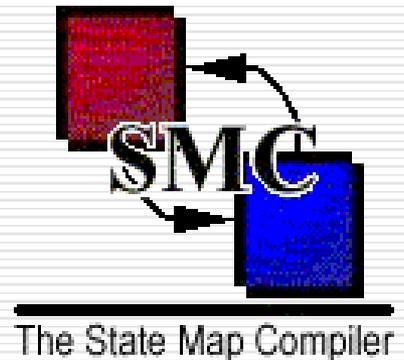
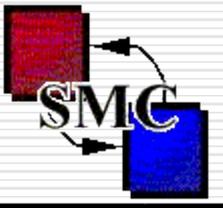


SMC

The State Machine Compiler (40')

François Perrad
francois.perrad@gadz.org

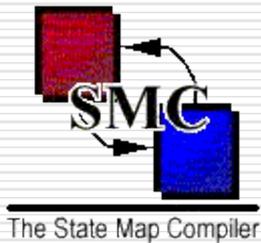




The State Map Compiler

The State Machine Compiler

- Introduction
- Basic concepts
- Advanced concepts
- More features
- A case study : a Telephone
- Conclusion

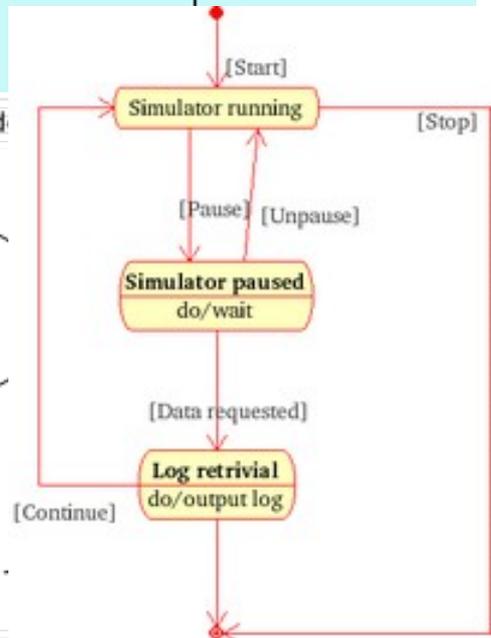
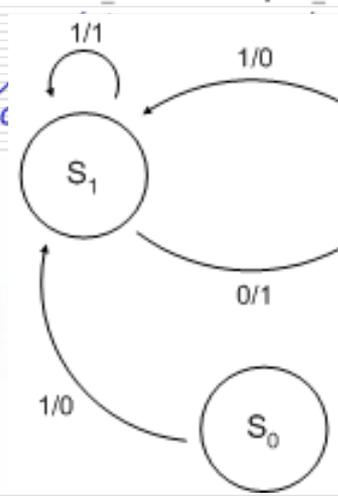
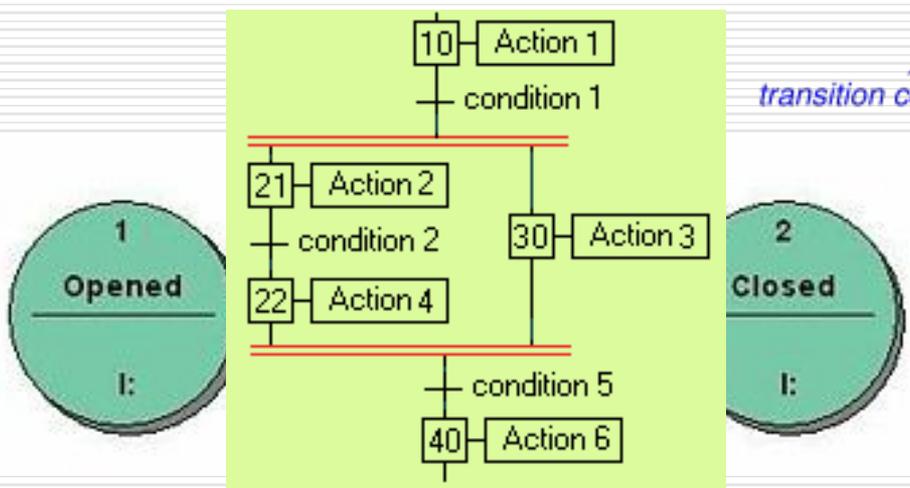
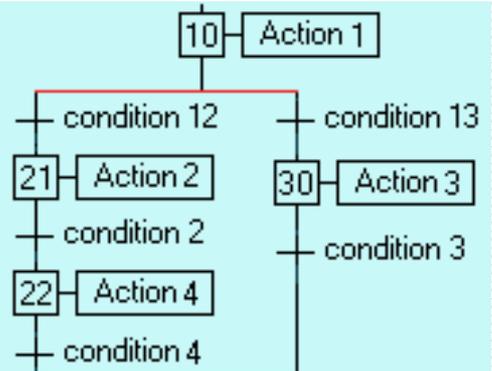
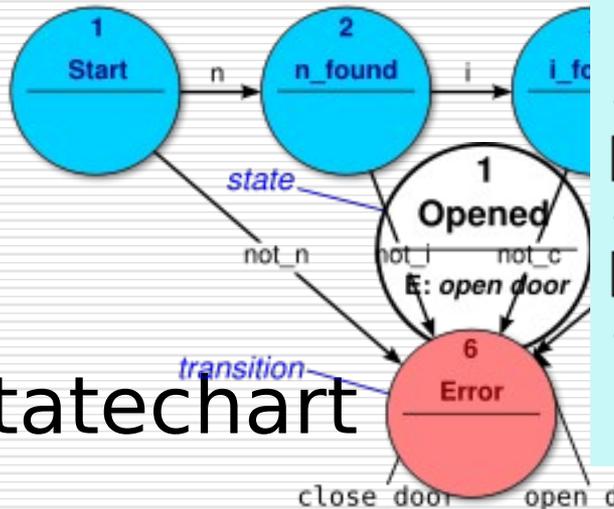


