



*Compass+*

# *Flora/C+*

- Объектно-ориентированная система программирования
- Программирование методом выращивания дерева объектов
- Быстрая разработка сложных приложений
- Приложения с развитой мультизадачностью
- Приложения с распределенными инстанциями
- Переносимое программное обеспечение

(Версия документа 1.4 от 20 января 2001)





<b>Версия</b>	<b>Дата</b>	<b>Комментарии</b>
1.0	06 января 2000	
1.1	12 марта 2000	
1.2	31 мая 2000	
1.3	30 июня 2000	
1.4	20 января 2001	

© ООО “*Compass Plus*”, 1996-2001

Настоящий документ и представленная в нем информация является собственностью ООО “*Compass Plus*” (*Compass Plus*). Этот документ как в целом, так в любой своей части не может быть воспроизведен или включен в другой документ или использован в целях, отличных от тех, для которых он был разработан, без специального письменного разрешения *Compass Plus*, или, если какая-либо часть настоящего документа является предметом контракта между *Compass Plus* и третьей стороной, то – только в соответствии с полномочиями, предусмотренными этим контрактом.

Следующие обозначения являются торговыми марками ООО “*Compass Plus*” и её партнеров–  
*Compass Plus, PromCard, A4M/C+, A4M BackOffice/C+, A4M FrontOffice/C+, A4M Telebanking/C+, A4M ATManager/C+, PDM/C+, Teller/C+, ITP/C+, SPDH/C+, VTBI/C+, Flora/C+, RCS/C+, FloraWare, TranzWare*

В настоящем документе используются также торговые марки других компаний.



## ОГЛАВЛЕНИЕ

<b>1. Введение</b>	<b>5</b>
<b>2. Ключевые свойства</b>	<b>5</b>
Объектно-ориентированное окружение ( <i>Object-Oriented Environment</i> )	6
Развитая мультизадачность ( <i>Powerful Multi-tasking Features</i> )	8
Поддержка распределенных экземпляров ( <i>Multi-Node Instance</i> )	9
Независимость от операционной системы ( <i>Operating System Independence</i> )	9
Переносимость на различные платформы ( <i>Cross-Platform Portability</i> )	10
Масштабируемость ( <i>Scalability</i> )	10
Развитые средства визуализации ( <i>Advance Visual Features</i> )	11
Открытая архитектура ( <i>Open Architecture</i> )	11
Поддержка многоязыковых приложений ( <i>Multi-Lingual Applications Support</i> )	12
<b>3. Системная архитектура</b>	<b>12</b>
<b>4. Объектно-ориентированная среда разработки приложений</b>	<b>15</b>
<b>5. Среда разработки мультизадачных приложений</b>	<b>19</b>
<b>6. Архитектура распределенных приложений</b>	<b>21</b>
<b>7. Масштабируемость решений</b>	<b>24</b>
<b>8. Концептуальные особенности методов разработки приложений</b>	<b>26</b>
<b>9. Свойства технологии разработки приложений</b>	<b>37</b>
<b>10. Инструменты разработки приложений</b>	<b>40</b>
Дизайнер ( <i>Designer</i> )	42
Редактор графических объектов ( <i>Painter</i> )	42
Менеджер приложений ( <i>Application Manager</i> )	42
Транслятор F++ ( <i>F++ Translator</i> )	42
Инспектор классов ( <i>Class Inspector</i> )	42
Отладчик ( <i>Debugger</i> )	43
Менеджер проектов ( <i>Project Manager</i> )	43
Журнал событий ( <i>Event Log</i> )	43
Мастер приложений ( <i>Application Wizard</i> )	43
Менеджер секций ( <i>Section Manager</i> )	44
Редактор иконок ( <i>Icon Editor</i> )	44
Центр разработки ( <i>Development Centre</i> )	44
<b>11. Заключение</b>	<b>45</b>

## СПИСОК ИЛЛЮСТРАЦИЙ

Рисунок 1.	Ключевые свойства системы программирования <i>Flora/C+</i>	7
Рисунок 2.	Системная архитектура <i>Flora/C+</i>	14
Рисунок 3.	Объектно-ориентированная среда разработки приложений <i>Flora/C+</i>	16
Рисунок 4.	Средства разработки мультизадачных приложений <i>Flora/C+</i>	20
Рисунок 5.	Архитектура распределенных приложений <i>Flora/C+</i>	23
Рисунок 6.	Масштабируемость решений на базе <i>Flora/C+</i>	25
Рисунок 7.	Особенности методов разработки приложений <i>Flora/C+</i>	27
Рисунок 8.	Свойства технологии разработки приложений <i>Flora/C+</i>	38
Рисунок 9.	Инструменты разработки приложений <i>Flora/C+</i>	41
Рисунок 10.	Официальные ссылки	46



## 1. Введение

Система программирования *Flora/C+* представляет собой программный комплекс, обеспечивающий технологическую среду, программные компоненты и инструментальные средства, необходимые для разработки и исполнения приложений различного типа. Изначально *Flora/C+* разрабатывалась как высокотехнологичная объектно-ориентированная система автоматизации разработки программ широкого класса – от обработки событий реального времени и массовых потоков транзакций до визуализации объектов реального мира и диалогового доступа к базам данных.

Средства *Flora/C+* не только предоставляют разнообразный набор инструментов, необходимых для эффективной разработки программ, но также определяют технологию организации вычислительного процесса, управляющего исполнением программ, в том числе – таких основополагающих его компонент как организация памяти, процессов, системы команд, прерываний и т.д. Последнее обстоятельство позволяет говорить о *Flora/C+* как о целостной системе программирования, основанной на оригинальном способе организации вычислительного процесса.

В основе идеологии организации вычислительного процесса *Flora/C+* лежит нелинейная структура памяти вычислительной системы, организованная в виде дерева объектов (*Objects Tree*), элементами которого могут быть элементарные типы данных и производные от них, встроенные объекты, объекты библиотек, пользовательские объекты (агрегаты), программы или задачи.

Управление деревом объектов выполняется объектной машиной (*Objects Engine*), осуществляющей контроль над всем вычислительным процессом, в том числе – порождением и переключением задач (процессов), синхронизацией доступа процессов к элементам дерева объектов, выполнением программ, включенных в дерево объектов, активизацией или прерыванием процессов при возникновении каких-либо событий и т.д.

В следующих главах рассматриваются основные свойства описанных выше и других компонент системы программирования *Flora/C+* в контексте её применения к решению задач разработки систем массовой обработки потоков транзакций (*OLTP Development Tool Kit*), каковой является реализованная с помощью средств *Flora/C+* подсистема процессингового обслуживания *A4M FrontOffice/C+*.

## 2. Ключевые свойства

Технологические достижения последних лет в области информационных технологий, в частности, в технологии объектно-ориентированного программирования позволяют на практике широко применять новые методы при разработке сложных программных систем в таких традиционных областях приложений, какими, например, являются системы массовой обработки потоков транзакций (*OLTP Systems*) или событий, возникающих в реальном масштабе времени (*Real Time Systems*), системы визуализации объектов реального мира (*SCADA Systems*) или системы управления и доступа к базам данных (*Data Management Systems*).

Вместе с тем, современные средства разработки таких систем должны обеспечивать поддержку реализации целого ряда жестких системных требований, определяемых особенностями конкретных областей приложения и режимов функционирования прикладных систем.

При разработке системы программирования *Flora/C+* ставилась задача удовлетворить многие из таких системных требований, вытекающих из особенностей реализации сложных программных комплексов, и одновременно, создать целостную систему программирования, основанную на современных представлениях об объектно-ориентированных вычислениях, методах разработки, проектирования, отладки и тестирования.

Рассмотрим ключевые свойства (*Key Properties*) *Flora/C+* (см. Рисунок 1, стр. 7).

### **Объектно-ориентированное окружение (*Object-Oriented Environment*)**

Основное свойство *Flora/C+* состоит в организации специального мультизадачного объектно-ориентированного окружения, в среде которого разрабатываются и исполняются все приложения. Любое приложение, созданное с помощью *Flora/C+* существует в форме дерева взаимодействующих между собой объектов (*Objects Tree*), погруженного в среду *Flora/C+*. В виде дерева объектов представлены также и сами элементы объектно-ориентированной среды *Flora/C+*, доступной каждому приложению, такие, например, как – системная консоль, устройства ввода/вывода, встроенные системные функции, библиотеки, константы. Инструментальные приложения – различные редакторы, Отладчик, Транслятор *F++*, Дизайнер, ряд менеджеров разработки и системные утилиты и т.д. – также существуют в виде дерева объектов *Flora/C+*.

В качестве элементов в дерево объектов приложений *Flora/C+* могут быть включены компоненты различных типов:

- скалярные константы и переменные (числовые, строковые и т.д.)
- агрегаты данных
- указатели на другие объекты
- графические элементы и фигуры
- окна
- диалоговые элементы
- каталоги (которые в свою очередь могут содержать объекты и другие каталоги)
- задачи
- каналы связи
- объекты управления внешними устройствами
- объекты ввода/вывода
- объекты взаимодействия с базами данных
- группы объектов (агрегаты)
- и т.д.



## Flora/C+

### OLTP Development Tool Kit

#### Key Properties

- Object-Oriented Environment
- Powerful Multi-tasking Features
- Multi-Node Instance
- Operating System Independence
- Cross-Platform Portability
- Scalability
- Advanced Visual Features
- Open Architecture
- Multi-Lingual Applications Support



E-mail:compass@compassplus.ru Tel: +7 (095)239-10-28, 230-68-43(fax), +7(3511)37-09-71, 37-05-03(fax)

**Рисунок 1. Ключевые свойства системы программирования *Flora/C+***





Список встроенных типов объектов открыт для расширения. На основе встроенного механизма наследования свойств объектов могут порождаться производные пользовательские объекты, так называемые агрегаты.

В качестве элемента в дерево объектов может быть помещена и пользовательская программа, написанная на встроенном языке программирования *Flora/C+*, называемом *F++* (по аналогии с *C++*).

Объектная машина *Flora/C+* (*Objects Engine*) автоматически отслеживает целостность всех связей между объектами, выполняет необходимые в мультизадачной среде блокировки доступа к данным и объектам. Объекты могут взаимодействовать между собой передачей друг другу сообщений или прямым доступом через указатели.

Библиотеки *Flora/C+* позволяют разработчику использовать в совокупности более 3,500 свойств и методов встроенных и предопределённых объектов.

Таким образом, средства *Flora/C+* предоставляют развитую, концептуально целостную объектно-ориентированную среду разработки и исполнения приложений на базе широкого набора встроенных типов и классов объектов, что значительно облегчает задачу разработки новых приложений.

Более детально объектно-ориентированные средства *Flora/C+* рассмотрены далее (см. раздел 4, стр. 15 и Рисунок 3, стр. 16).

### **Развитая мультизадачность (*Powerful Multi-tasking Features*)**

Одним из основополагающих свойств объектно-ориентированной среды *Flora/C+* является её концептуальная мультизадачность. Теоретически любой элемент или группа элементов объектного дерева может быть отдельной задачей. Возникающие проблемы и коллизии, связанные с обработкой данных в мультизадачной среде, решаются автоматически с помощью специальных встроенных средств объектной машины (*Object Engine*). С целью эффективной организации асинхронной обработки событий в задачах разработан специальный класс объектов, называемый рефлексам (*Reflexes*).

Таким образом, использование объектов типа «задача» и «рефлекс» в приложениях, в совокупности со встроенным аппаратом контроля блокировок *Flora/C+*, предоставляет возможность решать задачи создания сложных мультизадачных приложений с асинхронной обработкой данных. Решение таких задач с использованием *Flora/C+* становится по силам даже разработчику с невысокой квалификацией и опытом работы в этой области. Более того, простота и естественность организации асинхронной обработки данных в системе программирования *Flora/C+* провоцирует разработчика на широкое применение средств мультизадачности. Опыт показывает, что даже в приложениях среднего размера может порождаться несколько десятков и даже сотен параллельно исполняемых задач. Естественная мультизадачность среды исполнения приложений и самих приложений *Flora/C+* позволяет говорить о наличии концептуальных оснований для организации глубоко распараллеленного вычислительного процесса на базе технологии системы программирования *Flora/C+*.

Более детально мультизадачные средства *Flora/C+* рассмотрены далее (см. раздел 5, стр. 18 и Рисунок 4, стр. 20).



**Поддержка распределенных экземпляров (*Multi-Node Instance*)**

Программное обеспечение системы программирования *Flora/C+* существует в виде экземпляров, которые могут исполняться на разных процессорах, серверах или узлах кластерной платформы. Организация взаимодействия элементов дерева объектов одной инстанции с элементами другой инстанции осуществляется с помощью специальных сетевых средств (*Flora Net*), поддерживаемых объектной машиной (*Object Engine*). Предусмотрены два механизма взаимодействия объектов разных экземпляров: порты ввода/вывода, используемые для поточной передачи данных между объектами разных экземпляров; общее дерево объектов, элементы которого доступны с помощью прямого обращения, как если бы эти элементы были частью дерева объектов инстанции.

Таким образом, система программирования *Flora/C+* позволяет естественным образом создавать распределенные приложения, исполняемые на различных процессорах, серверах или узлах кластера. Механизмы, обеспечивающие взаимодействие распределенных приложений, поддерживают как слабо связанные объекты (взаимодействие только в режиме передачи данных) так и сильно связанные объекты (взаимодействие в режиме прямого обращения).

Более детально организация и взаимодействие распределенных экземпляров *Flora/C+* рассмотрены далее (см. раздел 6, стр. 21 и Рисунок 5, стр. 23).

**Независимость от операционной системы (*Operating System Independence*)**

Реализация системы программирования *Flora/C+* выполнена с применением специальных технологий, максимально обеспечивающих независимость её программного кода от операционной системы, в среде которой исполняется инстанция *Flora/C+*. В частности, использована технология микроядра, состоящая в выделении системно зависимых частей программного кода в специальном программном слое, называемом микроядром. Только на этом уровне программного обеспечения вызываются функции операционной системы, используемые инстанцией *Flora/C+*.

