CHAPTER 21 BRIEF HISTORY OF ASCEND

ASCEND is an acronym which stands for **Advanced System for Computations in ENgineering Design**[1]. The name ASCEND first appeared in print in 1978. The ASCEND programs are a series of modeling systems that Arthur Westerberg and his graduate students at Carnegie Mellon University have developed since that time.

ASCEND I

Dean Benjamin developed the first ASCEND system. It was an interactive system in Fortran. Chemical engineering students at Carnegie Mellon University used this system from about 1978 to 1982 to carry out multicomponent flash calculations. It supported the senior design project.

ASCEND II

Almost in parallel, Michael Locke developed the ASCEND II simulation system for his PhD thesis [1981]. ASCEND II allowed users to create models by configuring them using predefined types of parts. System maintainers defined the library of types, each in the form of seven handcrafted Fortran subroutines. These routines computed the space needed for the data when instancing a part, generated numerical values for the partial derivatives and the residuals of the equations that the part instance provided to the overall model, generated proper variable and equation scaling and the like. Michael Locke used this system to create models involving a few thousand equations to test variants of the Sequential Quadratic Programming algorithm. Tom Berna and he developed for optimizing structured engineering systems. Selahattin Kuru also used ASCEND II to generate and test solution algorithms for dynamic simulation that he subsequently developed for his PhD. Two companies used the software architectural design of ASCEND II to create their own internal equation-based modeling systems.

Experience at this time demonstrated that models involving several thousands of equations were solvable and could even be efficiently optimized. The question of interest moved from how to solve large equation-based models to how to aid an engineer to pose them, debug them and get them to solve.

---

1. ASCEND originally stood for “Advanced System for Chemical ENgineering Design” but the second generation system and following are not discipline specific, thus the name change.

In 1983 Dean Benjamin proposed the first version of a modeling language for posing complex models. Larry Gaydos and Art Westerberg further developed this language in the spring of 1984.

ASCEND III

In 1984 Peter Piela undertook a PhD project with Art Westerberg to "reduce the time needed to create and solve a complex model by one order of magnitude." He developed what became ASCEND III. He had the help of two Carnegie Mellon University undergraduate students, Tom Epperly and Karl Westerberg, and of Roy McKelvey, a member of the faculty in the Design Department in Fine Arts. This team developed this system on the Apollo workstation and in Pascal. It comprised three parts: a modeling language and compiler, an interactive user interface and a suite of solvers. The language used object-oriented principles, with the exception of hiding of information. Modelers define types to create model definitions. A type (called a model in ASCEND) is a collection of variables and complex parts whose types are previously defined and the definition of the equations that model is to supply. A model can also be the refinement of a previously defined type. The language fully supported associative arrays and sets. For example, a distillation column is an array of trays. It also supported deferred binding by allowing one to reach inside a part and alter one of its parts to be a more refined type. The language and its compiler obviated the need to have a system programmer write the seven subroutines needed in ASCEND II.

The interactive user interface supplied the user with organized access to the many tools in the ASCEND III system. There were tools to load model definitions, to compile them, to browse them, to solve them, to probe them, to manipulate the display units (e.g., ft/s) for reporting variable values, to create reports and to run methods on them. One could even point at a part and ask that it be made into a more refined type (triggering the compiler to restart). As previously solved values were not overwritten, they became the starting point for the more complex model. Thus one could creep up on the solution by solving more and more complex versions of a model. Many of the tools were there specifically to aid the user in debugging their models as they tried to solve them. A tool could tell a user that the model appeared to be singular and why. Another set of tools aided in picking a consistent set of variables to fix before solving. Browsing allowed the user to look at all parts of the model. It was easy to check the configuration of a model. One could ask that parts of a model be solved one at a time.

Experience by Peter Piela, Oliver Smith, Neil Carlberg and Art Westerberg with ASCEND III demonstrated very clearly that *skilled* modelers could develop, debug and solve very complex models much more rapidly than they could with previously available tools, easily

meeting the original target of a order of magnitude reduction in time required.

