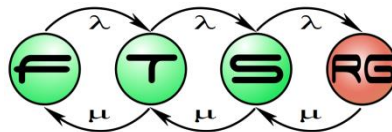


# Kódgenerálás

**Budapesti Műszaki és Gazdaságtudományi Egyetem**  
**Hibatűrő Rendszerek Kutatócsoport**

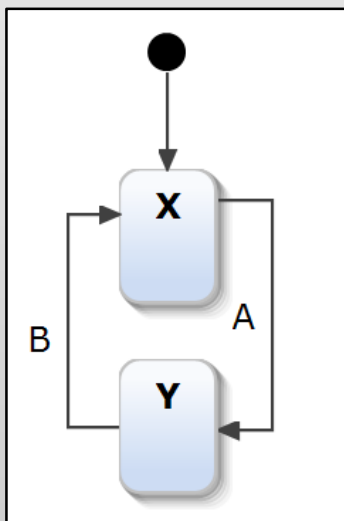


# MODELLEZŐ ESZKÖZ FUNKCIÓI

# Yakindu modellezési funkciói

Konkrét szintaxis  
(Szerkesztők)

Grafikus



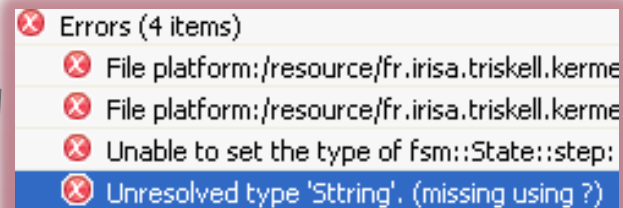
Szöveges

Demo  
**internal:**  
**event A**  
**event B**

Szintaxis → Szemantika

Modellező funkciók

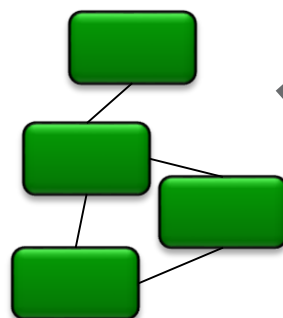
Ellenőrzés



Kódgenerálás

```
</membership>  
<profile defaultProvider="Sitefinity">  
<providers>  
<clear/>  
<add name="Sitefinity" connectionS<br/></providers>  
<properties>  
<add name="FirstName"/>  
<add name="LastName"/>  
<!-- SNP specific properties -->  
<add name="NickName" />  
<add name="Gender" />
```

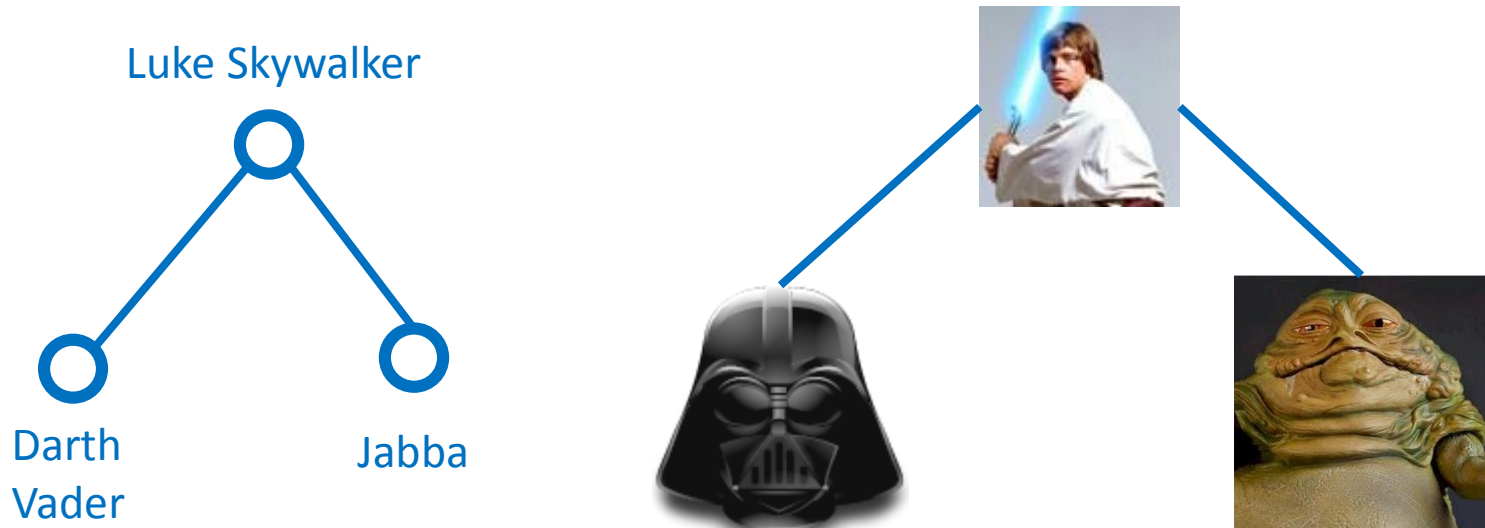
(Forráskód, dokumentáció,  
konfiguráció)