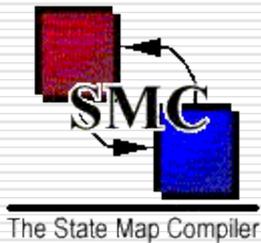
FSM are everywhere

- FSM : Finite State Machine
- Not a new technology
- Strong theoretical base
- Reactive systems / Transformational systems
- Event driven
- Applications :
 - Telephones, automobiles, communication networks, avionic systems, man-machine interface

FSM graphical view

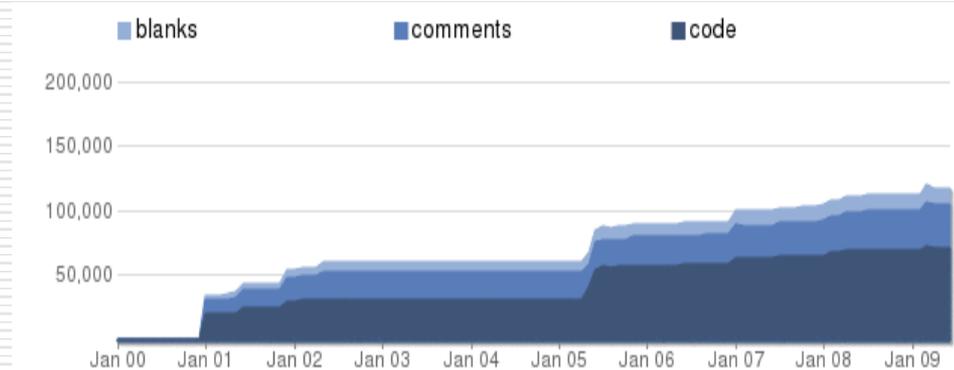
- Moore
- Mealy
- Grafcet
- UML = Harel statechart



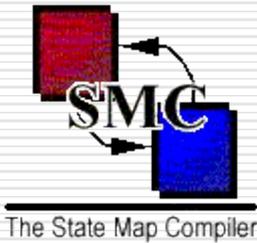


A SourceForge project

- Some facts :
 - registered in 2000
 - ~500 downloads / month
 - ~100 bugs (closed)
 - written in Java
 - mature codebase
 - well documented
 - 3 developers
 - Licence MPL

