

An Introduction to UML

The Dynamic Model

by Geoffrey Sparks

All material (c) Geoffrey Sparks 2001

www.sparxsystems.com.au

Table of Contents

THE DYNAMIC MODEL	3
INTRODUCTION TO UML	3
SEQUENCE DIAGRAMS.....	3
<i>Purpose</i>	3
<i>Notation</i>	3
ACTIVITY DIAGRAMS	5
<i>Purpose</i>	5
<i>Notation</i>	5
STATE CHARTS.....	8
<i>Purpose</i>	8
<i>Example Notation</i>	8
SUMMARY	9
RECOMMENDED READING.....	10

The Dynamic Model

This paper describes how to model the dynamic aspects of software systems using UML notation and semantics. The three topics covered are sequence diagrams, activity diagrams and state charts. An explanation is given of each and how they fit into the overall model structure.

Introduction to UML

The Unified Modelling Language (UML) is, as its name implies, a modelling language and not a method or process. UML is made up of a very specific notation and the related grammatical rules for constructing software models. UML in itself does not proscribe or advise on how to use that notation in a software development process or as part of an object-oriented design methodology.

This paper focuses on the modelling of dynamic behaviour using UML notation and semantics. Dynamic interaction and behaviour in UML is broken down into three main categories:

1. Interactions between object instances at run-time. This is modelled using Sequence diagram and/or Collaboration diagrams. This paper will only discuss Sequence diagrams, as Collaboration and Sequence diagrams are semantically identical.
2. General activity descriptions covering business process and user interaction. Activity diagrams and Business Process diagrams are used for this purpose.
3. State changes over time. UML supports State charts for modelling state changes.

You can find out more about UML from the books mentioned in the suggested reading section and from the UML specification documents to be found at the Object Management Groups UML resource pages: <http://www.omg.org/technology/uml/> and at <http://www.omg.org/technology/documents/formal/>

Sequence Diagrams

Purpose

Sequence diagrams are used to display the interaction between users, screens and object instances within the system. They provide a sequential map of message passing between objects over time. Frequently these diagrams are placed under Use Cases or Components in the model to illustrate a scenario, or common set of steps followed in response to an event that generates an outcome. The model includes what initiates activity in the system, what processing and changes occur internally and what outputs are generated. Often, the object instances are represented using special stereotyped icons - icons exist for boundary objects, controllers and persistent entities.

