ГОСУДАРСТВЕННАЯ КОРПОРАЦИЯ ПО АТОМНОЙ ЭНЕРГИИ «РОСАТОМ»

Конечные Автоматы. Машины Состояний FSM. Краткий обзор технологии SMI.

А.В. Курякин
к.ф.м.н., в.н.с.

Саров, 04.02.2022

alexei.kuryakin@cern.ch

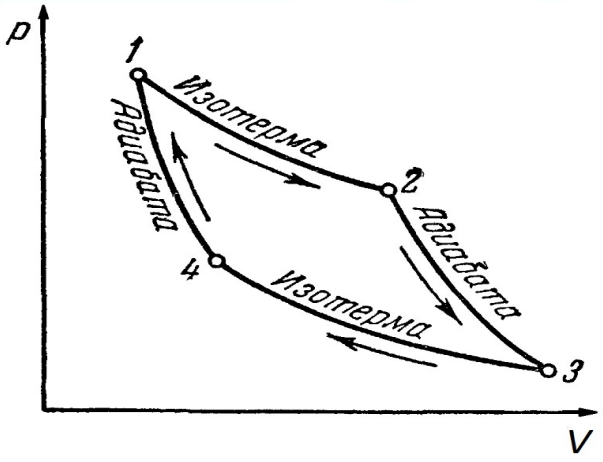
Конечные Автоматы или Машины Состояний
FSM = Finite State Machine

Конечные Автоматы – устоявшееся название логических машин, оперирующих **объектами** с **конечным** числом **состояний** – *Finite State Machine* (**FSM**), между которыми возможны переходы, именуемые **действиями**.

Следует различать **FSM** как:

- **Концепцию** (способ мыслить)
- **Описание** (как записывать/хранить)
- **Представление** (вид на бумаге/экране)
- **Реализацию** (**ПО**: языки,библиотеки,…)

FSM как физическая концепция



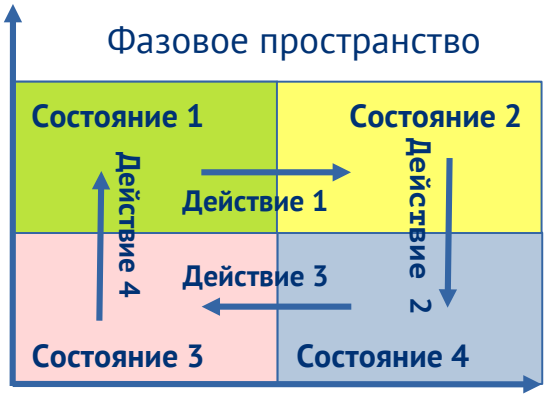
Цикл Карно на диаграмме P-V.

На примере термодинамики и статистической физики мы знаем, что физическую систему можно описать в терминах **траектории в фазовом пространстве**.

<== Например, цикл **Карно** двигателя внутреннего сгорания.

Вводя **области фазового пространства**, мы сразу переходим к понятию **Конечного Автомата – FSM**.

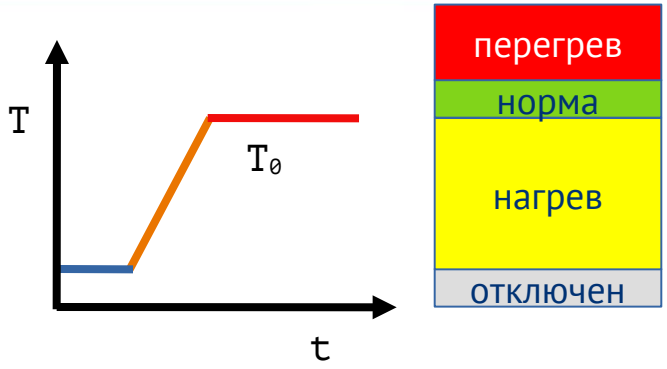
Логика работы **FSM** в этом случае - это просто **законы топологии** в фазовом пространстве.

