



ENOVIA Training
Foils

LCA Administration Advanced (6)

Lifecycle Customization

Version 5 Release 11
May 2003
EDU-ENOV-E-LAL-AF-V5R11

Copyright DASSAULT SYSTEMES 2003

1

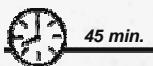
Course Presentation

Objectives of the Course

In this course, you will see how to modify lifecycles.

Targeted audience

ENOVIA LCA Programmers



45 min.

Prerequisites:

CAA V5 Programming

Copyright DASSAULT SYSTEMES 2003

2

Table of Contents

1. ENOVIA V5 : Lifecycle Customization	P.1
Objectives of the courses	P.2
Table of Contents	p.3
Planning	P.4
2. Lifecycle Objectives	p.5
Introduction	p.6
Lifecycle Management	p.7
State Machine	p.8
Lifecycle Graph	p.9
3. Lifecycle Functionalities	p.10
Lifecycle Object Model	p.11
Root Status	p.13
Status	p.14
Transition	p.15
Condition	p.18
Predicate	p.19
Operation	p.20
Command	p.21
4. How are Graphs used in GUI	p.22
Lifecycle Tree View	p.23
Lifecycle Graph View	p.24
Lifecycle Gate View	p.25
5. How to customize the default Graphs	p.26
Default Graphs	p.27
To Customize Default Graphs	p.28
6. To Sum Up	p.29

Copyright DASSAULT SYSTEMES 2003

3

Planning

In this course, you will learn:

- 🔗 Lifecycle objectives
- 🔗 Lifecycle functionalities
- 🔗 How are Graphs used in GUI
- 🔗 How to customize the default Graphs

Copyright DASSAULT SYSTEMES 2003

4

Lifecycle Objectives

Keep in mind some Concepts

- ▣ Introduction
- ▣ Lifecycle Management
- ▣ State Machine
- ▣ Lifecycle Graph

Copyright DASSAULT SYSTEMES 2003

5

Introduction

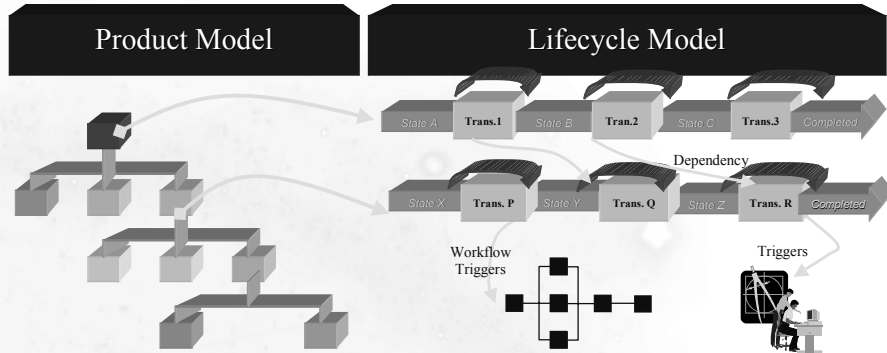
- The purpose of a lifecycle is to define a status progression of an object throughout object's existence. This status progression is largely dependent on the company's business practices, which makes the task of defining a single lifecycle to be universally used by everybody virtually impossible.
- ENOVIA LCA provides a user with a mechanism to define these lifecycles and attach them to any instance of any object.

Copyright DASSAULT SYSTEMES 2003

6

Lifecycle Management

Dependencies



- Lifecycle Model can be attached to different objects at different levels of the Product Tree
- Interdependencies between Lifecycle Models (managed through implemented "Conditions" to go across a gate)
- Lifecycle is the mega-flow of product development and Workflow is the micro-flow within that lifecycle model

Copyright DASSAULT SYSTEMES 2003

7

State Machine

- A State Machine is an oriented cyclic graph. It is composed of states linked between them using incoming and outgoing transitions. A transition can be associated with a condition and an operation. A transition can be triggered only if its condition is true, then it executes its associated operation which is composed of one or several commands.
- The commands of a state machine can be:
 - ◆ either standard commands provided by ENOVIA LCA
 - ◆ either commands programmed by the customization developer in the form of a late type extension

Copyright DASSAULT SYSTEMES 2003

8

Lifecycle Graph

- The Lifecycle Graph is a state machine which modelizes the different status of an object during its whole life. For example, a software can go through the following statuses: design, development, industrialization, maintenance, end of life. The different actors (designers, programmers, testers...) make it progress in the lifecycle by promoting it or regress by demoting it.
- ENOVIA LCA enables the definition of a lifecycle for each object.

