электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК 004.4'2

Разработка модуля выбора стальных профилей для NX 8.5 с помощью библиотеки NX Open

***Исаев Д.К.**, студент*
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана,
кафедра «Системы автоматизированного проектирования»

Научный руководитель: Мартынюк В.А., к. т. н., доцент
Россия, 105005, г. Москва, МГТУ им. Н. Э. Баумана
bauman@bmstu.ru

NX Open – набор средств программирования, позволяющий пользователям создавать свои собственные приложения для выполнения в САПР NX. Благодаря ему программисты располагают почти всем функционалом, доступным при взаимодействии со средой проектирования. Для написания приложений поддерживаются наиболее популярные на сегодня языки, такие как C, C++, C#, Java, Visual Basic. Наиболее удобно редактировать и компилировать код в Microsoft Visual Studio, куда система NX при установке встраивает свой шаблон проекта (при этом важно соответствие версий NX и Visual Studio; так, для NX 8.5 требуется Visual Studio 2010, в противном случае настройки проекта приходится прописывать вручную).

Для облегчения разработки диалоговых окон для пользовательских модулей предусмотрены два средства – Open User Interface Styler и Block Styler. Оба генерируют файл диалога и шаблоны текстов программы для указанного языка программирования. Первое средство использовалось в версиях NX до 6 и считается устаревшим. Поэтому для создания приложения будем использовать Block Styler.

Описание Block Styler

Интерфейс Block Styler показан на рисунке ниже.

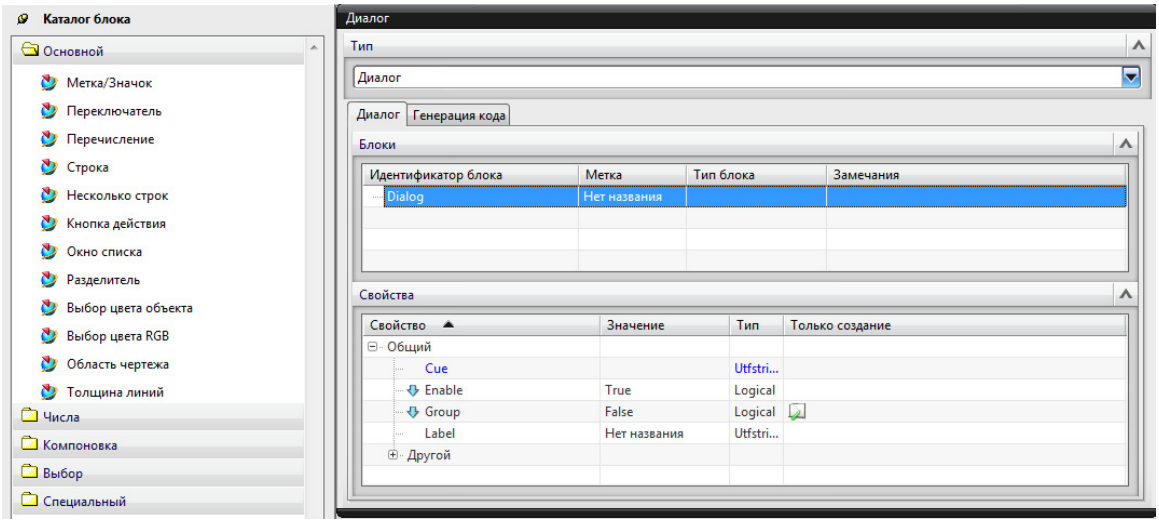

Рис. 1. Интерфейс Block Styler

В окне "Диалог" отображается текущая структура диалогового окна в виде дерева. Ниже перечисляются свойства выбранного в данный момент блока, которые можно изменять. На вкладке "Генерация кода" расположены настройки шаблона программы, такие как язык программирования, используемые точки входа, используемые кнопки внизу диалогового окна, отображение комментариев и др.

Окно "Каталог блока" содержит все доступные блоки, которые можно вставить в диалоговое окно. К ним относятся метки, поля ввода данных различного типа, таблицы, списки, блоки выбора объектов в трехмерном пространстве, блоки группирования других блоков и т. п. Как только пользователь добавляет первый блок, отображается заготовка диалогового окна в соответствии с используемыми настройками (рис. 2), а дерево структуры дополняется соответствующим блоком.