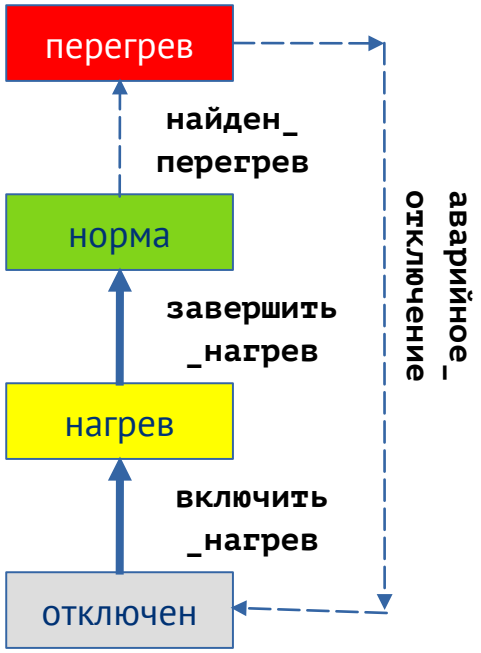


FSM – это **логическая модель** реальности. Как и всякая иная модель, она может быть **адекватной** или **неадекватной**. Формулирование адекватной модели **FSM** – сложная методическая задача.

Вопрос адекватности модели **FSM** следует четко отделять от вопросов её описания и практической реализации (интерфейса, язык описания, **ПО**).

Простой пример FSM: нагреватель





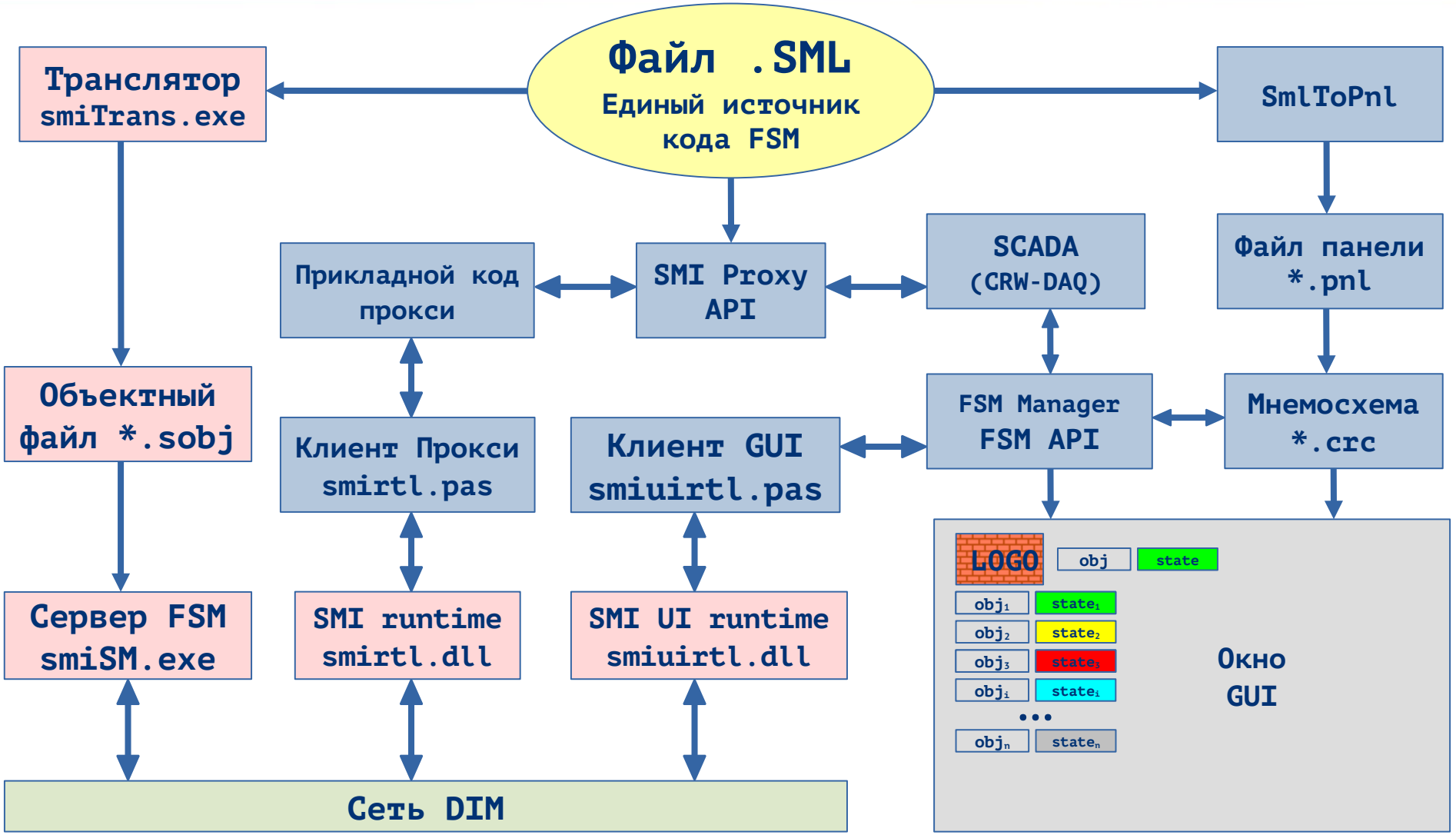
Имеется нагреватель (HEATER), который должен стабилизировать T около T_0, а также должен отключаться при перегреве выше нормы. Температура T – непрерывная величина, но мы можем попробовать смоделировать поведение системы по логике Конечного Автомата FSM – введя конечное число состояний с помощью диапазонов T.