Copyright DASSAULT SYSTEMES 2003

9

Lifecycle Functionalities

Graph description

- ▣ Lifecycle Object Model
- ▣ Root Status
- ▣ Status
- ▣ Transition
- ▣ Condition
- ▣ Predicate
- ▣ Operation
- ▣ Command

Copyright DASSAULT SYSTEMES 2003

10

Lifecycle Object Model (1/2)

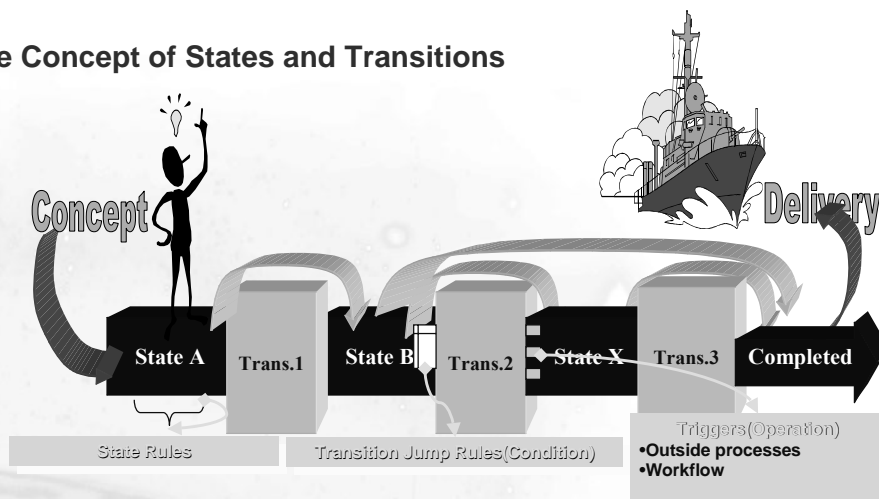
- ENOVIA Lifecycle is represented as a graph. This graph is defined by set of states and possible transitions. Each state can have multiple transitions attached to it.
- Any object can have a lifecycle graph attached to it. When promote action is invoked on the object, the Graph Manager will attempt to execute transitions associated with the current state of this object (method CATVpmGraphMng::StepForward). Transitions have a priority associated with them.
- Transitions with higher priorities will be tried first. Transition may have a condition associated with it. In this case, transition is executed only if condition (represented as a set of predicates) evaluates to true.
- An optional operation can be executed upon valid transition. Operation consists of a set of commands, each of which can have its own condition. If condition associated with a command is true, this command will be executed upon the execution of this transition.

Copyright DASSAULT SYSTEMES 2003

11

Lifecycle Object Model (2/2)

The Concept of States and Transitions



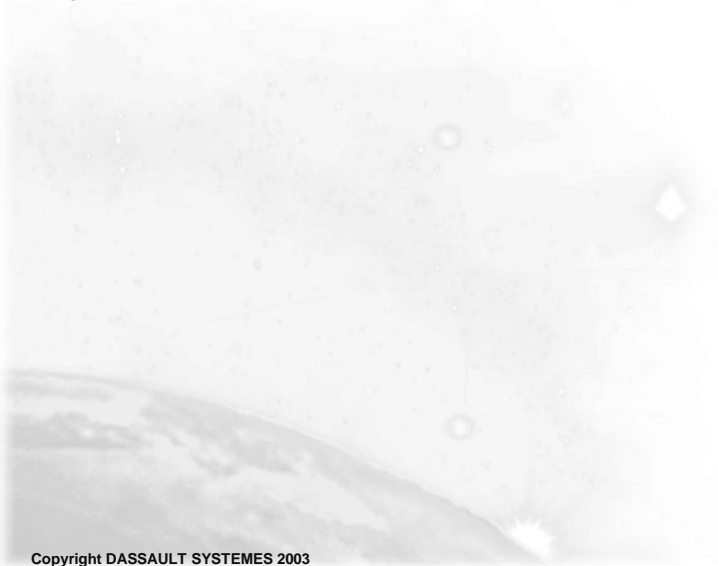
- Define product lifecycle into phases and gates
- Do not define the "How". It is too complex.
- Define "What" must be accomplished
- Define "Measurements" as to whether "What" has been accomplished or not. Defining rules to cross gates
- Break the product down into components and recursively define their (components) lifecycle
- Interlink the lifecycles from top to the bottom of the whole product definition
- When a product component moves to a higher maturity level, the system will automatically check "Gate" rules and the "Dependency" rules