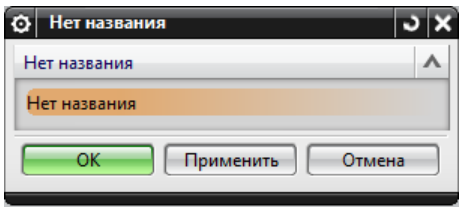

Рис. 2. Диалоговое окно с блоком типа "Метка/Значок"

Каждый блок помимо своих специфических свойств имеет и общие свойства. Во-первых, это настройки расположения относительно других элементов в диалоговом окне (группа "Присоединения"). Во-вторых, это параметры группы "Общие". Наиболее важным является свойство "BlockID", т. к. именно по этому имени происходит обращение к блоку в программном коде. Так что "BlockID" должен представлять собой недлинное, понятное имя блока с использованием символов латинского алфавита, цифр и подчеркивания. Свойства "Enable" и "Show" задают активность и отображение соответственно блока в момент запуска программы. Свойство "Label" задает текст, который может отображаться рядом с блоком. Свойство "Group" изменяет стиль отображения блока так, как будто он помещен в группу с использованием блока "Группа", с тем отличием, что структура дерева не меняется.

После того, как создано диалоговое окно и заданы все необходимые настройки, система сохраняет (в случае с C++) три файла: файл диалога *.dlg (имеющий формат XML), основной и заголовочный файлы исходного кода *.cpp и *.hpp. DLG-файл следует поместить в директорию application пользовательской директории. Чтобы иметь возможность пользоваться этой директорией, необходимо в системной переменной UGII_CUSTOM_DIRECTORY_FILE указать путь к текстовому файлу, который в свою очередь содержит полный путь к этой директории. Отметим, что создание такой директории необязательно, и все необходимые файлы можно хранить в путях, заданных в системных переменных UGII_BASE_DIR и UGII_ROOT_DIR (последнее может оказаться крайне неудобным и небезопасным, так как в этом каталоге хранится большое количество системных файлов NX).

Создание семейства деталей

Деталь профиля, на основе которой будет создаваться таблица семейства, представляет собой эскиз с примененной операцией выдавливания. В эскизе всем изменяемым размерам следует присвоить буквенное обозначение (рис. 3).

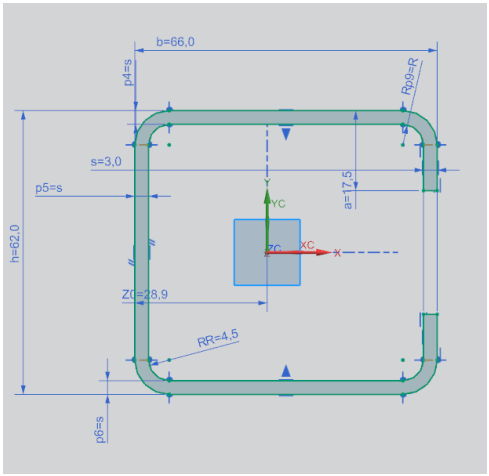

Рис. 3. Размеры С-образного профиля

Вызываем пункт меню Инструменты → Семейства деталей. В появившемся диалоговом окне добавляем в поле "Выбранные столбцы" все изменяемые размеры и нажимаем кнопку "Создать". Откроется электронная таблица в формате Microsoft Excel (поддерживается версия, начиная с 2007). Два первых обязательных столбца, DB_PART_NO и OS_PART_NAME, соответственно номер и названия членов семейства.Заполненная таблица имеет вид, показанный на рисунке ниже.

	A	B	C	D	E	F	G	H
1	*DB_PART_NO*	*OS_PART_NAME*	*h*	*b*	*a*	*s*	*R*	*Z0*
2	1	62-66	62	66	17,5	3	4,5	28,9
3	2	65-32	65	32	8	1	1,5	10,5
4	3	80-50	80	50	24	4	6	21,3
5	4	100-50	100	50	10	2	3	15,6
6	5	100-80	100	80	35	5	7,5	36,2
7	6	120-55	120	55	18	5	7,5	18,5
8	7	160-50	160	50	20	3	4,5	14,9
9	8	160-60	160	60	32	4	6	21,4
10	9	300-60	300	60	50	5	7,5	18,7
11	10	400-160	400	160	50	3	4,5	50,6
12	11	550-65	550	65	30	4	6	19,7
13	12	410-65	410	65	30	4	6	13,9

Рис. 4. Таблица семейства в формате Excel

Выделяем все строки таблицы, и во вкладке "Надстройки" (встраивается системой NX) в выпадающем списке "Семейство" выбираем "Сохранить семейство". Для нашего модуля сохраняем эту таблицу в формате TXT и размещаем в директории, прописанном в системной переменной UGII_BASE_DIR. Закрываем Excel, и нажимаем "OK" в диалоговом окне "Семейство деталей".

