Автоматы, модульность и программирование, управляемое данными

Дагаев Дмитрий Викторович dvdagaev@oberon.org

Парадигма автоматного программирования А.А.Шалыто

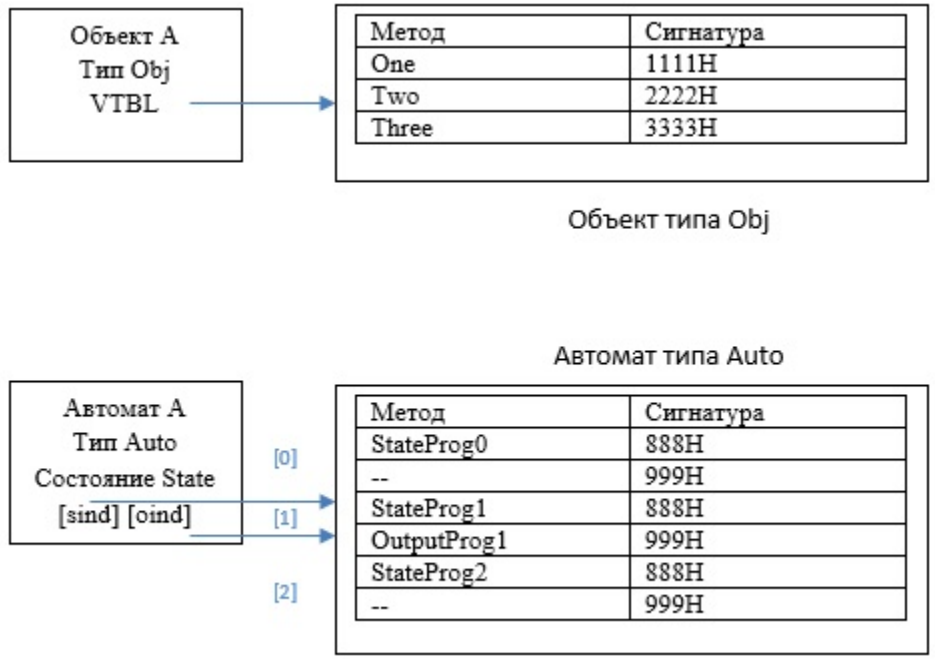
- «программы предлагается создавать так же, как производится автоматизация технологических (и не только) процессов»
- 24x7 и ответственные применения
- Возможность использования формальных методов проектирования и верификации
- Switch-технология, как удачная реализация (код справа)
- Паттерн State (отделенный от автомата, как улыбка от Чеширского кота) и классы ООП, как не слишком удачные.

```
switch( dg->y0 )
{
  case 0:
    A8( e, dg ) ;
    if( x220(dg) )                    dg->y0 = 4 ;
    else
    if( dg->y7 == 2 )                 dg->y0 = 7 ;
    else
    if( dg->y8 == 2 )                 dg->y0 = 1 ;
  break ;

  case 1:
    A4( e, dg ) ; A3( e, dg ) ; A1( e, dg ) ;
    if( dg->y4 != 0 )                 dg->y0 = 6 ;
    else
    if( dg->y7 == 2 || dg->y7 == 4 )  dg->y0 = 8 ;
    else
    if( dg->y3 != 0 )                 dg->y0 = 5 ;
    else
    if( dg->y1 == 0 )                 dg->y0 = 0 ;
    else
    if( dg->y1 == 3 )                 dg->y0 = 7 ;
    else
    if( dg->y1 == 2 )                 dg->y0 = 2 ;
  break ;
```

Отделение процесса от состояния

- Ремонт – это не процесс, а состояние
- Необходимо разделение уровней представления(о чем многократно говорил В.В.Лось): состояние (данные), процесс (алгоритмы) и привязка одного к другому (исполняющая машина).
- Вместо ООП реализуется привязка методов к состояниям на основе Liskov Substitution Principle. Принцип минимальной подстановки – соответствие набору состояния набора функций