Необходимо отметить также, что независимость *Flora/C+* от средств операционной системы обеспечивается также и минимальным набором используемых функций операционной системы. Применяются только базовые функции, такие как управление памятью, задачами, устройствами ввода/вывода, файловой системой и т.п. А, как известно, большинство операционных систем функционально совместимы на этом базовом уровне. Причиной этого является использования традиционной фон-неймановской архитектуры вычислительного процесса в подавляющем большинстве коммерчески используемых в настоящее время архитектур вычислительных систем.

Все следующие слои программного обеспечения *Flora/C+* реализованы на базе элементарных функций, предоставляемых микроядром.

Таким образом, обеспечивается практически полная независимость программного обеспечения *Flora/C+* и исполняемых в её среде приложений от свойств конкретной операционной системы, под управлением которой находится инстанция *Flora/C+*.

Более детально возможность использования *Flora/C+* в среде различных операционных систем рассмотрено далее (см. раздел 7, стр. 24 и Рисунок 6, стр. 25).

**Переносимость на различные платформы (*Cross-Platform Portability*)**

На основании тех же, описанных в предыдущем разделе, причин обеспечивается и независимость программного кода *Flora/C+* и от конкретной архитектуры используемой вычислительной платформы. Необходимо заметить, что собственно переносимость программного кода с платформы на платформу обеспечивается не только применением технологии микроядра, но также и использованием для реализации программ микроядра и объектной машины стандартного языка C++. Другие компоненты программного обеспечения *Flora/C+* полностью реализованы с помощью средств самой *Flora/C+* (для реализации этих компонент был использован так называемый метод «раскрутки») и, поэтому, являются переносимыми по определению.

Более детально возможность использования *Flora/C+* на различных платформах рассмотрено далее (см. раздел 7, стр. 24 и Рисунок 6, стр. 25).

**Масштабируемость (*Scalability*)**

Многие области приложений, особенно связанные с массовой обработкой событий реального мира или потоков транзакций, требуют применения так называемых масштабируемых решений, т.е. таких решений, которые бы позволяли увеличивать производительность системы пропорционально возросшему объёму обрабатываемых данных с помощью простого увеличения числа единиц вычислительных (процессорных, серверных, кластерных и т.д.) блоков, на которых исполняются приложения. Наиболее интересными с практической точки зрения являются линейно масштабируемые решения, поддерживаемые некоторыми серверными платформами.

С точки зрения применения системы программирования *Flora/C+* масштабируемость означает принципиальную возможность реализации масштабируемых решений в среде *Flora/C+*, а также приемлемую трудоёмкость реализации таких решений. Как уже указывалось выше, приложения *Flora/C+* могут исполняться в среде различных инстанций *Flora/C+*, которые в свою очередь могут исполняться под управлением различных процессоров, на различных серверах или узлах кластерного сервера. При этом с помощью средств поддержки распределённых приложений *Flora/C+* предоставляется возможность взаимодействия прикладных объектов, исполняемых в среде различных инстанций. Причем поддерживаются как слабо связанные, так и сильно связанные распределённые приложения.

Таким образом, масштабируемые решения поддерживаются системой программирования *Flora/C+* на базе средств поддержки распределённых приложений. Применение этих средств также обеспечивает и приемлемая трудоёмкость реализации масштабируемых решений.

Вообще говоря, масштабируемость в более широком смысле означает также и возможность исполнения одного и того же приложения без каких-либо модификаций и изменений на различных вычислительных системах – от самых простых и дешевых рабочих станций до высокопроизводительных кластерных систем и мэйнфреймов. Такая возможность также обеспечивается средствами *Flora/C+* за счет её свойств, связанных с переносимостью программного обеспечения, описанных выше.



Более детально масштабируемость решений на базе средств *Flora/C+* рассмотрена далее (см. раздел 7, стр. 24 и Рисунок 6, стр. 25).

### **Развитые средства визуализации (*Advance Visual Features*)**

Системно-независимая архитектура системы программирования *Flora/C+* предусматривает использование только одной элементарной базовой функции формирования графических изображений, поддерживаемой операционной системой, под управлением которой выполняется инстанция *Flora/C+*. Такой элементарной функцией является вывод массива пикселей на экран системной консоли. Все остальные элементы графического пользовательского интерфейса, включая мультиоконный интерфейс, формируются и поддерживаются средствами самой системы программирования *Flora/C+*.

Библиотеки *Flora/C+* предоставляют широкий набор объектов, предоставляющих разнообразные средства визуализации процессов реального мира, в частности такие, как окна и их элементы, диалоговые элементы (меню, поля ввода/вывода, полосы прокрутки, кнопки, списки, таблицы, переключатели, регуляторы, редактируемые текстовые окна и т.д. и т.п.), разнообразные графические элементы – линии, фигуры, изображения и т.д. Встроенный графический редактор позволяет создавать практически любые графические формы представления объектов реального мира, а также всевозможные формы и экраны пользовательских диалогов с помощью разнообразных инструментальных средств рисования.

Свойства и характеристики графических элементов, хранящихся в соответствии с идеологией *Flora/C+* в виде дерева объектов, доступны для их динамического изменения в процессе исполнения приложений другими объектами и программами. В том числе доступны для изменения и такие характеристики графических объектов как координаты местоположения на системном экране, форма и цветность фигур и их элементов и т.д. Все изменения моментально отражаются на системном экране. На основе этих простых и ясных для разработчика возможностей и средств могут быть реализованы сложные анимационные эффекты, значительно усиливающие графические возможности визуализации приложений.

Таким образом, средства визуализации *Flora/C+* позволяют создавать сложные графические пользовательские интерфейсы, в том числе – с элементами технической анимации и мультипликации.

### **Открытая архитектура (*Open Architecture*)**

Основанная на простых, логичных и четко сформулированных принципах, архитектура *Flora/C+* полностью прозрачна и открыта для разработчика. Ему доступны все системные элементы, необходимые для разработки приложений. Основной элемент архитектуры – дерево взаимодействующих объектов (*Object Tree*) содержит не только прикладные объекты разрабатываемой пользователем программной системы, но также – все системные объекты: устройства ввода/вывода (в том числе – мышь, клавиатуру, системный экран, устройство воспроизведения звуковых эффектов, часы, таймер и т.д.), стартовую панель, буфер обмена данными (*Clipboard*), системные переменные, функции и библиотеки, содержащие системные и прикладные объекты *Flora/C+*, с помощью которых строятся приложения и т.д. С точки зрения приложения,

разрабатываемого пользователем, обращение к системным функциям, переменным и другим объектам *Flora/C+* ничем не отличается от манипуляций с прикладными объектами, определёнными в самом приложении. Более того, дерево объектов содержит также и все инструментальные приложения самой системы программирования *Flora/C+*, в том числе: Дизайнер, Отладчик, Редактор графических объектов и другие, также разработанные в технологии *Flora/C+* на базе одних и тех же, доступных и разработчику, системных и прикладных библиотечных объектов.

Как уже отмечалось, библиотеки классов объектов открыты для расширения пользователем. Аппарат *Flora/C+* позволяет на основе библиотечных объектов строить производные библиотечные объекты – агрегаты, расширяя, таким образом, функциональность набора базовых объектов *Flora/C+*, отражающую особенности конкретной прикладной области или дополняющую систему объектов новыми библиотечными функциями, не поставляемыми разработчиком *Flora/C+*.

Отдельно необходимо сказать о встроенном языке программирования *F++*, который синтаксически подобен *C++* и *Java* и включает стандартный набор объектных расширений языка, конструкции структурного программирования (такие как *if*, *switch*, *for*, *while*), все операторы языка *C++*. Элементам дерева объектов, разрабатываемым с помощью средств *F++*, также доступны все системные и прикладные объекты, функции и переменные, содержащиеся в дереве объектов. Программы, разработанные на *F++*, позволяют манипулировать с данными всех базовых типов, читать и изменять свойства объектов, вызывать методы и функции. Язык *F++* предоставляет возможности описания локальных переменных и объектов, определения структур, обработки исключительных ситуаций и т.д.

Таким образом, система программирования *Flora/C+* обладает всеми признаками системы с открытой архитектурой и позволяет создавать программные системы в различных областях приложения, с различными характеристиками и режимами функционирования.

#### **Поддержка многоязыковых приложений (*Multi-Lingual Applications Support*)**

В основе средств *Flora/C+*, обеспечивающих поддержку многоязыковых приложений, лежит свойство объектов типа «символьная константа» иметь несколько значений в зависимости от значения некоторой системной переменной. На основе этого свойства могут быть построены приложения с несколькими различными языковыми интерфейсами.

Сама система программирования *Flora/C+* является таким многоязыковым приложением, поддерживающим в настоящее время два европейских языка: английский и русский и открыта для расширений.

### **3. Системная архитектура**

Архитектура программного обеспечения *Flora/C+* (*Flora System Architecture*) играет ключевую роль в обеспечении таких свойств как независимость, переносимость, портируемость и масштабируемость.

Программное обеспечение *Flora/C+* состоит из нескольких программных слоев (см. Рисунок 2, стр. 14), самым нижнем (базовым) из которых является микродро (*Micro-*



*Core*). Как уже указывалось выше, микроядро является единственной системно зависимой частью *Flora/C+* (*System Dependent Code*), непосредственно взаимодействующей с операционной системой, на которой исполняется инстанция *Flora/C+*. Этот слой программного обеспечения представляет также элементарные базовые объекты, свойства которых абстрагированы от свойств конкретной платформы и операционной системы. Эти базовые объекты используются на следующих уровнях программного обеспечения *Flora/C+* и поддерживают фундаментальные базовые функции операционной системы:

- управление файловой системой (*File System Management*)
- управление памятью (*Memory Management*)
- управление задачами (*Task Management*)
- операции ввода/вывода (*Input/Output*)
- управление мышью, клавиатурой и системным экраном (*Mouse, Keyboard, Screen*)

Общий объем программного кода микроядра составляет менее 1% от общего объема программного кода *Flora/C+*, что составляет около 150K программного текста, представленного в исходных кодах. Этим обстоятельством определяется невысокая трудоемкость переноса программного обеспечения *Flora/C+* на другие платформы или в среду других операционных систем.

На следующем уровне программного обеспечения *Flora/C+* находится объектная машина (*Object Engine*). Этот уровень программного обеспечения отвечает за организацию вычислительной среды *Flora/C+* и обеспечивает, в частности:

- графический пользовательский интерфейс (*Graphical User Interface*)
- мультиоконный интерфейс (*Multi-Windows Interface*)
- поддержку и обслуживание дерева объектов (*Object's Tree Support*)
- встроенный язык программирования *F++* (*Build-In Program Language – F++*)
- поддержку библиотек классов *F++* (*F++ Classes Library*)
- локальные и глобальные сетевые коммуникации (*Local & Global Networking*)
- взаимодействие распределённых приложений (*Flora Net*)
- интерфейсы с базами данных (*DBMS Interface*)

В организованной объектной машиной древовидной объектно-ориентированной вычислительной среде разрабатываются, хранятся и исполняются взаимодействующие между собой системные и прикладные программные объекты. На этом уровне поддерживается прикладная мультизадачность, механизмы синхронизации и блокировок задач, средства сетевого взаимодействия распределённых приложений (*Flora Net*), реализованы механизмы создания агрегатов объектов, поддерживается широкий набор базовых системных и прикладных классов объектов.

Общая доля программных кодов, реализующих функции объектной машины от общего объема программ *Flora/C+* составляет около 33%.

Вместе с микроядром (*Micro-Core*) объектная машина (*Object Engine*) образует инстанцию *Flora/C+* (*Flora Instance*) в среде которой исполняются все системные и прикладные модули следующих уровней







На следующем слое программного обеспечения находятся инструментальные средства разработки и программирования *Flora/C+* (*Development Tool Kit*). Большая часть этих инструментальных средств написана методом раскрутки (рекурсивного программирования), т.е. написана «сама на себе» (*Recursive Development Products*).

Занимающие по объему программного кода большую часть (примерно 66%), инструментальные средства *Flora/C+*, в частности, включают следующие приложения:

- Дизайнер (*Designer*)
- Редактор графических объектов (*Painter*)
- Менеджер приложений (*Application Manager*)
- Транслятор *F++* (*F++ Translator*)
- Инспектор классов (*Class Inspector*)
- Менеджер проектов (*Project Manager*)
- Журнал событий (*Event Log*)
- Отладчик (*Debugger*)
- Генератор типовых приложений (*Application Wizard*)
- Менеджер секций (*Section Manager*)
- Редактор иконок (*Icon Editor*)
- Центр разработки (*Development Centre*)

Программное обеспечение объектной машины (*Object Engine*) и инструментальных средств (*Development Tool Kit*) обладает свойством системной независимости программного кода от особенностей архитектуры платформы и операционной системы, под управлением которых выполняется инстанций *Flora/C+*.

