Saint-Petersburg University of Aerospace Instrumentation

SystemC and SDL Co-Modelling Methods

Irina Lavrovskaya, Valentin Olenev, Alexander Stepanov, Alexey Rabin

Nov, 6th 2009

SDL↔SystemC co-modeling

There are 3 ways of SystemC and SDL co-modeling in a one communication system:

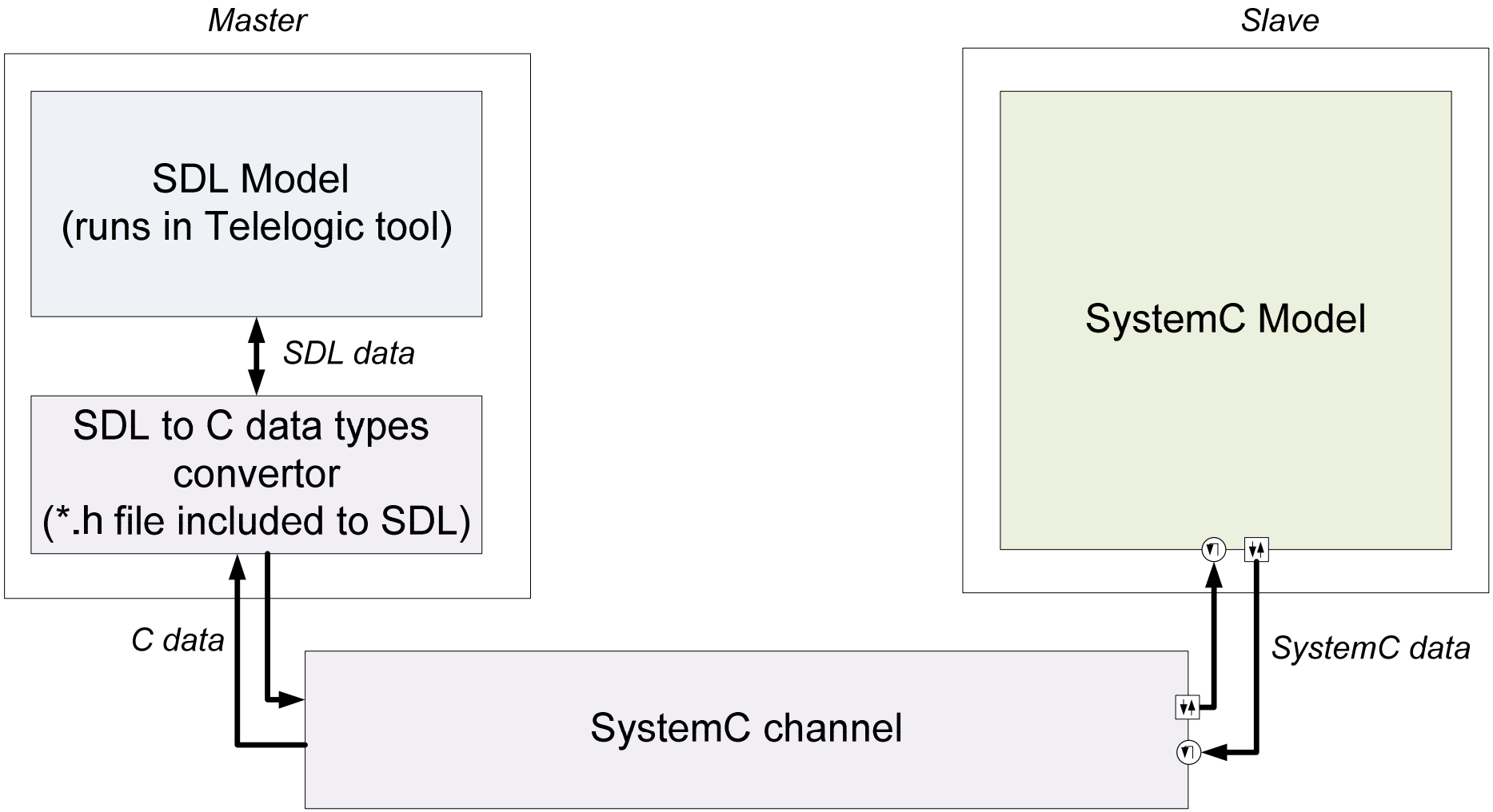
1. Insert SystemC into SDL (into the SDL Tools)
2. Insert SDL into SystemC (by compiling SDL to C code)
3. Run SDL and SystemC independently in an operating environments by using a specific tool with an SDL & SystemC interfaces

Insert SystemC into SDL

SDL can understand C code.
It is a feature of the Telelogic SDL Tool. So the SDL/SystemC models integration could be broken into a number of stages:

- writing an SDL model
- writing a pure C code to *.h file
- include this *.h file to SDL model
- writing a patch to convert SDL data types to C types

SystemC→SDL


Master
Slave
SDL Model
(runs in Telelogic tool)
SDL data
SDL to C data types
convertor
(*.h file included to SDL)
SystemC Model
C data
SystemC data
SystemC channel

