



Московский государственный технический  
университет им. Н.Э. Баумана

# Цикл лекций по дисциплине «Методология программной инженерии»

Автор: Романова Татьяна Николаевна,  
доцент кафедры «Программное обеспечение  
ЭВМ и информационные технологии»,  
кандидат физико-математических наук.

[rtn@bmstu.ru](mailto:rtn@bmstu.ru)

Россия, Москва - 2015



## *Лекция 1*

**Основные понятия  
программной инженерии.  
Принципы проектирования  
сложных программных систем**

Технология – сложный комплекс, в основе которого лежит применение различных орудий, инструментов и аппаратов, использующий наработанные человечеством знания и умения [1].

В широком смысле под *технологией программирования* будем понимать *технологию разработки программных средств*, включая в нее все процессы, начиная с момента зарождения идеи этого средства и до момента изъятия из эксплуатации. Сюда же включаются все процессы, связанные с созданием необходимой программной документации.

**Информационная технология** – система методов и способов сбора, получения, накопления, хранения, обработки, анализа и передачи информации с использованием средств ЭВМ. Применяется для повышения эффективности, защищенности и оперативности производственных процессов.

**Методология** – совокупность механизмов, применяемых при разработке программных систем и объединённых единым философским подходом.

**Метод** – концептуальное описание правил построения моделей системы, представляющих разные взгляды на проект с использованием специальных графических нотаций, которые определяют изобразительные средства и состав документации по проекту.

Главное различие между *технологией программирования* и *программной инженерией* как дисциплинами для изучения заключается в способе рассмотрения и систематизации материала.

В *технологии программирования* акцент делается на изучении процессов разработки ПС и порядке их прохождения. Методы и инструментальные средства, используемые для разработки ПС, образуют технологические процессы, которые рассматриваются в дисциплине *технология программирования*.

*Программная инженерия* изучает

различные методы и инструментальные средства разработки ПС с точки зрения достижения определенных целей.

Методы и средства, изучаемые в данной дисциплине, могут использоваться в разных технологических процессах и в разных технологиях программирования.

*В технологии программирования* методы рассматриваются

с точки зрения организации технологических процессов.

*В методологии программирования* методы

рассматриваются с точки зрения основ их построения.

*Методология программирования* определяется как

совокупность механизмов, применяемых в процессе

разработки программного обеспечения и объединенных

одним общим философским подходом.(Г. Буч. Объектно-

ориентированное проектирование с примерами применения. - М.:

Конкорд, 1992).

Например, *надежность* является неотъемлемым атрибутом ПС. Будем рассматривать *технологии программирования* как технологию разработки **надежных ПС**. Это означает:

- ❖ мы будем рассматривать все процессы разработки ПС, начиная с момента возникновения замысла ПС;
- ❖ нас будут интересовать не только вопросы построения программных конструкций, но и вопросы описания функций и принимаемых решений с точки зрения их человеческого (неформального) восприятия;

❖ в качестве продукта технологии принимается надежная (далеко не всегда правильная) ПС.

Такой взгляд на технологию программирования будет существенно влиять на организацию технологических процессов, на выбор в них методов и инструментальных средств.

# Основные требования к методикам и методам проектирования ПО

- ❖ Метод должен отражать специфику подхода (парадигму программирования).
- ❖ Метод должен быть освоен всеми участниками проекта и должен быть наглядным с точки зрения полученных результатов.
- ❖ Должны существовать формальные переходы от этапов анализа к этапу проектирования и обратно.
- ❖ Должны существовать инструментальные средства, поддерживающие все эти методы

## Понятие сложной системы

Международная организация по стандартизации (ИСО) в области науки о компьютерах и ИТ определила понятие "система" следующим образом:

**"Система - это множество элементов и отношений между ними, рассматриваемых, как единое целое".**

Таковыми элементами могут быть как материальные, так и логические объекты, а также результаты деятельности людей (например, организационные формы предприятий, математические методы и языки программирования).

# Классификация ИС

Классификации всегда относительно. Так в детерминированной системе можно найти элементы стохастических систем.

Цель любой классификации ограничить выбор подходов к отображению системы и дать рекомендации по выбору методов.

