

Some ideas regarding GNUe configuration

1. Introduction

The purpose of this document is to present some ideas fleshing out the concept of templating for GNUe configuration. The basic idea of templating is to simplify the definition of GNUe configurations for various kinds of enterprises and vertical market industries. The concept is that each kind of enterprise will need certain configuration features, and that GNUe should make it easy for that enterprise to define those features and accomplish the configuration. GNUe is intended to be highly modular and not monolithic. Examples of configuration considerations for various kinds of enterprises include:

- A small, home-based business does not need functions to manage numerous warehouses.
- Some businesses maintain only selected subsidiary journals of their accounting systems and outsource maintenance of their general ledgers and other journals to their accountants.
- Municipal governments, self-supporting municipal revenue authorities, and similar entities will need some functionality different from that required by businesses.
- Utilities (such as electric power, communications, and water) will need extensive customer information functions linked to the systems that manage their service delivery operations.
- Some businesses can use a system that displays and reports in only one language, but others will need to accommodate multiple languages.
- Businesses that operate in international trade need a capability for managing multiple currencies.

In addition, it appears that a common set of infrastructure may be able to both aid in implementing the templating concept and provide other important functions needed in GNUe.

1.1. Terminology

We will use the following terms:

- GNUe-module - A nearly stand-alone unit comprised of focused business functionality and containing various kinds of information. The basic unit of the Enterprise Application is the GNUe-Module (which can be shortened to ge-module for convenience and referred to as just a “module” if there is no possibility of confusion). All ge-modules can be mixed and matched according to the dependencies (which should be minimized) and the need of the customer. Currently, a ge-module could contain the following kinds of information:
 - GNUe class definition files (GCD),
 - GNUe report definition files (GRD),
 - GNUe form definition files (GFD),
 - GNUe security definition files (GSD),
 - GNUe data definition files (GDD), and
 - documentation (DOCBOOK) files.
- GNUe-Packages - An arbitrary physical grouping of Modules (primarily for development and functionality explanations). The term can be shortened to ge-package for convenience and referred to as just a “package” if there is no possibility of confusion
- GNUe-Template - An arbitrary physical grouping of Modules and Module modification instructions (GTD- GNUe Template Definition) focused on use by an industry or a type of enterprise and used primarily for distribution and configuration of GNUe software. A template may have its own GCD, GRD, GFD, GSD, GDD, and DOCBOOK files as needed for specializing standard GNUe-modules for an industry, type of enterprise, or environment.
- Program-module and program-package - These refer to modules of Python, C, Java, or other programming languages, and packages that comprise groupings of the program-modules. These terms can also be shortened to p-module and p-package.
- Business process - a business or enterprise-oriented view of the purpose for which GNUe is being used. A business process can be done in a single step using GNUe or involve multiple steps, some of which use GNUe and some of which are either manual or done using some other automated support.

2. Types of information to be configured

The types of information to be organized for configuring GNUe include:

- Program-modules and program-packages. P-modules and p-packages are contained in files and directories, respectively. P-modules will include:
 - Base p-modules that are used by all configurations
 - Extended p-modules that build on the base p-modules and are used by some configurations
 - Alternative p-modules that replace either base or extended p-modules in certain circumstances
- Definition data used in configuration and processing, and generally contained in files. These particularly include:
 - GNUe class definition files (GCD) - required during system configuration and may also be used for some purposes during processing
 - GNUe report definition files (GRD) - used during processing
 - GNUe form definition files (GFD), - used during processing
 - GNUe security definition files (GSD) - used for system configuration. These files define the basic setup of the operating system security rules and provide the security parameters (such as specific file and user permissions) that implement these rules within the particular GNUe configuration.
 - GNUe data definition files (GDD) - that define information to be loaded into database tables as part of the system configuration or for initializing common data. Examples of data managed by GDD's include language codes, postal codes, and accounting system parameters.

3. Business process definition as a possible core approach

The factor that most distinguishes different kinds of enterprises and different uses of GNUe is the definition of the business processes that GNUe will be used to support.

Defining a business process in terms of GNUe involves identifying:

- What is being done in a business sense
- What parts of GNUe are being used to support the process
- Any particularly important parameters that need to be considered at particular points in the process

The same considerations -- viewed from slightly different perspectives -- affect a number of aspects of GNUe architecture:

- What program modules and GNUe-modules need to be loaded to accomplish the configuration, i.e., the GNUe Template
- What program modules and GNUe-modules a user needs to use/access to accomplish any particular task, i.e., the integration required to establish a main menu and overall control program for GNUe.
- Control over the sequence-of-events in use of GNUe, i.e., the business process
- In some cases, presently including the accounting system, an opportunity to improve the modularity by introducing parameters in a business process context.