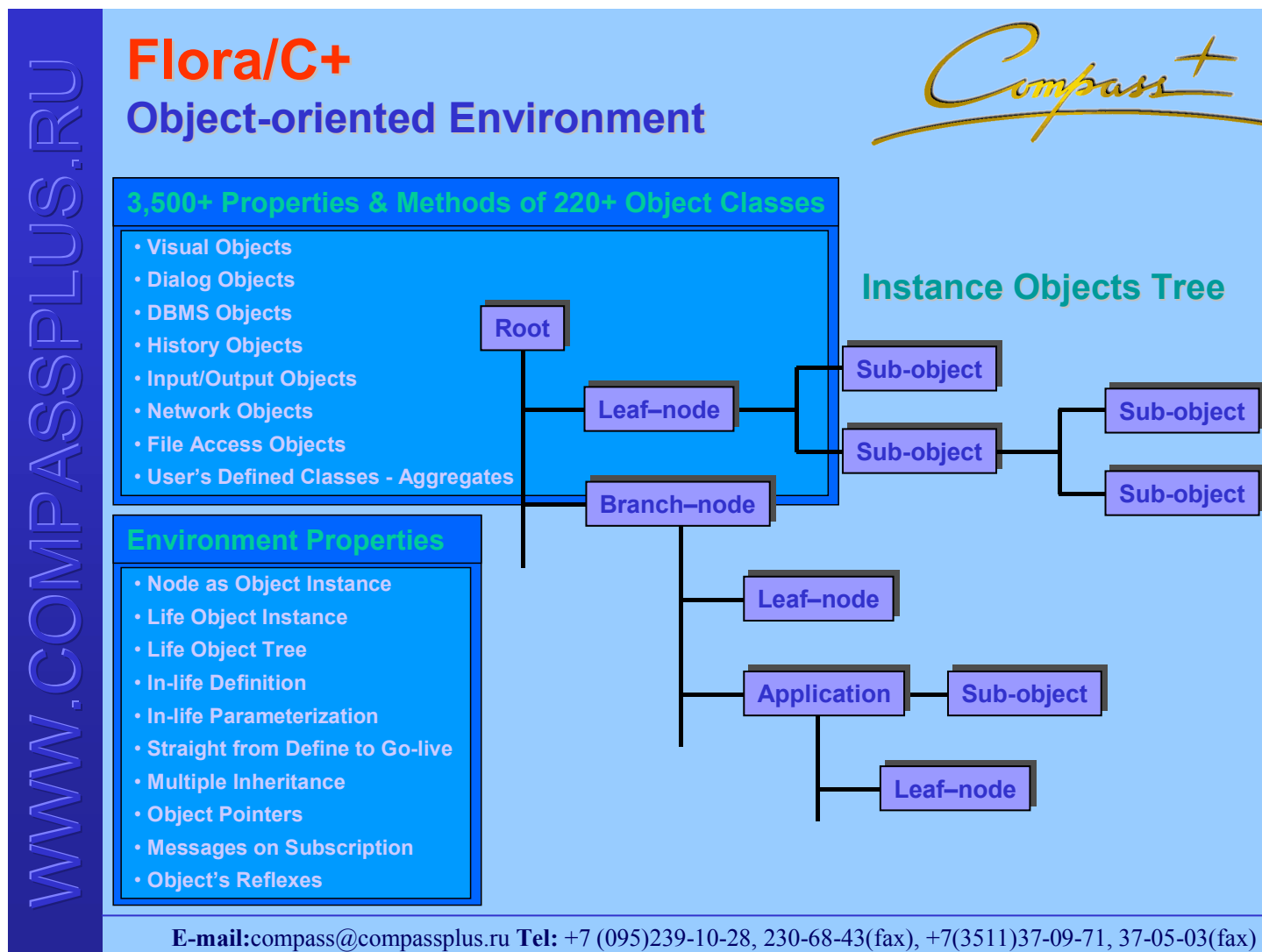
На последнем уровне программного обеспечения находятся пользовательские приложения, разработанные, отлаженные и скомпонованные с помощью инструментальных средств *Flora/C+*, выполняющиеся в среде и под управлением объектной машины *Flora/C+*. Этот слой программного обеспечения обладает свойствами полной переносимости программного кода (*Completely Portable Codes*).

#### **4. Объектно-ориентированная среда разработки приложений**

Как уже отмечалось выше основным элементом системы программирования *Flora/C+* является объектно-ориентированная вычислительная среда (*Flora Object-Oriented Environment*), организованная в виде дерева взаимодействующих между собой объектов (см. Рисунок 3, стр. 16). Рассмотрим свойства дерева объектов более подробно.

Каждая инстанция *Flora/C+* содержит единое дерево объектов (*Instance Object Tree*), содержащее как системные так и пользовательские приложения. Дерево объектов имеет единственный корень (*Root*). Любое приложения *Flora/C+* (*Application*) существует в виде части дерева объектов инстанции и в форме поддеревя объектов.





**Рисунок 3. Объектно-ориентированная среда разработки приложений *Flora/C+***

Узлы дерева объектов сами являются объектами. Существует два типа таких объектов – «ветви» (*Branch-node*) и «листья» (*Leaf-node*). «Листья» не содержат других подчинённых объектов, а «ветви» могут содержать произвольное количество других объектов типа «листья» и «ветви». «Ветви» и «листья» дерева называются узловыми объектами. В узлах могут содержаться объекты, подчиненные узловому объекту. И тогда они называются «подобъектами» (*Sub-object*). «Подобъекты» некоторых классов в свою очередь могут содержать другие «подобъекты» и т.д.

Таким образом, каждый объект приложения существует в виде узла или подчиненного объекта («подобъекта») дерева объектов. Здесь мы подошли к ключевому отличию свойств (*Environment Properties*) объектного окружения *Flora/C+*, выделяющему её среди других объектно-ориентированных средств разработки, предлагаемых на рынке в настоящее время. Таким отличием является сам способ существования объектов в виде дерева объектов, предполагающий концептуальное объединение описания объекта и его инстанции. Действительно, как только описание объекта определено, а определено оно может быть только в форме узла дерева объектов, сразу порождается инстанция этого объекта (*Node as Object Instance*). Иными словами, формой существования объекта в объектно-ориентированной среде *Flora/C+* является его инстанция. Это принципиальное свойство порождает массу весьма интересных и перспективных точек зрения, свойств и следствий. Одним из них является взгляд на объект как на «живую инстанцию» (*Life Object Instance*), а на дерево объектов как на «живое» дерево объектов (*Life Object Tree*). Несмотря на кажущуюся на первый взгляд искусственность аналогий дерева объектов с живыми формами жизни, такой взгляд на дерево объектов весьма точно отражает суть технологических принципов создания и существования приложений в среде системы программирования *Flora/C+* и является во многом весьма продуктивным (отсюда, кстати и название продукта – *Flora*).

В соответствии с этим взглядом приложения в среде *Flora/C+* не пишутся (стадия написания отсутствует как таковая), а «выращиваются», как выращивается дерево – ветвь за ветвью, лист за листом. При этом каждый новый элемент начинает «жить» как только он добавляется в узел дерева объектов (*In-life Definition*) и его появление отражается на других объектах, если определены какие-либо связи между ними (*Straight from Define to Go-Live*). Так, например, добавленные в дерево объектов окно, меню, диалоговый элемент или другие графические фигуры тут же появляются на системном экране и с ними уже можно производить какие-либо манипуляции. Таким же образом сразу после добавления в дерево объектов инициируются новые задачи, активизируются порты ввода/вывода, запускаются программы, написанные на языке *F++* и т.д. *Flora/C+* предлагает также средства, позволяющие пользователю самому управлять процессом запуска и остановки инстанций объектов.

Аналогично, если существующий в дереве объект имеет сложную структуру с возможностью параметризации, изменение его свойств может быть выполнена «в живую» (*In-life Parameterization*) разработчиком с помощью встроенного редактора или динамически из других объектов и программ.

Каждый объект, включенный в дерево объектов, является экземпляром некоторого класса. Классы связаны между собой отношениями множественного наследования (*Multiple Inheritance*). У каждого базового класса может быть несколько производных

классов и у каждого производного - несколько базовых классов. Каждый класс определяет совокупность свойств и методов, которыми обладают объекты этого класса. Свойства – это значения и/или массивы значений, которые задают и характеризуют текущее состояние объекта, а методы - это функции, которые могут быть выполнены объектом. Производные классы наследуют свойства и методы базовых классов. Часть классов *Flora/C+* являются абстрактными. Они служат только для определения свойств и методов, которые наследуются производными от них классами. Инстанции абстрактных классов не могут существовать в дереве объектов. Свойства всех объектов относятся к одному из четырех базовых типов: целое число, действительное число, строка произвольной длины и указатель на объект. Практически все классы являются производными от базового класса *Object*. Этот класс определяет два свойства: имя (*Name*) и статус (*Status*). Статус определяет состояние инстанции объекта (остановлена, загружена, инициирована, связана, запущена). При запуске приложения статус входящих в его состав объектов поэтапно поднимается, а при закрытии приложения - уменьшается.

С помощью указателей на другие объекты (*Object Pointers*) организуются прямые связи между объектами дерева объектов. Указатели объектов вместе с механизмом передачи сообщений (*Messages on Subscriptions*) предоставляют разработчику средства прямого взаимодействия объектов и задач.

Отдельно необходимо сказать об объектах, входящих в классы «рефлексов» (*Object's Reflexes*). Классы этой группы порождают отдельные задачи и предназначены для описания реакции объектов на различные события и сообщения. Количество классов «рефлексов» достаточно велико. Приведем здесь в качестве примера только следующие из них:

- |                             |  |
|-----------------------------|--|
| • <i>Reflex View Pos</i>    | визуализация изменений числа в виде перемещения фигуры |
| • <i>Reflex View Rotate</i> | визуализация изменений числа в виде вращения фигуры    |
| • <i>Reflex Set Int</i>     | установка целого значения при нажатии фигуры           |
| • <i>Reflex Hint</i>        | привязка к фигуре всплывающей подсказки                |
| • <i>Reflex Disable</i>     | управление блокировкой диалогового элемента            |

Кроме приведенных в настоящем разделе классов объектов библиотеки *Flora/C+* содержат более 220 других классов, предоставляющих разработчику в целом свыше 3,500 свойств и методов (*3,500+ Properties & Methods of 220+ Object Classes*), в частности, следующих:

- визуальные объекты (*Visual Objects*)
- диалоговые объекты (*Dialog Objects*)
- объекты работы с базами данных (*DBMS Objects*)
- объекты работы с историей изменения переменных и параметров (*History Objects*)
- объекты ввода/вывода (*Input/Output Objects*)
- объекты сетевого взаимодействия (*Network Objects*)
- объекты доступа к файловой системе (*File Access Objects*)
- пользовательские объекты – агрегаты (*User's Define Classes - Aggregates*)

Таким образом, *Flora/C+* предоставляет разработчику не только широкий набор объектно-ориентированных средств, но также создает уникальную объектно-ориентированную среду разработки и исполнения приложений, позволяющую не



писать программные модули, как это традиционно производится, а «выращивать» (собирать, связывать, настраивать и т.д.) эти модули в специально организованной среде. Более чем на 80% написанная «сама на себе» *Flora/C+* наглядно демонстрирует эффективность концептуально нового подхода к организации вычислительного процесса в виде дерева взаимодействующих объектов.

### 5. Среда разработки мультизадачных приложений

Так же как и объектно-ориентированная среда, развитая мультизадачность (*Flora Build-In Multi-tasking Features*) является одним из основных свойств системы программирования *Flora/C+* (см. Рисунок 4, стр. 20).

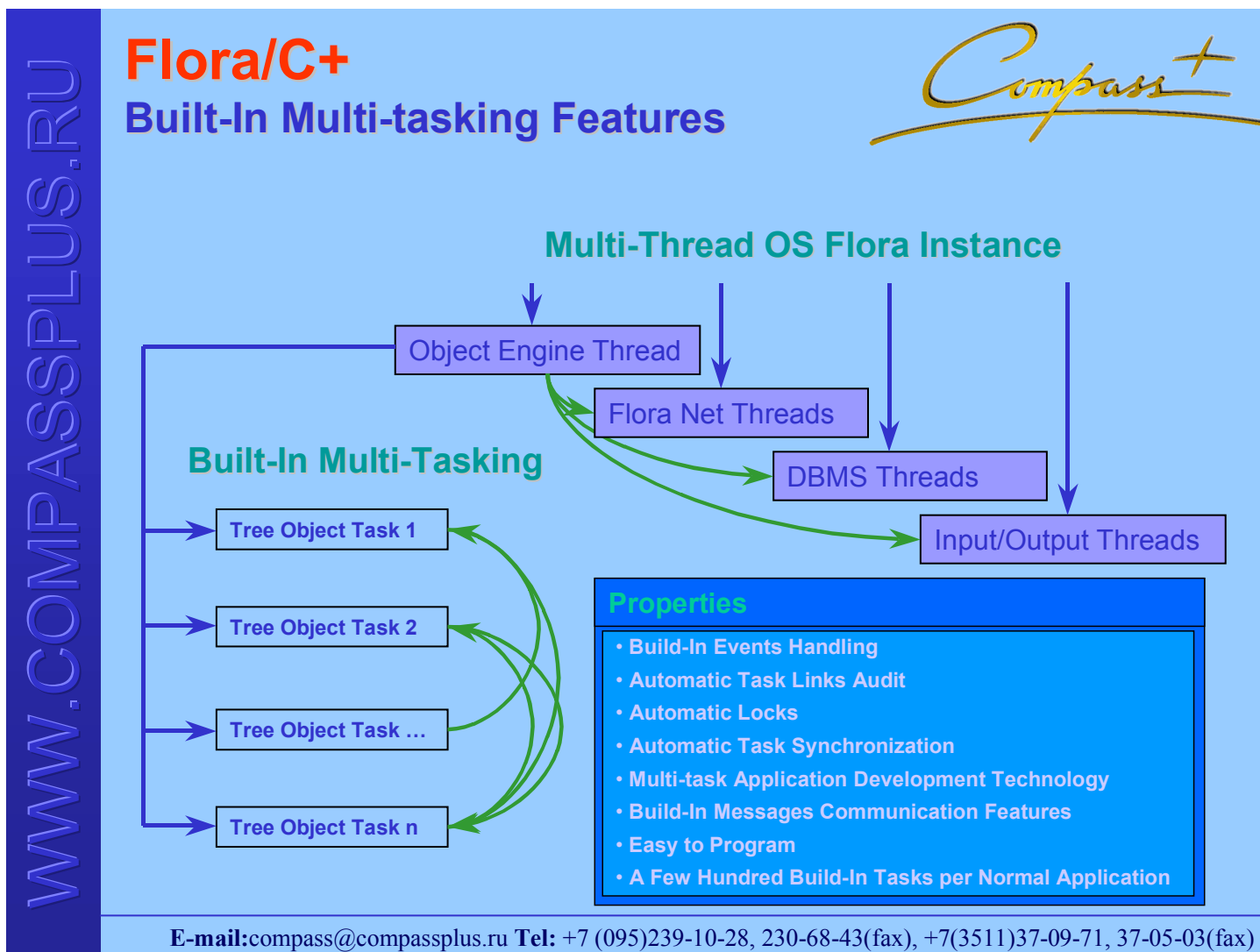
Мультизадачность не только является внутренним способом реализации программного обеспечения самой системы программирования *Flora/C+*, но и мощным инструментом проектирования приложений в руках пользователя. Наличие большого количества параллельно выполняющихся задач для приложений *Flora/C+* является скорее правилом, чем исключением. При этом *Flora/C+* берет на себя все заботы по обеспечению взаимных защит и блокировок, необходимых для надежной работы мультизадачных приложений.