## Системы классифицируются следующим образом:

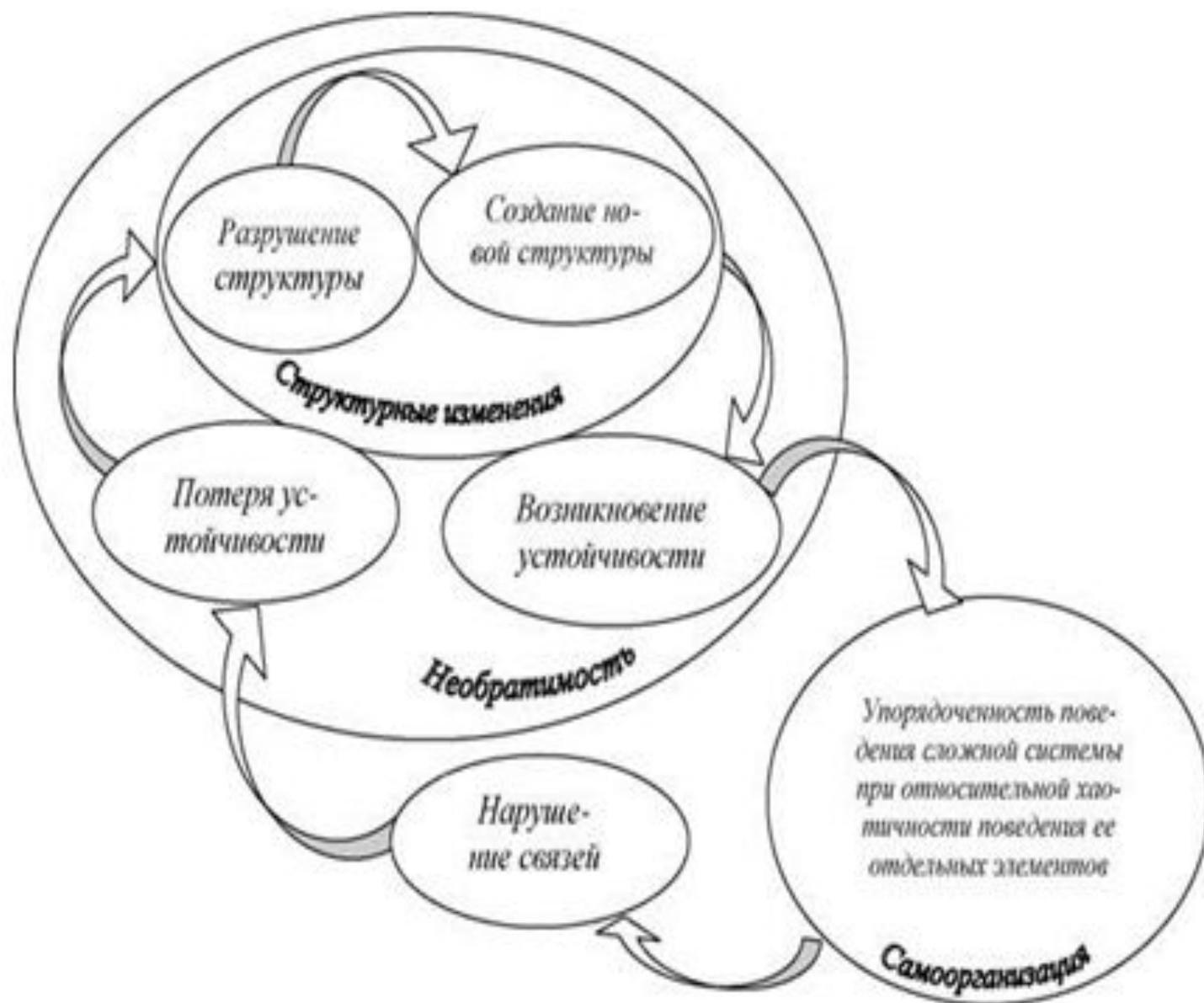
- ❖ по виду отображаемого объекта—технические, биологические и др.;
- ❖ по виду научного направления — математические, физические, химические и т. п.;
- ❖ по виду формализованного аппарата представления системы — детерминированные и стохастические;
- ❖ по типу целеустремленности-открытые и закрытые;
- ❖ по сложности структуры и поведения — простые и сложные;
- ❖ по степени организованности — хорошо организованные, плохо организованные (диффузные), самоорганизующиеся системы.