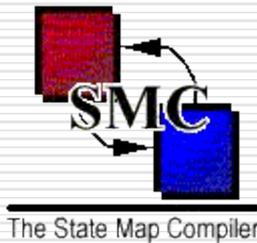


- See :
<http://www.ohloh.net/projects/7339?p=SMC>



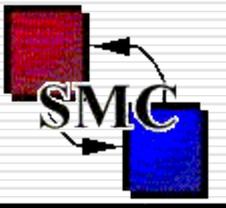
The State Machine Compiler

- Introduction
- Basic concepts
- Advanced concepts
- More features
- A case study : a Telephone
- Conclusion



A Compiler

- A input source .sm (yacc-like syntax)
 - A output source (readable) in your language
 - Currently 14 target languages :
 - C, C++, C#, Groovy, Java, Lua, Objective-C, Perl, PHP, Python, Ruby, Scala, Tcl and VB.net
 - An Object Oriented design :
 - your class has a member which is the FSM generated class
 - A small RunTime Library
-

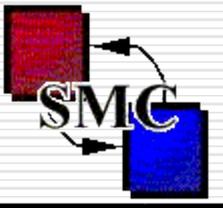


The State Map Compiler

A Simple Transition

```
// State
Idle {
  // Trans Next State Actions
  Run          Running  {}
}
```

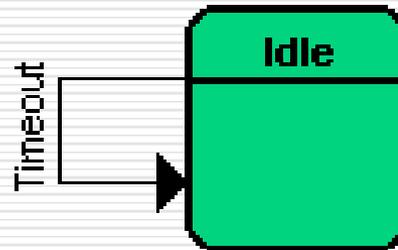


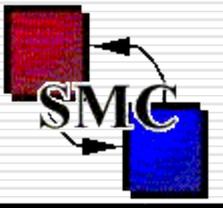


The State Map Compiler

A Reflexive Transition

```
// State
Idle {
    // Trans Next State Actions
    Timeout Idle {}
}
```

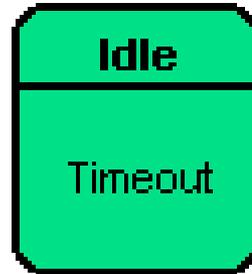


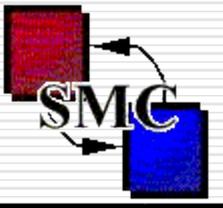


The State Map Compiler

A Internal Event

```
// State
Idle {
    // Trans Next State Actions
    Timeout nil {}
}
```





The State Map Compiler

A Transition with Actions

```
// State
```

```
Idle
```

```
{
```

```
    // Trans
```

```
    Run
```

```
        // Next State
```

```
        Running
```

```
            // Actions
```

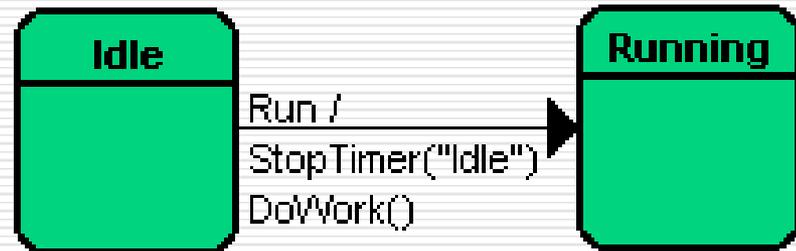
```
            {
```

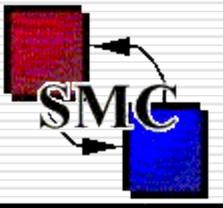
```
                StopTimer("Idle");
```

```
                DoWork();
```

```
            }
```

```
    }
```



