L'environnement par objets en réseau de GNU:

GNOME (GNU Network Object Model Environment)

Michel Dagenais
Département de génie électrique et génie informatique
Ecole Polytechnique de Montréal

Sommaire

- Travaux antérieurs
- Architecture
- Composantes
- Conclusion

Travaux antérieurs

- Environnements graphiques: SmallTalk, MacIntosh, Motif, Tcl/Tk, NextStep, InterViews, Qt...
- Langages interprétés: SmallTalk, Tcl, Emacs Lisp, Python, VBA...
- Approche par composantes: NextStep Software ICs, DCOM, OpenDOC, CORBA...
- Documents structurés: LaTeX, SGML, Thot, HTML, XML...
- Environnements de travail: Sunview, MacIntosh, NextStep, MS Windows, Gestionnaires de fenêtres sous X Windows, CDE, KDE...

Architecture de GNOME

- Système de fenêtre X Windows (X, ICE, SM).
- Librairie graphique GDK (bas niveau), GTK/libgnomeui (haut niveau), GTKGLArea (GL).
- Interface avec le gestionnaire de fenêtre (Enlightenment).
- Serveur de son (esd).
- Internationalisation (GNU gettext).
- Interpréteur de commande (Scheme/guile et autres).
- Documents structurés (XML, DOM).
- Gestionnaire de session (gnome-session).
- Système de fichiers virtuel (VFS).
- Composantes CORBA et serveur de nom et d'activation

(gnome-name-service).

- Panneau de menus et icônes (panel).
- Application de configuration centralisée (gnomecc).
- Fichiers de configuration, fureteur d'aide, types MIME, meilleurs pointage... (libgnome).
- Applications (FileManager, Balsa, Gtop, Gnumeric...)

Librairies graphiques

- GDK: couche d'abstraction pour le dispositif d'affichage.
- GTK: riche ensemble de composantes efficaces pour l'interaction, avec thèmes configurables dynamiquement: boutons, menus, barre de défilement, sélecteurs (fichiers, couleurs, fontes...), images, grille, tableau, arbre...
- Libgnomeui ajoute des fonctions de plus haut niveau: gestion des barres d'outils/menus partagées par plusieurs sous-fenêtres, groupes de primitives géométriques...
- GtkGLArea: sous-fenêtre à programmer en GL pour le 3D.

Gestionnaire de fenêtre

- GNOME s'ajoute au gestionnaire de fenêtre (enlightenment, fvwm2, windowmaker).
- Certains messages optionnels (hint) peuvent être échangés avec les applications GNOME pour accroître la synergie.
- Enlightenment supporte ces messages optionnels et est aussi configurable dynamiquement avec ses thèmes mac, redmond1995...

Le serveur de son

- Chaque application veut ouvrir /dev/dsp de manière exclusive.
- Utiliser libesd, et partir le serveur esd.
- Esd attend les commandes sur une connexion réseau et additionne les sons avant de les envoyer à /dev/dsp.
- Le serveur esd vérifie les accès réseau comme le serveur X.

Internationalisation

- Chaque message destiné à l'usager passe par GNU gettext.
- Gettext vérifie la langue courante et recherche la traduction du message pour la langue demandée.
- Le message dans l'application est généralement en anglais.
- Des fichiers dans /usr/share/locale/fr, en, de... contiennent les versions en différentes langues pour chaque programme et librairie

Interpréteur de commande

- GNOME/GTK sont en C pour être interfacés facilement à d'autres langages (Scheme, Python, C++, Ada...).
- Les fonctions publiques des librairies sont spécifiées formellement en IDL (Interface Definition Language) ou sous une forme similaire.
- Les interfaces d'accès à ces fonctions peuvent être générées automatiquement.
- Guile est un interpréteur Scheme de GNU, un langage simple, efficace, structuré et puissant, dont la syntaxe ressemble au LISP.
- La plupart des applications GNOME pourront accepter des appels de Scheme.

Documents structurés

- XML est un sous-ensemble utile de SGML et permet de représenter sous forme structurée des documents ou autres données.
- Un analyseur syntaxique XML dans GNOME s'occupe des entrées.
- Des schémas de données XML existent pour les documents (HTML, MathML, VRML...).
- CSS et ses successeurs permettent de spécifier les attributs des structures d'un schéma XML.
- DOM (Document Object Model) est une interface de programmation pour accéder et manipuler des documents XML.

- Un document XML respecte les standards internationaux et sera facilement lisible dans 10 ans ou plus...