Copyright DASSAULT SYSTEMES 2003

12

Root Status

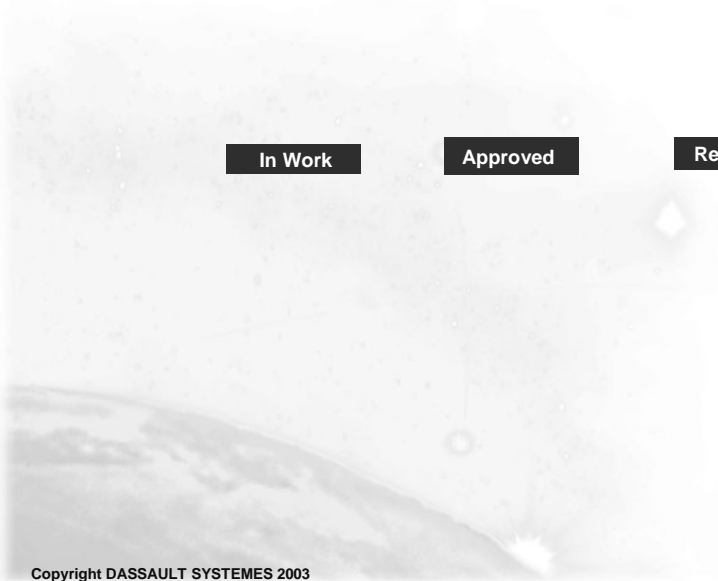
- **Root status entity contains the name of the graph, and reference to the initial state of the lifecycle. The name of the graph must be unique. These parameters are defined inside the entities: GIMaster and GIVersion.**



13

Status

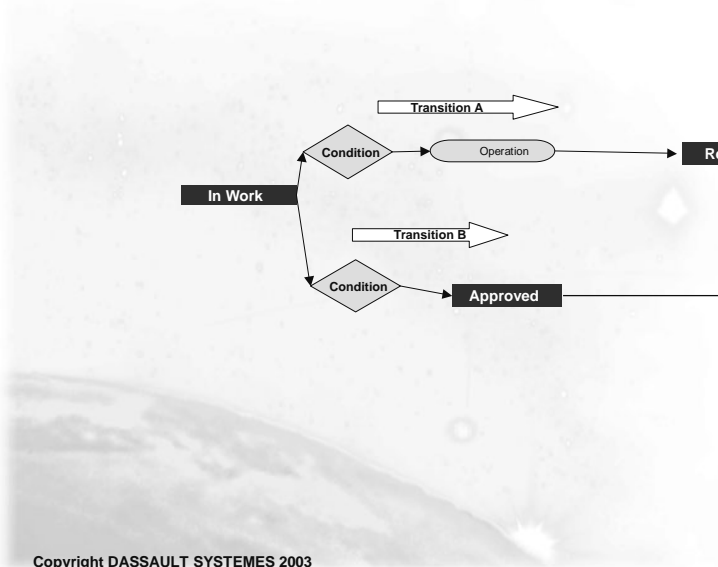
- **Status entity contains the set of available transitions and the name of the state. A State can be associated to multiple transitions.**



14

Transitions (1/3)

- Mechanism to advance an object from one state to the next.
- Can have a condition and/or post-transition operation



15

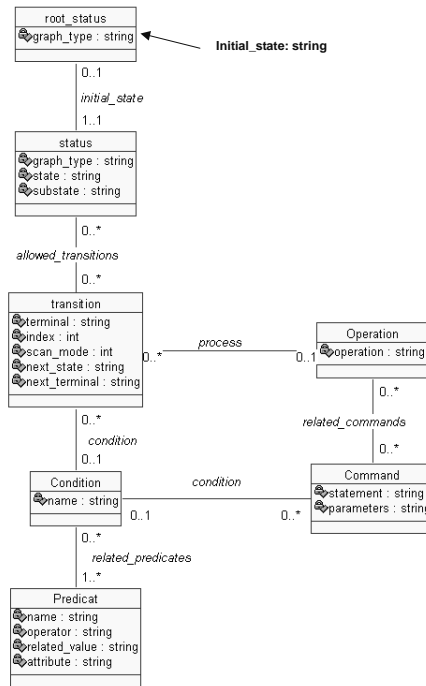
Transitions (2/3)

- Terminal attribute of a transition entity contains transition entity name.
- Next_state denotes the destination state of this transition.
- Index determines priority of the transition. It is possible to have multiple transitions from the same state with the same name, and in this case transition with the highest priority (smallest index) is executed first. If this transition cannot be executed, transition with the next highest priority is tried.
- Condition associated with a transition determines whether this transition will be executed. Operation (process) associated with transition determines actions taken if the transition is successfully executed.



