Особенности программного обеспечения цифровых систем коммутации

- На стадии разработки системы коммутации создается **универсальная библиотека программных модулей**, которые могут потребоваться при дальнейшей эксплуатации системы **в разных областях применения**.
- Такая библиотека имеет очень большой объем, поэтому **на каждой действующей системе все это хранить нет никакого**

- Отдельный узел коммутации на действующей сети связи **имеет свои особенности** в зависимости от конкретного места на сети:
  - перечень выполняемых функций,
  - индивидуальный состав и объем оборудования,
  - взаимосвязи с другими элементами сети и др.
- Поэтому в соответствии с проектом конкретного объекта сети приходится готовить **загрузочный пакет ПО**.
- Более детальная адаптация этих программ к конкретному объекту управления происходит с помощью ***информационного обеспечения***.

Информационное обеспечение цифровой системы коммутации

Информационное обеспечение (ИО) – это набор данных, которые необходимы для работы системы управления.

I) оперативные данные о состоянии элементов системы

- состояния типа «занят/свободен»
- технические состояния

❖ ACTIVE – в работе;

❖ NON-ACTIVE – в резерве, на тестировании, в ремонте и др.

II) полупостоянные данные

- полное описание текущей конфигурации системы;
- характеристики «окружающей среды», т.е. данные по

Полупостоянные данные

1. **Станционные данные** – отображают комплектацию оборудования, структурную схему системы и параметры отдельных элементов
2. **Сетевые данные** – описание связей с другими элементами сети:
   - данные по системе нумерации;
   - характеристики всех направлений связи (какие каналы относятся к направлению, используемая система сигнализации и др.);
   - данные по маршрутизации трафика.

Размещение ПО и ИО в памяти системы управления

Организуется **многоуровневая виртуальная память**, которая включает в себя:

- оперативную память (ОЗУ);
- накопители на жестких магнитных дисках;
- архивные накопители (оптические диски, стриммеры, магнитные ленты и т.п.).

Использование виртуальной памяти облегчает поиск компромисса в условиях следующих противоречивых требований:

- большой объем памяти;
- высокое быстродействие устройств памяти;
- минимум затрат на хранение программ и данных.

В ***оперативной памяти*** **постоянно** размещаются:

- **резидентные программы**, которые всегда нужны для работы системы

(например, ядро операционной системы);

- **оперативные данные** о состоянии оборудования.

Часть ***оперативной памяти*** выделяется **динамически**:

- **нерезидентным программам**, которые загружаются сюда с жесткого диска перед их запуском;
- для хранения данных при обмене информацией с устройствами внешней памяти.

Полный пакет программ (рабочая версия ПО) хранится ***на жестких дисках.***

Он используется для загрузки в оперативную память:

- резидентных программ (при запуске системы);
- нерезидентных программ (при дальнейшей работе).

На жестких дисках поддерживается также обновляемая резервная копия рабочей версии ПО.

Это нужно на случай, если в основном пакете программ возникают серьезные ошибки, которые мешают устойчивой работе системы.

Архивные носители используются для хранения статистических данных и вспомогательных программ, которые не требуются при решении срочных

Организация информационного обеспечения

- ИО организуется в виде БД, которая в полном объема размещается на жестком диске.
- Средствами СУБД обычно поддерживается текущая БД и ее резервная копия.

- Для систем с распределенным управлением применяется архитектура ***распределенных баз данных*** (РБД).
- В этом случае отдельные фрагменты БД (таблицы или их части) размещаются **в разных модулях** системы управления
- Любой фрагмент БД может существовать в **нескольких экземплярах**

(это называют ***репликацией данных***).

- Средства СУБД ***автоматически*** поддерживают ***идентичность копий***.
- При этом существует два варианта:

- ***синхронная репликация*** – процедура обновления входит в состав каждой транзакции;
- ***асинхронная репликация*** – при транзакции обновляется только главная копия, а затем происходит рассылка обновлений для изменения других

Этапы жизненного цикла ПО

Жизненный цикл (ЖЦ) продукта – это интервал времени, который начинается при появлении замысла о создании продукта и завершается изъятием продукта из обращения.

ЖЦ ПО включает в себя следующие укрупненные фазы:

1. Анализ требований со стороны заказчика
2. Проектирование ПО
3. Реализация проекта и испытания ПО
4. Производство ПО
5. Эксплуатация (сопровождение) ПО

1. Анализ требований

На этой фазе основная задача – определить, какие функциональные возможности нужно заложить в систему, чтобы она успешно конкурировала с аналогичными системами других производителей

2.1. Структурное проектирование

а) Определить состав программ, их функции и порядок взаимодействия.

б) Предварительный выбор состава и структуры данных

в) Формулировка конкретных требований к отдельным программным компонентам

(составляются технические задания на разработку отдельных элементов ПО)

2.2. Алгоритмическое проектирование

Основная задача – в виде **строгих алгоритмов** описать все функции, выполняемые системой управления.