Особенность мультизадачности *Flora/C+* по сравнению с мультизадачностью операционных систем состоит в том, что её задачи тесно связаны друг с другом, имеют доступ к одним и тем же объектам и не ограничены такими жесткими механизмами взаимодействия, как сообщения, каналы, семафоры и т.п.

Сама по себе инстанция *Flora/C+* является мультинитевым приложением (*Multi-Thread OS Flora Instance*) с точки зрения операционной системы, в которой она исполняется. Существуют по крайней мере следующие параллельно исполняемые нити инстанции: нить объектной машины (*Object Instance Thread*), нити обслуживания сетевого взаимодействия (*Flora Net Threads*), нити обслуживания взаимодействия с базами данных (*DBMS Threads*), нити обслуживания операций ввода/вывода (*Input/Output Thread*).

На основе встроенных средств мультизадачности *Flora/C+* (*Build-In Multi-Tasking*) строятся асинхронно выполняющиеся приложения в объектно-ориентированной среде *Flora/C+*.

Теоретически любой объект или группа объектов могут быть отдельной задачей. Возникающие проблемы и коллизии, связанные с асинхронной обработкой данных в мультизадачной среде, решаются автоматически объектной машиной *Flora/C+*. Объекты класса «задача» содержат программу на языке *F++* и подобъекты – переменные и внешние ссылки на объекты в дереве объектов. Запуск задачи на выполнение может происходить периодически и при изменении значений внешних ссылок. Одновременно и асинхронно может выполняться множество таких задач.



**Рисунок 4. Средства разработки мультизадачных приложений *Flora/C+***

Другим типом задач являются так называемые «рефлекс-задачи», порождаемые классами «рефлекс». Являясь подобъектами для объектов некоторых классов, эти задачи запускаются при возникновении определенных событий, относящихся к объекту – владельцу подобъекта (*Build-In Events Handling*). Будучи, например, подобъектом для визуального диалогового объекта "кнопка", «рефлекс-задача» может быть запрограммирована на выполнение определенных операций при нажатии кнопки пользователем.

Простота организации асинхронной обработки данных, с одной стороны, когда функции контроля связей между задачами (*Automatic Task Link Audit*), контроля блокировок (*Automatic Locks*) и синхронизации задач (*Automatic Task Synchronization*) выполняется автоматически, а также широкий набор объектно-ориентированных средств организации мультизадачных приложений, с другой стороны, позволяет говорить о технологии *Flora/C+* как о технологии разработки мультизадачных приложений (*Multi-task Application Development Technology*).

Объекты дерева объектов, принадлежащие разным задачам, могут получать сообщения друг от друга или из внешнего мира, обрабатывать их и генерировать сообщения для других объектов (*Build-In Messages Communication Features*). Работа приложения представляет собой совокупность реакций его объектов на внешние события. Пути передачи сообщений могут быть как статическими, так и динамически изменяющимися в ходе работы.

Перечисленные свойства внутренней мультизадачной среды *Flora/C+* принципиально облегчают программирование мультизадачных приложений (*Easy to Program*), что на практике провоцирует разработчика к созданию приложений с развитыми средствами асинхронной обработки данных. Разработка асинхронных приложений становится по силам даже разработчику с невысокой квалификацией и опытом работы в этой области. Повторим еще раз, что наличие большого количества параллельно выполняющихся задач в приложениях, разработанных в среде *Flora/C+*, является правилом (*A Few Hundred Build-In Tasks per Normal Application*).

В заключении необходимо отметить, что мультизадачность среды исполнения приложений и самих приложений *Flora/C+* является концептуальным основанием организации глубоко распараллеленного вычислительного процесса на базе технологии системы программирования *Flora/C+*.

## **6. Архитектура распределенных приложений**

Как было описано ранее, программное обеспечение микроядра и объектной машины *Flora/C+* образуют инстанцию, представляющую собой программный модуль, исполняемый под управлением операционной системы. Инстанция *Flora/C+* обеспечивает вычислительную среду, внутри которой разрабатываются и исполняются приложения. Инстанции *Flora/C+* могут быть размещены на нескольких различных, объединенных сетью передачи данных, компьютерах, серверах или узлах кластерного сервера. В этом случае *Flora/C+* образует распределённую инстанцию (*Flora Multi-Node Instance*), которая может поддерживать распределённые приложения, отдельные





модули которого исполняются в разных узлах сети передачи данных (см. Рисунок 5, стр. 23).

Взаимодействие между приложениями (*Application*), находящимися в разных экземплярах *Flora/C+* (*Flora Instance*), выполняется на основе сетевых средств *Flora/C+* - *Flora Dual Ports* и *Flora Net*.

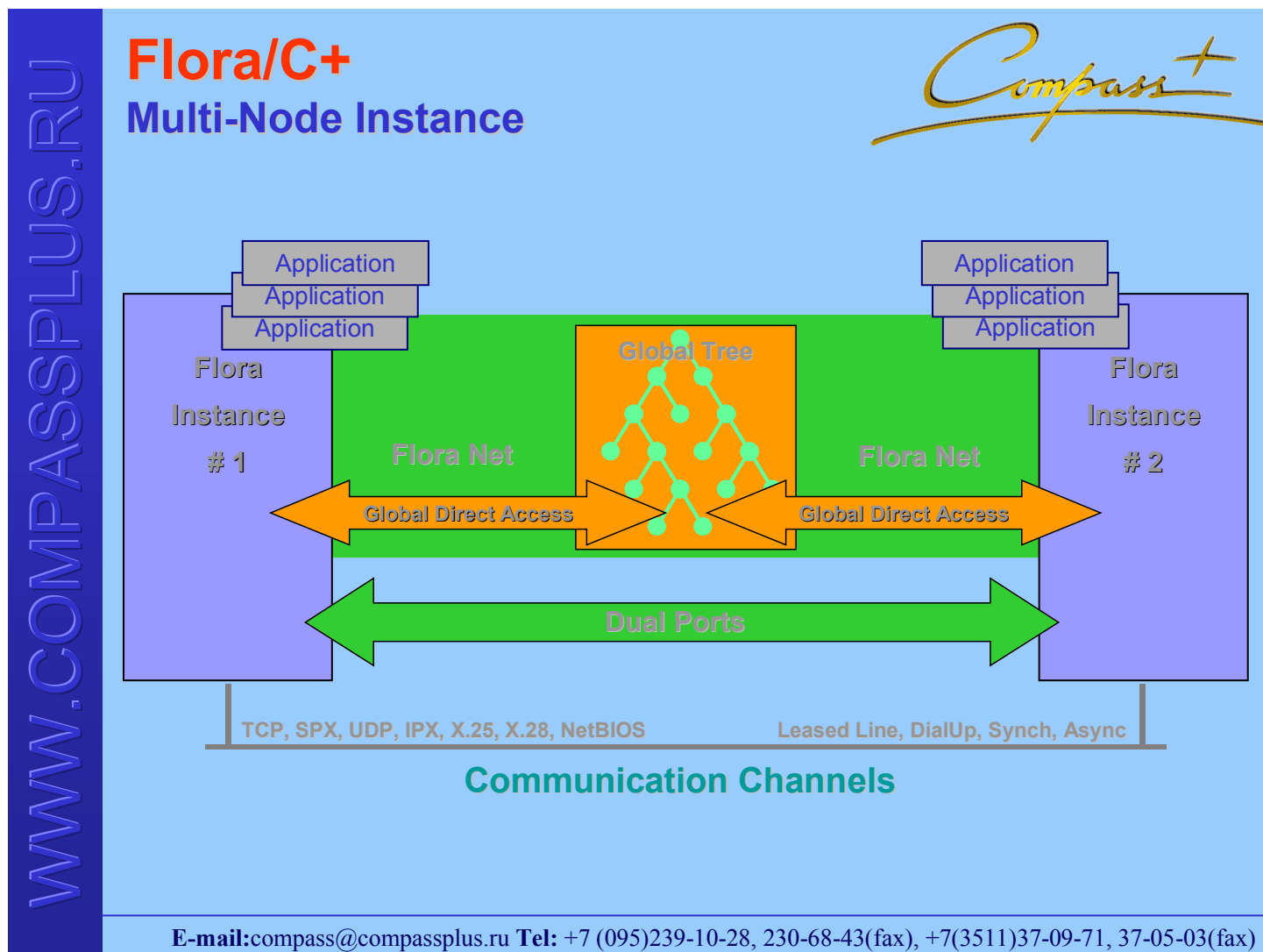
Средства *Flora Dual Ports* реализуют базовый уровень ввода/вывода и поддерживают различные коммуникационные протоколы, в том числе – *TCP*, *SPX*, *UDP*, *IPX*, *X.25*, *X.28*, *NetBIOS* и методы *Leased Line*, *DialUp*, *Sync*, *Async*. На базе этих средств могут быть организованы как локальные так и глобальные сетевые конфигурации распределённых приложений. Порты ввода/вывода используются для поточной передачи данных между объектами разных экземпляров.

Средства *Flora Net* обеспечивают создание общего дерева объектов (*Global Tree*), элементы которого доступны с помощью прямого обращения, как если бы эти элементы были частью дерева объектов каждой из участвующих в сетевом обмене экземпляров. Типология сети и потоки передаваемых данных описывается с помощью объектов специального типа «карта сети», «каталог на карте сети», «библиотека на карте сети». С помощью этих объектов организуется прямой обмен данными в сети (чтение, запись данных целого, вещественного или строкового типа) или удаленный вызов функций в сети (*Global Direct Access*).

Таким образом, сетевые средства системы программирования *Flora/C+* позволяет естественным, прозрачным для разработчика способом создавать распределенные приложения, исполняемые в различных узлах сети. Механизмы, обеспечивающие взаимодействие распределенных приложений, поддерживают как слабо связанные объекты, взаимодействующие только в режиме передачи данных (*Flora Dual Ports*) так и сильно связанные объекты, взаимодействующие в режиме прямого обращения (*Global Direct Access*).

На основе сетевых средств *Flora/C+* могут быть разработаны масштабируемые приложения, поддерживающие в том числе отказоустойчивые решения на базе специальных кластерных серверных конфигураций.





**Рисунок 5. Архитектура распределенных приложений *Flora/C+***



## 7. Масштабируемость решений

Возможность масштабирования решений *Flora/C+* (*Flora Scalability*) базируется на следующих описанных ранее свойствах (см. Рисунок 6, стр. 23) – переносимости (*Cross-Platform Portability*), системной независимости (*Operating System Independence*) и распределённости (*Multi-Node Instance*), которые обеспечивает архитектура инстанций *Flora/C+*.

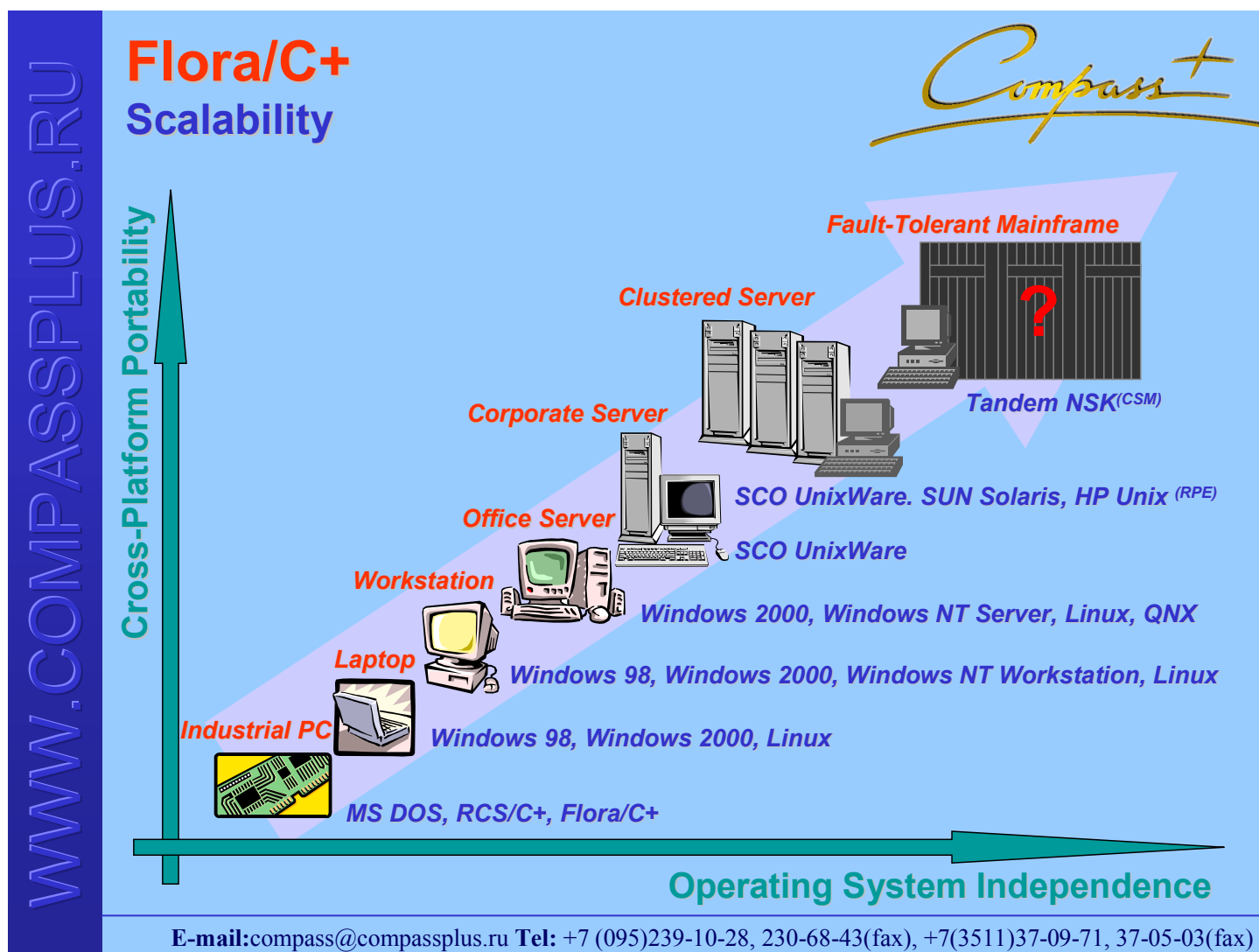
Масштабируемость приложений *Flora/C+* в классическом понимании обеспечивается возможностью наращивания производительности системы пропорционально возросшему объёму обрабатываемых данных с помощью простого увеличения числа единиц находящихся на разных узлах серверной платформы обрабатывающих инстанций, в среде которых исполняются приложения.