SystemC→SDL: Possible implementation problems

- SDL could execute an *.h file written only in pure C, so using of classes, inheritance and other features of C++ and SystemC imposes difficulties
- SDL data types can not be easily converted to C types. For connection of this two models kind of “type convertor” should be implemented.
- Hard to use additional C and C++ libraries (e.g. SystemC)

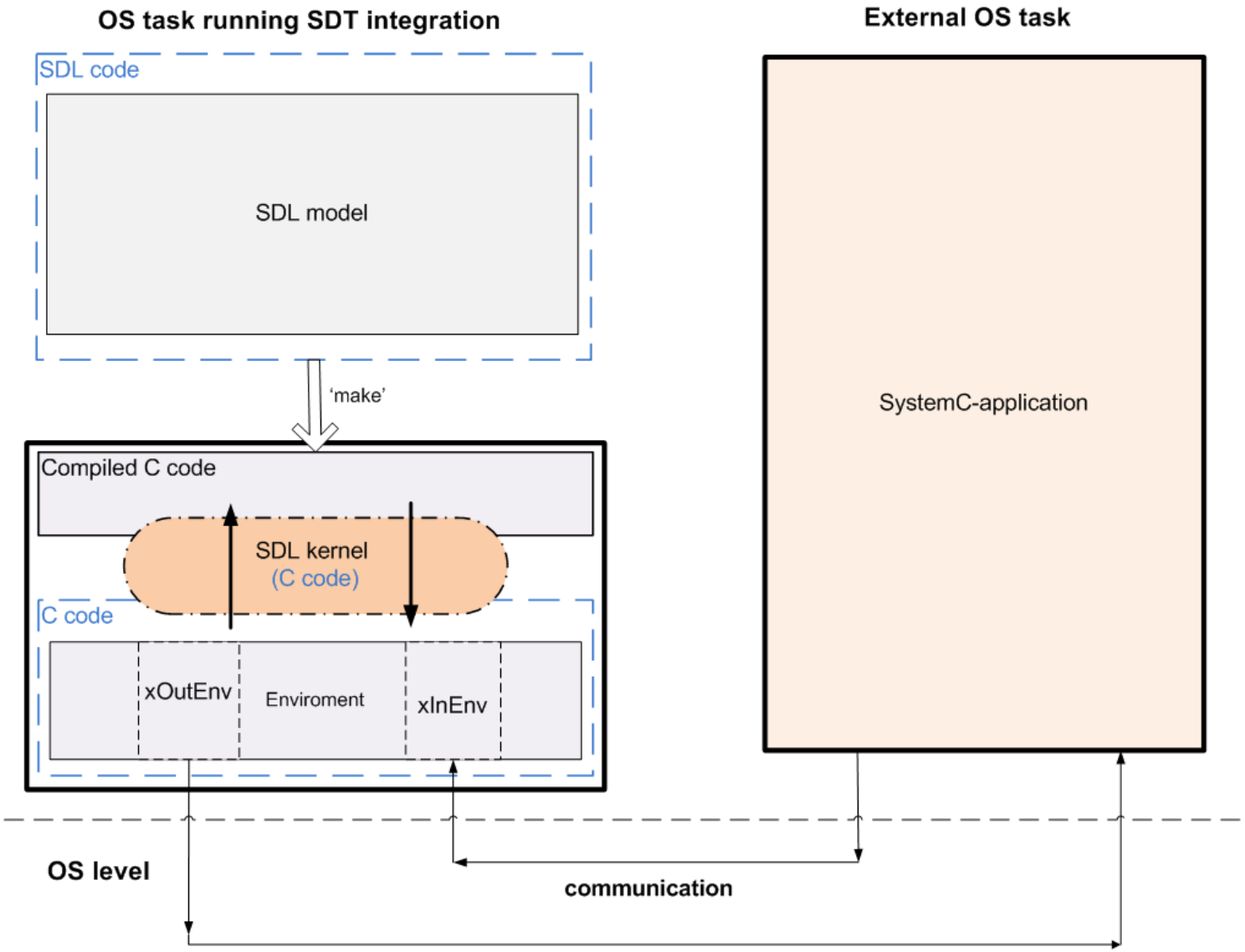
SystemC→SDL: Usability problems

- SDL model could be changed, but if there is a change in a “bottom” part of the model, possibly some updates would be needed for “type convertor”
- A big percent of changes in the SystemC model will cause a change in a channel which makes a call of methods

Insert SDL into SystemC

- During the debug SDL is compiled into a pure C code and environment which could be used for the connection to C++/SystemC
- Connection of the SDL model and OS/application is made by the modification of the environment
- An executable file is created by using of C-code of the model, environment and additional libraries
- In dependence of the “make” mode we can create MSC

SDL→SystemC


OS task running SDT integration
External OS task
SDL code
SDL model
'make'
Compiled C code
SDL kernel
(C code)
C code
xOutEnv
Enviroment
xInEnv
SystemC-application
OS level
communication

