Verifying OS components

Mateus Krepsky Ludwich

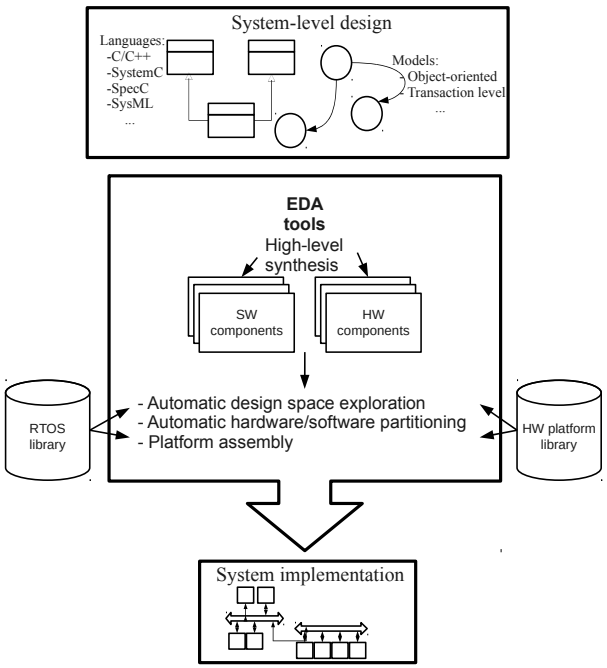
Laboratory for Software and Hardware Integration (LISHA)
Federal University of Santa Catarina (UFSC)
Florianópolis, Brazil
mateus@lisha.ufsc.br

Computer Architecture Group Presentations

Why to Verify?

- Increasing the use of computational systems thus, increasing the need of them to be correct
  - “A system must do what it is *specified* to do”

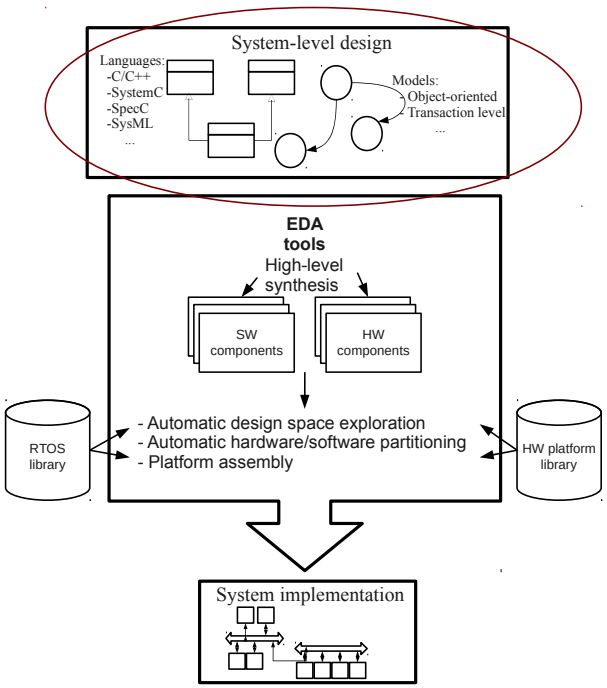
What to Verify?



- Verification at System Level Design: regardless whether a component is gone to be implemented in SW or HW
- Chosen language: C++
  - HLS allows for “writing” HW in C++

Figure: Tomorrow’s System-Level Design Flow

What to Verify?



- Verification at System Level Design: regardless whether a component is gone to be implemented in SW or HW
- Chosen language: C++
  - HLS allows for “writing” HW in C++

Figure: Tomorrow’s System-Level Design Flow

How to Verify?