# *Классификация систем по сложности*

Г. Н. Поваров в зависимости от числа элементов,

входящих в систему, выделяет четыре класса систем:

- малые системы ( $10 \dots 10^3$  элементов),
- сложные ( $10^4 \dots 10^7$  элементов),
- ультрасложные ( $10^7 \dots 10^{30}$  элементов),
- суперсистемы ( $10^{30} \dots 10^{200}$  элементов).



## Понятие сложной информационной системы

1. Система, которая разрабатывается не одним человеком, а группой разработчиков (более 5 человек).
2. Если количество строк исходного кода исчисляется сотнями тысяч или даже миллионами.
3. Если сложную задачу можно декомпозировать на более простые задачи (которые будут реализованы более мелкими подсистемами).
4. Если в требованиях встречаются *взаимоисключающие требования*. Например, нужно обработать огромные информационные потоки, и время отклика должно быть минимальным. Следует найти компромисс.

## *Сложная программа обладает следующими свойствами:*

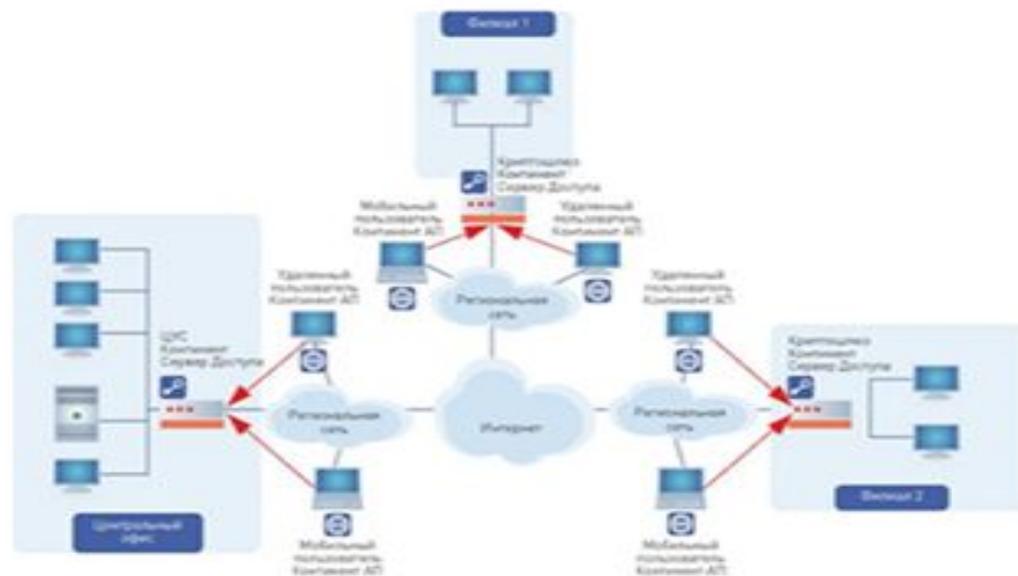
- Она решает одну или несколько связанных задач, зачастую сначала не имеющих четкой постановки, настолько важных для каких-либо лиц или организаций, что те приобретают значимые выгоды от ее использования.
- Существенно, чтобы она была удобной в использовании. В частности, она должна включать достаточно полную и понятную пользователям документацию, возможно, также специальную документацию для администраторов, а также набор документов для обучения работе с программой.
- Ее низкая производительность на реальных данных приводит к значимым потерям для пользователей.
- Ее неправильная работа наносит ощутимый ущерб пользователям и другим организациям и лицам, даже если сбои происходят не слишком часто.
- Для выполнения своих задач она должна взаимодействовать с другими программами и программно-аппаратными системами, работать на разных платформах.

