Strawman Productline Strategy/Priorities

December 12, 1986

Table Of Contents

1. Introduction

This document is intended to outline an overall product line strategy for 1987/1988. Each major product area is identified. The basic strategy and minimum requirements for each area are identified. More detailed items in each product area merge into tactics and are very likely to change when the product teams develop detail product plans.

There are several references to the long-term system architecture. This system architecture is described separately (see Rational System Architecture).

Items are marked with A, B, or C to give a gross indication of priority. Items marked A* are more or less in the current budget resources are available to implement them. Besides the fact that there are insufficient resources to accomplish even A items, all of the schedules indicated here are quite aggressive.

2. Machines

During 1987 customers will require at least minimal support for development in classified environments. The Series 200 family must be evolved in this timeframe to provide this support.

From a hardware technology point of view, the current processor is three to six years behind competitive products. A second generation product exploiting improved hardware technology is required as soon as possible. A new processor must provide increased performance (about twice the current processor performance) to support increasing interactive user demands, must improve cost/performance by greater than a factor of two, and must increase packaging flexibility to allow a range of products to be developed in support of the long term system architecture. To be practical, the new processor must be available for Beta shipment before the end of 1988 and must fit within 1987 budgetary constraints.

A 2.1. Secure Support

Basic approach is to provide removable media (see below) and then throw field resources at the problem on a case-by-case basis. Basically driven by program needs during 1987 and early 1988.

A* 2.2. Removable Disks

Adopt removable (200+ insertion life) 8" disks in Model 20 and Model 40 by 4Q87. Requires 3-6 man months of development and approximately $100K for prototype materials. The materials would contribute to the second generation machine.

A* 2.3. Second Generation Processor

4Q88 Beta test, Announce 1H88.
2x Cpu performance
32-128 MB memory, 2KB-4KB page size,
0.8x cost of 200/5 for similar configuration.
Budget + 40K 1H87 + 300K 2H87 (2.5-3M$ total R&D)
Integrated cpu/controller/disk backplane
Smaller board size (60% of current boards)
Gate Array or Semi-custom logic
Largely microcode compatible
Macro-architecture compatible

Series 200 peripherals & controllers.
Series 200 basic package.

B 2.4. Small Node

Develop a small processing node (200/10 minus comm lines and tape drive, includes ethernet) in the Series 200 family. Supports long term system architecture. Requires E0 environment release. Any decision on this product has been deferred until mid-87.

C 2.5. Network Comm Server

Provide a network server that connects asynch lines to the network. Required to support long term system architecture. Most likely would OEM Bridge Boxes and similar products. Not required before availability of small processing nodes.

3. Environment Releases

There are two major steps in the evolution of the environment planned for the next 2-3 years. First, the initial Epsilon release must be completed and shipped as quickly as possible. This release must include the new operating system (KKOM) with its performance and operational improvements, eliminate Gamma/Delta spec views, and must provide better support for multi-machine development.

The second major step, the Zeta release, is to provide increased distribution in the environment to fully support the long term system architecture. As part of the "Rational Trajectory", both of these major releases include significant improvements in compilation and configuration management.

Pragmatically, these two steps are too large to take without planning intermediate maintenance releases to fix emergency problems and address the most serious customer issues. The goal is to minimize the scope of the maintenance releases to allow resources to be devoted to the major releases.

3.1. Delta_1 Maintenance Release (2Q87 Beta, 3Q87 production)

Ideally this release would only address emergency fixes discovered with field use of Delta_0. However, there are several major problems which can not necessarily be deferred until the availability of Epsilon.

A 3.1.1. Multi-Machine Support

Support rapid release of views to other machines, including support for debugging. CCA requires this by the end of 1Q87. Two main approaches proposed are to extend relocation to work between machines or to automatically propogate incremental changes between machines. TRW looking at alternatives. If the relocation approach is taken, going fully native is possible with Delta.

A 3.1.2. Loader Performance

Given the improvements in the Delta editor, the two biggest performance problems in the system are the loader and the debugger. In both cases the problem is a paging problem, not a CPU performance problem. While other Delta improvements (the editor, incremental coding, etc.) allow customers to put more users on the system during design and coding, during test and integration the problems with the loader and the debugger reduce the number of user that can be supported by roughly a factor of two

(BLB estimate based on field experience, seems consistent with internal development experience).

The Delta_0 loader has performance improvements which should reduce paging by approximately a factor of two. Further improvements would require some incremental linking strategy. This could be done in Delta, but would ideally be deferred until Epsilon. A limited Delta form of incremental loading for main programs would require about 2 man months of development.

A 3.1.3. Debugger Performance

As described above, multi-user working set problems with the debugger are of high priority. While there are probably several problems, one known problem is that much of the debugger is elaborated on a per user rather than per system basis. This increases each users working set by several hundred pages. This situation could be greatly improved by restructuring the debugger to look like other parts of the system (compiler, editor, etc.) which are reentrant and only require one shared elaboration for the whole system. This would require 3-6 man months of development and would impact target tool schedules.

