```
 SSSS   U    U   BBBB     SSSS   Y    Y    SSSS   TTTTT   EEEEE   M    M    SSSS
S       U    U   B   B   S       Y    Y   S         T     E       MM  MM   S
S       U    U   B   B   S        Y Y     S         T     E       M M M M   S
 SSS    U    U   BBBB     SSS      Y       SSS      T     EEEE    M    M    SSS
    S   U    U   B   B       S     Y          S     T     E       M    M       S
    S   U    U   B   B       S     Y          S     T     E       M    M       S
SSSS    UUUUU   BBBB    SSSS      Y     SSSS      T     EEEEE   M    M   SSSS
```

```
        222
       2   2
           2
          2
         2
  ..    2
  ..   22222
```

START Job DESIGN Req #976 for EGB Date 29-Apr-85 9:37:29 Monitor: //, TOPS
File RM:<SYSTEM.SPEC>SUBSYSTEMS..2, created: 24-Feb-85 17:44:47
        printed: 29-Apr-85 9:38:44
Job parameters: Request created:29-Apr-85 9:37:27 Page limit:261 Forms:NORMAL
File parameters: Copy: 1 of 1 Spacing:SINGLE File format:ASCII Print mode:ASC

1. Separate specification and implementation.

The principal difference between worlds and subsystems is the ability to reference the visible parts of packages whose implementation is not present in the particular world. This is accomplished by allowing one or more libraries dedicated to subsystem interface specification visible parts (specs) to be included in the world. These specs are used in compilation as with any visible part, but their implementation is hidden.

Subsystem interface libraries are identified at the time they are built. All subsystem interface libraries precede implementation libraries in the library sequence. The particular limitations on subsystem specs depends on target-specific characteristics. For the R1000 target, the ability to have "closed" private parts reduces the need to have specs for packages only used in the private part. Targets that do not support truly private types may not support this feature. Similarly, the inlining/generic instantiation characteristics of the particular target may require selected bodies to be included in the spec library.

2. Program execution.

Separation of specs and implementation requires a loading mechanism, similar to EE08, to specify which implementation is to be executed to represent specs.

Initial R1000 implementations have treated each imported subsystem as an atomic unit. For more diverse development and for other targets, it may be necessary to load a smaller closure of the specs referenced.

3. Verifying specs.

Dissociation of specs and implementation introduces opportunities for violations of the Ada typing mechanisms. Facilities, possibly optional, for checking spec-implementation compatibility will be required. Initial implementation could be for equivalence, then for simple upward compatibility. These checks may be target-specific.

4. Acquiring and maintaining specs.

Specs are representatives of an implementation that is proceeding in parallel. Explicit action is required to acquire newly referenced or changed specs.

For a particular system, there is a set of current versions of subsystems from which specs can be derived. For each version of a subsystem, there can be a specification of particular versions of worlds from which to acquire specs for a particular subsystem. Operations exist to find a spec that has been used for the first time, get a new version of a spec that is known to have changed, assure that all specs are the same as the ones currently in use, and to get new specs for a specific subsystem. All of these are fairly simple, except that no easy mechanisms currently exists to detect equivalence of different version of the same unit. This primarily affects situations that call for getting specs that have changed.

5. Parallel development.

At any particular time, there may be more than one version of each of the active subsystems. These are grouped into sets, called activities, that allow tracking of these subsystems and their relations. Specs are updated from other subsystems in the activity; the current version of a subsystem

is relative to the activity of interest (though there is a "current" activity). This allows multiple threads of development within the same basic system structure and makes the tools as easy to use for new activities as for established and released activities.