Создание диалогового окна

Общий вид требуемого диалогового окна представлен на рисунке ниже.

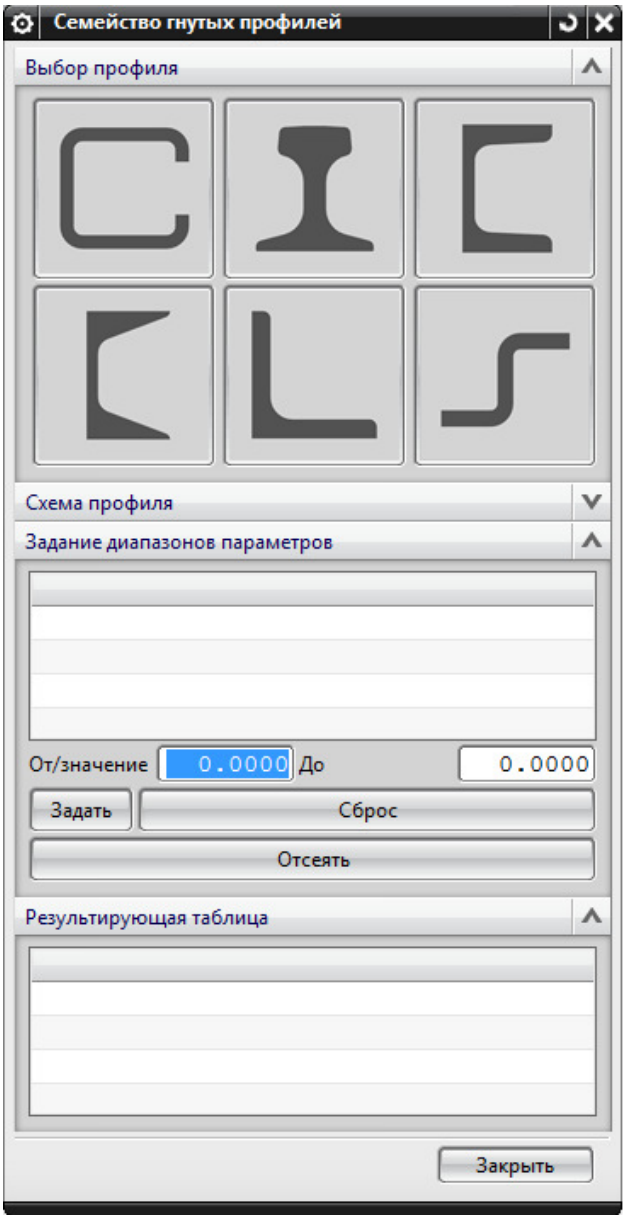

Рис. 5. Диалоговое окно выбора стальных профилей

Блоки в диалоге объединены в четыре группы: "Выбор профиля", "Схема профиля", "Задание диапазонов параметров" и "Результирующая таблица". Первая группа включает в себя шесть кнопок действия. Свойство "Bitmap" кнопки позволяет отображать любое изображение в формате BMP. Так, было предварительно подготовлено шесть изображений различных профилей, которые были размещены в директории пользовательских ресурсов (там же должен размещаться и DLG-файл). В поле "Bitmap" достаточно задать только имя изображения (без расширения). Каждая кнопка

предназначена для загрузки данных о соответствующем профиле в блоки ниже. Группа "Схема профиля" предназначена для вывода детальной схемы профиля с буквенными обозначениями размеров. Третья группа содержит блок типа "Дерево списка", два поля ввода вещественных чисел и три кнопки. В группе "Результирующая таблица" – один блок "Дерево списка". В этот блок будут вноситься данные о семействе профиля, подготовленные в табличном виде. В таблицу выше в первый столбец будут заноситься доступные для изменения параметры. Пользователь выбирает размеры, на которые желает наложить ограничения (можно выбрать сразу несколько), нажимает кнопку "Задать", после чего во втором и третьем столбцах в соответствующих строках отображается диапазон, введенный в поля ввода вещественных чисел. Для подтверждения отсева необходимо нажать кнопку "Задать", после чего из исходной таблицы исчезнут строки, не попадающие в заданные диапазоны. Последовательное нажатие кнопок "Сброс" и "Задать" должно возвращать таблицу семейства к исходному виду.