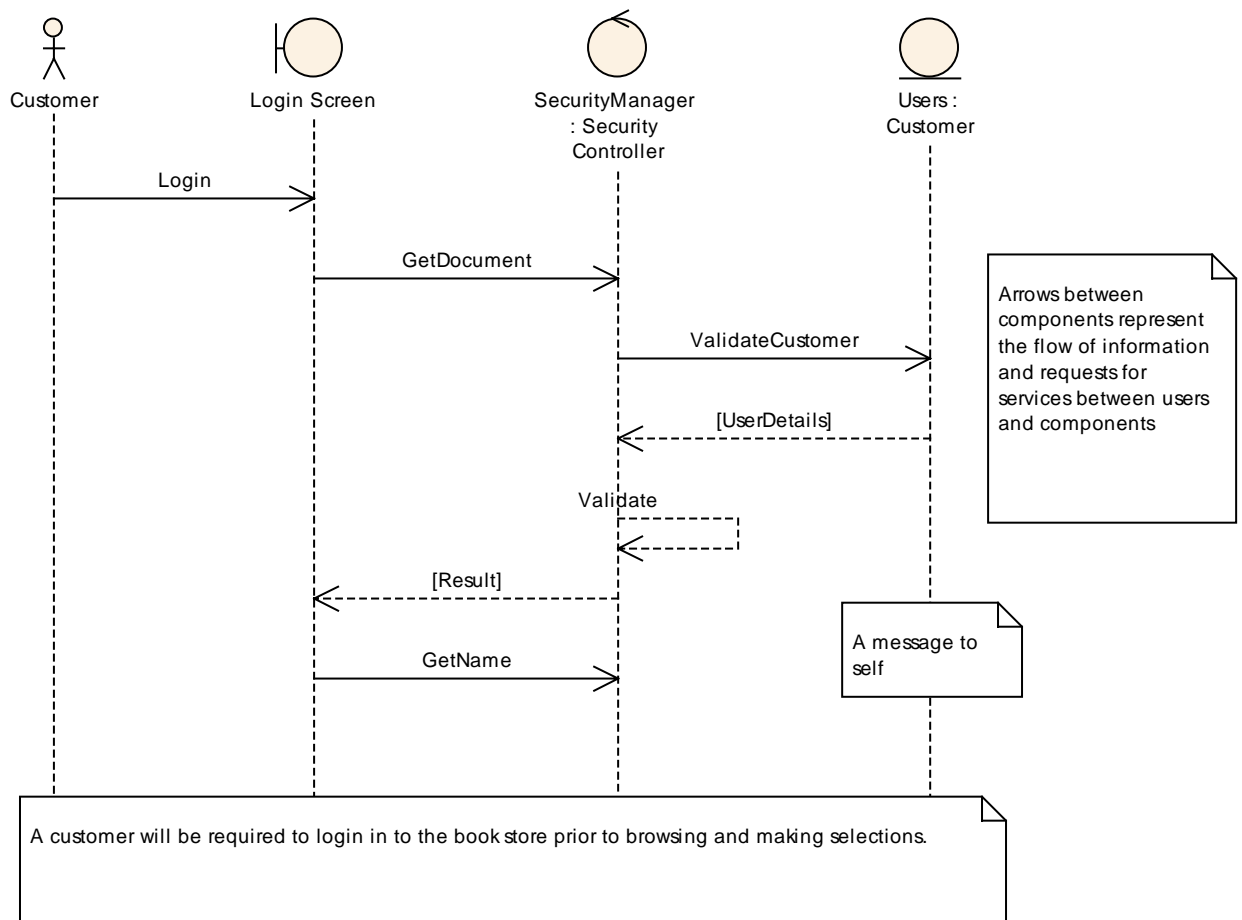
Notation

The notation used is typically a horizontally deployed set of actors and object instances, each having a vertical lifespan bar. Messages (usually method calls but may also

represent messages passed using message queuing services and other events) are drawn from one object to another with an arrow indicating the direction of flow.

A Sequence diagram is representation of the messages passed between object instances, so generally the messages will map during the design phase of the project to class operations. The initial Sequence diagrams indicate what public behaviour is needed for objects to get work done and cooperate, and during design, Sequence diagrams are used to show what actual responsibilities and operations are assigned to which classes.

The example diagram below demonstrates some features of Sequence diagrams



Note the use of stereotyped icons to display particular objects: for example the user interface (Login Screen) is displayed with a Boundary stereotype and the User as an Entity stereotype. These help visually differentiate object roles during analysis.

Activity Diagrams

Purpose

Activity diagrams are used to show how different work flows or processes in a system are constructed, how they start, the many decision paths that can be taken from start to finish and where parallel processing may occur during execution.

An Activity diagram generally does not model the exact internal behaviour of a software system (like a Sequence diagram does) but rather it shows the general processes and pathways at a high level. Often it is used to model business activities (such as Selling Books or Manage Inventory), and may be at a very high level. Activities will generally be realised by one or more Use Cases, the Activity describes the process that is undertaken and the Use Case how an Actor will use the system to realise all or part of an Activity.