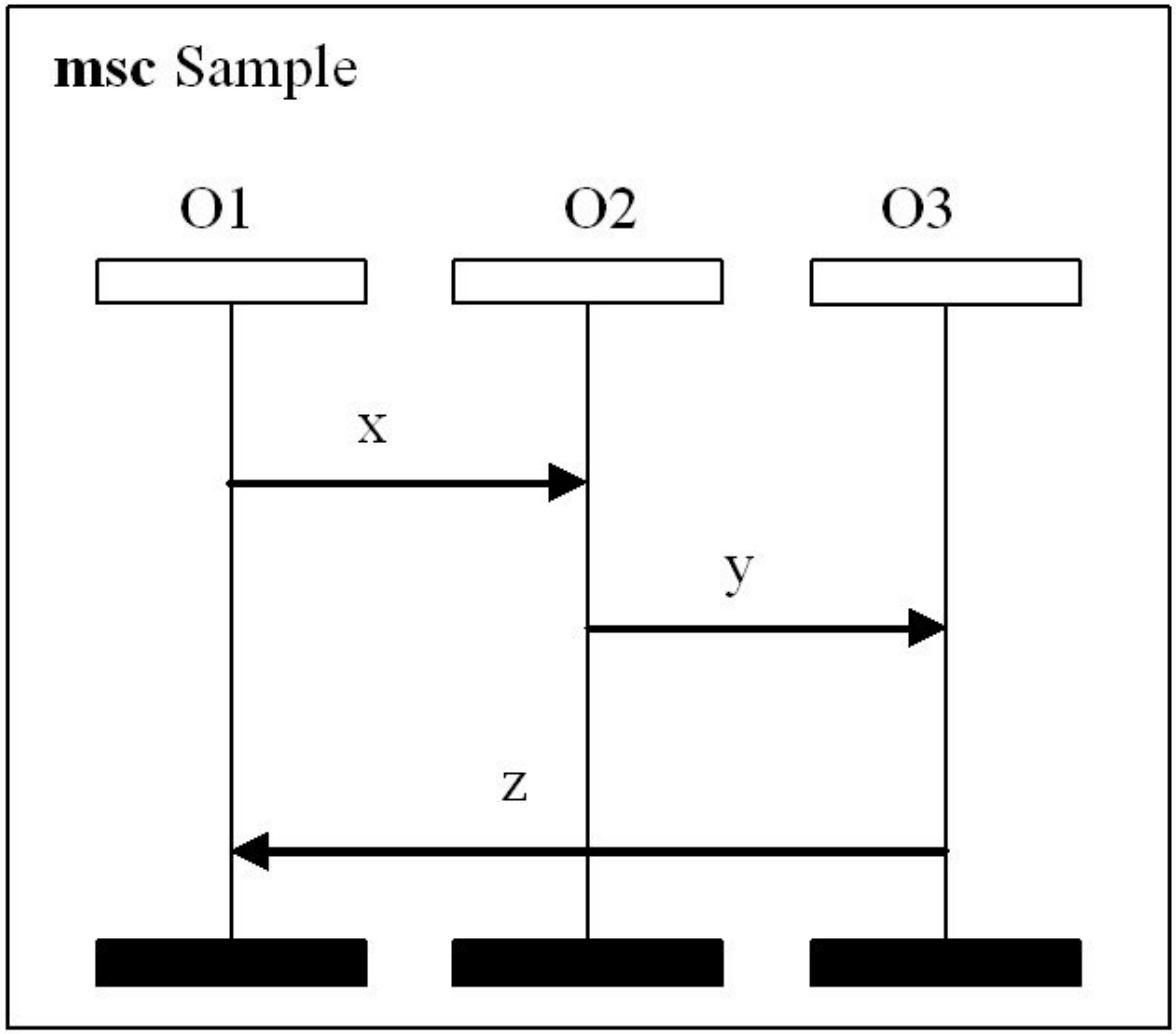
Для этой цели применяются **специализированные** языковые средства, которые позволяют описывать процессы **взаимодействия** в режиме **реального времени**.

1.Язык MSC (Message Sequence Charts)

Позволяет строить «стрелочные диаграммы» для **упрощенного** отображения процессов взаимодействия некоторых объектов.

