In this contribution, we report on a diploma thesis project which combined two major aims:
(1) to try out UML as a specification language for a specific software process model, and
(2) to study the general aptness of UML for the software process modelling task. Due to
the limited space we can only sketch our work and results here, for a longer version of this
article cf. [BH03], for the diploma thesis [Be01].

The EOS\(^1\) model, developed by the second author [He97, He03], was taken as subject of
our experiment, since there was no prior elaborated formalisation (which might have bi-
ased the modelling), it has special features (systematics, orthogonality, transparent struc-
ture) supporting such a formalisation, and because there is a good mixture of static and
(non-linear, sufficiently complex) dynamic process features. The skeleton for all EOS
process elements is the **building block hierarchy** consisting of systems, components (in-
cluding subcomponents of arbitrary depth) and modules. Building blocks are the central
anchor for development cycles, activities, artifacts, management and quality assurance ac-
tions; they determine the repository structure and project documentation. The recursive,
scalable structure of this hierarchy induces a similar structure of software processes. **De-
velopment cycles** (the central feature for grouping process elements) always consist of the
four development phases **analysis**, **design**, **implementation**, and **operational use**. They can
hierarchically be decomposed into further cycles according to the system structure.

The EOS metamodel containing its central concepts and their interdependencies is de-
picted in fig 1 as a UML static structure diagram.

For modelling the dynamic aspects of the EOS model we used activity diagrams in the
first place. The control flow of serial or parallel EOS activities could be shown well, but
problems occurred with highly flexible dynamic EOS constructs. Cycles for subordinate
building blocks can be started whenever needed. Activity diagrams of UML 1.3 could not
express this dynamic enactment independent of the current activity and with an arbitrary
multiplicity. Furthermore, splitted threads of control flow have to be synchronized later
on. We chose to model this feature by defining a special transition stereotype with the
required meaning. Among other problems we had to deal with very large diagrams, and
we had liked to use labelled elements for connecting items between partial diagrams.

A feature for which state diagrams proved useful is the capability of a building block to
be split and to enact new development cycles for the descendant (new) bulding blocks. It
was not possible to draw direct dependency links between the different state models of

\(^1\)Evolutionary, Object-oriented Software development
building blocks, but the dependencies could be expressed using constraints. Other UML
diagram types were used for more detailed modelling tasks, e.g. sequence diagrams for
the splitting of development cycles into sub-cycles. Use case, component, and deployment
diagrams were not used because they either would have added too much redundancy or
turned out as less suitable for the given task of process modelling and design.

Our resume: In general UML is well-suited for software process modelling tasks. The
various diagram types of UML support a multi-perspective view on the heterogeneous
aspects of software processes and helped us to find various errors in the EOS specs and
in earlier versions of the diagram package. The extension mechanisms of UML were
thoroughly and successfully used. However, problems arise whenever complex situations
like cross-hierarchical associations or highly dynamic structures are to be modelled.

In the upcoming UML version 2.0 several dynamic features have been overworked, promising
to cope well with some of the indicated problems. For future work which might cover
further details of the EOS model we therefore plan to try out UML 2.0.

References

[Be01] Beyer, M.: Verwendung von UML zur Modellierung von Software-Entwicklungsprozes-

http://www.mathematik.uni-marburg.de/~hesse/papers/bh03.pdf.


[He03] Hesse, W.: Dinosaur meets archaeopteryx? or: Is there an alternative for Rational’s Unified