Добавление исполняемого кода к шаблону диалога

Заголовочный файл подключает библиотеки NX Open и другие библиотеки, а также объявляет главный класс программы Sem. В соответствии с этим объявлением программа содержит конструктор Sem(), деструктор объекта класса ~Sem(), и три callback-функции, вызываемые в определенные моменты времени: initialize_cb() (вызывается при инициализации модуля), dialogShown_cb() (вызывается сразу после показа диалогового окна) и update_cb() (вызывается всякий раз, как изменяется состояние какого-либо блока). Каждому типу блока в NX Open соответствует определенный класс пространства имен NXOpen::BlockStyler. Например, для кнопок существует класс Button, для метки – Label, для группы – Group, для поля ввода вещественного числа – DoubleBlock, для дерева списка – Tree. Через соответствующие объекты классов осуществляется доступ к данным конкретного блока, поэтому для каждого блока объявляется свой объект. Название его совпадает со значением свойства BlockID.

Файл *.cpp содержит определения методов класса Sem, стандартную точку входа ufusr(), функция очистки и функция определения способа завершения модуля (ufusr_cleanup() и ufusr_ask_unload() соответственно). Содержание конструктора, деструктора, callback-функций initialize_cb() стандартно и обычно не подлежит изменению. Определения ufusr(), ufusr_ask_unload() и ufusr_cleanup() в данном случае также оставим неизменными.

Первым делом необходимо задать состояние окна, когда в него еще ничего не загружено. Во-первых, группа "Схема профиля" пока не нужна и должна быть свернута. Целесообразно также задать названия столбцов списка отсеивания. Наконец, кнопки "Задать", "Отсеять" и "Сброс" должны быть недоступны для нажатия, так как данные для них еще не подготовлены. Для управления скрытием/отображением группы используется метод SetExpanded(bool expanded). Для скрытия аргумент expanded должен иметь значение false. Ниже показан соответствующий фрагмент кода.

```
group2->SetExpanded(false);
```

Здесь group2 – название указателя на объект класса Group. Для активации/деактивации кнопок используем метод SetEnabled(bool enable). Чтобы задать столбцы списка, используем метод InsertColumn(int columnID, const char* columnTitle, int columnWidth). Здесь columnID – номер столбца, начиная с 1, columnTitle – название столбца, columnWidth – его ширина. Соответствующий фрагмент кода показан ниже.

```
tree_control0->InsertColumn(1, "Размер", 70);
tree_control0->InsertColumn(2, "От/значение", 95);
tree_control0->InsertColumn(3, "До", 50);
```

Все эти операции удобно прописать в теле функции dialogShown_cb().

В функцию update_cb() передается указатель на блок, состояние которого изменилось. Используется конструкция if-else, которая узнает, какой именно блок вызвал callback-функцию и выполняет соответствующие операции. Для кнопок группы "Выбор профиля" выполняются следующие действия.

1) Раскрывается группа "Схема профиля" и загружается соответствующее изображение.

2) В случае, если данные профиля уже загружены, очищается таблица семейства и список отсеивания.

3) Открывается таблица семейства нужного профиля в формате TXT и загружается во внутренний массив программы.

4) Список отсеивания заполняется доступными для отсеивания размерами.

5) Заполняется таблица семейства в диалоге.

Для работы с файлом и данными перед загрузкой их в диалог используются функции стандартной C-библиотеки. Хотя данные в блоке Tree представлены в виде

таблицы, технически они представляют собой список узлов (объектов типа Node). Из-за этого работа с таблицей несколько усложняется. Для создания нового узла используется метод CreateNode(const char *displayText). Название узла одновременно будет и значением в первом его столбце. Чтобы узел отобразился в списке, используется метод InsertNode(). Фрагмент кода, иллюстрирующий данный подход, показан ниже. Новый узел вставляется в конец списка.

```
NewNode = tree_control0->CreateNode(cur_tok);
tree_control0->InsertNode(NewNode, NULL, NULL, tree_control0->NodeInsertOptionLast);
```

Чтобы задать значение в определенном столбце узла, используется метод SetColumnDisplayText(int columnID, const char* columnDisplayText), где первый аргумент – номер столбца, второй – строковое значение.