Код описания на языке SML может выглядеть так:

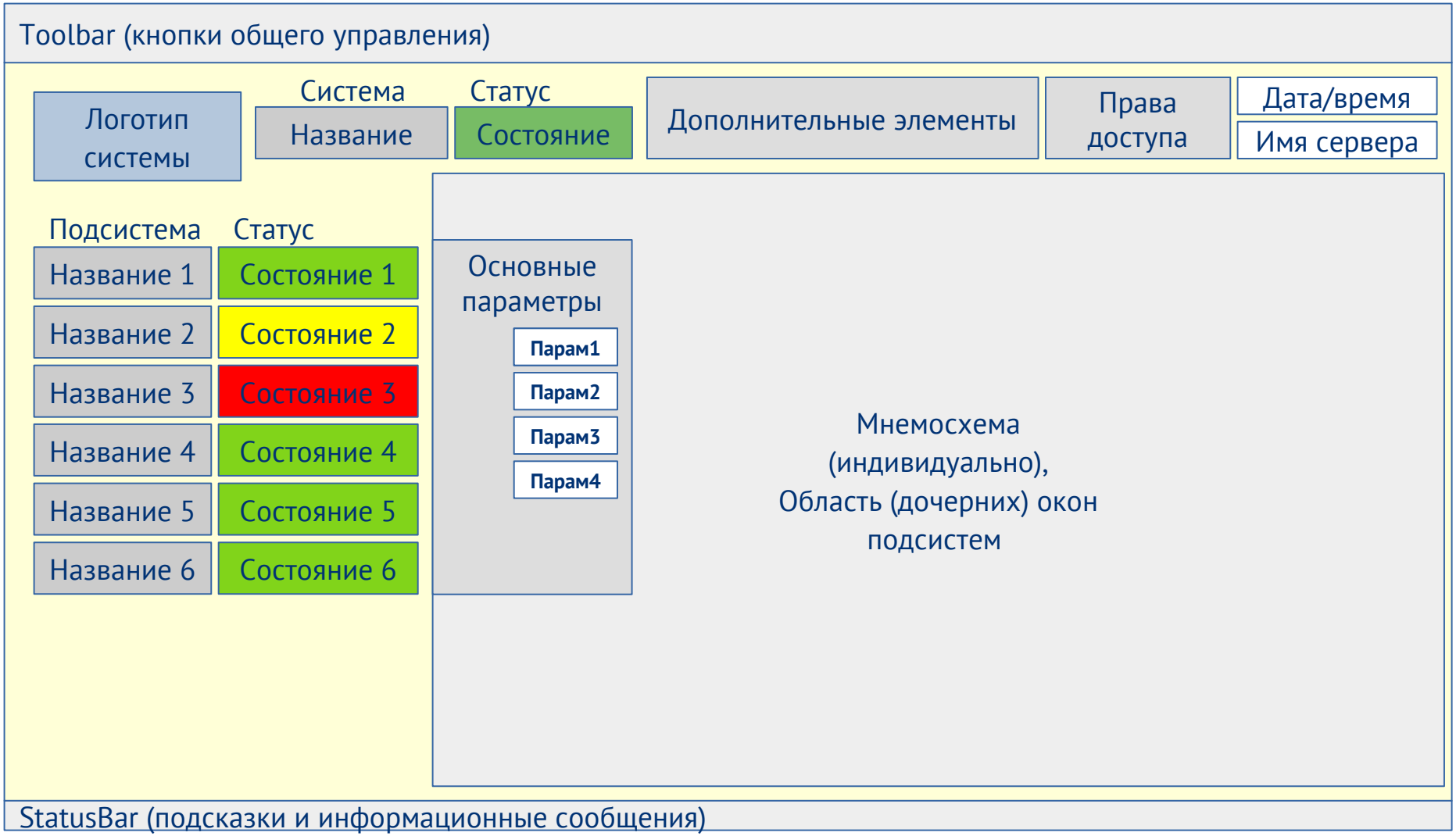
```
object: HEATER
    parameters: float T0, float dT=5.0
    state: OFF /initial                 !color: silver
        action: SWITCH_HEAT(float T0) !visible: 1
    state: HEATING                      !color: yellow
        when (T>T0-dT) do
            move_to_state NORMAL
        end
        action: END_HEAT                !visible: 0
            move_to_state NORMAL
        action: BREAK_HEAT              !visible: 1
            move_to_state OFF
    state: NORMAL                       !color: green
        when (T>T0+dT) do
            move_to_state OVERHEAT
        action: END_WORK                !visible: 1
            move_to_state OFF
    state: OVERHEAT                     !color: red
        action: ABORT                   !visible: 0
            alarm
            move_to_state OFF
```

