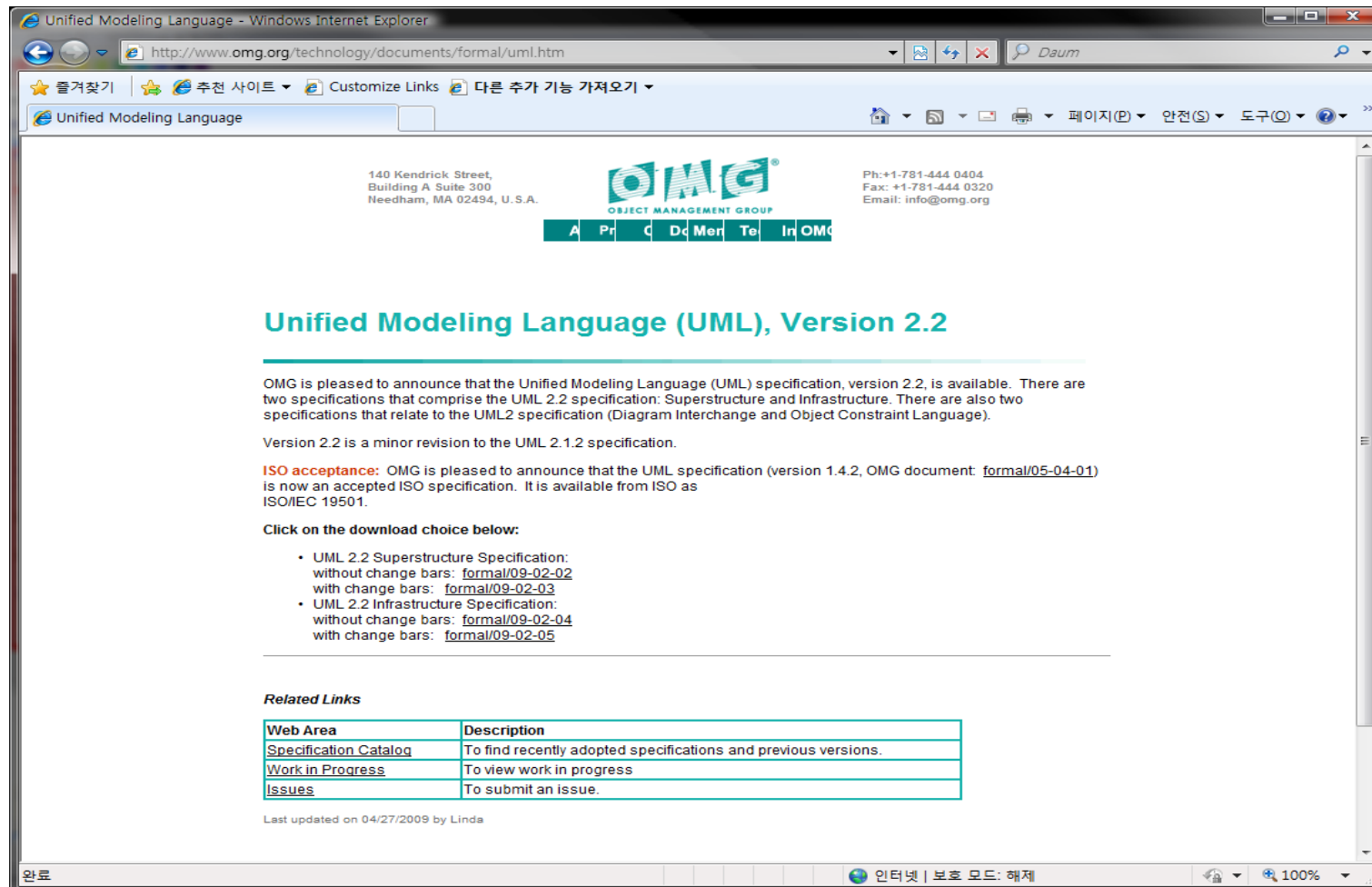


□ Pan & Zoom

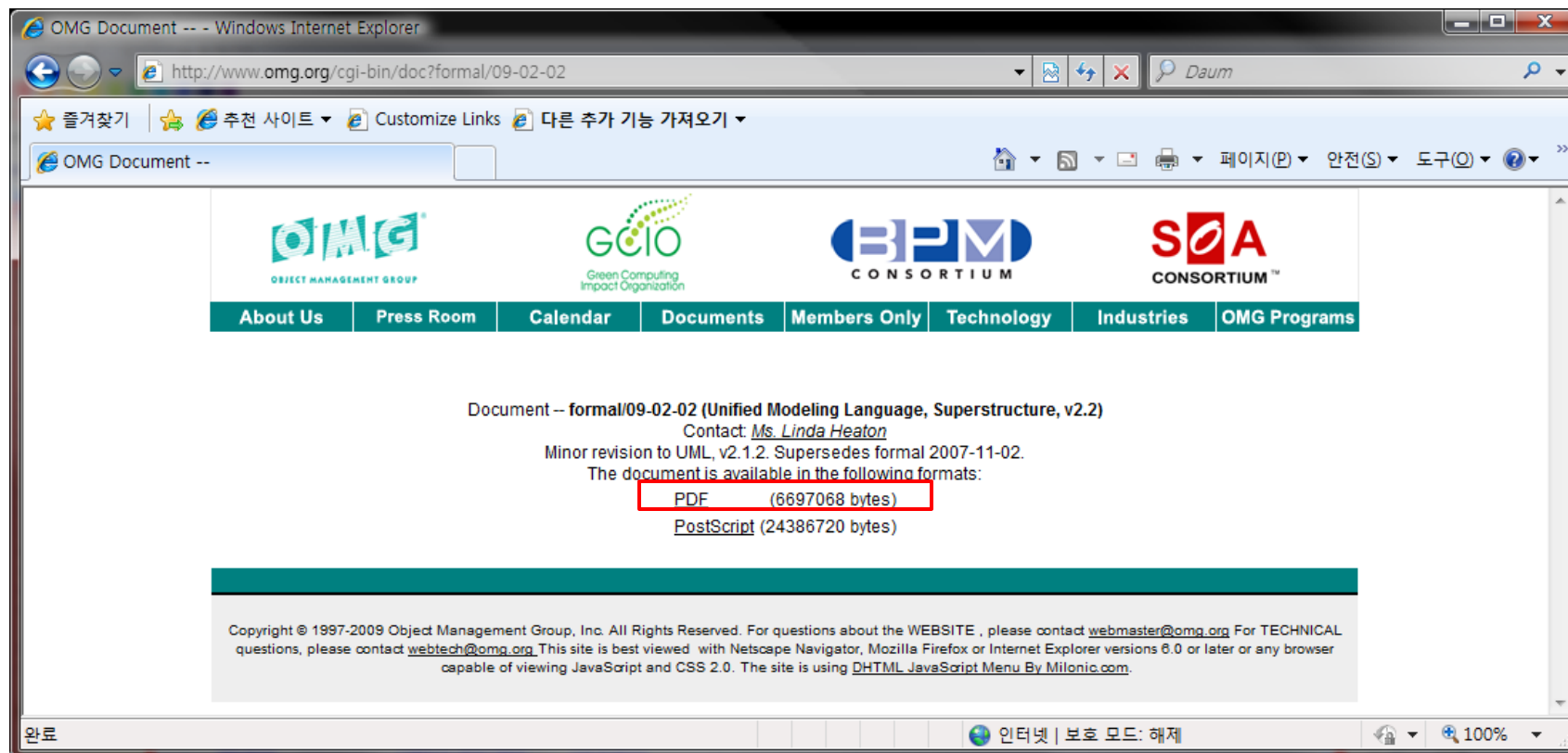




□ <http://www.omg.org/technology/documents/formal/uml.htm>



□ PDF 저장



09-02-02.pdf - Adobe Reader

File Edit View Document Tools Window Help

39 / 740 100%

Find

Bookmarks

Options

1 Scope

2 Conformance

3 Normative References

4 Terms and Definitions

5 Symbols

6 Additional Information

Part I - Structure

7 Classes

8 Components

9 Composite Structures

10 Deployments

Part II - Behavior

11 Actions

12 Activities

13 Common Behaviors

14 Interactions

15 State Machines

16 Use Cases

Part III - Supplement

17 Auxiliary Constructs

18 Profiles

Part IV - Annexes

Annex A - Diagrams

Annex B - Keywords

Annex C - Standard Stereotypes

Annex D - Component Profile Examples

Annex E - Tabular Notations

Annex F - Classifiers Taxonomy

Annex G - XML Serialization and

7 Classes

7.1 Overview

The Classes package contains sub packages that deal with the basic modeling concepts of UML, and in particular classes and their relationships.

Reusing packages from UML 2 Infrastructure

The *Kernel* package represents the core modeling concepts of the UML, including classes, associations, and packages. This part is mostly reused from the infrastructure library, since many of these concepts are the same as those that are used in, for example, MOF. The *Kernel* package is the central part of the UML, and reuses the *Constructs* and *PrimitiveTypes* packages of the InfrastructureLibrary.