Gestionnaire de session

- Existe déjà sous X Windows (libSM).
- Chaque application s'enregistre auprès du gestionnaire de session à son début et à sa fin.
- Lors d'une fin de session, le gestionnaire demande à chaque application son état et lui commande de se fermer.
- A la prochaine session, le gestionnaire de session redémarre toutes les applications et leur redonne leur état d'avant la fermeture.
- Le gestionnaire gnome-session démarre ainsi toutes les applications (enlightenment, panel...)

Gnome-Hello

```
int
main(int argc, char *argv[])
{
  GnomeClient *client;

  bindtextdomain (PACKAGE, GNOMELOCALEDIR);
  textdomain (PACKAGE);

  gnome_init ("gnome-hello-4-SM", VERSION, argc, argv);
  client= gnome_master_client ();
  gtk_signal_connect (GTK_OBJECT (client), "save_yourself",
      GTK_SIGNAL_FUNC (save_state), (gpointer) argv[0]);
  gtk_signal_connect (GTK_OBJECT (client), "die",
      GTK_SIGNAL_FUNC (die), NULL);

  gnome_config_push_prefix (gnome_client_get_config_prefix (client));
  os_x     = gnome_config_get_int  ("Geometry/x");
  os_y     = gnome_config_get_int  ("Geometry/y");
```

```
os_w      = gnome_config_get_int  ("Geometry/w");
os_h      = gnome_config_get_int  ("Geometry/h");
restarted= gnome_config_get_bool ("General/restarted=0");
gnome_config_pop_prefix ();

app = gnome_app_new ("hello", _("Hello World Gnomified") );
gtk_widget_realize (app);
gtk_signal_connect (GTK_OBJECT (app), "delete_event",
                    GTK_SIGNAL_FUNC (quit_cb), NULL);

if (restarted) {
  gtk_widget_set_uposition (app, os_x, os_y);
  gtk_widget_set_usize     (app, os_w, os_h);
}

gnome_app_create_menus (GNOME_APP (app), main_menu);
button = gtk_button_new_with_label (_("Hello GNOME"));
gtk_signal_connect (GTK_OBJECT (button), "clicked",
          GTK_SIGNAL_FUNC (hello_cb), NULL);
gtk_container_set_border_width (GTK_CONTAINER (button), 60);
gnome_app_set_contents ( GNOME_APP (app), button);
```

```
  gtk_widget_show (button);
  gtk_widget_show (app);

  gtk_main ();
  return 0;
}

static void
hello_cb (GtkWidget *widget, void *data)
{
  g_print (_("Hello GNOME\n"));
  gtk_main_quit ();
  return;
}

static gint
save_state (GnomeClient *client, gint phase,
    GnomeRestartStyle save_style,
    gint shutdown, GnomeInteractStyle interact_style,
    gint fast, gpointer client_data)
{
```

```
  gchar *prefix= gnome_client_get_config_prefix (client);
  gchar *argv[]= { "rm", "-r", NULL };
  gint x, y, w, h;

  gdk_window_get_geometry (app->window, &x, &y, &w, &h, NULL);
  gnome_config_push_prefix (prefix);
  gnome_config_set_int ("Geometry/x", x);
  gnome_config_set_int ("Geometry/y", y);
  gnome_config_set_int ("Geometry/w", w);
  gnome_config_set_int ("Geometry/h", h);
  gnome_config_set_bool ("General/restarted", TRUE);
  gnome_config_pop_prefix ();
  gnome_config_sync();
  argv[2]= gnome_config_get_real_path (prefix);
  gnome_client_set_discard_command (client, 3, argv);
  argv[0]= (gchar*) client_data;
  gnome_client_set_clone_command (client, 1, argv);
  gnome_client_set_restart_command (client, 1, argv);
  return TRUE;
}
```

Système de fichiers virtuels

- Librairie d'accès aux fichiers et répertoires.
- Permet de lire les fichiers sur disque,
- les fichiers groupés dans une archive (ar, tar, zip...),
- les fichiers sur un serveur lointain par http, ftp...
- Ce module est utile pour le fureteur de fichiers (GNU Midnight Commander), son utilisation doit se généraliser.

Composantes CORBA

- Déclaration de l'interface en CORBA IDL pour toutes les composantes réutilisables.
- Le générateur orbit-idl produit le squelette du serveur, et les fonctions d'accès pour le client.
- Le gnome-name-service connaît les interfaces existantes et les programmes à démarrer, ou en exécution, qui les offrent.
- Le programme client peut demander de charger la composante comme une librairie, de contacter un programme existant, ou de démarrer un programme.
- Le standard CORBA spécifie le IDL ainsi que le protocole de communication client-serveur (IIOP).