Общая схема взаимодействия компонентов FSM

Язык SML – единый способ описания FSM



Интерфейсы пользователя на базе FSM
Общий принцип построения GUI



Основные элементы (или сущности - Entity) Терминология FSM API

Сущность	**Entity**	Любой элемент **FSM**
Менеджер	**Manager**	Корневой элемент **FSM**, главный контейнер для всех элементов
Домен	**Domain**	«Среда обитания» (подсистема, установка, …) контейнер для логических объектов
Класс	**Class**	Прототип объекта, используемый для описания свойств однотипных объектов
Объект	**Object**	Логическая модель реального объекта, основная единица **FSM**
Состояние	**State**	Поименованное состояние объекта
Действие	**Action**	Переход из одного состояния в другие
Набор объектов	**ObjectSet**	Группа объектов для совместных действий с ними
Функция	**Function**	Набор инструкций
Параметр	**Parameter**	Скалярные данные: **int**, **float**, **string**

FSM API: иерархия элементов

Элементы ***entity*** образуют дерево, его корнем ***root*** является Менеджер **FSM** ***manager***, содержащий домены ***domain***, включающие объекты ***object***, содержащие состояния ***state***, включающие действия ***action***. Домены также содержат наборы объектов ***objectset***.

Дерево наследования классов

- **TFsmEntity** — Любой элемент **FSM**
  - **TFsmParameter** — Любой параметр
    - **TFsmParamInt** — Параметр **int**
    - **TFsmParamFloat** — Параметр **float**
    - **TFsmParamString** — Параметр **string**
  - **TFsmParent** — Любой «родитель»
    - **TFsmManager** — Менеджер **FSM**
    - **TFsmDomain** — Домен **FSM**
    - **TFsmClass** — Класс (прототип)
      - **TFsmObject** — Объект **FSM**
    - **TFsmState** — Состояние объекта
    - **TFsmAction** — Действие
    - **TFsmObjectSet** — Набор объектов
    - **TFsmFunction** — Набор инструкций

Константы описания типов элементов FSM

Признак отсутствия:

- fsm_type_nil = 0

Параметры:

- fsm_type_int
- fsm_type_float
- fsm_type_string

«Родители»:

- fsm_type_manager
- fsm_type_domain
- fsm_type_class
- fsm_type_object
- fsm_type_state
- fsm_type_action
- fsm_type_objectset
- fsm_type_function

Абстрактные типы:

- fsm_type_parameter
- fsm_type_parent
- fsm_type_entity
- fsm_type_any

Дерево порождения элементов

(корневой **root** элемент - **Manager**)

- **Manager**
  - **Parameter(s)**
  - **Domain(s)**
    - **Parameter(s)**
    - **Class(es), Object(s)**
      - **Parameter(s)**
      - **State(s)**
        - **Parameter(s)**
        - **Action(s)**
          - **Parameter(s)**
    - **ObjectSet(s)**
      - **Parameter(s)**
    - **Function(s)**
      - **Parameter(s)**