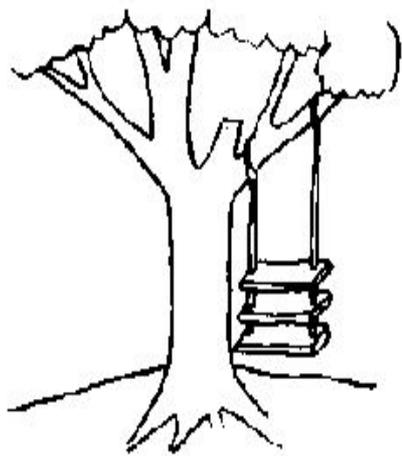
- Пользователи, работающие с ней, приобретают дополнительные выгоды от того, что программа развивается, в нее вносятся новые функции и устраняются ошибки. Необходимо наличие проектной документации, позволяющей развивать ее, возможно, вовсе не тем разработчикам, которые ее создавали, без больших затрат на обратную разработку (реинжиниринг).
- В ее разработку вовлечено значительное количество людей (более 5-ти человек). «Большую» программу практически невозможно написать с первой попытки, с небольшими усилиями и в одиночку.
- Велико количество ее возможных пользователей.

Математика делает то, что *можно*, так, как *нужно*.

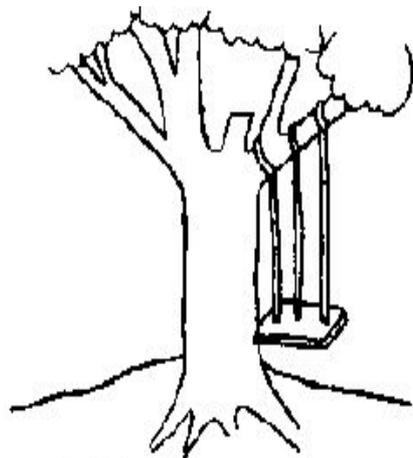
Информатика делает то, что *нужно*, так, как *можно*!

## Программистский фольклор

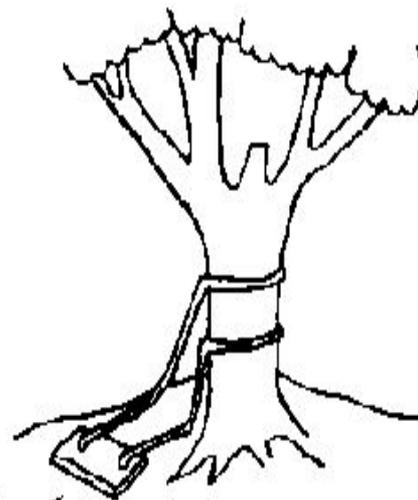




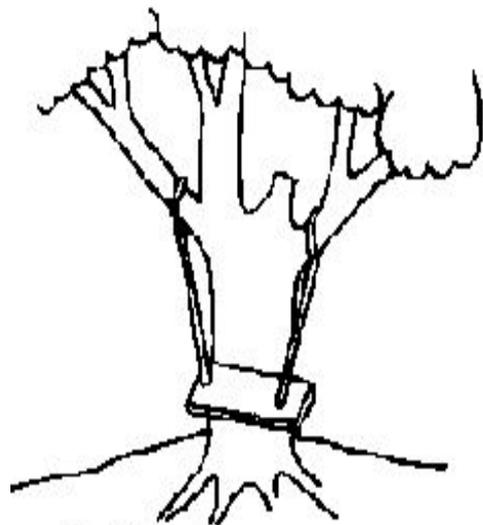
1. Как было предложено организатором разработки



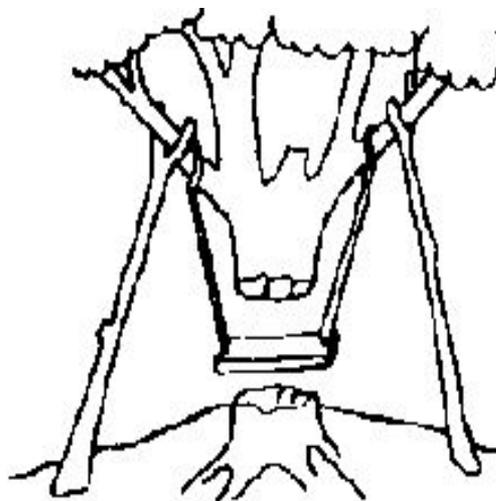
2. Как было описано в техническом задании