Полное описание этого языка содержится в

Пример MSC-диаграммы



2. Язык SDL (Specification and Definition Language)

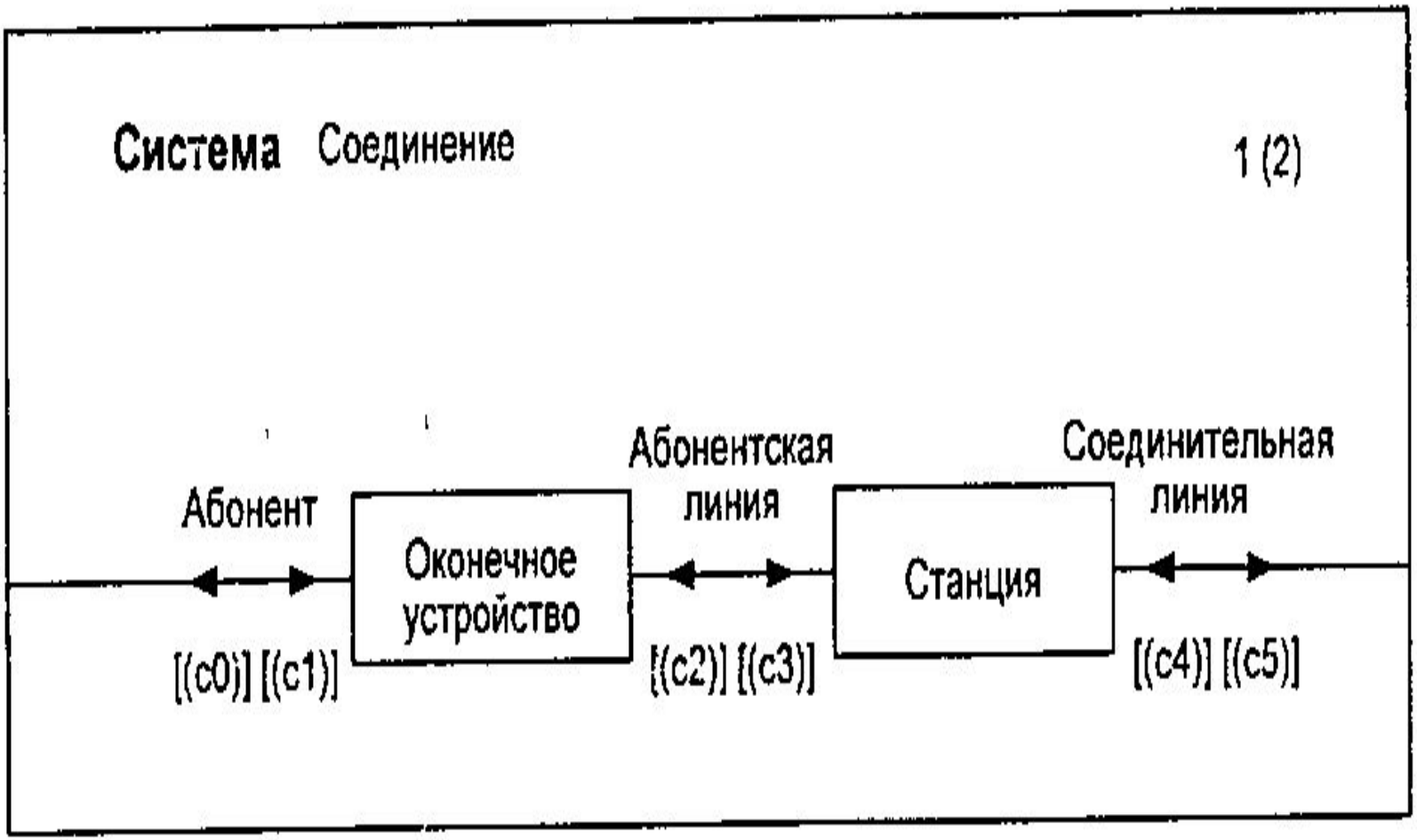
Язык SDL (**язык спецификаций и описаний**) обладает гораздо более широкими возможностями.

Позволяет строить **два типа** моделей для сложных систем реального времени:

1) **Структурная модель** (структурное описание) – отображает элементы системы и взаимосвязи между ними.
2) **Функциональная модель** (функциональное описание) – отображает поведение элементов системы в виде последовательности выполняемых действий.

Полное описание этого языка содержится в Рекомендации ITU-T **Z.100.**

Пример SDL-диаграммы для структурного описания системы



3.1. Написание программ (кодирование)

Цель этого этапа – преобразование алгоритмов в программы с использованием языков программирования.

Могут применяться языки разных уровней:

1) языки низкого уровня (типа Assembler);
2) **универсальные** языки высокого уровня (например, C++);
3) **специализированные** языки высокого уровня для применения в области телекоммуникаций:

- фирменной разработки (например, ERIPASCAL и PLEX фирмы ERICSSON);
- язык CHILL, принятый в качестве **международного**

Язык CHILL разрабатывался по заказу МККТТ (Международный Консультативный Комитет по Телефонии и Телеграфии, в настоящее время МСЭ-Т).

Полное формальное описание языка содержится в Рекомендациях МСЭ-Т серии Z.200.
Имеет много общего с ***языком Ада*** (создавался по заказу Министерства обороны США для программирования бортовых систем управления военными объектами – корабли, самолёты, танки, ракеты и т.п.).
Характерная особенность языка CHILL – наличие развитых средств описания и реализации параллельных действий в режиме реального времени.
Используется в системах коммутации фирмы Siemens (система EWSD) и фирмы ALCATEL (система S-12).