Сетевые средства *Flora/C+* позволяют создавать также и наиболее эффективные с практической точки зрения линейно масштабируемые приложения при условии, что аналогичными свойствами обладает сама серверная платформа.

В более широком смысле масштабируемость также означает и возможность исполнения приложения без каких-либо модификаций на различных вычислительных системах – от самых простых и дешевых рабочих станций до высокопроизводительных кластерных систем и мэйнфреймов.

Задуманная как переносимая система программирования *Flora/C+* в настоящее время доступна для использования на целом ряде платформ.

На самом нижнем уровне вычислительных систем, на которых возможно использование описываемой технологии, находятся *PC* промышленного исполнения (*Industrial PC*) и процессорные платы, применяемые в системах автоматизации технологических процессов, диспетчерского управления и сбора данных (*SCADA*). Для компьютеров этого уровня предназначена специально разработанная *SCADA*-система, называемая *RCS/C+*, совместимая по интерфейсам с *Flora/C+* и исполняемая в среде *MS DOS*. Для промышленных *PC* с приемлемыми ресурсами (*P-II*, 32М) возможно применение непосредственно средств *Flora/C+*, исполняемых под управлением операционной системы реального времени *QNX<sup>(DS)</sup>*. Необходимо заметить, что для запуска и функционирования инстанций *Flora/C+* вообще не требуется подключения системной консоли к исполнительному компьютеру. Функции системной консоли, если это потребуется, можно обеспечить на удаленной рабочей станции с помощью так называемого «тонкого клиента» *Flora/C+*. Это приложение позволяет имитировать системную консоль на основе команд, поступающих от исполняемой на удаленном компьютере инстанции *Flora/C+* по низкоскоростному каналу передачи данных.



**Рисунок 6. Масштабируемость решений на базе Flora/C+**



Применение *Flora/C+* возможно также и на мобильных компьютерах (*Laptop*) и рабочих станциях (*Workstation*), работающих под управлением операционных систем *Windows 98*, *Windows NT for Workstation*, *Windows 2000*, *Linux*.

Поддерживаются серверные платформы офисного уровня (*Office Server*) под управлением *Windows NT Server*, *Windows 2000*, *Linux*, *QNX<sup>(DS)</sup>*.

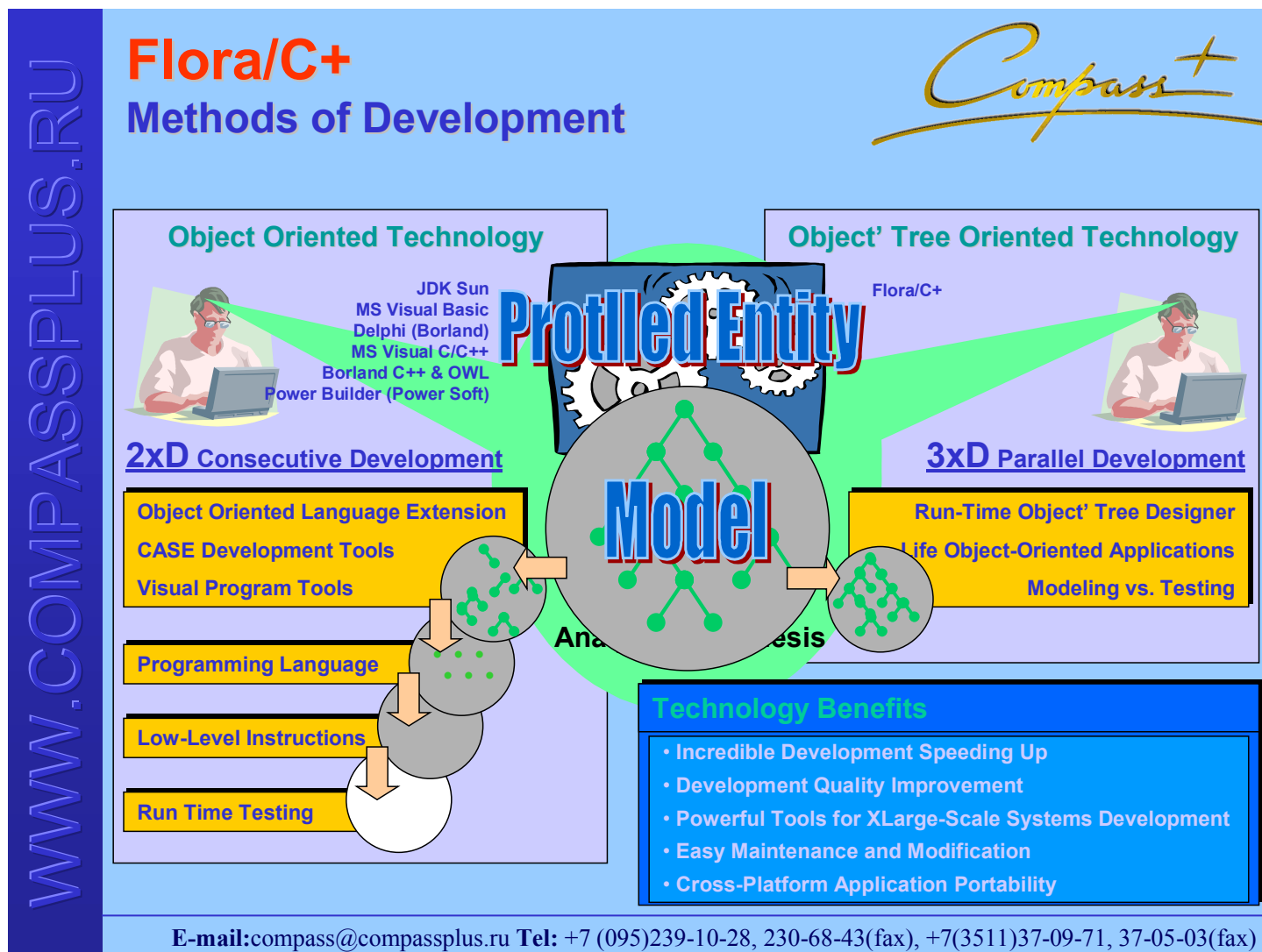
Инстанции *Flora/C+* могут исполняться на серверах корпоративного уровня (*Corporate Server*) под управлением операционной системы *SCO UnixWare* или кластерных серверах (*Clustered Server*), работающих под управлением *SCO UnixWare*, *SUN Solaris* и *HP Unix<sup>(RPE)</sup>*.

Специалистами компании *Compass Plus* проведен анализ возможности переноса программного обеспечения *Flora/C+* на серверную платформу уровня отказоустойчивого мэйнфрейма (*Fault-Tolerant Mainframe*) – *Himalaya*, работающую под управлением операционной системы *Guardian Tandem Nonstop System Kernel<sup>(CSM)</sup>*. Перенос запланирован в соответствии со стратегией развития продуктов компании и может быть выполнен по заказу клиента.

## 8. Концептуальные особенности методов разработки приложений

Как уже не раз отмечалось в предыдущих разделах, вычислительная среда системы программирования *Flora/C+* существенно отличается от традиционных представлений о технологии организации вычислительного процесса. Одним из следствий этого является также и значительное отличие методов разработки приложений *Flora/C+* (*Flora Methods of Development*) в сравнении с общепринятыми методами программирования (см. Рисунок 7, стр. 27). *Flora/C+* предлагает разработчику не только технологические и инструментальные средства программирования, но и отличный от традиционного способ представления программ и приложений, оригинальную объектно-ориентированную архитектуру вычислительной среды с нелинейной организацией памяти, принципиально отличающуюся от господствующих фон-неймановских представлений о вычислительных системах.

Таким образом, не тривиальность самой концепции организации вычислительной среды *Flora/C+* и вытекающих из неё принципиальных следствий, касающихся технологии разработки программных систем, заслуживают отдельного детального рассмотрения. Это рассмотрение тем более важно, что оно позволяет прояснить наиболее существенные концептуальные свойства *Flora/C+* и снять многие вопросы, а также психологические барьеры, возникающие у неподготовленного читателя, особенно обременённого предыдущим опытом использования традиционных объектно-ориентированных инструментов.



**Рисунок 7. Особенности методов разработки приложений *Flora/C+***

Основным понятием *Flora/C+* является дерево объектов (*Object Tree*), представляющее собой иерархически организованный набор разнородных программных объектов, в ходе исполнения и взаимодействия которых решаются задачи приложения. Собственно говоря дерево объектов в терминологии *Flora/C+* – это и есть одновременно суть разрабатываемое и исполняемое приложение. Подчеркнем, что именно одновременное существование приложения в состоянии разрабатываемого и исполняемого приложения без каких-либо других известных промежуточных стадий (генерация, компиляция, сборка, исполнение, тестирование, отладка и т.д.) является наиболее существенным следствием принятой технологии организации вычислительного процесса в системе программирования *Flora/C+*. (Заметим здесь в скобках, что более внимательные рассуждения показывают, что совмещение стадии разработки и исполнения приложений в среде *Flora/C+* является следствием более фундаментального свойства объектов дерева объектов – совмещения определения объекта с его экземпляром, т.е. фактически с ним самим. Однако речь о свойствах объектов *Flora/C+* пойдет несколько ниже.)

Традиционные современные инструментальные средства разработки предлагают проектировщику объектно-ориентированные (*Object Oriented*) средства программирования, представленные, как правило, в виде некоторого расширения возможностей традиционного языка программирования. Такими, широко используемыми в настоящее время средствами являются, например, *JDK Sun*, *Microsoft Visual Basic*, *Delphi* (Borland), *Microsoft Visual C/C++*, *Borland C++ & OWL*, *Power Builder* (Power Soft) и другие. Заметим, что крайне плодотворная идея объектно-ориентированного подхода к разработке программ, родившаяся в начале 70-х и приведшая к началу 90-х годов к полному пересмотру методов процедурного программирования, продолжает существовать в перечисленных выше средствах программирования и через 30 лет лишь в виде надстройки над традиционными языками программирования. Существует только как их языковое расширение!

В отличие от такой упрощенной реализации объектно-ориентированной идеи, *Flora/C+* предлагает не только и даже не столько объектно-ориентированные расширения языка программирования, а нечто гораздо более существенное – «объектно-ориентированно» организованную вычислительную среду, внутри которой приложение может существовать исключительно в своей объектно-ориентированной форме.

Более того, технология *Flora/C+* делает еще один существенный шаг – в основу организации памяти (хранилища объектов) положена другая крайне плодотворная концепция – идея древовидной иерархической структуры.

Остановимся на обсуждении этой концепции более подробно.

Вообще говоря, понятие древовидной (иерархической) структуры универсально и крайне продуктивно само по себе. Это понятие, широко применяемое во всех областях человеческого знания, отражает фундаментальные закономерности реального мира и лежит в основе формального описания природы многих сущностей, процессов и явлений. Древовидные (иерархические) модели применяются повсюду: при описании биологических и социальных систем, процессов развития, роста, деградации и смерти, устройства и функционирования технических систем и систем управления, при построении математических и логических моделей, при описании структур данных в

вычислительных системах, в математических и физических теориях, при описании строения материи и вселенной и т.д. и т.п.

Даже в основе самого процесса познания объектов реального мира человеческим разумом лежат основополагающие методы, связанных с понятием древовидной структуры: метод анализа и метод синтеза. Первый метод предполагает разделение целого на составляющие части и последовательное исследование и изучение этих частей, которые затем в свою очередь подвергаются аналогичной процедуре. Таким образом, метод анализа можно сравнить с движением по дереву снизу вверх (выращиванием) – от корня к ветвям от ветвей к другим ветвям и далее к листьям. Метод синтеза напротив предполагает умозрительную попытку получения целого на основе составляющих его частей. И в этом смысле его можно сравнить с движением по дереву сверху вниз – от листьев к ветвям и от ветвей к другим ветвям и далее к корню. В применении к процессу проектирования эти два метода находят отражение, например, в известных методах структурного программирования «сверху вниз» (процесс анализа) и «снизу вверх» (процесс синтеза).

Таким образом, обобщая сказанное, можно утверждать, что использование дерева в качестве местоположения объектов *Flora/C+* дает разработчику идеальную среду для декомпозиции приложений и, в конечном итоге – для создания моделей объектов реального мира.