Для очистки таблиц используется следующая схема: сначала получаем указатель на корневой узел (метод RootNode()). Получаем указатель на следующий узел методом NextSiblingNode() и удаляем текущий узел (DeleteNode()). И так продолжаем, пока NexSiblingNode() не возвратит NULL. Пример кода:

```
Node* tableNodeDel = tree_control01->RootNode();
while (tableNodeDel != NULL)
{
	Node* nodeForDelete = tableNodeDel;
	tableNodeDel = tableNodeDel->NextSiblingNode();
	tree_control01->DeleteNode(nodeForDelete);
}
```

По нажатии кнопки "Задать" в поля таблицы отсеивания заносятся введенные значения вещественных чисел уже известными нам методами. Нажатие кнопки "Сброс" просто присваивает всем полям в столбцах диапазона пустые строки ("").

Схема действий по нажатии кнопки "Отсеять" следующая.

1) Удалить все узлы из таблицы.
2) Заново заполнить ее в первоначальном виде.
3) Удалить узлы, не попадающие в указанные диапазоны.

Порядок работы с программой

Выбираем желаемый профиль по нажатию кнопки. Загружается таблица семейства, схема профиля, в списке отсеивания отображаются доступные размеры. Вводим желаемый диапазон размеров в поля ввода "От" и "До". Выбираем размеры, для которых необходимо установить этот диапазон и нажимаем кнопку "Задать", а затем "Отсеять".

Результаты работы программы представлены на рисунке ниже.

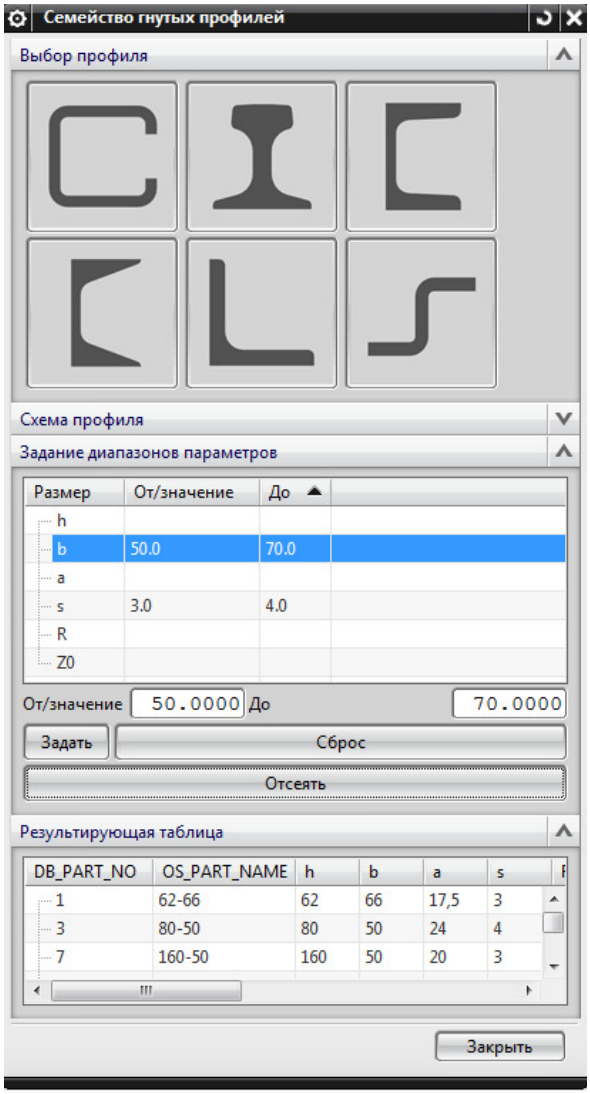

Рис. 6. Промежуточное состояние программы

Для очистки полей диапазонов используем кнопку "Сброс". В любое время можно переключить профиль нажатием кнопки, при этом результаты предыдущих операций стираются.

Список литературы

1. Справка NX. Siemens Product Lifecycle Management Software Inc., 2012. 1 электрон. опт. диск (DVD-ROM).
2. Тихомиров В.А. Разработка приложений для Unigraphics на языке С. Владивосток, Дальнаука, 2011. 422 с.
3. Гончаров П.С., Ельцов М.Ю., Коршиков С.Б., Лаптев И.В., Осиюк В.А. NX для конструктора-машиностроителя. М.: ДМК, 2010. 504 с.