These considerations suggest that a definition of the business processes of an enterprise can serve as the core of the GNUe Template Definition.

To illustrate this approach, consider the definition of the business processes for the enterprise by a table containing the following types of data:

- Identification of a business process and a step within that process.
- A description of the step
- Information on the nature of the step, including whether the step is to be accomplished using GNUe, any requirements for sequencing of the step with respect to other steps, and similar considerations
- The data objects affected by the step
- The forms/reports used for performing the step
- Any audit trail requirements (for recording who did what when)
- The control account, if any, for recording the accounting results of the step
- The business rules to be applied, including any non-base program-modules implementing the rules

By analyzing this table, the following configuration information can be determined:

- A list of the form definitions (gfd) and report definitions (grd) required by the enterprise configuration
- A list of the data objects required by the enterprise configuration
- A list of the non-base program-modules required by the enterprise configuration

In addition to establishing the configuration itself, the following elements of the system could be established:

- The main menu of the configuration could be established by organizing the list of business processes and steps identified in the table
- The main control program could operate by building calls to GNUe-Forms, GNUe-Reports, and other functions using information contained in the table
- A step could be identified as resulting from the actions and triggers of another step, and the step could be initiated by reference to the table.
- Maintenance and customization of the table could be performed as an ordinary function of GNUe-Forms

This definition can be prepared in the form of a table. A gcd for one possible definition of the table could be as follows:

```
char busproc(25) # The top level business process
char subproc(25) # The subsidiary business process
char altproc(25 # An alternative within the business process
int step # the step of the process
char descr(100) # A description of the step
char step_type # Is this step automated, manual, or really handled elsewhere
char form(50) # The name of the gfd file for the form used in performing the step
char report(50) # The name of the grd file for the report resulting from the step
int acct_1 # The control account for recording the result of the step
int acct_2 # Another account involved in the step
char method(50) # The name of the business logic program implementing the rules
bool audittrail # Does this person and datetime for this step need to be recorded
char policy(100) # A pointer to the enterprise policy manual section for help
char tables[] # A list of the gcd-defined tables affected by the step
char security(50) # The name of a file containing the security rules for the step
```

4. GNUe Template Definition File (GTD) description

This section attempts to answer the questions “What does a GTD look like?” and “How does a GTD work?”

A GTD could consist of a combination of:

- A business process definition table for an enterprise type or industry
- A default chart of accounts for the same enterprise type or industry, with segments conditionally included if the relevant data structures are included
- Collections of ancillary data needed to build a (possibly nested) set of scripts for loading relevant program-modules, program-packages, GNUe-modules, and GNUe-packages onto a system and configuring its security.

A GTD-management program may also be eventually needed. Such a program could have some functionality in common with a “package manager” such as Red Hat Package Manager” (RPM), that performs functions such as:

- Providing brief descriptions of ge-modules, ge-packages, p-modules, and p-packages so the user can determine their purpose and applicability
- Managing inter-dependencies among ge-modules, ge-packages, p-modules, and p-packages
- Allowing the configuration to be queried to determine its status

The processing of the GTD would result in several separate scripts.

- A script that installs p-modules and p-packages into the appropriate directories, possibly in conjunction with an operating system program package manager.
- A script that installs GNUe definition files into the appropriate directories and causes tools to be run to execute any necessary steps in defining databases, such as conversion of gcd’s into sql and execution of the sql to set up the databases.
- A script that loads configuration-relevant datasets defined by GDD’s into the appropriate databases. An example of a configuration-relevant dataset would be one

that loads a database table used for storing configuration parameters of the accounting system.

- A script that sets up the operating system, DBMS, and middleware security rules. The form and content of this script will depend on the security features of the operating system, DBMS, and middleware.
- A script that configures the security access controls on the installed and configured directories, files, databases, and other entities.
- A script that builds the main menu of the system (or layer or other part of the system).

5. Implementation considerations

Initial implementation of the GTD management system is likely to be by manually editing the basic scripts, running tools that perform the include functions, and running tools that perform the other functions from the generated scripts.

Eventually, a GUI-type system will be developed that allows the user to view module and package descriptions(both GNUe and program), to select the modules and packages desired for the configuration, and to initiate automated processing of the selections, building the necessary scripts, and executing the scripts.