Modell  
(Absztrakt szintaxis)

# Absztrakt szintaxis

- **Definíció:** a szerkesztés alatt álló rendszer strukturális modellje.
- Modellező program kezeli
- Emlékeztető: strukturális modell = **gráf**
  - **csomópontok, élek és tulajdonságok gráfja**

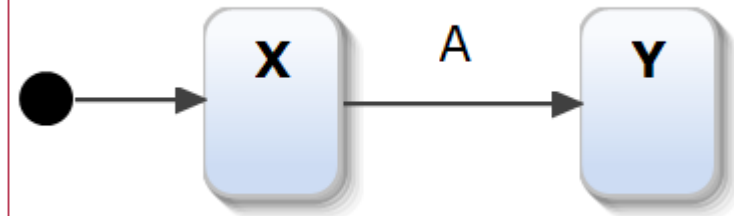


# Absztrakt szintaxis példa: Yakindu

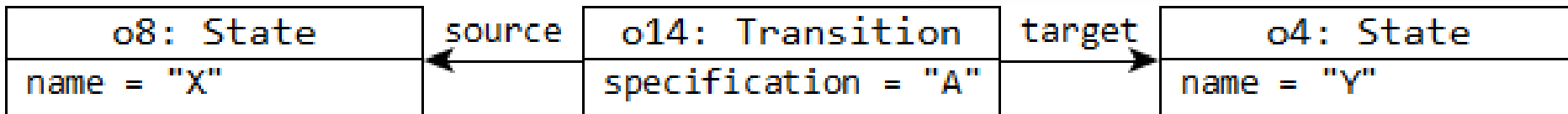
## Kérdés:

Hogyan készítenénk modellező programot?

## Példa: Yakindu modell



## Abstract Syntax

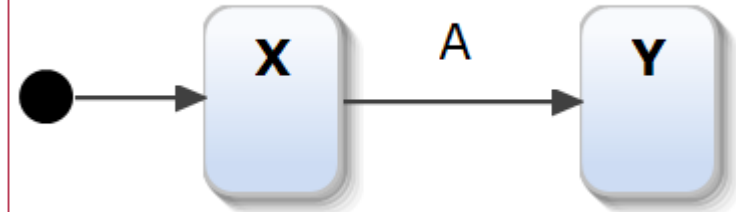


# Absztrakt szintaxis példa: Yakindu

**Kérdés:**

Hogyan készítenénk modellező programot?