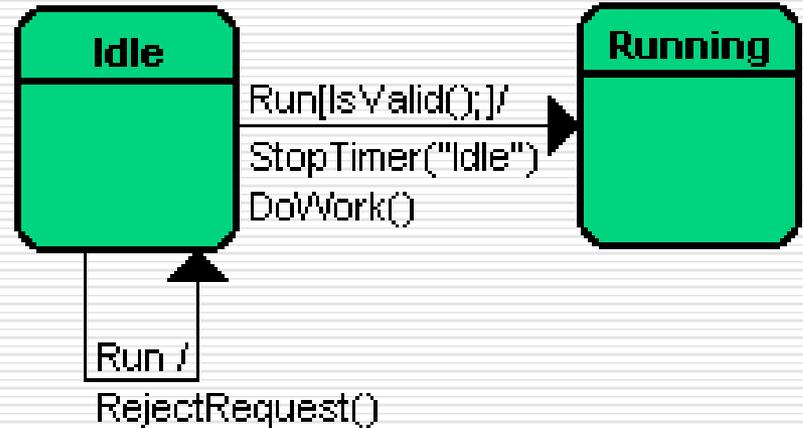


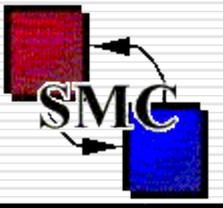
The State Map Compiler

Transition Guards

```
// State
Idle
{
    // Trans
    Run
    // Guard condition
    [ctxt.isValid()]
    // Next State
    Running
    // Actions
    {
        StopTimer("Idle");
        DoWork();
    }
}

Run      Idle { RejectRequest(); }
```





The State Map Compiler

Transition Arguments

```
// State
```

```
Idle
```

```
{
```

```
    // Transition
```

```
    Run(msg: const Message&)
```

```
    // Guard condition
```

```
    [msg.isValid()]
```

```
    // Next State
```

```
    Running
```

```
    // Actions
```

```
{
```

```
    StopTimer("Idle");
```

```
    DoWork(msg);
```

```
}
```

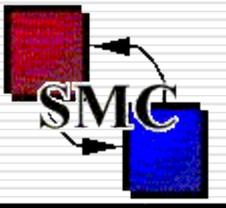
```
Run(msg: const Message&)
```

```
    // Next State Actions
```

```
    Idle { RejectRequest(msg); }
```

```
}
```



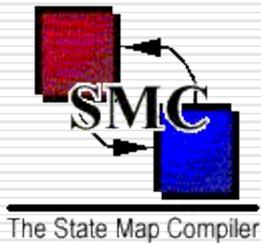


The State Map Compiler

Entry and Exit Actions

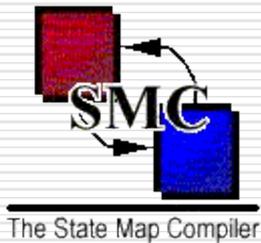
```
// State
Idle
Entry { StartTimer("Idle", 1); CheckQueue(); }
Exit  { StopTimer("Idle"); }
{
    // Transitions
}
```





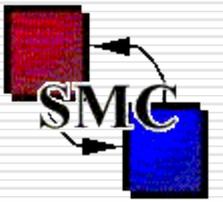
the State Machine Compiler

- Introduction
- Basic concepts
- **Advanced concepts**
- More features
- A case study : a Telephone
- Conclusion