B 3.1.4. Exceptions in Generics

Implementing Ada semantics for exceptions exported from generics is required by CCA and eventually will show up in a validation test. Fixing this would require approximately 2 man months of development. If CCA multi-machine support is provided, may be able to defer this until Epsilon.

A 3.1.5. Emergency Problems

Some mid-life Delta maintenance release will be required to fix emergency problems.

3.2. Epsilon_0 (E0) Environment Release (4Q87 Beta, 2Q88 production)

The goal is to get KKOM into the field as quickly as possible. Minimum requirements include the basic KKOM facilities (conventional permanence model, no snapshots and no disk GC, improved Backup/Recovery, etc.) plus elimination of Gamma/Delta spec views and multi-machine development (cross-machine copiable Diana). Schedule is a primary driver.

The basic strategy is to provide Delta functionality on top of Epsilon mechanisms, with no user visible changes other than the ones mentioned above.

A* 3.2.1. KKOM

The new operating system (KKOM) must be included in the Epsilon release. In particular, the following capabilities are required.

No snapshots or disk GC.
Immediate permanence.
Individual object recovery from backups.
No object daemons.

A* 3.2.2. Eliminate Hard Diana Pointers

The current system relies upon hard (machine architecture) pointers between Ada objects. Eliminating this and introducing the Epsilon Diana design (partially introduced in Delta) would eliminate many compilation and configuration management problems. The minimum requirement is to eliminate Gamma/Delta spec views (with all of their associated obsolescence and configuration management problems) and to support efficient transfer of libraries between machines.

A* 3.2.3. CMVC Support for Multi-Machine Development

E0 model where all active development on a subsystem is on a single machine and releases are distributed to others. Also, support for archiving CMVC database and "important" views to a archive server (see system architecture discussion).

B 3.2.4. New Obsolescence Model

Propogate obsolescence on the declaration level rather than unit level. Defer propogation of obsolescence until a unit is installed, and detect when the new version of an object still exports all required entities.

B 3.2.5. Differential Views

Comes more or less for free with KKOM, but may not be user visible.

B 3.2.6. Differential Release between Machines

Determine minimum set of units to move between machines, based on which views already exist at the destination.

3.3. Zeta_0 (Z0) Environment Release (4Q88 Beta, 2Q89 Production)

Main goal is to support more distributed development along the lines of the long term system architecture. Also extend and complete Epsilon functionality.

A 3.3.1. New Distribution Model

Allow multiple active machines cooperatively working on different views of the same subsystem. Fully differential behavior across the network in the sense of only moving the minimum number of objects when moving views between machines (explointing differential view mechanism on both source and target machines). Lots more thought required here.

A 3.3.2. New Obsolescence Model (if not in E0)

B 3.3.3. Edit-in-Place for Installed Units

4. Workstations

Expand on compatibility theme during 1987 to provide professional, high-quality support for common workstations as R1000 terminals.

Secondarily, identify high-leverage opportunities to improve positioning with respect to workstations. Examples include working with Sun, simple mouse support, etc. Exploit one or more of these as resources permit.

A* 4.1. Good terminal emulator support (2Q87)

B 4.2. NFS-type Capability (2Q87)

B 4.3. Simple Mouse (4Q87)

C 4.4. Move Functionality to Workstation

C 4.5. Support Standard Window Model (2Q87)

5. Networking

A 5.1. Target Tool Support

A 5.2. Asynch lines

Provide point-to-point file transfer over asynch lines.

6. Lifecycle tools

Exploit momentum generated by announcement of Design Product. Deliver Design Product into Beta as early as possible. Deliver production version 9/1987 with expanded functionality to exceed expectations and establish trajectory.

Provide R1000 based document formatting with same release timing as design product.

Other lifecycle tools should be investigated, but for 1987 the primary focus is design and documentation.

A 6.1. Implement Announced Design Support

A 6.2. Editor Support for Annotations

B 6.3. Simple Graphics Generation

C 6.4. Real Graphics

A 6.5. Document Preparation

7. Target Tools

Highest priority is deliverying announced products to beta customers and making those customers referenceable as quickly as possible. Until there are satisfied customers, target tools are still just a promise.

Resources for new target tool iniatives in 1987 should be focused on the IBM 370.

A* 7.1. Deliver announce products ASAP

A 7.2. IBM Target Build Product

A 7.3. IBM 370 Product

370 C3I tool set jointly with CCA (M204&Adaplex) Target tools for "portable" Ada, ignore hard runtime issues. Announce and ship data sheets in volume 7/87. Deliver 4Q88. Drive real runtime issues from program needs.

B 7.4. Intel Product