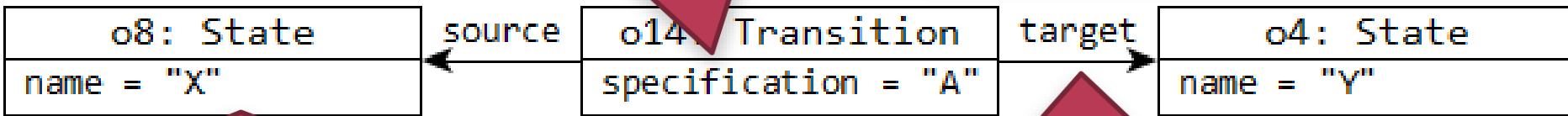
Példa: Yakindu modell



Neveket String-ként tároljuk

```
name = "A"
```

Abstract Syntax



Modellelemeket objektumként

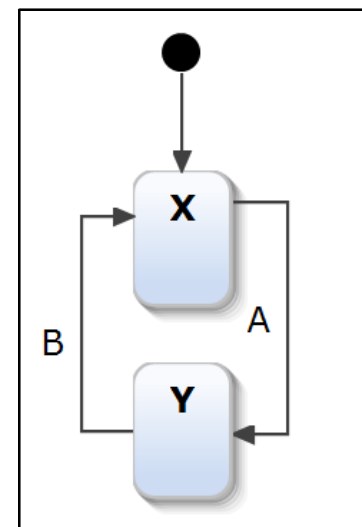
Kapcsolatokat pointerekkel

**Válasz:** Egyszerű objektum-orientált program

# Konkrét szintaxis

- **Cél:** konkrét megjelenítés  $\Leftrightarrow$  mögöttes modell
- Szöveges szintaxis (programozási nyelv)
  - Feladat: Szöveg  $\rightarrow$  Modell
  - Szabályok alapján (egyébként nehéz!)  
**<Digit> ::= "1" | "2" | "3" | "4" | ... | "9" | "0"**  
**<Number> ::= <Digit> | <Digit> <Number>**
- Grafikus szintaxis (Diagram)
  - Könnyebben átlátható, nehezebben írható
  - **Itt is:** vannak saját technológiák

Demo  
**internal:**  
**event A**  
**event B**



**Megfelelő módszerekkel mindenki tud saját programozási nyelvet csinálni!**

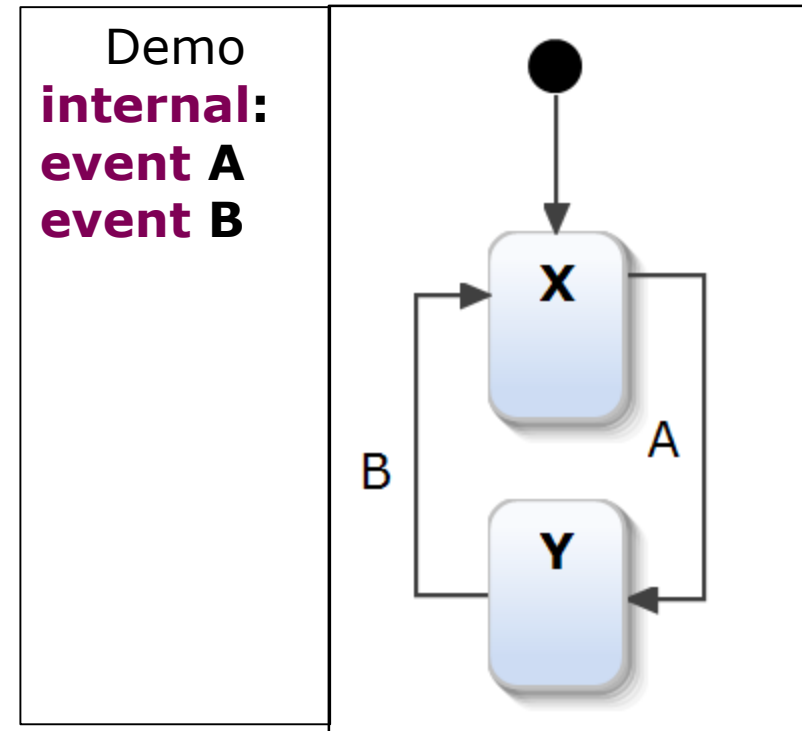




# KÓDGENERÁTOR

# Kódgenerátor példán keresztül:

- **Feldadat:**  
Generáljunk C kódot  
Yakindu állapotgépekből
- Írjunk olyan függvényt:  
→ kap egy Modell objektumot  
← visszaad egy szöveget
- A szöveg egy „Demo.c” fájlba kerül
- Fordító lefordítja