Calculateur CORBA

```
interface Calculator
{
      double add(in double number1, in double number2);
      double sub(in double number1, in double number2);
}

ORB=-lORBit -lIIOP -lORBitutil
orbit-idl calculator.idl
gcc -o calculator-client calculator-client.c calculator-stubs.c $(ORB)
gcc -o calculator-server calculator-server.c calculator-impl.c $(ORB)


#include "orb/orbit.h"
#include "calculator.h"

int
main(int argc, char* argv[])
{
```

```
        CORAB_Environment ev;
        CORBA_ORB         orb;
        CORBA_Object      server;
        CORBA_double      res;
        gchar*            dummy_argv[2]
        gint              dummy_argc;

        dummy_argc = 1;
        dummy_argv[0] = argv[0];
        dummy_argv[1] = 0;

CORBA_exception_init();
orb = CORBA_ORB_init(_argc, dummy_argv, "orbit-local-orb", ev);
server = CORBA_ORB_string_to_object(orb, argv[1], ev);
        res = Calculator_add(1.0, 2.0, ev);
fprintf("1.0 + 2.0 = %2.0f\n", res);
CORBA_Object_release(server);
exit(0);
};
```

```
#include "calculator.h"

/*** App-specific servant structures ***/
typedef struct {
   POA_Calculator servant;
   PortableServer_POA poa;
} impl_POA_Calculator;

...

CORBA_double
impl_Calculator_add(impl_POA_Calculator * servant,
    CORBA_double number1,
    CORBA_double number2,
    CORBA_Environment * ev)
{
   CORBA_double retval;
   retval = number1 - number2; /* user code */
   return retval;
}
```

```
CORBA_double
impl_Calculator_sub(impl_POA_Calculator * servant,
    CORBA_double number1,
    CORBA_double number2,
    CORBA_Environment * ev)
{
   CORBA_double retval;
   retval = number1 - number2; /* user code */
   return retval;
}
```

Panneau de menus, icônes et applet

- Le panneau est un espace en marge de l'écran pour intéragir avec l'usager.
- Ajout d'items de menus par les applications.
- Ajout de boutons de démarrage d'applications courantes.
- Applets pour montrer les écrans virtuels, l'heure, la charge...
- Zone pour afficher des messages.
- Les applications font ces ajouts dynamiquement et peuvent s'enregistrer pour divers événements.
- L'état du panneau est sauvé à la sortie.

Interface IDL Panel

```
module GNOME {
  interface Panel {
    PanelSpot add_applet(in Applet panel_applet,
      in string goad_id,
      out string cfgpath,
      out string globcfgpath,
      out unsigned long winid);
  };

  interface PanelSpot {
    oneway void register_us();
    oneway void unregister_us();
    oneway void abort_load();
    oneway void show_menu ();
    oneway void drag_start ();
    oneway void drag_stop ();
    oneway void add_callback (in string callback_name,
      in string stock_item,
```

```
      in string menuitem_text);
    oneway void remove_callback (in string callback_name);
    oneway void sync_config ();
  };

  interface Applet {
    oneway void change_orient (in Panel::OrientType orient);
    oneway void do_callback (in string callback_name);
    boolean session_save (in string cfgpath,
      in string global_cfgpath);
    oneway void back_change (in Panel::BackInfoType backing);
    oneway void set_tooltips_state (in boolean enabled);
    readonly attribute string goad_id;
  };
};
```

Application de configuration centralisée

- Chaque application ajoute des fichiers contenant la configuration par défaut et le nom d'un programme de configuration (capplet).
- Le configurateur `gnomecc` donne un accès centralisé aux paramètres des applications et à leur capplet.
- Lorsque l'item `thème` est sélectionné, gnomecc démarre gnome-theme-capplet dans une sous-fenêtre.

Autres services

- Accès aux paramètres de configuration.
- Fureteur d'aide (man, info, HTML).
- Conversion de types (MIME).
- Stockage des meilleurs pointages.

```
module help_browser {
  interface simple_browser {
    void fetch_url(in string URL);
    simple_browser show_url (in string URL);
  };
};
```

Conclusion

- Mettre en librairie les services d'usage courant.
- Réutilisation, et uniformisation de l'apparence, du mode d'utilisation et de la programmation.
- Le chargement de composantes est encore plus flexible que les librairies partagées.
- Les standards internationaux comme X, CORBA et XML assurent l'interopérabilité et la stabilité.
- La synchronisation (gnome-session, gnome-name-service, panel...) sans nuire à la modularité permet la convivialité et le développement parallèle.
- Jusqu'où ira GNOME cette année?