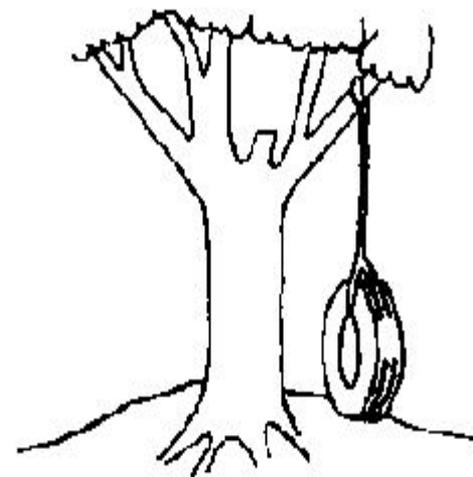
3. Как было спроектировано ведущим системным специалистом



4. Как было реализовано программистами



5. Как было внедрено



6. Чего хотел пользователь

# Общие принципы построения распределенных систем

Создание распределенных систем высокого качества является одной из наиболее сложных задач по разработке ПО. Технологии типа J2EE и .NET создаются как раз для того, чтобы сделать разработку широко встречающихся видов распределенных систем — так называемых бизнес приложений, поддерживающих решение бизнес-задач некоторой организации, — достаточно простой и доступной практически любому программисту.

Основная задача, которую пытаются решить с помощью распределенных систем — обеспечение как можно большему числу пользователей максимально простого доступа к возможно большему количеству ресурсов.

Наиболее важными свойствами такой системы являются **прозрачность, открытость, масштабируемость и безопасность.**

## **Прозрачность (transparency).**

**Прозрачностью** называется способность системы скрыть от пользователя физическое распределение ресурсов, а также аспекты их перераспределения и перемещения между различными машинами в ходе работы, репликацию (т.е. дублирование) ресурсов, трудности, возникающие при одновременной работе нескольких пользователей с одним ресурсом, ошибки при доступе к ресурсам и в работе самих ресурсов.

Степень прозрачности может быть различной, поскольку скрывать все эффекты, возникающие при работе распределенной системы, неразумно. Кроме того, прозрачность системы и ее производительность обычно находятся в обратной зависимости — например, при попытке преодолеть отказы в соединении с сервером большинство Web-браузеров пытается установить это соединение несколько раз, а для пользователя это выглядит как сильно замедленная реакция системы на его действия.

*Открытость системы (openness) определяется как полнота и ясность описания интерфейсов работы с ней и служб, которые она предоставляет через эти интерфейсы.*

Такое описание должно включать в себя все, что необходимо знать для того, чтобы пользоваться этими службами, независимо от реализации данной системы и платформы, на которой она развернута.

Открытость системы важна как для обеспечения ее переносимости, так и для облегчения использования системы и возможности построения других систем на ее основе. Распределенные системы обычно строятся с использованием служб, предоставляемых другими системами, и в то же время сами часто являются составными элементами или поставщиками служб для других систем.

Именно поэтому использование компонентных технологий при разработке практически полезного распределенного ПО неизбежно.

***Масштабируемость системы (scalability). — это зависимость изменения ее характеристик от числа ее пользователей, числа подключенных ресурсов, а также от степени географической распределенности системы.***

В число значимых характеристик при этом попадают функциональность, производительность, стоимость, трудозатраты на разработку, на внесение изменений, на сопровождение, на администрирование, удобство работы с системой.

Система хорошо *масштабируема по производительности*, если параметры задач, решаемых ей за одно и то же время, можно увеличивать достаточно быстро (лучше — линейно или еще быстрее, но это возможно не для всех задач) при возрастании количества имеющихся ресурсов, в частности, отдельных машин.

**Большую роль играет административная масштабируемость системы** — зависимость удобства работы с ней от числа административно независимых организаций, вовлеченных в ее обслуживание.

При реализации очень больших систем (поддерживающих работу тысяч и более пользователей, включающих сотни и более машин) хорошая масштабируемость может быть достигнута только с помощью децентрализации основных служб системы и управляющих ею алгоритмов.

**Вариантами такого подхода являются следующие:**

1. Децентрализация обработки запросов за счет использования нескольких машин для этого.
2. Децентрализация данных за счет использования нескольких хранилищ данных или нескольких копий одного хранилища.
3. Использование, где это возможно, *асинхронной связи* — передачи сообщений без приостановки работы до прихода ответа.