# Sablon alapú kódgenerátor (Xtend)

- Cél: Állapotok → Enum

Összevágunk egy char\*-ba a kimenetet,  
%s helyére írjuk az X,Y neveket

- Megoldás: C program

```
printf(result,  
"enum states {\n\tState%s,\n\tState%s\n};",  
state1->name,  
state2->name);
```

```
enum states {  
    StateX,  
    StateY  
};
```

Ez egyből működik!  
☹ nehezen átlátható.

- Sablon (Xtend):

```
'''  
enum states {  
    State«state1.name»,  
    State«state2.name»  
}'''
```

Megadjuk a változások helyét

Megírjuk a sablonosan

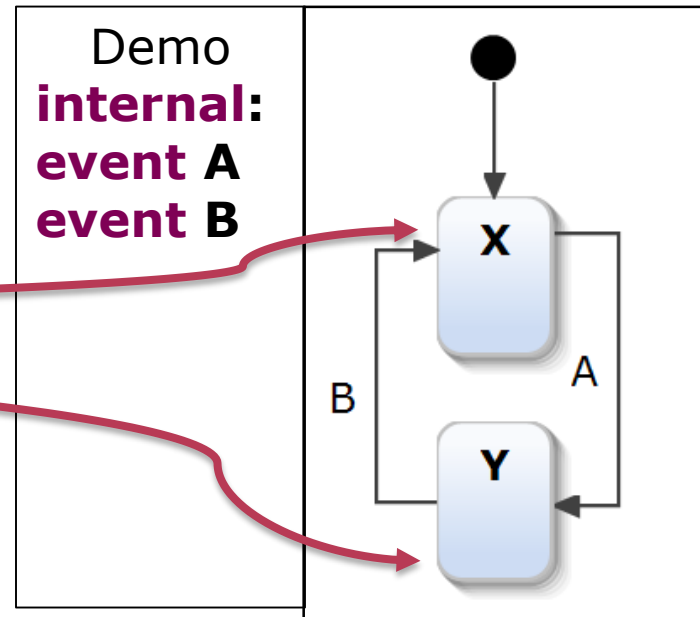
😊 Könnyebb írni  
😊 Átláthatóbb  
😊 Könnyű módosítani  
☹ +1 technológia

# Kódgenerátor példa: Állapotok

## ■ Várt C kód:

```
//States of the statemachine  
enum states {  
    StateX,  
    StateY  
};
```

Lehetséges állapotok:  
Enumként felsorolva



## ■ Sablon:

```
//States of the statemachine  
enum states {  
«FOR state : states»  
    State«state.name»  
«ENDFOR»  
};
```

- 1.Összes állapoton végigmegyünk
- 2.Vesszővel elválasztva írjuk ki:

**State**«név»

Pl: **StateX**

# Kódgenerátor példa: kezdőállapot

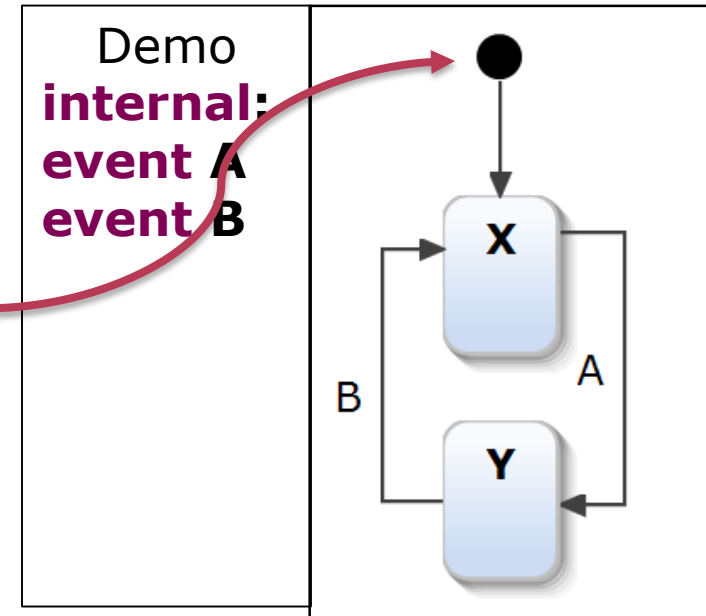
## ■ Várt C kód:

```
// The actual state  
// First = entry state.  
enum states actualState = StateX
```

Aktuális állapot = kezdőállapot

## ■ Sablon:

```
// The actual state  
// First = entry state.  
enum states actualState = State«findEntry(states).name»
```



1. Megkeressük a kezdőelemet
2. Kiírjuk a nevét

# Kódgenerátor példa: Állapotátmenet

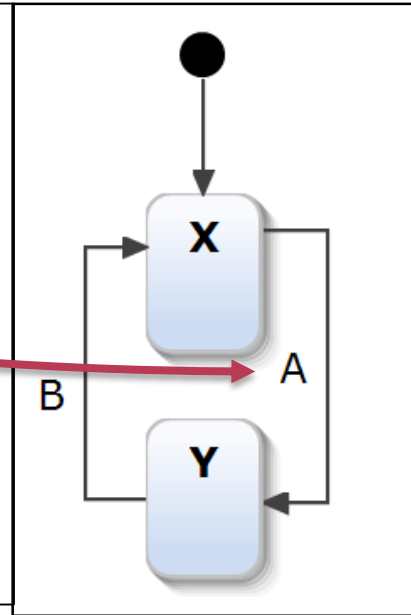
## ■ Várt C kód:

```
// Execute "A" event  
void doA{  
  switch(actualState) {  
    case StateX:  
      actualState = StateY;  
      break;  
    case StateY:  
      break;  
  }  
}
```

A / X → Y

A / -

Demo  
**internal:**  
event A  
event B



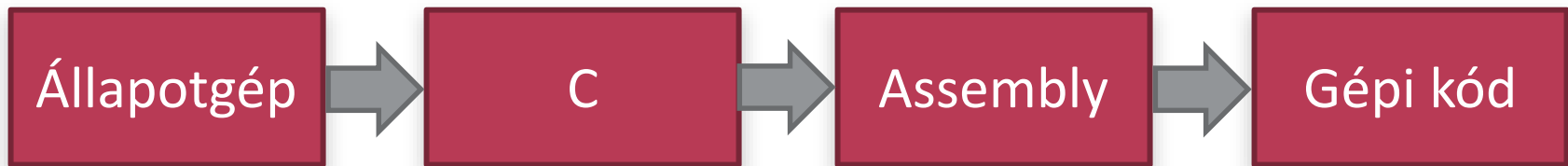
## ■ Sablon (vázlatosan):

1. Minden eseményhez egy `do«Esemény neve»` függvény
2. A tranzíciók alapján mást csinál a függvény

Egy (egyszerű) állapotgéphez ennyi a kódgenerátor!