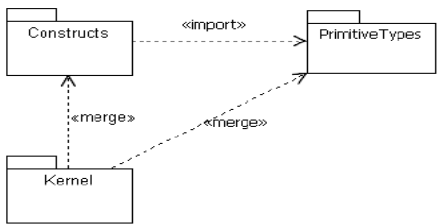
In many cases, the reused classes are extended in the *Kernel* with additional features, associations, or superclasses. In subsequent diagrams showing abstract syntax, the subclassing of elements from the infrastructure library is always elided since this information only adds to the complexity without increasing understandability. Each metaclass is completely described as part of this clause; the text from the infrastructure library is repeated here.

It should also be noted that *Kernel* is a flat structure that like *Constructs* only contains metaclasses and no sub-packages. The reason for this distinction is that parts of the infrastructure library have been designed for flexibility and reuse, while the *Kernel* in reusing the infrastructure library has to bring together the different aspects of the reused metaclasses.

The packages that are explicitly merged from the InfrastructureLibrary are the following:

- PrimitiveTypes
- Constructs

All other packages of the InfrastructureLibrary::Core are implicitly merged through the ones that are explicitly merged.



```

graph TD
    Constructs -- «import» --> PrimitiveTypes
    Kernel -- «merge» --> Constructs
    Kernel -- «merge» --> PrimitiveTypes
    
```

Figure 7.1 - InfrastructureLibrary packages that are merged by Kernel (all dependencies in the picture represent package merges)