There are at least three levels of abstraction to a
simulation programming language. At the highest
level, we ask questions such as "How does the con-
tceptual framework of the language aid in formulating
the model" or "Are we aid in focusing attention
on the essential elements and relationships which
must be modelled?"

With an affirmative answer to these questions, we
proceed to model implementation. At this level we
might ask: "Does the language provide a natural
and convenient way of expressing the model? If a
particular feature of the model does not map well
into the language 'world-view' are there provisions
for language modification or extension?"

Assuming the language does fit our conceptual needs
and is convenient to expressing the problem, we come
to a final and very important, though often over-
looked, level of abstraction. "What aids does the
language provide in the area of model debugging?"
Large simulation models (and even small ones!)
don't often execute properly the first time they are
run. "What consistency checks are provided in
the execution of the model? What diagnostic aids are
provided for pinpointing trouble spots?"

SIMSCRIPT II.5 is making significant progress on all
three levels of abstraction.

First, at the conceptual level, the long-established
"world-view" -- consisting of entities, attributes,
sets and discrete events -- has been extended to
include processes and resources. A process combines
the concepts of an entity and a sequence of discrete
events into one element. A resource provides a
means of modelling competition among processes for
objects which are in short supply. Included in the
resource concept are automatic queueing of processes
for unavailable resources and automatic restart of
processes for which the resource later becomes avail-
able.

Second, and still at the conceptual level, there are
many systems which are more readily viewed as con-
tinuous systems or, in the more general case, as
combined discrete/continuous systems. The constructs
required to add this capability to SIMSCRIPT II.5
have been implemented as a user-level extension.
This includes the capability to do numerical inte-
gration of differential equations, restart integra-
tion at discrete event times, and prepare graphical
output.

At the model implementation level, in addition to
the new statements which support processes and re-
sources, considerable attention has been given to
the structure of programs. In particular the "IF"
statement has been restructured to conform to "mod-
er" or "structured" programming conventions. These
improvements greatly increase, the potential for self-
documenting programs when combined with the free-
form of the language (making possible indentation,
etc.)

Finally, at the execution/debugging level, much
effort has been put into giving the user the tools
and information he needs to develop and verify the
correctness of large simulation models. Such de-
velopment aids as full cross-reference listings of
modelling elements, on-line diagnostics for time-
sharing users, interactive model execution and ex-
tended compatibility with FORTRAN are all imple-
mented as an attempt to reduce the elapsed time
from model conceptualization to fully working sim-
ulation programs.

As the various simulation programming languages
continue to evolve, it appears that they are coming
closer together at the conceptualization level.
The major differences would appear to be in the
method of implementation, which in turn more di-
rectly affects the user in his day-to-day use of the
language.

REFERENCES

Delfosse, C. M., "Continuous Simulation and Com-
 bined Simulation in SIMSCRIPT II.5." C.A.C.I., Inc.,
1976.

Kiviat, P. J., H. M. Markowitz and R. Villanueva,
"SIMSCRIPT II.5 Programming Language." C.A.C.I.,

Russell, E.C., "Simulating with Processes and