```
U   U  TTTTT   III   L        III   TTTTT   III   EEEEE   SSSS
U   U    T      I    L         I      T      I    E      S
U   U    T      I    L         I      T      I    E      S
U   U    T      I    L         I      T      I    EEEE    SSS
U   U    T      I    L         I      T      I    E          S
U   U    T      I    L         I      T      I    E          S
UUUUU    T     III   LLLLL    III     T     III   EEEEE  SSSS
```

```
        222   55555
       2   2  5
           2  555
         2       5
        2         5
  ..   2     5   5
  ..  22222   555
```

START Job UTILIT Req #159 for EGB Date 28-Sep-82 22:04:08 Monitor: Rational
File RM:<RPE.DOC>UTILITIES..25, created: 10-Dec-81 11:57:25
      printed: 28-Sep-82 22:04:08
Job parameters: Request created:28-Sep-82 22:03:50 Page limit:18 Forms:NORMA
File parameters: Copy: 1 of 1 Spacing:SINGLE File format:ASCII Print mode:

This is a list of operations and capabilities to be provided in the RPE. Some of these features may be integrated into the basic editor/interface while others are separate programs.

Editor/User interaction

The ada support environment provides a variety of facilities including, but not limited to: syntax supported editing, automatic compilation, automatic integration and preparation, and source level debugging.

The standard user interface provides a multi-window editor as the front end. This editor allows the display and manipulation of objects of various types, including text, ada programs, and other objects of basic ada types.

Multiple windows can be created and deleted.

Searches for structures and text patterns is done by the editor. The result of the searches can form a collection that can be operated on in various ways.

Debugging is done by editing the appropriate scope.

Help info is generally associated with individual programs.
Help information should be invoked uniformly in any context.
Interactive help facility can be invoked at any time without loss of current context.

There will be self-teaching programs for editor/debug operations

Access Control

Login will give access to last environment

Login simply means to link a terminal to ones editor.

Logout deletes the terminal link.

Editor state is preserved from one login to the next.

Access to multiple packages simulataneously. Access can be released.

A utility should ease creation of new user packages and consistent update of associated system tables.

Users can change their password and establish more elaborate access control if desired.

Each line has a default terminal type. The terminal type can also be changed after log on. The terminal profile includes information about terminal-specific characteristics.

User can configure the terminal controller to know about a particular terminal.

Language processing

Any program is stored as components of the user package in the proper environment.

Compilation will be invisible to the user. Depending on resources compilation may be invoked after each change or only when required, i.e. user wants to see semantic errors or execute.

Various compiler options are available to compile debug statements into programs, e.g. proc traces, exception traces, single stepping statements.

Diana will provide a cross reference, i.e. it will show the declaration of an object or all uses of a declaration (visibility if observed). There may be a program to produce conventional XREF listings.

Compilers for other languages. Preferably, we should only provide front-ends compiling into diana wherever possible.

Program execution

Programs are started by evaluating a procedure or function call or by elaborating a task (variable of task type) declaration.

Programs can be run in background, on a separate window, or on an existing window.

Normal ada tasking primitives are used to have programs execute at arbitrary times. There is no special command or facility required.

In addition to Ada semantics user may change priority of task. Null priority will stop task temporariely.

The debugger is invoked by explicit break points (call to a system function) (see editor for debug operations)

Other debug features are available via compiler options.

Data storage

Data is stored in Ada variables. Encapsulation in packages and tasks allows flexible protection.

There will be generic tasks and packages modelling conventional operating system concepts, i.e. files with append, copy, delete, close etc.

Lists of user objects are obtained by editing the package containing the current (or desired) context.

Declarative items in a unit have special information associated, i.e. creation date, reference date, creator etc. This information can be accessed, i.e. declarative items may be displayed in time order etc.

Version control will allow to keep fixed number of old versions; back up a subtree to old versions; delete old versions etc.

Close operation is performed automatically by the system in certain situations, e.g. logout. Close may be invoked automatically by the user.

Rename will allow to change the name of an object without unelaborating it.

An archive utility has to allow symbolic dump of certain user data.

Archiving/restoring objects

A print utility will print programs and data. Print is part of the editor, i.e. uses the same pretty-printing routines. Printoutput may be conventional or may be structured (i.e. one package per page with ellipses)

Tools for text manipulation, editor (knowledge about sentences, words etc) formatting program, spelling correction/checking

Status information

Calendar utility will provide time and date information

Status - details have to be decided.

Automatic display of system messages, notifications etc.

System Status Inquiries, Reports on resource utilization

A list of processes can be displayed with the editor. The normal procedure of editing is used to delete or change the state of a process.

Accounting information is accumulated based on the individual user and project. The exact mechanism is to be decided.

Communication

An advise/talk feature would be nice, e.g. have two users share windows.

There is a mail facility. We need nothing fancy initially. Has to include Broadcast messages, (Data/"File" transfer ?)

Misc. Utilities

Command files are simply ada procedures. Most of the poblems of configuring one environment upon login will disappear since we keep the edit environment around.

IO task will make operations of IO devices available to the user. there will be the notion of assigning a device. Operations will be those provided by the IOP's operating system. Operations on system disks will not be available to the user.

A tool to compare two similay objects and report on their differenced is available.

Graphics, there will initially be none. Possibly graphics will be used locally in high resolution terminals (framing of windows etc.)