Возвращаясь к рассмотрению технологии разработки программных систем, отметим, что в самом общем смысле цель создания приложения заключается в изготовлении действующей в виртуальном мире (роль которого играет вычислительная среда) модели объекта реального мира (роль которой играет приложение). Эта модель в данном контексте и является результатом процесса познания разработчиком автоматизируемых им объектов и процессов реального мира.

В процессе проектирования представление об объекте моделирования (*Protitled Entity*) – см. Рисунок 7, стр. 27, складывается в голове разработчика в виде некоторой умозрительной модели (*Model*). Интуитивно применяя методы анализа (*Analysis*) и синтеза (*Synthesis*), разработчик изучает свойства частей объекта моделирования и определяет связи между ними. Очевидно, что в соответствии с отмеченными выше свойствами этих методов модель объекта, складывающаяся в результате процесса проектирования в голове разработчика, получает древовидную структуру. Безусловно, чёткость этой структуры зависит от опытности самого разработчика, последовательности соблюдения им правил принятой технологии проектирования и установленных парадигм (рамки свободы действий) и т.д. Кстати говоря, роль упомянутых выше методов структурного проектирования «сверху вниз» и «снизу вверх» как раз и состоит в навязывании разработчику правил строгого следования в процессе проектирования иерархии составных элементов объекта моделирования, а эффективность применения этих методов на практике лишний раз подтверждает правомерность утверждения о иерархическом характере интуитивной модели проектируемого объекта, складывающейся в голове разработчика.

Однако, ограничения физических возможностей мозга человека не позволяют даже опытному проектировщику детально представить, а тем более полностью проработать модель любого нетривиального объекта. Известный факт из психологии творчества – в



кратковременной («оперативной») памяти человека, т.е. памяти используемой в процессе рассуждений, может помещаться не более семи объектов. Здесь, кстати, таится один из самых распространенных источников ошибок неопытного проектировщика – если сущность состоит из более чем семи объектов, некоторые объекты и их связи могут ускользать от внимания разработчика в процессе анализа. Простой способ обойти это ограничение кратковременной памяти – использование «внешних запоминающих устройств». В качестве такого неструктурированного запоминающего устройства может использоваться, например, обычная бумага, на которую переносятся представления проектировщика об объекте моделирования, его частях и связях между ними. Эти представления обычно оформляются с применением широко известных методов хранения информации – рисунков, графиков, таблиц, текстов спецификаций, пиктограмм, формул, блок схем, алгоритмов и т.д.

При проектировании современных программных систем, когда целью является получение действующей модели объекта реального мира, в роли таких «внешних запоминающих устройств» выступают инструментальные средства, языки и системы программирования, используемые разработчиком. Обычные, применяемые на практике инструменты разработки программ, в том числе – и современные средства, перечисленные в начале данного раздела, являются такими же неструктурированными «запоминающими устройствами» с точки зрения представления интуитивной модели объекта в голове разработчика. В этом смысле они ничем существенным не отличаются от обычного листа бумаги. Более того, они даже усложняют разработчику задачу, поскольку требуют неестественного с точки зрения структуры модели объекта последовательного описания этого объекта в виде текста программ на каком-либо языке. Вообще говоря, текст программы какого-либо приложения, написанный на традиционном пусть даже объектно-ориентированном языке программирования, в такой же мере чужд самому приложению, как и, например, последовательность нажатия клавиш и движений мыши, проделанные разработчиком при написании текста программы этого приложения. Более того, применение каких бы то ни было объектных расширений языка (*Object Oriented Language Extension*), всевозможных средств автоматизированного проектирования и создания программ (*CASE Development Tools*), визуальных средств разработки (*Visual Program Tools*) в этом смысле ничего не меняет! Они лишь играют роль метаязыка (метасистемы). Описанная с их помощью модель объекта остается с одной стороны «мертвой» (не живой, неисполняемой), а с другой стороны, неадекватной интуитивно представляемой разработчиком древовидной модели объекта, поскольку, как показывает опыт, любая имеющая практическое значение модель, созданная в терминах метаязыка, является неполной и требует доработки, выполняемой разработчиком, как правило, вручную. Поэтому во всех современных средствах проектирования предусматривается либо возможность изменения сгенерированного метасистемой программного текста, либо добавление некоторых программных вставок вручную самим разработчиком.

Таким образом, используя традиционные средства автоматизации разработки приложений даже на стадии описания модели объекта в терминах предлагаемого этими средствами метаязыка (метасистемы), интуитивная древовидная модель объекта разрушается (см. рисунок). И обычно требуются некие подпорки в виде добавлений или исправлений сгенерированных метасистемой текстов программ для того, чтобы

получаемое в результате разработки приложение хотя бы выполняло требуемые функции. Более того, в результате дальнейшей обработки модели объекта, описанной разработчиком с помощью метасистемы, а именно – генерации текстов программ на каком-либо языке программирования (*Programming Language*) – *C++*, *Pascal*, *Java*, *Basic* и т.п., последующей трансляции этого текста в исполнимый на компьютере код, сборки исполняемого модуля приложения, постепенно разрушаются даже те немногие элементы структуры модели объекта, которые были описаны с помощью метаязыка. В результате на стадии исполнения приложения существуют только машинные коды (*Low-Level Instruction*), никоим образом не отражающие структуру объекта моделирования. (Если не веришь – проведи эксперимент на своем компьютере: останови любую программу и посмотри в каком виде она существует в оперативной памяти!). Львиная доля трудозатрат разработчика в процессе тестирования и отладки (*Run-Time Testing*) такой программы расходуется на интерпретацию машинных кодов в терминах своего интуитивного представления об объекте моделирования. Справедливости ради необходимо отметить, что жизнь программистов значительно облегчают символьные средства отладки. Однако применение этих средств особенно ограничено при разработке больших систем и мультизадачных приложений. Предназначены они исключительно для использования на стадии тестирования и отладки. Кроме того, максимально, на что способны эти средства, – обеспечение интерпретации исполняемого приложения в терминах исходного кода языка программирования, но никак не в терминах метасистемы.

Добавим к этим недостаткам традиционных средств автоматизации разработки программ во многих случаях практическую невозможность или значительные неудобства возврата к описанию модели в терминах метаязыка после отладки приложения или при необходимости внесения каких-либо изменений. Это делает их применение еще более ограниченным с точки зрения поддержки приложений на протяжении всего времени жизни и сводит роль этих средств к роли генераторов грубых первоначальных шаблонов программ. По сути дела они служат лишь для визуализации исходных текстов программ, написанных средствами языков программирования общего назначения! Собственно ожидать большего от традиционных средств автоматизации разработки и не приходится, поскольку в этом и заключается сверхзадача их роли!

В противоположность традиционным средствам, технология *Flora/C+* предусматривает единственную форму существования приложения – в виде «живого» («исполняющегося») дерева объектов, что в точности соответствует структуре интуитивной модели объекта, находящейся в голове разработчика. Приложение в этой форме без каких-либо изменений существует на всех стадиях его жизни, включая разработку, тестирование и эксплуатацию. Собственно говоря дерево объектов – это и есть единственная форма существования приложения в среде *Flora/C+*, какие бы действия с ним не производились и в какой бы то ни было стадии жизни оно не находилось. (Если не веришь – проведи эксперимент: запусти приложение *Flora/C+* (демонстрационную версию можно найти на [www.compassplus.ru](http://www.compassplus.ru)) и посмотри (для этого выполняющееся приложение даже не нужно останавливать!) в каком виде оно существует в памяти объектной машины)

Таким образом, древовидная объектно-ориентированная технология (*Object' Tree Oriented Technology*) *Flora/C+* является дальнейшим развитием объектно-ориентированного подхода к созданию приложений, новым качественным шагом по сравнению с формами реализации этого подхода, принятыми в традиционных объектно-ориентированных средствах разработки программ. Согласно терминологии *Flora/C+* – в процессе разработки создается не программа, а приложение, процесс разработки это – не написание программ, а создание моделей объектов реального мира. И важнейшей из используемых при этом технологических парадигм, создающих идеальную среду для декомпозиции приложений, является иерархическая структура, эффективно отражающая устройство объектов и процессов реального мира, моделирование которых и осуществляет создаваемое приложение. В этом смысле технология системы программирования *Flora/C+* последовательно реализует еще один известный и весьма плодотворный подход структурного программирования, так называемое «программирование от данных», также создающий стимулирующие ограничения на «неорганизованную мышление» разработчика. Собственно говоря приложение *Flora/C+* ничего, кроме иерархически организованных данных (объектов), и не содержит. Все свойства этих данных определены и содержатся (инкапсулированы) в них самих.

Вернемся, однако, к рассмотрению свойств моделей объектов реального мира, выполненных в технологии *Flora/C+*. Под объектом мы, как и ранее, будем понимать любую сущность, процесс или явление, моделирование которых выполняется в процессе разработки приложения. Как уже указывалось, с точки зрения технологии *Flora/C+* в создании модели объекта и заключается цель процесса разработки приложения и его результат. Вообще говоря, самой точной моделью объекта является только сам объект. Однако, этот объект до начала проектирования является в некотором роде «непознанной сущностью» (такой же, впрочем, как и модель этого объекта, представленная в виде традиционной программы!) Задача разработчика модели реального объекта состоит в познании сущности объекта реального мира и реализации его модели. С этой точки зрения сам процесс разработки приложения как модели объекта принципиально состоит из двух стадий. На первой стадии реальный объект анализируется разработчиком, выявляется его структура, компоненты, части, связи, закономерности, определяются алгоритмы его функций, исключительные ситуации, область существования параметров, ограничения, пределы и т.д. и т.п. На второй стадии, на основе полученных данных выполняется синтез собственно модели объекта. Задачи первой стадии процесса разработки относятся, вообще говоря, к проблематике задач искусственного интеллекта и слабо поддаются автоматизации с помощью современных информационных технологий. Решение задач синтеза программных моделей объектов реального мира является основной областью приложения современных средств автоматизации разработки программных систем и, в частности, областью приложения системы программирования *Flora/C+*.

Очевидно, что для облегчения решения задачи синтеза модели объекта разработчику необходима специализированная технология, обеспечивающая естественную среду воспроизведения этой модели, а также набор специальных инструментов, позволяющих ему эффективно и легко моделировать структуру объекта, описывать свойства его частей и их функции, определять связи между этими частями, устанавливать

процедуры реакции на исключительные ситуации и т.д. Такой средой и является дерево объектов *Flora/C+*, состоящее из совокупности взаимодействующих между собой элементарных программных объектов (или просто – объектов), которые и составляют суть приложение, моделирующее объект реального мира.

Во время проектирования приложения разработчик постепенно собирает модель объекта реального мира из программных объектов *Flora/C+*, выбирая последние из библиотек классов объектов (системных или пользовательских) и помещая их в дерево объектов. Объект, помещённый в дерево объектов, незамедлительно порождает свою инстанцию (конкретный экземпляр) и, в этом смысле, описание объекта в дереве объектов совмещено с самим объектом (его инстанцией). При этом используются как предопределённые классы объектов, которых насчитывается, как уже указывалось выше, более 220, так и объекты собственных, определяемых самим разработчиком классов (в терминологии *Flora/C+* – агрегатов). Частным случаем объекта, помещаемого в дерево объектов, является объект типа «программа». Объекты этого типа создаются с помощью встроенного языка программирования *F++*. В этом языке нет средств описания данных. Более того, кроме локальных переменных в программах вообще нет данных. Все данные, которыми манипулирует программа, находятся вне программы, а именно – в дереве объектов. Точнее говоря, сами объекты дерева объектов являются данными, которыми манипулирует программа. При этом программа не владеет этими данными, она лишь может управлять ими на уровне свойств объектов. Объекты типа «программа» не являются обязательными для использования в проектируемых в среде *Flora/C+* приложениях. Большинство «врождённых» действий выполняется самими объектами. Встраиваемые же в дерево объектов программы задают какие-либо нестандартные действия. На практике текст программ редко превышает несколько строк. Программы являются обычными объектами в дереве объектов, ими также можно манипулировать.

Помещённый таким образом объект в дерево объектов *Flora/C+* сразу начинает «жить» (в том числе, в зависимости от своих свойств и местоположения – контактировать с другими объектами дерева объектов и окружающим миром), т.е. становится частью и выступает как часть создаваемой модели объекта реального мира. Например, как только определен какой-либо графический элемент, он активизируется и становится доступным, его можно видеть на системной консоли, им можно управлять. Другие примеры: если определен объект типа «задача», порождается соответствующий асинхронный процесс; объекты ввода/вывода активизируют порты, коммуникационное и сетевое оборудование, устанавливают логические соединения; объекты базы данных устанавливают соединение с базой данных, активизируют курсоры; объект типа «встроенная программа» получает управление и начинает исполняться и т.д. и т.п. Таким образом, как и в реальном мире, в среде *Flora/C+* нет разделения понятий определения объекта и конкретной инстанции объекта. Каждый объект целен сам по себе и является одновременно и определением (самого себя) и (своей) инстанцией. В нем нет разделения на «команды» и «данные». Все свои свойства объект носит с собой.

Таким образом, создаваемая проектировщиком в среде *Flora/C+* модель объекта сразу начинает «жить» (в традиционной терминологии – выполняться). Отсюда – использование термина «выращивание приложения» вместо традиционного – «написание программы». Образно говоря, технологию создания приложений в среде

*Flora/C+* можно сравнить с 3-х мерным параллельным процессом (*3xD Parallel Development*) выращивания «живой» модели реального объекта в противоположность 2-х мерному последовательному процессу (*2xD Consecutive Development*) описания этого объекта, соответствующему традиционному способу написания программ.

Отложив на несколько страниц вперед обсуждение особенностей отладки и тестирования приложений в технологии *Flora/C+*, отметим здесь только тот факт, что созданное таким методом приложение без каких-либо изменений согласно идеологии *Flora/C+* существует, может использоваться и действительно используется в реальном режиме эксплуатации.