- Software Model Checking
- CBMC (C Bounded Model Checking)
  - Pointers
  - Arrays bound
  - User defined assertions

Approach/Methodology

- Write the contract of the program/design to be verified
- Instrument the program with this contract
- Execute CBMC over the program to verify whether it follows its contract

Writing the contract

- Hoare Logic: *{P} C {Q}*
- Example (D's style)

```
void insert(T obj)
in
{
   size_at_pre = size();
   assert(! ready.contains(obj));
}
out
{
   assert(size() == size_at_pre + 1);
   assert(ready.contains(obj));
}
body
{
   // Method's implementation
}
```

Instrumenting the program with its contract

- Directly in the code
- Or using scenario adapters:

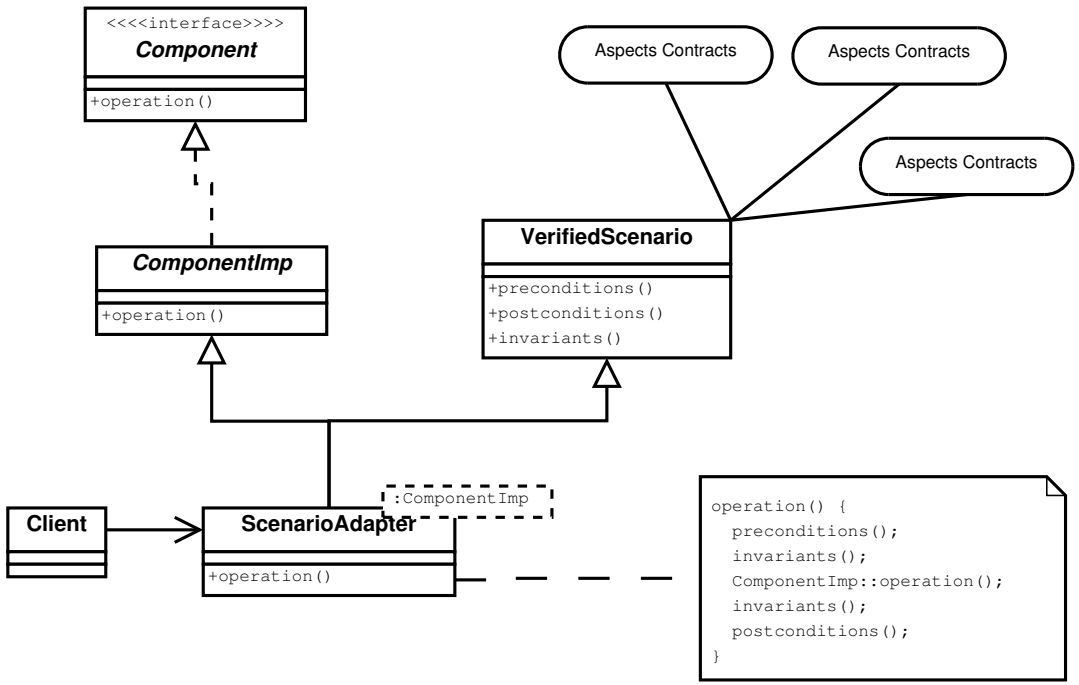


Figure: Verified Scenario

Execute CBMC over the program to verify whether it follows its contract

- “Verification SUCCESSFUL”
- Or a counter-example, showing where the specification is not followed