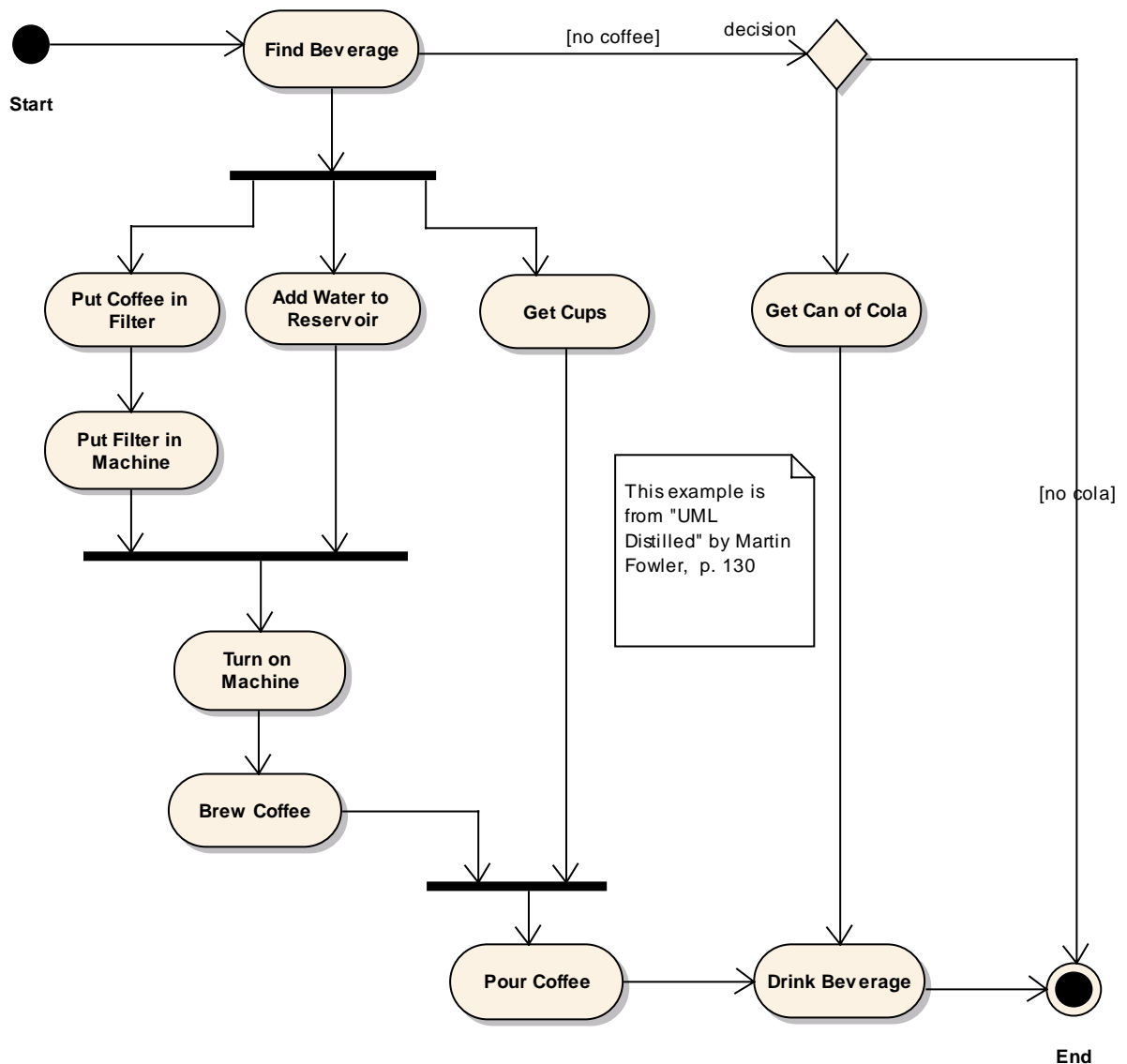
Notation

Activity Diagram Notation

Standard UML notation uses a rectangle with rounded corners to depict an Activity. Activities may be joined by process flows or events. In addition, a Decision node can model divergent behaviour based on a condition. Typically a Start and End node are defined to complete the full Activity representation. Synchronisation points may also be defined to illustrate how processing may be carried out in parallel, then synchronised at a point before further activity is undertaken.

The example below illustrates most of these points – it describes the process surrounding the acquiring of a beverage from a vending machine. The rounded rectangles are Activities, the diamonds are Decision points and the horizontal black bars are synchronisation points.