# Kódgenerátor Összefoglalás

- Kódgenerálás = Fordító
- Ugyanaz a lépés:



- A problémát a saját nyelven: Produktivitás ++
- Sok unalmas, bonyolult kód automatikusan Teljesítmény ++
- Ellenőrizzük a saját nyelven: Megbízhatóság ++
- Tanszékünkön fejlesztett projektek: akár **95%** generált kód

# PÉLDA ECLIPSE ALAPÚ MODELLEZŐ ESZKÖZÖK



# XMind: Mindmap tervező

The screenshot displays the XMind 2007 application window. The main workspace shows a mind map with a central node 'Sitemap' and several branches:

- Solutions**: Individual (Presentation, Decision Maker, Writing Helper, Time Management), Education (Innovation, Teaching, Key Notes), Teamwork (Brainstorming).
- Products**: Overview, Screenshots, Tech Notes, Downloads, Purchase.
- Features**: A list of 11 features, with a dashed box highlighting items 8 through 11: 8. Filter and delamination, 9. Powerful workbook and assoc..., 10. Seamless integration with of..., 11. Import other mindmaps to sa...
- Support**: FAQ, Install, Forum.
- About**: Pictures, Map, Privacy.
- Analytics**: Pageviews, Visits, P/V.
- Next Version**: Solutions, Tech Notes, Downloads, Cases.
- Four Special Features**: A separate node with a smiley icon.

The right sidebar contains an 'Outline' panel showing a hierarchical tree of the map's nodes, and a 'Markers' panel with various icons for filtering and marking.

At the bottom, the 'Templates' panel shows several pre-defined mind map styles: Default Template, XMIND Classic, XMIND Simple, XMIND Business, XMIND Academese, and XMIND Comic.

# Bonita: Üzleti folyamatok modellezése

**Bonita Studio**

Process Edit Run View Extensions Look'n'feels Data types Connectors Contexts Forms BAM Simulation Repository Help

New Open Save Print Import Export Copy Paste Run Debug User XP Application Developer User guidance Preferences Help Welcome

Fibra\_Spenta\_Richiasta (1.1) x

The diagram illustrates a business process for 'Fibra Spenta Richiesta'. It starts with a start event leading to 'Inserimento Richiesta', followed by 'Mail Richiesta Studio Fattibilità'. A decision point 'Gate1' branches based on 'Fattibilità KO' (leading to 'Chiusura Fattibilità KO' and 'Fine Richiesta NON Fattibile') and 'Fattibilità OK' (leading to 'Produzione Offerta Commerciale' and 'Attesa Risposta Cliente'). A second decision point 'Gate3' branches based on 'Cliente Rifiuta' (leading to 'Chiusura Rifiuto Cliente') and 'Cliente Accettato' (leading to 'Richiesta Fattibilità Fibra', 'Mail Studio Fattibilità Completato', and back to 'Gate1').

Overview

General x Application Appearance Simulation

**Fibra Spenta Richiesta**

Pool	Name	Fibra Spenta Richiesta (Fibra_Spenta_Richiasta)	Edit...
	Version	1.1	

# Kalypso: Vízügyi tervezés + szimuláció

The screenshot displays the Kalypso software interface for water management simulation. The main window shows a map with a red-shaded catchment area and a network of blue and orange lines representing water flow paths. The interface is divided into several panels:

- Navigator:** Shows the current style 'Subcatchments' and various settings like 'Regel', 'Titel', 'MinDenom', 'MaxDenom', 'Symbolizer', and 'Legende'.
- Tree View:** Displays the hierarchical structure of the model, including 'CatchmentCollection1' and its members like 'Catchment100', 'Catchment101', 'Catchment102', etc.
- Map View:** Shows the spatial distribution of the catchments and flow paths on a street map. Catchments are labeled with numbers like 100, 101, 102, 103, 104, 105, 106.
- Data Table:** A table titled '\*Tabelle\_Teilgebiete.gtt' showing parameters for different catchments.

TG-Nummer (in...)	Versiegelu...	Anstie...	TG-Flaeche (fla...	Faktor ...	Anfangsinhalt...
100	0.188		1123080		
101	0.0080				
102					
				2.0	
				1.5	

<http://www.kalypso-simulation-platform.org>