#### 4. Децентрализация алгоритмов работы за счет использования «уникальных» алгоритмов,

- не требующих полной информации о состоянии системы,
- способных продолжать работу при сбое одного или нескольких ресурсов системы,
- не предполагающих единого хода времени на всех машинах, входящих в систему.

## 5. Использование комбинированных систем организации взаимодействия, основанных на следующих схемах:

- Иерархическая организация систем, хорошо масштабирующей задачи поиска информации и ресурсов.
- *Репликация* — построение копий данных и их распределении по системе для балансировки нагрузки на разные ее элементы — и ее частном случае, *кэшировании*, организующем хранение результатов наиболее часто используемых запросов как можно ближе к клиенту.
- Взаимодействие точка-точка (peer-to-peer, P2P), обеспечивающем независимость взаимодействующих машин от других машин в системе.

## Безопасность (safety).

Так как распределенные системы вовлекают в свою работу множество пользователей, машин и географически разделенных элементов, вопросы их безопасности получают гораздо большее значение, чем при работе обычных приложений, сосредоточенных на одной физической машине. Это связано как с невозможностью надежно контролировать доступ к различным элементам такой системы, так и с доступом к ней гораздо более широкого и разнообразного по своему поведению сообщества пользователей.

## Понятие безопасности включает следующие характеристики:

- ❖ ***Сохранность и целостность данных.***  
При обеспечении групповой работы многих пользователей с одними и теми же данными нужно обеспечивать их сохранность, т.е. предотвращать исчезновение данных, введенных одним из пользователей, и в тоже время целостность, т.е. непротиворечивость, выполнение всех присущих данным ограничений.

Это непростая задача, не имеющая решения, удовлетворяющего все стороны во всех ситуациях. При одновременном изменении одного и того же элемента данных разными пользователями итоговый результат должен быть непротиворечив, и поэтому часто может совпадать только с вводом одного из них. Как будет обработана такая ситуация и возможно ли ее возникновение вообще, зависит от дополнительных требований к системе, от принятых протоколов работы, от того, какие риски — потерять данные одного из пользователей или значительно усложнить работу пользователей с системой — будут сочтены более важными.

## ❖ Защищенность данных и коммуникаций.

При работе с коммерческими системами, с системами, содержащими большие объемы персональной и бизнес-информации, с системами обслуживания пользователей государственных ведомств очень важна защищенность, как информации, постоянно хранящейся в системе, так и информации одного сеанса работы.

Для распределенных систем обеспечить защищенность гораздо сложнее, поскольку нельзя физически изолировать все элементы системы и разрешить доступ к ней только



## *Отказоустойчивость и способность к восстановлению после ошибок.*

Одним из достоинств распределенных систем является возможность построения более надежно работающей системы из не вполне надежных компонентов. Однако для того, чтобы это достоинство стало реальным, необходимо тщательное проектирование систем с тем, чтобы избежать зависимости работоспособности системы в целом от ее отдельных элементов.

Перед разработчиками систем, удовлетворяющих перечисленным свойствам, встает огромное количество проблем.

Решать их все сразу просто невозможно в силу ограниченности человеческих способностей. Чтобы хоть как-то структурировать эти проблемы, их разделяют по следующим аспектам:

- Организация связи и передачи данных между элементами системы.
- Поддержка идентификации и поиска отдельных ресурсов внутри системы.

- ❑ Организация работ в рамках процессов и потоков.
- ❑ Синхронизация параллельно выполняемых потоков работ.
- ❑ Поддержка целостности данных и непротиворечивости вносимых изменений.
- ❑ Организация отказоустойчивой работы.
- ❑ Организация защищенности данных и коммуникаций.

## Литература

1. С.И. Ожегов. Словарь русского языка. - М.: Советская энциклопедия, 1975.
2. Ф.Я. Дзержинский, И.М. Калиниченко. Дисциплина программирования. Концепция и опыт реализации методических средств программной инженерии. - М.: ЦНИИ информации и технико-экономических исследований по атомной науке и технике, 1988.
3. В. Турский. Методология программирования. - М.: Мир, 1981.
4. Г. Буч. Объектно-ориентированное проектирование с примерами применения. - М.: Конкорд, 1992.