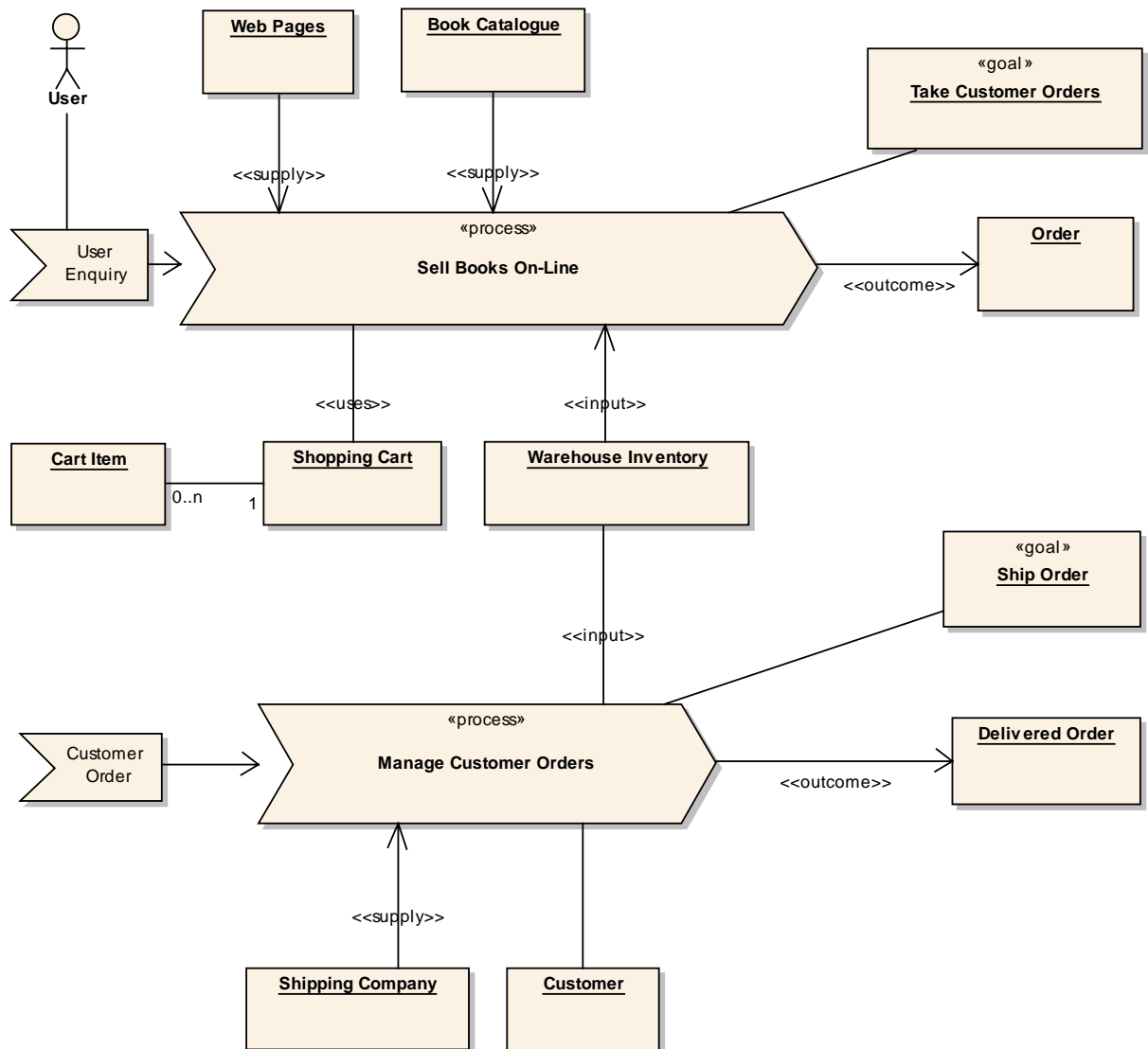
Note that if you were building drink vending machine software, only some of the activities are relevant, although the diagram as a whole provides a good picture of what the total process of getting a beverage is all about. Further analysis would be needed to determine which parts of this model could be implemented or supported in software.