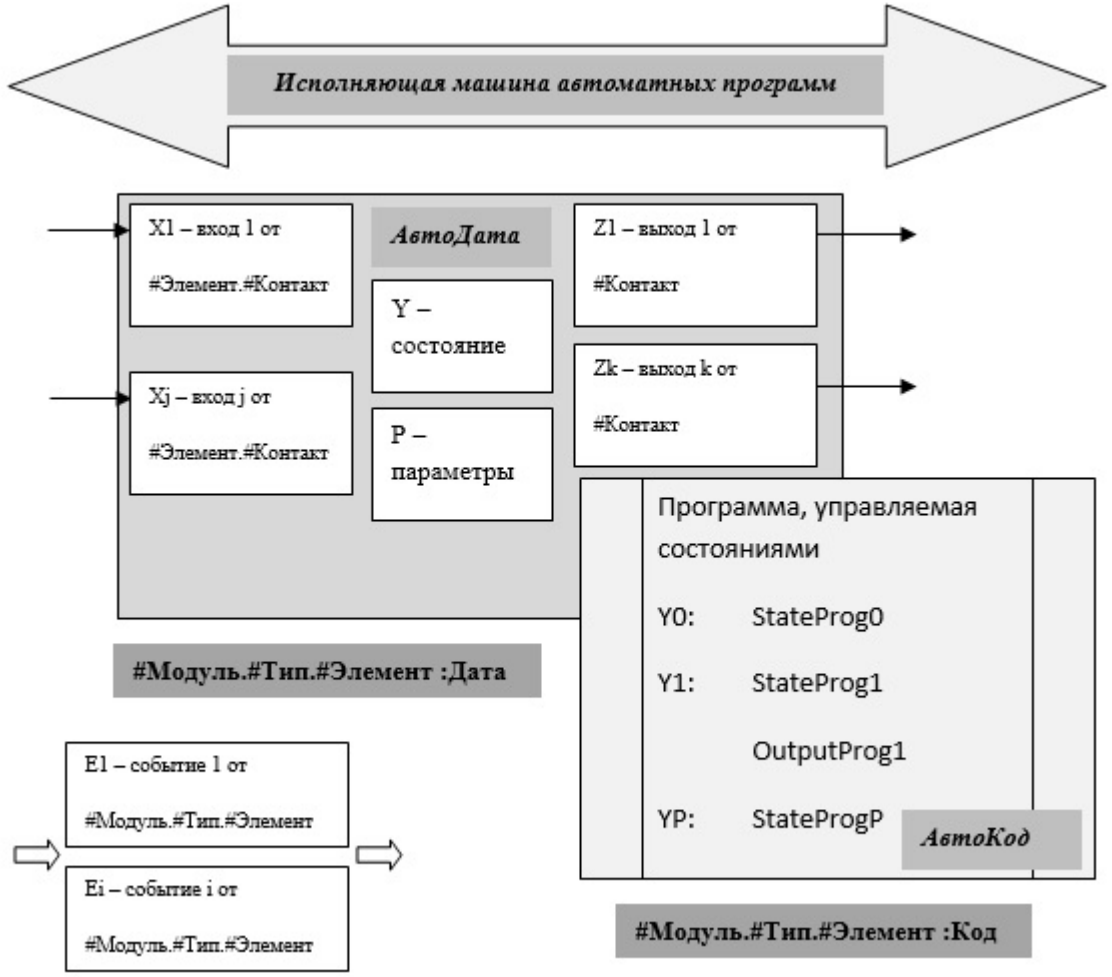


Программирование, управляемое данными DOP (Евгений Кузнецов, IBM)

- Предоставление данных и метаданных. Прямо противоположно инкапсуляции в ООП — данные о состоянии ПО предоставляются в рамках прав доступа программных компонентов
- Скрытие подробностей кода. DOP скрывает программный код и основывается на сообщениях между компонентами. Интерфейс является некритическим свойством системы.
- Сепарация кода от данных является также принципом DOP.
- Обработка данных должна осуществляться на основе метаданных реализующей платформы.