Прервемся здесь ненадолго, чтобы ответить на сакраментальный вопрос прагматичного, осведомленного и нетерпеливого читателя – а какова эффективность исполнения приложений *Flora/C+*? Заметим на это, что проблемы производительности не связаны напрямую с рассматриваемыми в настоящем разделе методами разработки приложений, а более относятся к эффективности реализации собственно технологии *Flora/C+*, обсуждение которой выходит далеко за рамки настоящего документа. Тем не менее, ответ на поставленный вопрос состоит в том, что реализация способа организации вычислительного процесса *Flora/C+* и исполнения приложений не требует применения классических методов интерпретации, используемых в подобных случаях и катастрофически снижающих эффективность исполнения приложений. Одной из основных причин этому является тот факт, что все методы стандартных библиотечных классов объектов *Flora/C+* встраиваются в код самой объектной машины. Поэтому исполнение приложений *Flora/C+* практически равноценно по производительности исполнению программ, полученных с помощью эффективного транслятора *C++*.

Однако вернемся к технологии разработки приложений в среде *Flora/C+*.

Как уже было отмечено, разработка приложений производится методом «выращивания» «живого» дерева объектов. В соответствии с этим, а также в силу того, что описание объектов и сами объекты в дереве объектов *Flora/C+* совмещены, процессы разработки, тестирования и исполнения приложений также оказываются совмещенными, т.е. выполняются одновременно. Собственно же процесс «выращивания» нового приложения, равно как и любая корректировка или модификация объектов и дерева объектов уже созданного приложения выполняется в среде *Flora/C+* с помощью специального графического приложения – Дизайнера, который, таким образом, по своей сути оказывается средством разработки приложений времени исполнения (*Run-Time Object's Tree Designer*). Создаваемые с его помощью приложения являются «живыми» (исполняющимися) объектно-ориентированными приложениями (*Live Object-Oriented Applications*). Дизайнер *Flora/C+* можно также рассматривать в качестве контекстного графического редактора нелинейно-организованной памяти объектно-ориентированной машины *Flora/C+*.

В заключении рассмотрим некоторые особенности тестирования и отладки приложений в среде *Flora/C+*.

В первую очередь необходимо отметить выявившуюся в практике реализации больших и сложных приложений *Flora/C+* устойчивую тенденцию применять методы моделирования внешних по отношению к разрабатываемому приложению объектов автоматизации вместо использования традиционных методов и процедур тестирования



(*Modeling vs. Testing*). Модели внешних объектов создаются разработчиками с помощью средств самой же *Flora/C+*. Причем, в силу свойств разрабатываемых «в живую» приложений *Flora/C+* необходимость создания моделей объектов внешнего мира возникает уже на самых ранних стадиях «выращивания» приложения. В дальнейшем эти модели развиваются и совершенствуются по мере продвижения разработки создаваемого приложения. Таким образом, технология *Flora/C+* предлагает еще одну весьма эффективную технологическую парадигму – необходимость в процессе разработки приложения параллельной разработки моделей внешних объектов автоматизации. Эффективность такого «навязываемого» свойствами приложений *Flora/C+* технологического требования заключается в том, что разработчик вынужден вникать в суть устройства каждого внешнего объекта автоматизации, моделировать логику его функционирования, разрабатывать модули внешних интерфейсов, описывать внутреннее устройство и связи между частями объекта. Другими словами – в процессе моделирования внешних объектов разработчик детально познает внутреннее устройство внешних объектов, взаимодействие с которыми будет осуществляться на практике при использовании разрабатываемого им приложения. Следствием такого подхода является значительное улучшение качества проектирования самих приложений, сокращение времени, необходимого для отладки и тестирования разрабатываемых систем, экономия средств на организацию тестовых стендов и выполнение в некоторых случаях сложных и дорогостоящих процедур тестирования.

Кроме того, как выяснилось, приложения, реализующие функции моделей, или их составные части могут быть использованы в дальнейшем при реализации сервисных функций, связанных с основным приложением. Так, например, при реализации драйвера крайне сложных в управлении устройств автоматического банковского обслуживания в системе процессингового обслуживания *A4M FrontOffice/C+* использовалась модель реального банкомата вплоть до последних стадий тестирования. Такая модель не только позволила сэкономить массу времени разработчиков, но также после незначительных доработок стала применяться в коммерческих целях в качестве программного обеспечения устройства автоматического обслуживания. Другой пример – разработка приложений *A4M FrontOffice/C+*, связанных с *Online*-взаимодействием с внешними платежными системами (*VISA*, *EP/MC* и т.д.). Созданные в процессе разработки модели соответствующих коммуникационных сетей платежных систем были использованы в дальнейшем в приложениях, связанных с конфигурированием, мониторингом и журнализацией событий, происходящих при взаимодействии с этими платежными системами.

Что касается средств отладки приложений *Flora/C+*, то они предоставляют разработчику традиционный набор возможностей символьной отладки, в частности, таких как останов выполнения в указанной точке отлаживаемого приложения (в частности, в указанной строке программного кода объектов типа «программа»), останов по событию или исключительной ситуации, выполнение программного кода объектов типа «программа» по шагам, наблюдение за состоянием объектов в динамике (функция «*Watch*»), всплывающие подсказки с информацией о текущих значениях объектов, просмотр исходного кода объектов типа «программа» и их внутреннего представления и т.д. и т.п.

Необходимо отметить, что принятая в соответствии с общей идеологией *Flora/C+* организация хранения данных в виде объектов, располагающихся в дереве объектов, значительно упрощает сам процесс просмотра этих данных, способ доступа к ним, а, при необходимости, и модификации этих данных во время исполнения отлаживаемого приложения что называется «на лету». Более того, все встроенные редакторы библиотечных объектов показывают актуальное (т.е. динамически изменяющееся) значение объектов и их переменных, что само по себе обеспечивает легкость наблюдения за текущим состоянием объектов, входящих в отлаживаемое приложение.

Заметим также, что сам Отладчик является обычным приложением *Flora/C+* и исполняется как любое другое приложение. Его функционирование не влияет на функционирование других объектов дерева объектов, кроме тех, для которых пользователем указаны какие-либо отладочные действия.

Подведем итоги и отметим некоторые наиболее существенные следствия описанной организации вычислительного процесса объектно-ориентированной машины *Flora/C+*.

Во-первых, отказ от исходного текста программы как воплощение метода реализации и «материального носителя» приложения позволило вообще исключить этап трансляции в процессе создания прикладной системы (заметим, что к большинству объектов *Flora/C+* понятие трансляции вообще не применимо!). Включаемый разработчиком в дерево объектов новый объект тот час же становится частью разрабатываемого им приложения, начинает «жить» (исполняться). К процессу создания приложения в технологии *Flora/C+* более применимо понятие «выращивание приложения» а не традиционное «написание программы». Объединение среды разработки и исполнения приложений позволяет иметь единый набор инструментов и методов управления объектом на протяжении всей жизни приложения. Все применяемые в процессе создания приложений инструментальные средства с точки зрения *Flora/C+* являются её обычными приложения.

Во-вторых, наличие единого иерархического хранилища объектов и данных освобождает разработчика от рутинной работы по управлению данными, предоставляет исчерпывающие средства контроля и управления данными. Концептуально реализованное главенство данных над программами и прекрасные визуальные средства управления данными создают для разработчика полную иллюзию материальности объектов дерева данных. Прохождение объекта, при работе приложения, через несколько состояний, и способность выполнять некоторые «врожденные» действия позволяет говорить о наличии у объектов *Flora/C+* некоторых признаков жизни, свойственных только объектам реального мира!

В-третьих, отказ от описания данных избавляет от огромного количества ошибок, связанных с несоответствием между данными и их описанием.

В-четвертых, предложенная структура дерева объектов идеально подходит для решения задачи масштабируемости приложений. Увеличение мощности достигается простым добавлением элемента в дерево объектов. Проблема синхронизации параллельно выполняющихся процессов решается практически без вмешательства проектировщика. Пока процесс работает со своим узлом дерева объектов, он – владелец. При обращении к чужому узлу автоматически включается механизм синхронизации.





В-пятых, создание приложения производится путем сборки, настройки и связывания объектов. Все операции производятся «в живую», т.е. любое внесенное изменение тут же отражается на функционировании приложения и может быть проверено и отлажено. При таком подходе отсутствует обычное разделение проектирования на фазы написания, трансляции и отладки, что резко сокращает цикл разработки. Приложения *Flora/C+* в ходе функционирования могут создавать, уничтожать, связывать и перенастраивать объекты, из которых они состоят. Способность приложений к динамической самонастройке открывает перед проектировщиком уникальные возможности.

Практический результат применения перечисленных преимуществ *Flora/C+* состоит в:

- беспрецедентном увеличении скорости разработки прикладных систем (*Incredible Development Speed Up*), несравнимой ни с одним из широко применяемых в настоящее время на практике объектно-ориентированных инструментов программирования
- повышении качества разрабатываемых приложений (*Development Quality Improvement*)
- возможности реализации на основе предлагаемой *Flora/C+* технологии больших и сверхбольших прикладных систем (*Powerful Tools for XLarge-Scale Systems Development*)
- легкости поддержания, развития и модификации используемых приложений (*Easy Maintenance and Modification*)
- возможности создания полностью переносимых на различные платформы приложений (*Cross-Platform Application Portability*)

Опыт разработки одного из таких приложений – полнофункциональной переносимой подсистемы процессингового обслуживания *A4M FrontOffice/C+*, является наглядным и ярким подтверждением сказанного. Программное обеспечение *A4M FrontOffice/C+*, состоящее из более чем 90,000 объектов, было реализовано небольшой группой разработчиков за неполные 2 года.

## 9. Свойства технологии разработки приложений

Резюмируя описанные выше концептуальные особенности методов разработки приложений *Flora/C+* (см. раздел 8 стр. 26), приведем краткую характеристику наиболее существенных особенностей технологии разработки приложений (см. Рисунок 8, стр. 38), отличающих, с нашей точки зрения, систему программирования *Flora/C+* от других подобных систем.



## Flora/C+ Technology of Programming

### Key Properties

- “Grow Tree” Application Development Method
- Go-life in Development Stage
- Single Development and Run Time Environment
- Object Modeling
- Debugging and Executing in Object Context
- Data Structure Based Programming
- Easy Multi-Thread / Multi-Task Application Development



WWW.COMPASSPLUS.RU

E-mail:compass@compassplus.ru Tel: +7 (095)239-10-28, 230-68-43(fax), +7(3511)37-09-71, 37-05-03(fax)

**Рисунок 8. Свойства технологии разработки приложений *Flora/C+***

Основное свойство объектно-ориентированной технологии разработки приложений в среде *Flora/C+* состоит в том, что приложение не пишется (или программируется) в общепринятом понимании этого слова, а выращивается в виде иерархической структуры – дерева взаимодействующих между собой объектов (*“Grow Tree” Application Development Method*). В этом смысле идея визуального программирования в среде *Flora/C+* значительно развита в сравнении с другими традиционными средствами. Все составные части (объекты) разрабатываемого приложения, включая не только графические элементы диалоговых интерфейсов, но также – и все другие, не имеющие визуального представления элементы (в том числе – программные коды), тем не менее наглядно представляются на его экране разработчика в форме графического изображения иерархической структуры – дерева объектов и набора соответствующих графических редакторов этих объектов, входящих в состав основного средства разработки приложений *Flora/C+* – Дизайнера приложений. Разработчик использует все эти средства для просмотра, редактирования и визуальной отладки элементов (объектов) приложения.

Другим важным свойством технологической среды разработки приложений *Flora/C+* является поддержка процесса выращивания дерева объектов создаваемого приложения «в живую». На практике это означает, что разработчик, создавая приложение, практически всегда имеет дело с исполняющейся инстанцией этого приложения. Все действия по модификации дерева объектов, свойств объектов, входящих в дерево объектов, производятся над исполняющимися объектами этой инстанции. Таким образом, создаваемое разработчиком приложение одновременно находится и в стадии разработки и в стадии использования (*Go-life in Development Stage*).

Более того, технологическое, программное и операционное окружение приложения, находящегося в стадии разработки и в стадии использования, полностью совпадают (*Single Development and Run Time Environment*), что исключает какие-либо проблемы, обычно возникающие в следствии переноса приложения из среды разработки в реальную среду исполнения. Кроме того, это свойство технологии *Flora/C+* обеспечивает приложениям, разработанным в её среде, свойство полной независимости от свойств различных операционных систем и платформ, на которых эти приложения будут исполняться после завершения этапа разработки (портируемость приложений).

Технология разработки приложений «в живую» вызывает необходимость разрабатывать в процессе проектирования действующие модели внешних по отношению к приложению объектов автоматизации (*Object Modeling*) – каналов, процессов, интерфейсов, протоколов, устройств, агрегатов, контроллеров, сторонних систем автоматизации и других объектов материального мира. Такое требование технологии *Flora/C+* вынуждает разработчика более тщательно изучать свойства этих объектов в процессе моделирования, обеспечивая тем самым более высокий уровень качества всей разрабатываемой системы.