Business Process Notation

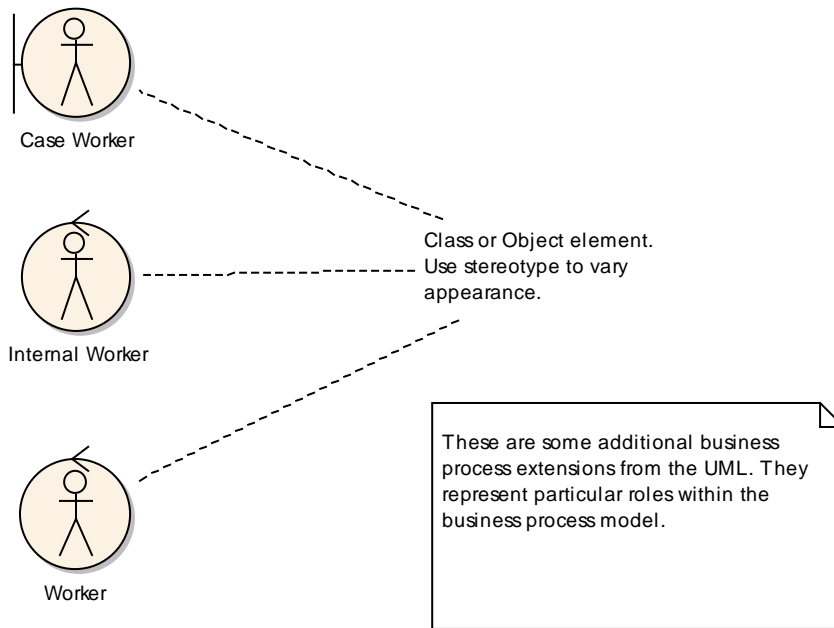
In addition to the standard Activity diagram notation as described in UML, several extensions have been defined to more accurately portray business processes. In particular, the extensions proposed by Hans-Erik Eriksson and Magnus Penker (see suggested reading) provide a useful way of approaching this aspect of system analysis.

These extensions provide further stereotypes such as business process (represented as an elongated rectangular arrow), goal, input and output objects. The example below shows how these elements may be combined to produce a high level picture of business activity.



The business process model lets you model the main business activities. A 'Process' can cut across many departments or divisions of a larger entity. It describes something a business does as part of its normal activity; focussing mainly on the inputs, outputs, goals and key events that drive the process.

UML also defines some additional stereotyped icons for use in business process modelling. The example below shows some of these. Note how these icons re-use the boundary, controller and entity icons from the Sequence diagrams.



State Charts

Purpose

State Charts are the third dynamic model that UML uses to capture system changes over time. Any object at run-time that has non-constant instance variables has some potential state. The actual state of an object depends on the values of its instance variables. Typically State charts are associated with particular classes (often a class may have one or more state charts to fully describe its potential states).