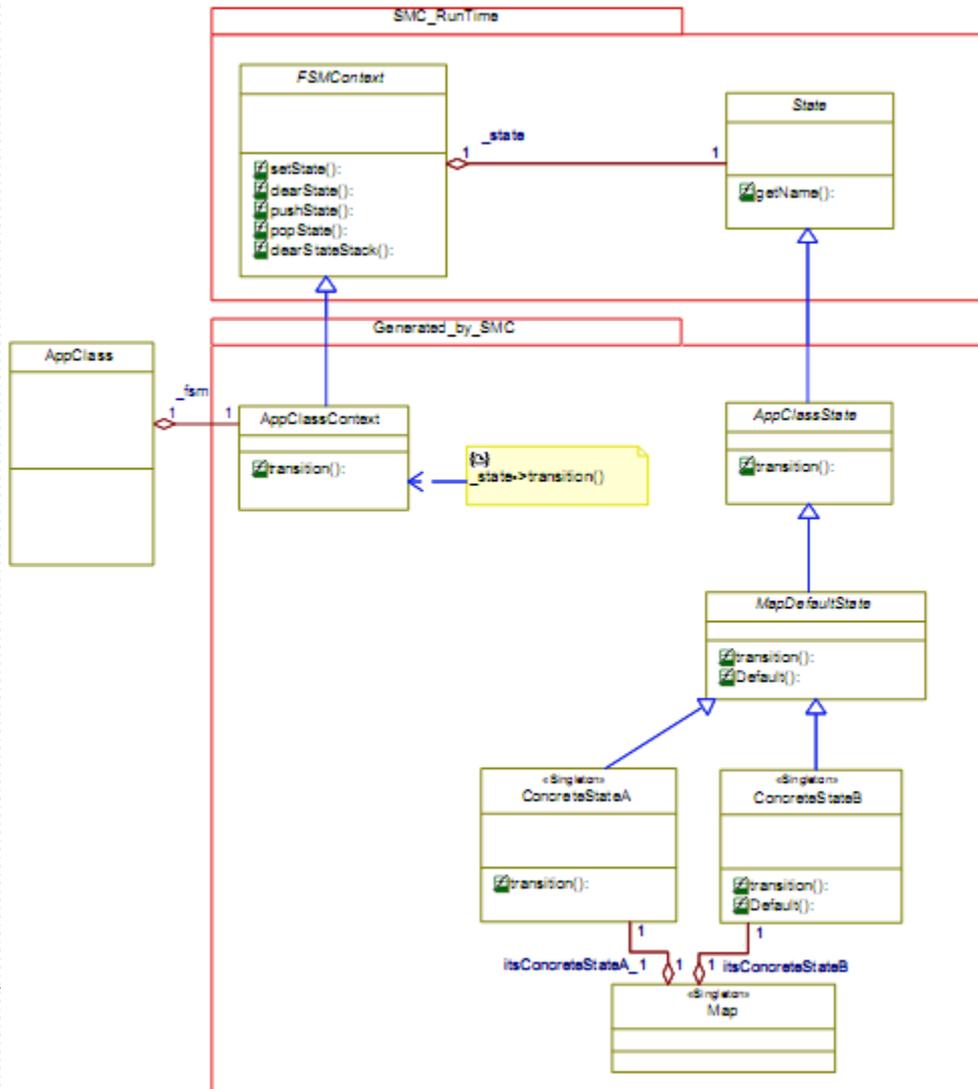
Advanced Features

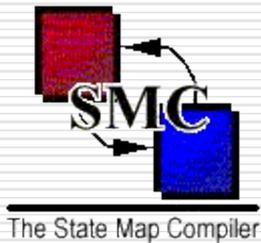
- Map : state container
 - only one level (multiple with UML)
- Push/Pop
 - with stack context
 - see UML History
- Default state
 - factorisation of common behavior in a map
- No concurrency (ie //)



The State Map Compiler

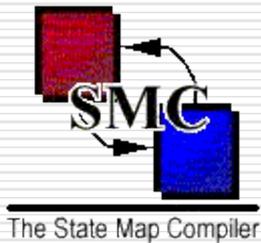
The Design Pattern





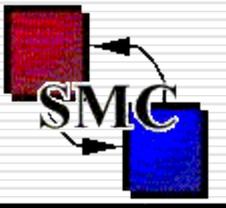
the State Machine Compiler

- Introduction
- Basic concepts
- Advanced concepts
- [More features](#)
- A case study : a Telephone
- Conclusion