SDL→SystemC: Possible implementation problems

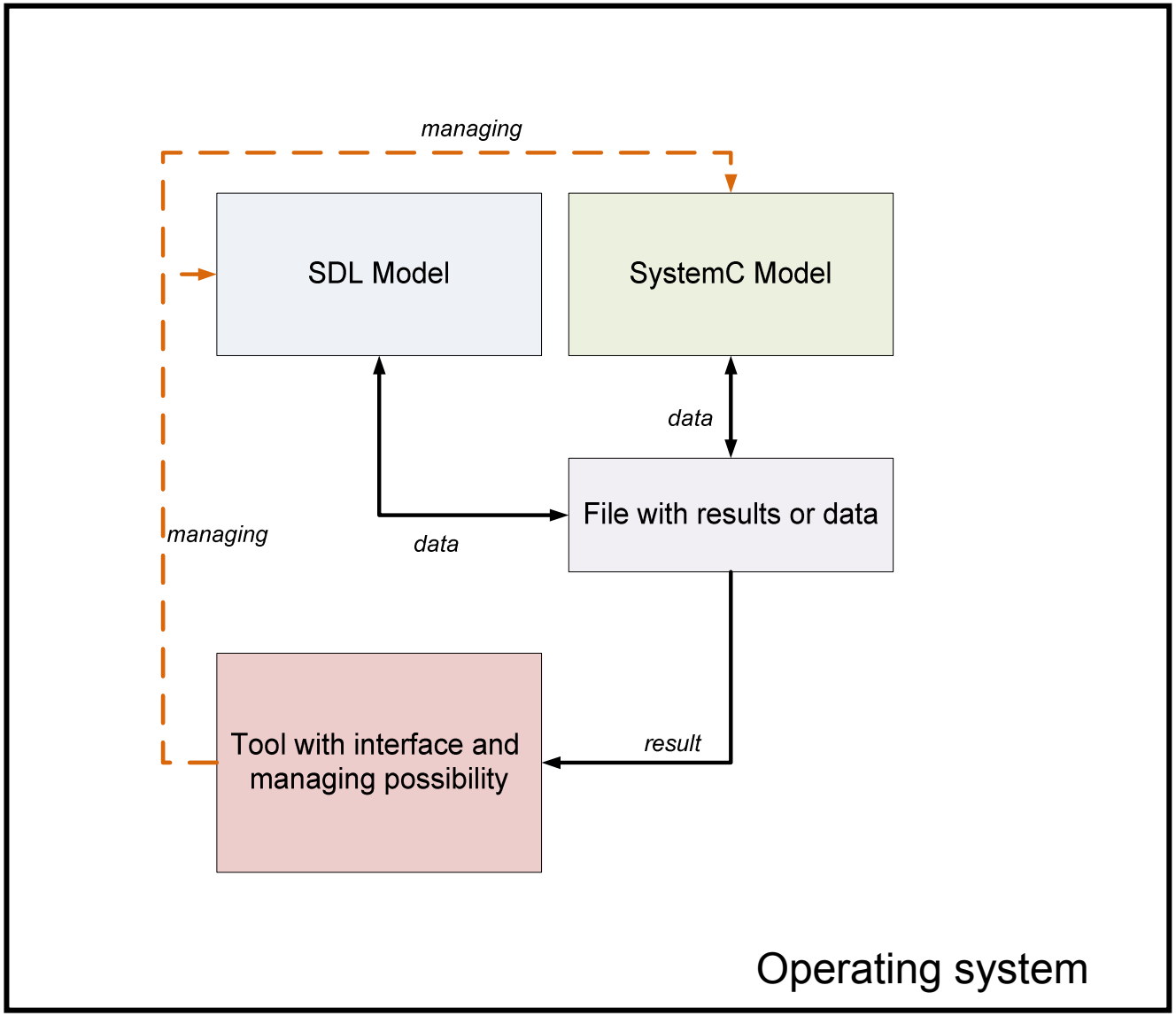
- Any modification of an SDL model needs to create again all C-code and thereafter to make again the whole project

- We only create an executable file. There are no other ways provided to work with the project (e.g. Simulator)

Use SDL and SystemC independently

- Could be implemented by passing the results through a file
  - Writing results to a file and reading them depends on SystemC or SDL clocks.
- A special tool with interface will manage the SDL model and SystemC model, read the file and show the results.
  - Still a question to further research, how to manage SDL and SystemC tools correctly
- This tool could handle point-to-point work of two SDL/SystemC systems.

Use SDL and SystemC independently



Conclusions

- Insert SystemC to SDL:
  - SDL could not use SystemC and C++ directly
  - SDL data types could not be easily converted to C types
- Insert SDL to SystemC:
  - Whole SDL project is compiled into a single huge C file
  - Every change in SDL causes the change in a system
- Run SDL and SystemC using special tool:
  - More programming, less research issues
  - Needs more resources from the computer
  - All the modeling results in SystemC and SDL parts could be seen
  - The most flexible way: changes in one model won't touch the other parts

Thank you!