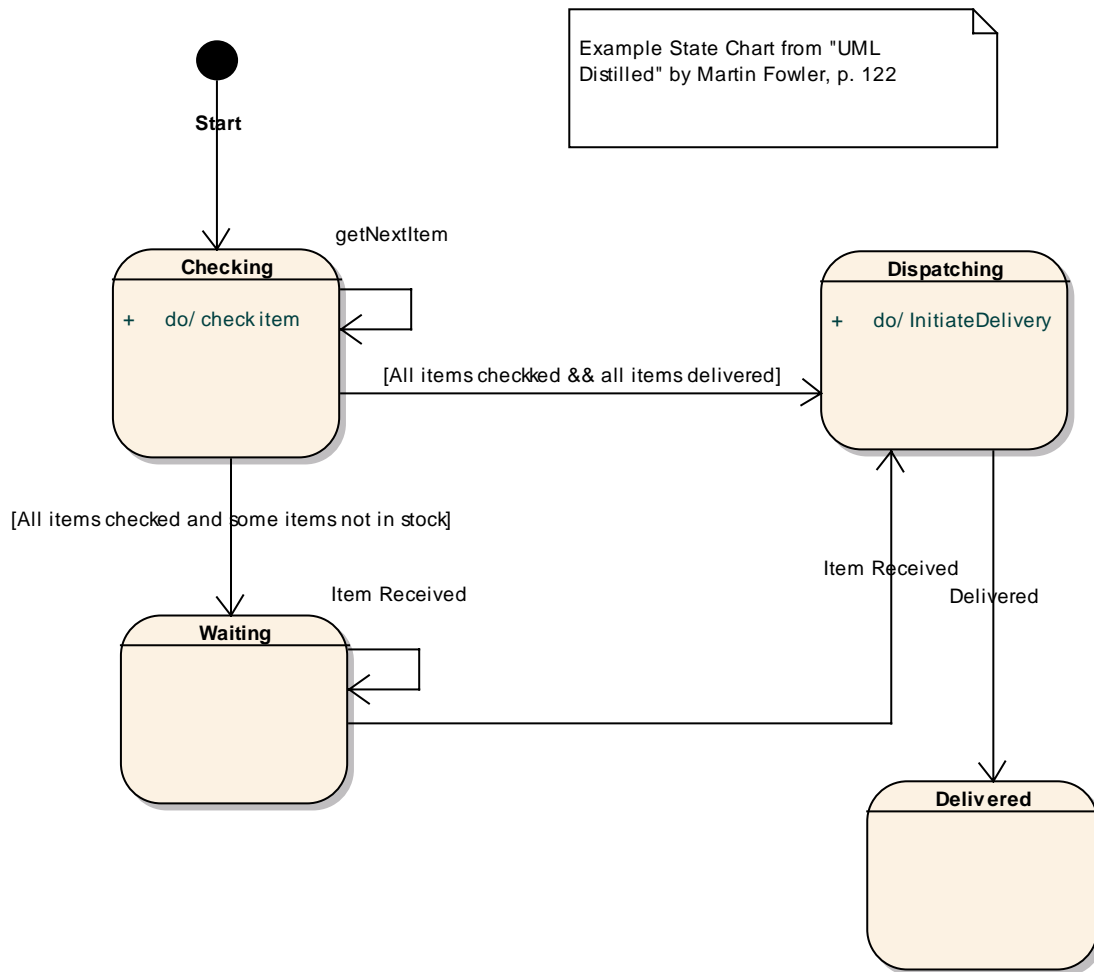
In UML a State is displayed as a rounded rectangle with optional compartments for attributes, events and internal activities. State flows or transitions are drawn between States, usually with guard conditions and rules governing how and when an object may transition from one state to another.

Careful use of State charts will help reveal the instance variables required to fully maintain the states of an object and the pre-conditions necessary to transition (update instance variables) to another state.

States are usually named according to their condition – for example ‘Checking’, ‘Waiting’ and ‘Dispatching’ are all active conditions an object can be in while waiting to transition to another state or end the cycle completely.

Start and end nodes represented as solid and empty circles are used to represent the beginning and end of all transitions.

Example Notation



Summary

UML provides good support for dynamic modelling of software systems, from the early analysis (Activity diagrams and Business Process modelling) through to Use Case functionality (Sequence diagrams) and class behaviour (State charts). Each diagram type helps to capture information about the system as a whole and further refine the design and implementation details necessary to complete the software system.

Recommended Reading

Sinan Si Alhir, *UML in a NutShel*.

ISBN: 1-56592-448-7. Publisher: O'Reilly & Associates, Inc

Doug Rosenberg with Kendall Scott, *Dynamic Driven Object Modeling with UML*.

ISBN:0-201-43289-7. Publisher: Addison-Wesley

Geri Scheider, Jason P. Winters, *Applying Dynamics*

ISBN: 0-201-30981-5. Publisher: Addison-Wesley

Ivar Jacobson, Martin Griss, Patrik Jonsson, *Software Reuse*

ISBN:0-201-92476-5. Publisher: Addison-Wesley

Hans-Erik Eriksson, Magnus Penker, *Business Modeling with UML*

ISBN: 0-471-29551-5. Publisher: John Wiley & Son, Inc

Peter Herzum, Oliver Sims, *Business Component Factory*

ISBN: 0-471-32760-3 Publisher: John Wiley & Son, Inc