ASCEND IIIc

In the fall of 1992, Kirk Abbott and Ben Allan approached Art Westerberg and said they wanted to convert the ASCEND III system from Pascal into C. They would also use Tcl/Tk for the interface. With these changes, the system would then run on most Unix workstations. Tom Epperly and Karl Westerberg had already created a C version for the compiler and solver. Abbott and Allan wanted to do this conversion even after they were warned that converting the system would take time that they could be using to do more apparently relevant work to complete their PhD theses. They insisted[2]. They were aided by Tom Epperly who, although located remotely, worked with them on the compiler. In eight months and putting in excessively long hours, they had a working system that could could mimic most of the capabilities of the ASCEND III system.

Several students and a few people outside CMU could now use the system for modeling. Bob Huss and Boyd Safrit performed the hardest testing when they used ASCEND IIIc to model nonideal distillation processes. They developed and solved models involving up to 15,000 equations. Using a rudimentary capability for solving initial value problems, Safrit also solved dynamic models.

With use came the recognition of a need for improvements.

Attempts to teach ASCEND to others showed that it was a great system to speed up the modeling process for experts. Nonexperts found it nearly impossible to reuse models contained in the ASCEND libraries. The library for computing the thermodynamic properties of mixtures was particularly elegant but almost impossible to reuse. Modelers would reinvent their own properties models quickly, unable to use the library models.

Models larger than about 17,000 equations took more space than our largest workstation could provide. The models by Huss and Safrit were pushing the limits. Abbott and Allan established the goal to increase the size possible by a factor of at least ten, i.e., to about a quarter of a million equations. ASCEND needed to solve models more quickly. Without counting the increases coming from faster and larger hardware, the goal here too was a factor of ten. If solving were to be that fast, then compiling would stand out as unacceptably slow. The goal: ten times faster.

Abbott, with Allan, exposed a new style for modeling in ASCEND. He created prototypes of the various repeating types that occur in a model.

---

2. It should be understood that Art Westerberg was tbrilled they insisted on doing this conversion.

The compiled equations and other data structures to define these prototypes then became available for all subsequent instances of parts that were of the same type as the prototype. Only the instance data needed to be developed separately. Demonstrated impact on compile times was dramatic.

Abbott, with Allan, looked at how to speed up the solving times. The new twist was to use the model structure as defined by the model definition to expose a global reordering for the model equations before presenting them for solution. The time to solve the linear Newton equations as the inner loop of solving nonlinear equations dropped by factors of 5 to 10.

ASCEND IV

Ben Allan has taken a lead role and worked with Mark Thomas, Vicente Rico-Ramirez and Ken Tyner to produce the next version of ASCEND, ASCEND IV. Playing the role of tester, an undergraduate, Jennifer Perry, demonstrated that Allan's introduction of parameterized types dramatically increased the reusability of the model libraries, converting it into an almost automatable exercise. Adding language constructs to permit the modeler to state what constitutes misuse of a model leads to the system generating diagnostics the model itself defines. Allan also completely revised the data structures and the interface between ASCEND and its solvers so that adding new solvers is much less work and so the solvers in ASCEND themselves become separable from ASCEND and usable by others.

Allan also defined the addition of NOTES to ASCEND which are like methods except they are not understood by the ASCEND system itself. Rather they can be passed to programs outside ASCEND. An example includes documentation notes which a documentation manager can use to compose answers to queries about what is in an ASCEND model. Another is a note that contains a bitmap description of a part that an external package could use to draw a symbol of that part.

ASCEND IV can now handle discrete variables and constants (logical, binary, symbolic, and integer). It supports the solver directing that parts of the model be excluded when solving such as when solving using implicit enumeration (dynamic model modification). CONOPT is now attached for optimization. The standard solver is rapidly becoming much more robust. ASCEND IV can generate a GAMS model corresponding the ASCEND model, giving access to solvers GAMS has that ASCEND does not.

While not quite there just yet, the goal to compile and solve 250,000 equations on a 150 megahertz workstation having about 250 megabytes of fast memory in a few tens of minutes is in sight.