FSM API — основной набор (встроенный в DaqPascal)

fsm:=**fsm_new**	Создать Менеджер **FSM**, корневой (**root**) элемент — контейнер для остальных
fsm_free(fsm)	Удалить Менеджер **FSM**, созданный
fsm_ref(ref)	Проверить/вернуть ссылку
fsm_type(ref)	Возвращает тип элемента со ссылкой **ref**, например, fsm_type_object
fsm_root(ref)	Возвращает по ссылке **ref** корневой элемент — Менеджер **FSM**
fsm_parent(ref)	Возвращает родительский элемент, которому принадлежит элемент **ref**
fsm_name(ref)	Возвращает имя элемента **ref**
fsm_path(ref)	Возвращает путь элемента **ref**
fsm_ctrl(ref,arg)	Читает/задает внутренние поля fsm_ctrl(ref,'name=value')
fsm_count(ref,typ)	Счетчик дочерних элементов элемента **ref** с типом **type**
fsm_items(ref,typ,i)	Ссылка дочернего элемента с типом **type** и индексом **i**=0…fsm_count-1
fsm_get_iparam(ref) **fsm_set_iparam**(ref,data)	Чтение (**get**) и запись (**set**) параметра целого типа fsm_type_int
fsm_get_fparam(ref) **fsm_set_fparam**(ref,data)	
fsm_get_sparam(ref) **fsm_set_sparam**(ref,data)	
fsm_add(ref,type,name) **fsm_find**(ref,type,name)	