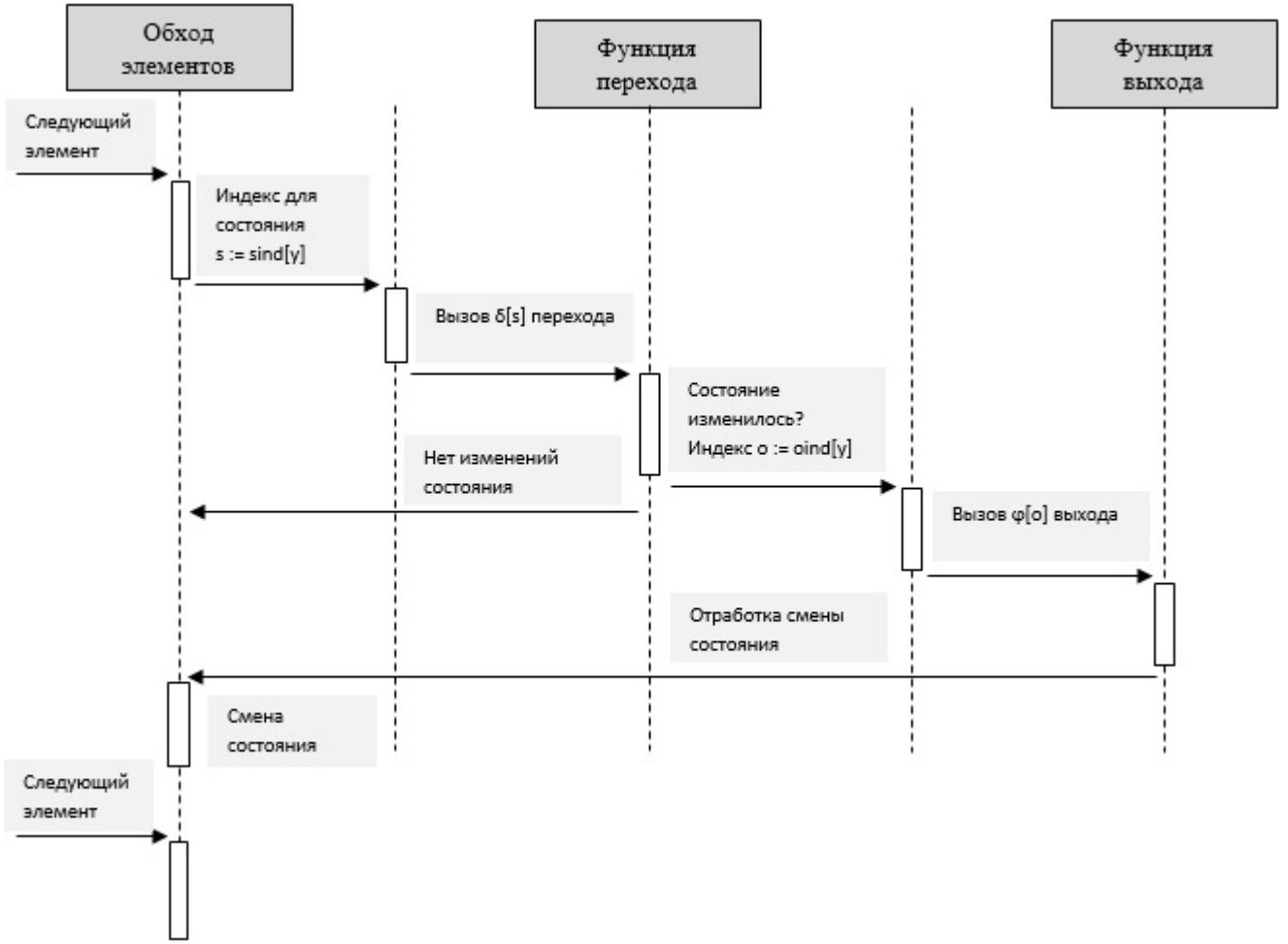
Структура автомата из модулей

- Сепарация модулей данных от модулей кода
- Отдельный модули типа, содержащие метаданные;
- Необходимость поддержки модульности в языке программирования;
- Использование средств Kernel в BlackBox.



Обход модулей средствами ИМАП

- Исполнительная машина автоматных программ нужна для обхода автоматов по дереву метаданных;
- При необходимости выполняются функции вызова, перехода;
- Состояние сохраняется в модулях АвтоДата.