16

Transitions (3/3)



Copyright DASSAULT SYSTEMES 2003

17

Conditions

- Condition consists of a set of predicates that are evaluated (if this condition is attached to transition), or executed if this condition is attached to command.
- Condition contains late type of a class to evaluate predicates

Copyright DASSAULT SYSTEMES 2003

18

Predicates

- Predicate consists of four parts: object name, operator, related_value and attribute. Related_value, attribute and operator allows one to specify expressions similar to "quantity >= 4", where attribute is "quantity", operator is ">=", and related_value is "4". Note that in this case the predicate is evaluated and it uses comparison operator. If the predicate is executed, this operator is assignment.
- Usually, if the name is NULL, the evaluated attribute is taken on the object to which this lifecycle is attached. The object name denotes the instance in a pool on which this attribute is evaluated. Each time a graph is called, the objects on which occurs the current operation are put in an object pool. Objects can be placed into pools under specific names to permit graph conditions and commands to retrieve it. It is up to the application to make sure that the object with a given name is in the pool when the predicate is evaluated.
- Predicates are Boolean expressions on current or related objects.

Predicate1: 'Release Approval.status == "complete"
Predicate2: 'Design Maturity > 85%'

Operation

- Operation contains a set of commands that are executed when transition had taken place.

Command

- **Command has a statement, parameters and condition. Statement can be a late type of a class which will execute this command. Condition contains a set of predicates that are executed as described above.**

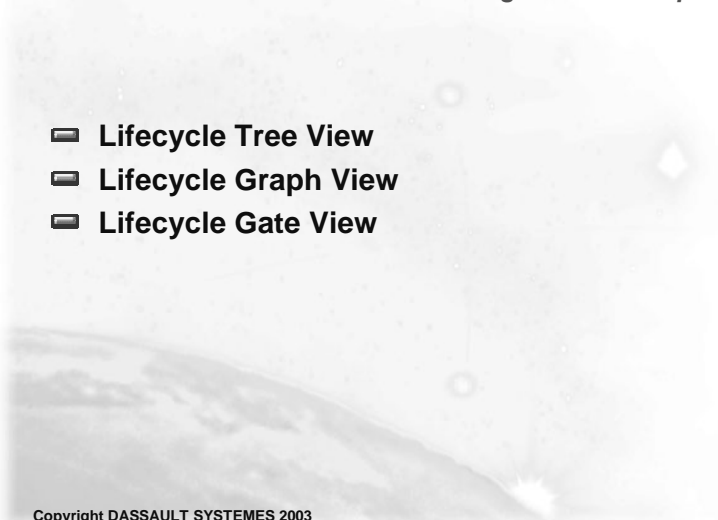


Copyright DASSAULT SYSTEMES 2003

21

How are Graphs used in GUI

Lifecycle GUI is implemented as a navigator View with three major areas: tree of lifecycles, lifecycle graph definition and lifecycle gate definition. These views can be resized at will or minimized using one-touch expand



- ▣ Lifecycle Tree View
- ▣ Lifecycle Graph View
- ▣ Lifecycle Gate View

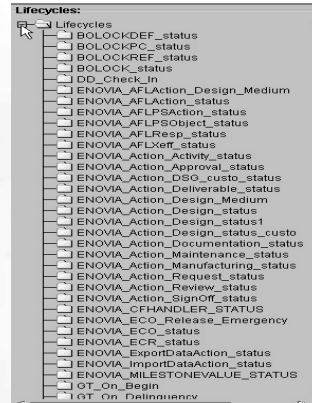
Copyright DASSAULT SYSTEMES 2003

22

Lifecycle Tree View

- Lifecycle tree view represents a list of all available lifecycles. It also provide a user with a popup menu which will enable him/her to create new lifecycle graphs.

- Expand the folder and the list of lifecycles now shows up.



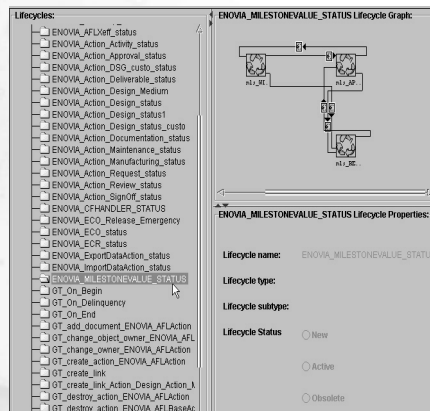
Copyright DASSAULT SYSTEMES 2003

23

Lifecycle Graph View

- At this time, lifecycle graph is displayed as its transition matrix. In future, a graphical representation should also be available.
- The states where transitions originate from are located in the first column of the table, transition destinations are located in the first row. Cells at their intersection display a gate name if the gate from From state to To state exists.