fsm_get_state(ref)	
fsm_set_state(ref,state)	

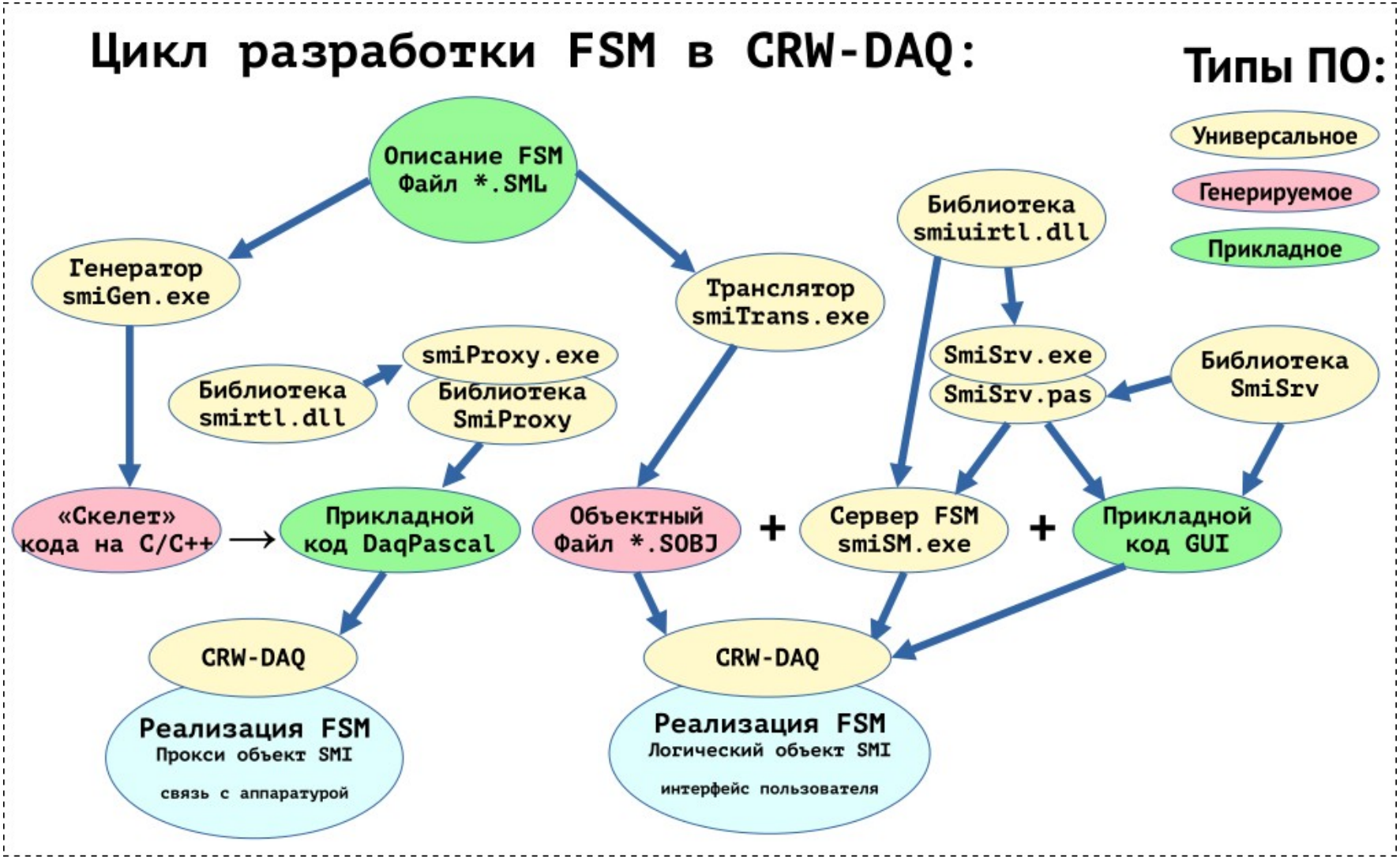
FSM API — примеры использования

```
fsm:=fsm_new;                                          // Создает FsmManager
dom:=fsm_add(fsm,fsm_type_domain,'DEMO');              // Создает элемент типа домен
par:=fsm_add(dom,fsm_type_int,'RUN_NUMBER');           // Создает параметр в домене (dom)
obj:=fsm_add(dom,fsm_type_object,'LOGGER');            // Создает объект в домене (dom)
sta:=fsm_add(obj,fsm_type_int,'RUN_NUMBER');           // Создает состояние в объекте (obj)
res:=fsm_set_iparam(par,12345);                        // Задает значение параметра (par)
if fsm_get_iparam(par)=12345 then {OK};                // Читает значение параметра (par)
typ:=fsm_type(par);                                    // Узнать тип элемента (par) => fsm_type_int
man:=fsm_root(par); ref:=fsm_parent(par);              // Узнать ссылку Менеджера man и «родителя» ref
for i:=




res:=fsm_free(fsm);                                    // Удаляет Менеджер (fsm) и все его элементы
```