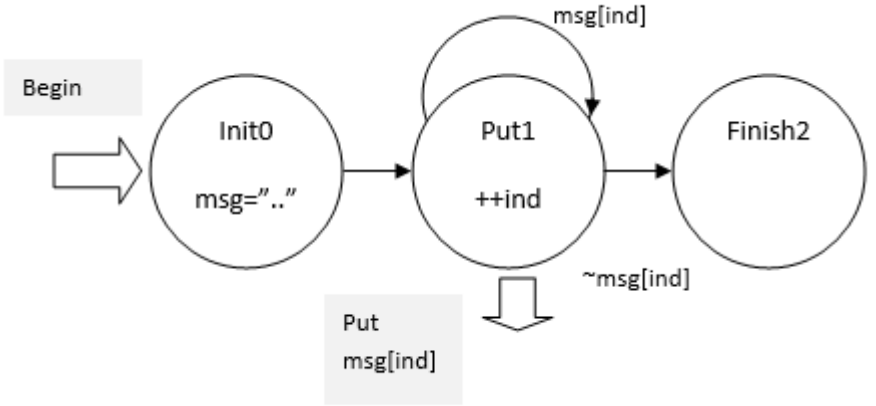


Пример с побуквенной печатью

Объект АвтоДата, Состояния и выходы 0, 1, 2

```
MODULE AbpeObxHelloData;
IMPORT A := AbpeAuto;
TYPE
        Hello* = RECORD (A.Data)
                msg*: ARRAY A.N_STR OF CHAR;
                ind*: INTEGER;
        END;

        HelloStates_0_1* = RECORD (A.OnlyStates) END;
        HelloOutputs_1_2* = RECORD (A.OnlyOutputs) END;
END AbpeObxHelloData.
```



Реализация АвтоКода

```
(* ST_RUN = 1 *)
PROCEDURE Put1* (VAR t: D.Hello; IN e: A.Event; VAR y: A.State);
        VAR m2: ARRAY 2 OF CHAR;
BEGIN
        WITH e: A.End DO
                y := 2;         (* END *)
        ELSE
                IF t.msg[t.ind] # 0X THEN
                        m2[0] := t.msg[t.ind];
                        m2[1] := 0X;
                        t.SendSimpleMsg(e.from, m2);
                        INC(t.ind);
                ELSE
                        t.SendSimpleMsg(e.from, "\n");
                        y := 2;         (* END *)
                END;
        END
END Put1;
```

```
(* ST_RUN = 1 *)
PROCEDURE Started1* (VAR t: D.Hello; y: A.State);
BEGIN
        IF A.traceLevel >= A.TRACE_LEVEL_NORMAL THEN
        t.SendParamMsg(T.sysRef, "%r started\n",
        t.ref.rec.typfp, t.ref.rec.ind, 0, 0)
        END
END Started1;
(* ST_END = 2 *)
PROCEDURE Finished2* (VAR t: D.Hello; y: A.State);
BEGIN
        IF A.traceLevel >= A.TRACE_LEVEL_NORMAL THEN
        t.SendParamMsg(T.sysRef, "%r finished\n",
        t.ref.rec.typfp, t.ref.rec.ind, 0, 0)
        END
END Finished2;
```

+1. Реализация сборочного программирования для автоматов

Схема, как программный модуль

```
MODULE AbpeObxCounterSchema;
VAR
  theCounter-: AbpeObxCounterData.Counter;
  theAverage-: AbpeObxCounterData.Average;
BEGIN
  (* setting static parameters *)
  theAverage.minnumber := 3;
  theAverage.maxnumber := 10;
  theAverage.count.Link(theCounter.count);
  (* Connecting out->in theCounter.count-
        >theAverage.count *)
END AbpeObxCounterSchema.
```

Схема как данные

```
V AbpeObxCounterData.Counter theCounter
V AbpeObxCounterData.Average theAverage
S theAverage.minnumber 5
S theAverage.maxnumber 16
L theAverage.count theCounter.count
```

+2. Лучше, чем 155ЛА3 – горячая замена

- Спасибо Б.В.Рюмшину за поставленный вопрос.
- Микросхема: каждый автомат – микросхема или несколько. Заменил – работает.
- Модуль: каждый автомат – модуль данных и модуль программы. Заменил – работает.
- В отличие от 155ЛА3 состояние сохраняется, даже при отключении питания.

+3. Автоматный подход подтверждает концепцию семантических ограничений

Чистое автоматное программирование можно обеспечить ограничениями RESTRICT

- Запрет динамической памяти;
- Запрет рекурсии;
- Запрет циклов с переменным числом итераций;
- В модулях не нужны указатели (только в исполняющей машине).

Использование близких подходов

- Модель Акторов;
- Слабо-связанные коммуникационные системы DDS;
- Использование МикроСервисов.

Преимущества Оберон-подходов

- Полная реализация модульности, без которой DOP не будет эффективно работать;
- Удобное применение метаданных.