Важным свойством технологии *Flora/C+* является и тот факт, что отладка и исполнение приложений производится в объектно-ориентированном окружении (*Debugging and Executing in Object Context*). Свойства приложений и входящих в их состав объектов не теряются, остаются доступными для просмотра, анализа и модификации как на стадии создания и отладки приложения, так и на стадии исполнения находящегося в

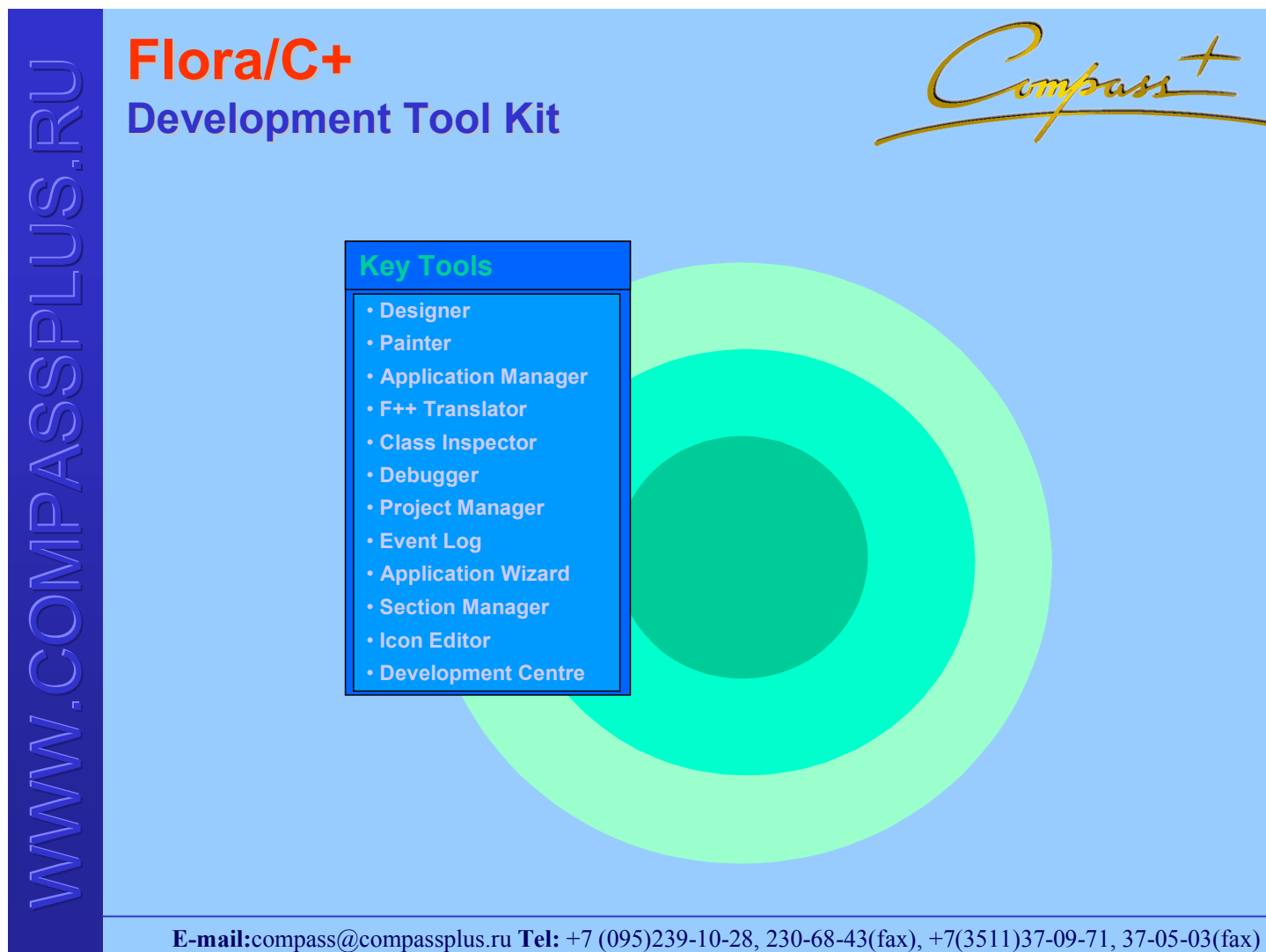
эксплуатации приложения. Тем самым обеспечивается свойство самодокументируемости приложений, сохранение и доступность свойств, взаимосвязей и конфигураций объектов, составляющих приложение, на протяжении всех фаз жизни приложений.

Фундаментальным свойством технологии *Flora/C+* является также и тот факт, что сама методология проектирования приложений в среде *Flora/C+* реализует подход, при котором в основе структуры создаваемого приложения всегда находятся данные, представляемые в виде дерева объектов (*Data Structure Based Programming*). Реально дерево объектов отражает структуру объекта реального мира, модель которого и создается в процессе создания приложения. Способ создания приложения, в основе строения которого находится структура данных (в терминологии *Flora/C+* – структура объекта моделирования), положен в фундамент технологии *Flora/C+* и одновременно по общему мнению является наиболее плодотворной идеей методологии проектирования вообще и программирования в частности.

Следующее важное свойство технологии разработки приложений *Flora/C+* состоит в концептуальной и легко реализуемой «массовой» мультизадачность создаваемых приложений (*Easy Multi-Thread / Multi-Task Application Development*). Все части объекта реального мира, моделирование которого осуществляется приложением, реально действующие автономно (асинхронно) по отношению друг к другу частям этого объекта, представляются в объектно-ориентированной модели *Flora/C+* в виде независимых (асинхронных) процессов. Например, все диалоговые элементы пользовательского интерфейса (окна, кнопки, панели, элементы меню, другие активные диалоговые элементы) представлены в дереве объектов приложения в виде независимых процессов. В этом смысле исполнение приложения в среде *Flora/C+* подобно моделированию процессов в аналоговых вычислительных машинах, которые, как известно, более адекватно отражают физические свойства моделируемых объектов реального мира, нежели цифровые последовательные вычислители архитектуры Фон-Неймана, на основе которых строятся традиционные системы программирования.

## **10. Инструменты разработки приложений**

Последняя глава описания системы программирования *Flora/C+* посвящена инструментам разработки приложений (см. Рисунок 9, стр. 41). В состав этих приложений входит ряд инструментальных средств, обеспечивающих автоматизацию и удобство разработки, отладки, проектирования, документирования, сборки и поддержки приложений *Flora/C+*.



The diagram features a light blue background with a large, stylized green and cyan circular graphic on the right. On the left, a vertical dark blue bar contains the text 'WWW.COMPASSPLUS.RU' in white. In the top right corner, the 'Compass+' logo is displayed in gold. The main title 'Flora/C+ Development Tool Kit' is positioned in the top left of the main area. A central box titled 'Key Tools' lists the following components:

- Designer
- Painter
- Application Manager
- F++ Translator
- Class Inspector
- Debugger
- Project Manager
- Event Log
- Application Wizard
- Section Manager
- Icon Editor
- Development Centre

At the bottom, a dark blue bar contains the contact information: 'E-mail:compass@compassplus.ru Tel: +7 (095)239-10-28, 230-68-43(fax), +7(3511)37-09-71, 37-05-03(fax)'.

**Рисунок 9. Инструменты разработки приложений *Flora/C+***



### **Дизайнер (*Designer*)**

Дизайнер является основным приложением *Flora/C+*, обеспечивающим минимально необходимые функции для создания приложений. Главное назначение Дизайнера – предоставление разработчику средств манипуляции с объектами дерева объектов и с самим деревом объектов. С помощью Дизайнера можно выполнять навигацию по узлам дерева объектов, фокусироваться на отдельных объектах дерева объектов, а также создавать, удалять, показывать или редактировать объекты и их свойства. Редактирование объектов и их свойств выполняется с помощью специально разработанных для каждого типа объекта редакторов, встроенных в тело Дизайнера.

Дизайнер также способен выгружать фрагменты дерева объектов в файлы специального формата (*OTS*-файлы). В его состав также входит несколько сервисных приложений, описанных в следующих главах.

### **Редактор графических объектов (*Painter*)**

Входящий в состав Дизайнера, Редактор графических объектов, предназначен для повышения удобства разработки диалоговых графических приложений и обеспечивает естественный способ выполнения этой работы, используя альбом встроенных графических элементов, мышь, приемы «drag-and-drop» и т.п. С помощью средств Редактора графических объектов также как и с помощью средств Дизайнера можно создавать, удалять, редактировать графические объекты и их свойства, в частности – цветовые свойства, формы фигур, их месторасположение на экране, относительно друг друга и в дереве объектов.

### **Менеджер приложений (*Application Manager*)**

Также входящий в состав Дизайнера Менеджер приложений предназначен также для работы объектами дерева объектов. Однако в отличие от самого Дизайнера, который в каждый конкретный момент способен работать только с объектом, на который он сфокусирован, Менеджер приложений может выполнять групповые операции, операции поиска по дереву объектов, генерировать сводные таблицы свойств и состояний объектов (в том числе – системных), таблицы перекрестных ссылок и связей, статистические справки и таблицы состояния дерева объектов, его фрагментов и частей.

### **Транслятор *F++* (*F++ Translator*)**

Транслятор кодов функций, входящих в объекты типа «программа» и написанных на встроенном языке *F++*, является частью Дизайнера и выполняет перевод исходных кодов этих функций в исполнимое представление.

### **Инспектор классов (*Class Inspector*)**

Инспектор классов является сервисным приложением, предоставляющим разработчику дополнительную информацию об объектах, их взаимосвязях, свойствах, методах, системных функциях и пр.





### **Отладчик (*Debugger*)**

Отладчик предоставляет широкий набор средств, обеспечивающих эффективную символьную отладку приложений: останов выполнения в указанной точке отлаживаемого приложения (в частности, в указанной строке программного кода объектов типа «программа»), останов по событию или исключительной ситуации, выполнение программного кода объектов типа «программа» по шагам, наблюдение за состоянием объектов в динамике (функция «*Watch*»), всплывающие подсказки с информацией о текущих значениях объектов, просмотр исходного кода объектов типа «программа» и их внутреннего представления и т.д. и т.п.

### **Менеджер проектов (*Project Manager*)**

Приложение Менеджер проектов специально разработано для поддержки реализации сложных и больших проектов, включающих множество подсистем и компонент, реализуемых несколькими отдельными разработчиками или группами разработчиков. Менеджер проектов поддерживает файлы специального формата (*PRJ*-файлы), в которых хранятся описания реализуемых проектов, компонент, из которых они состоят, в том числе – *OTS*-файлов (созданных Дизайнером), отдельных секций *OTS*-файлов (частей дерева объектов), объектов дерева объектов, агрегатов, связей, контактов, версий и т.п. С помощью Менеджера проектов можно создавать, удалять и редактировать элементы, входящие в проект, вести версии проекта и его элементов, хранить данные об авторах, аннотации и комментарии разработчиков.

Средствами Менеджера проектов на основании специальных описателей можно генерировать различные дистрибутивы разработанного приложения. Кроме того, имеется возможность создания пакетов *Upgrade* и соответствующих скриптов к ним, необходимых для обновления уже установленного у пользователей приложения, включая необходимые процедуры корректировки структуры базы данных используемой в приложении СУБД (в настоящее время поддерживается только *Oracle*).

### **Журнал событий (*Event Log*)**

Информация о целом ряде системных событий, возникающих в ядре *Flora/C+* (в частности – сведения об ошибках и прерываниях в объектах приложения), помещается в специальный Журнал событий, доступный разработчику. В этот же Журнал событий помещается информация о событиях, генерируемых многими библиотечными объектами. Записи Журнала событий доступны для поиска и просмотра. Помещение информации о том или ином событии в Журнал событий может быть заблокирована пользователем, если в приложении указан соответствующий обработчик этого события.

### **Мастер приложений (*Application Wizard*)**

Мастер приложений предназначен для генерации по указанию разработчика типового приложения, реализующего общепринятые функции, используемые в прикладных оконных интерфейсах, а также для включения в разрабатываемое приложение типовых широко используемых на практике функций, например – редактирование и просмотр файлов и элементов базы данных, работа с каталогами и файлами операционной системы и т.п.



### **Менеджер секций (*Section Manager*)**

Менеджер секций обеспечивает необходимые средства по работе с *OTS*-файлами, содержащими информацию о дереве объектов приложения, созданного с помощью Дизайнера. *OTS*-файл может содержать несколько секций, в каждой из которых находится дерево объектов приложения или его фрагменты. С помощью Менеджера секций можно исключить секцию из *OTS*-файла, скопировать или перенести какую-либо секцию в другой *OTS*-файл, вести несколько версий секций и также манипулировать этими версиями (исключать, сравнивать, освобождать место в *OTS*-файле, занятое удаленными секциями и т.п.).

### **Редактор иконок (*Icon Editor*)**

Редактор иконок предоставляет стандартные возможности растрового графического редактора файлов.

### **Центр разработки (*Development Centre*)**

Менеджер разработки организует интегрированную среду разработки приложений, использующую возможности и средства Дизайнера (*Designer*), Менеджера проектов (*Project Manager*) и Менеджера секций (*Section Manager*). Интегрированная среда Менеджера разработки позволяет пользователю работать с различными объектами приложения – деревом объектов, файлами приложений (*OTS*-файлами), секциями *OTS*-файлов (фрагментами дерева объектов), файлами проектов (*PRJ*-файлами) и т.д., в едином контексте без прямого обращения к другим инструментам разработки приложений



### 11. Заклучение

Дополнительную информацию, касающуюся программных продуктов, услуг и решений, предлагаемых *Compass Plus*, а также консультации и ответы на запросы и предложения можно получить по официальным каналам доступа к службам компании (см. Рисунок 10, стр. 46).



WWW.COMPASSPLUS.RU

Compass Plus

Compass+

Officials

**Compass Plus LTD**  
**JSC Research and Development Institute of Information Technologies**  
**Moscow (Russia):**  
Compass Plus, 2, Leninsky Pr., Moscow, 117936, Russia  
Tel: +7(095) 239-10-28, 230-68-43  
Fax: +7(095) 230-68-43  
**Magnitogorsk (Russia):**  
Compass Plus, 68, Lenin St., Magnitogorsk, 455044, Russia  
Tel: +7(3511) 37-09-71, 37-04-46  
Fax: +7(3511) 37-05-03  
**Augsburg (Germany):**  
Kirch Str. 13 , 86438 Kissing/Augsburg, Germany  
Tel: +49(8233) 847-155  
Fax: +49(8233) 847-157  
**E-mail:**  
compass@compassplus.ru  
**Http://**  
www.compassplus.ru  
www.compass.ru  
**Ftp://**  
ftp.compassplus.ru

E-mail:compass@compassplus.ru Tel: +7 (095)239-10-28, 230-68-43(fax), +7(3511)37-09-71, 37-05-03(fax)

**Рисунок 10. Официальные ссылки**