Название слайда



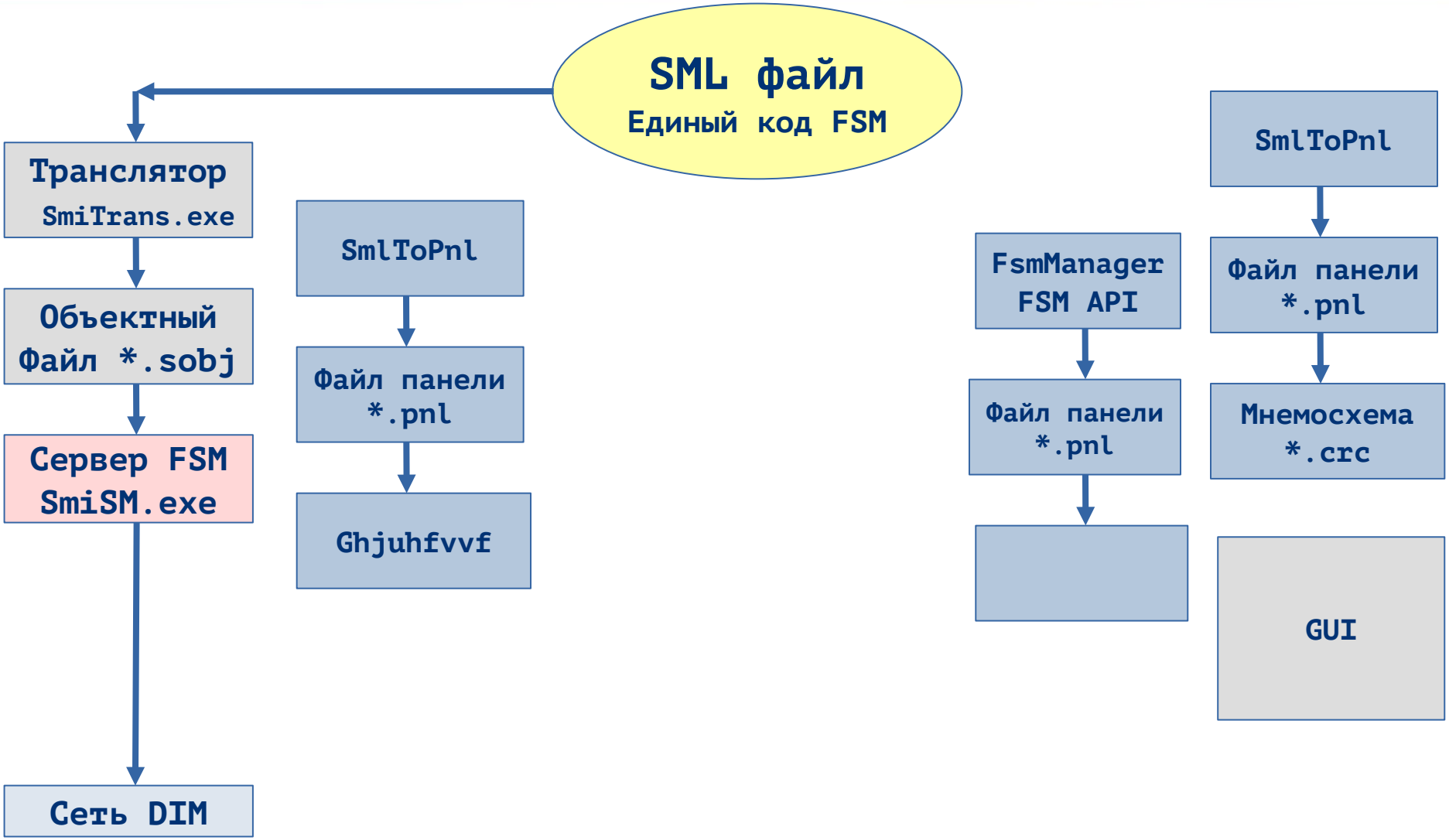
Название слайда

Название слайда

Название слайда

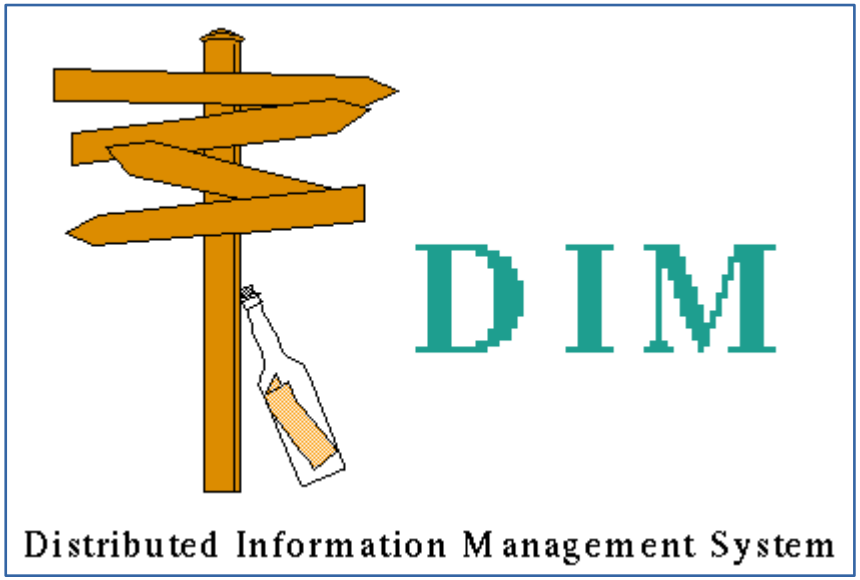
Название слайда

Язык SML – единый способ описания FSM
State Machine Language



Назначение. Происхождение. Доступность.

DIM – сетевая программная технология для создания распределенных систем управления в неоднородных (смешанных) средах, работающих на разных аппаратных и программных платформах. Она предоставляет переносимый, прозрачный для сети механизм взаимодействия между процессами, работающими как локально, так и на разных машинах сети.



DIM разработан в **CERN**. Доступен в исходных кодах под свободной лицензией **LGPL** на сайте http://dim.web.cern.ch.

Применяется с 199x в ряде крупных экспериментов **LHC**.

Многоплатформенность: **Unix**, **Linux**, **Windows**, ...

Многоязычность: **C/C++**, **Java**, **Pascal**, ...