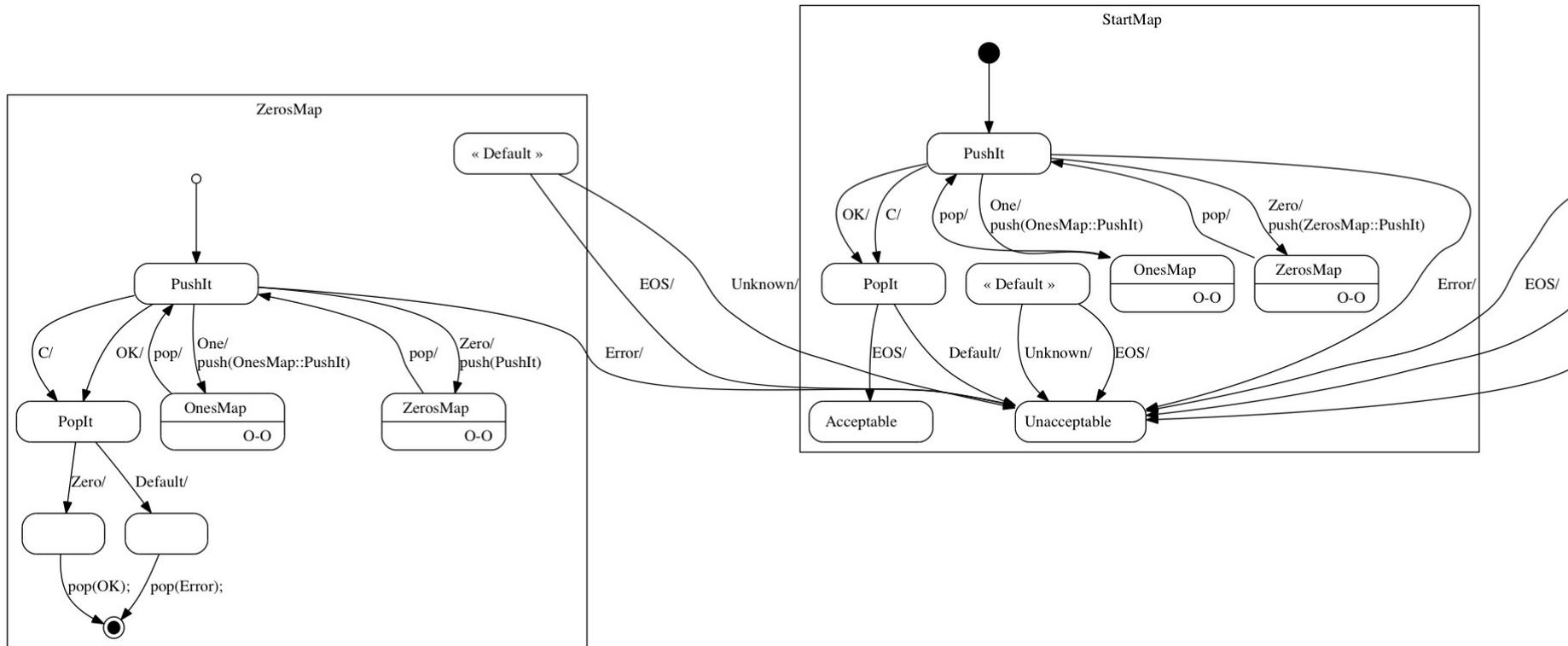
More features

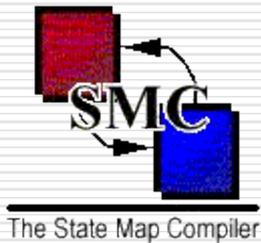
- Event management is yours
- Graphviz output generation
- HTML table generation
- Dynamic trace
- Namespace support
- Reflection (for MMI)



The State Map Compiler

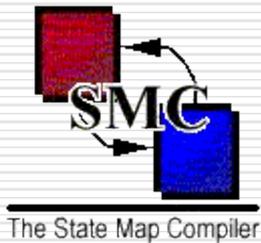
Graphviz output





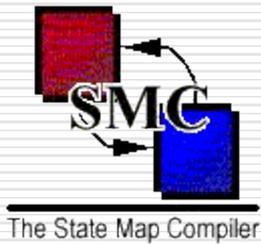
the State Machine Compiler

- Introduction
- Basic concepts
- Advanced concepts
- More Features
- **A case study : a Telephone**
- Conclusion



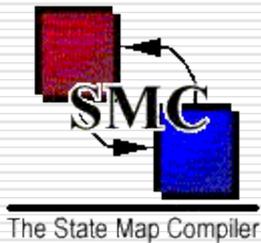
A Telephone

- Go to the WEB
 - Play with the demo (Applet Java)
- @ <http://smc.sourceforge.net/SmcDemo.htm>



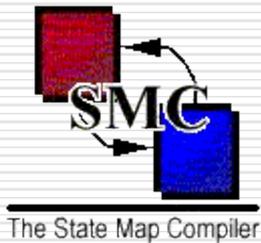
the State Machine Compiler

- Introduction
- Basic concepts
- Advanced concepts
- More Features
- A case study : NSquare
- **Conclusion**



all contributions welcomed

- Eclipse plugin
- Debian packaging
- Pluggable language support
- New target language
- Regression test
- ...



Bibliography / Webography

- SMC : <http://smc.sourceforge.net/>
- Robert C. Martin, "Agile Software Development"
- http://en.wikipedia.org/wiki/Finite_state_machine
- <http://en.wikipedia.org/wiki/Statechart>
- D. Harel, "Statecharts: A Visual Formalism for Complex Systems"
- <http://www.uml.org/>
- <http://fr.wikipedia.org/wiki/Grafcet>
- NF C03-190 - Diagramme fonctionnel "GRAFCET"
- http://en.wikipedia.org/wiki/Augmented_transition_network