- View the lifecycle graph for an existing Lifecycle.



Copyright DASSAULT SYSTEMES 2003

24

Lifecycle Gate View

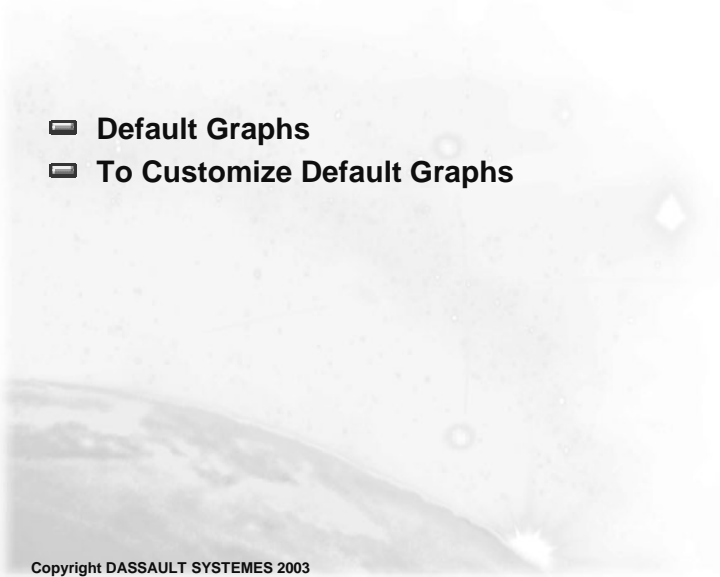
- Lifecycle gate view is a notebook that allows user to view or enter complete definition of a state.



25

How to customize the default Graphs

You will be able to import your new Graphs



26

Default Graph

- For the **PRODUCT** Package a "default" customization graph is available for all the objects
- That file is delivered in the `$OS/reffiles/sample` directory. It is called `VPM_VPMObject.VGraph`
- All the **PRODUCT** instances inherit this default graph if a new one is not attached to its

Copyright DASSAULT SYSTEMES 2003

27

To Customize Default Graphs

- Using the **ENOVIA LCA GUI**, copy / paste the `VPM_VPMObject_V_status`
- Change its name, for instance: `VPMItemInstance_V_status`
- **Export it:**
 - **Connect as dictionary owner**

```
catstart -run "VPMGRAPHADM Export VPMItemInstance_V_status  
exportdirectory/exportfilename"
```
- **Edit it to add a new state and transitions for instance**
- **Import the new graph**

```
catstart -run "VPMGRAPHADM Import exportfilename"
```

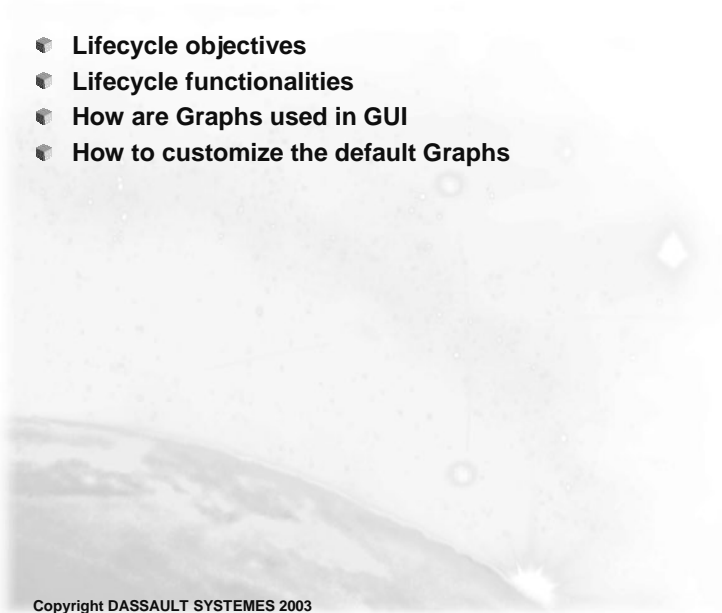
Copyright DASSAULT SYSTEMES 2003

28

To Sum Up

In this course you have seen :

- Lifecycle objectives
- Lifecycle functionalities
- How are Graphs used in GUI
- How to customize the default Graphs



Copyright DASSAULT SYSTEMES 2003

29