Case study: OS Scheduler

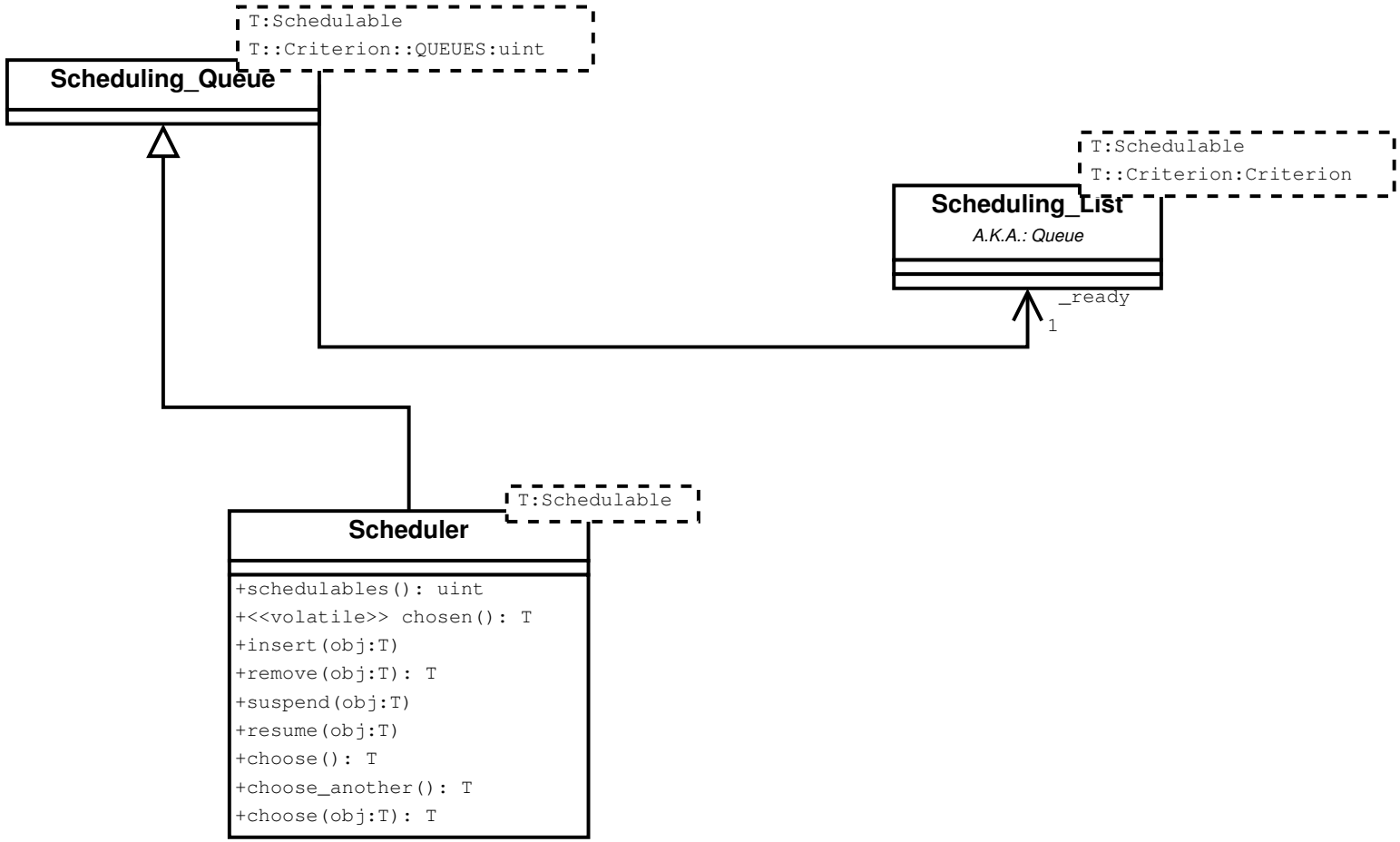


Figure: EPOS's Scheduler

Scheduler's Contract

- Class invariants
  - **assert**(Base::size() >= 0)

Scheduler's Contract

Method	Preconditions	Postconditions
unsigned int schedulables()	None	None
T volatile chosen()*	None	**assert**(ready.contains(var_chosen))
void insert(T obj)*	**assert**(! ready.contains(obj))	**assert**(Base::size() == size_at_pre + 1) **assert**(ready.contains(obj))

T remove(T* obj)*	None	**if** ready_at_pre.contains(obj)) **then**
		assert(Base::size() == size_at_pre - 1) **else**
		assert(result = null)
		endif
		assert(! ready.contains(obj))

void suspend(T obj)*	None	**if** ready_at_pre.contains(obj)) **then**
		assert(Base::size() == size_at_pre - 1)
		endif
		assert(! ready.contains(obj))

*void resume(T * obj)*	*same as insert*	*same as insert*
T choose()*	**assert**(ready.contains(var_chosen))	**assert**(ready.contains(var_chosen))
T choose_another()*	**assert**(ready.contains(var_another))	**assert**(ready.contains(var_another))
*T * choose(T * obj)*	**assert**(ready.contains(obj))	**assert**(ready.contains(var_chosen))