Lucas Heer | MDE project reading presentation | 15.12.2016

Challenges

- Safety → testing, verification

Challenges

- Safety → testing, verification
- Performance → low-level

Challenges

- Safety → testing, verification
- Performance → low-level

- Problem: C as programming language

```c
int main(void)
{
    int *ptr = (int *)0;
    *ptr = 0;
    return 0;
}
```

mbeddr

„Set of integrated and extensible languages for embedded software engineering, plus an IDE“

(Builds on JetBrains MPS language workbench)

Extensions

- C99 core
  - preprocessor
  - headers
  + module system

Extensions

- C99 core
  - preprocessor
  - headers
  + module system
- Decisions tables
- State machines

Extensions

- C99 core
  - preprocessor
  - headers
  + module system
- Decisions tables
- State machines
- Requirements, unit testing, documentation, ...

Extensions

Decision Table

main

model NewSolution.main.main

constraints

imports SIUnits

```
uint32 get_points(uint32 speed, uint32 altitude) {
  uint32 points = 0;
  points +=
```

	speed > 200	speed > 500	otherwise 0;
altitude < 1000	1	2	
altitude < 500	3	4	

```
  return points;
} get_points (function)

exported int32 main(int32 argc, string[] argv) {
  uint32 points = get_points(300, 700);
  return 0;
} main (function)
```

Extensions

State machine - Textual

```
statemachine SM initial = idle {
  in event customer_arrive() <no binding>
  in event make_pause() <no binding>
  in event back_to_work() <no binding>
  in event customer_finished() <no binding>

  var uint32 customer_served = 0

  state idle {
    on customer_arrive [ ] -> serve
    on make_pause [customer_served > 10] -> pause
  } state idle
  state serve {
    entry { customer_served++; }
    on customer_finished [ ] -> idle
  } state serve
  state pause {
    entry { customer_served = 0; }
    on back_to_work [ ] -> idle
  } state pause
}
```

Extensions

State machine - Tabular

```
statemachine SM initial = idle {
```

		Events			
		customer_arrive()	make_pause()	back_to_work()	customer_finished()
States	idle	[ ] -> serve	[customer_served > 10] -> pause		
	serve				[ ] -> idle
	pause			[ ] -> idle	

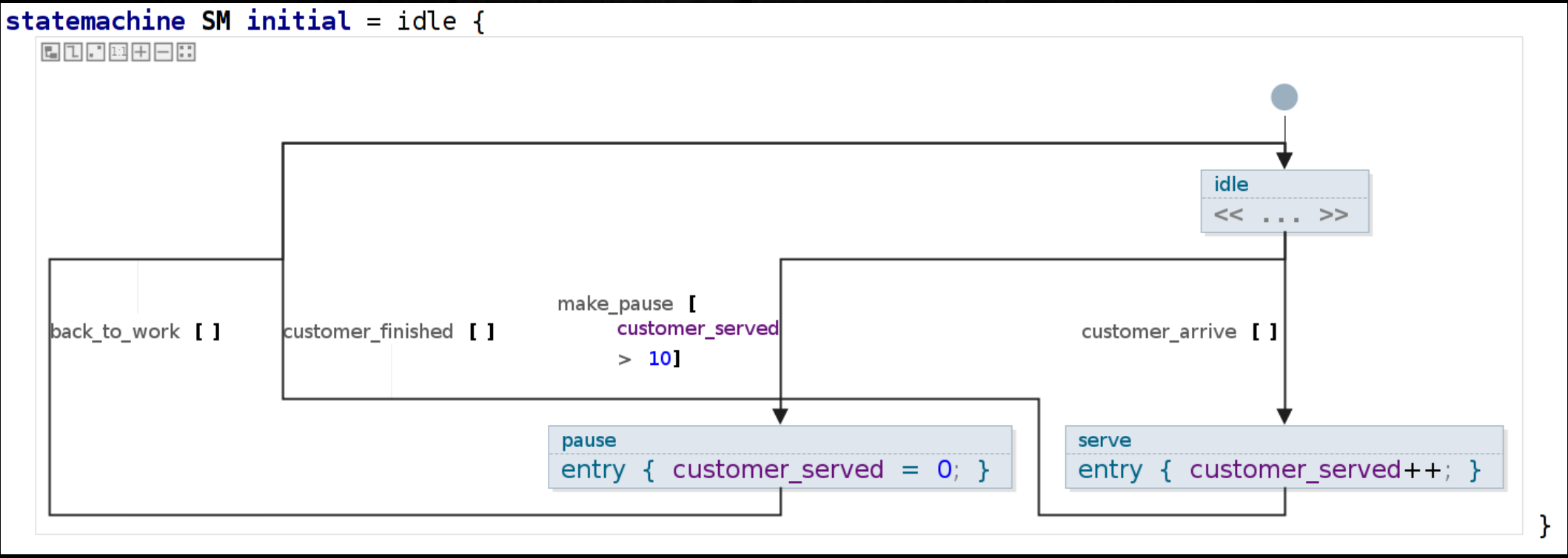
```
}

exported testcase test_SM {
  SM state_machine;
  state_machine.init;

  for (uint8 i = 0;i < 10; i++ ) {
    state_machine.trigger(customer_arrive);
    state_machine.trigger(customer_finished);
  } for
  state_machine.trigger(make_pause);
  assert(0) state_machine.isInState(pause) == true;
} test_SM(test case)
```

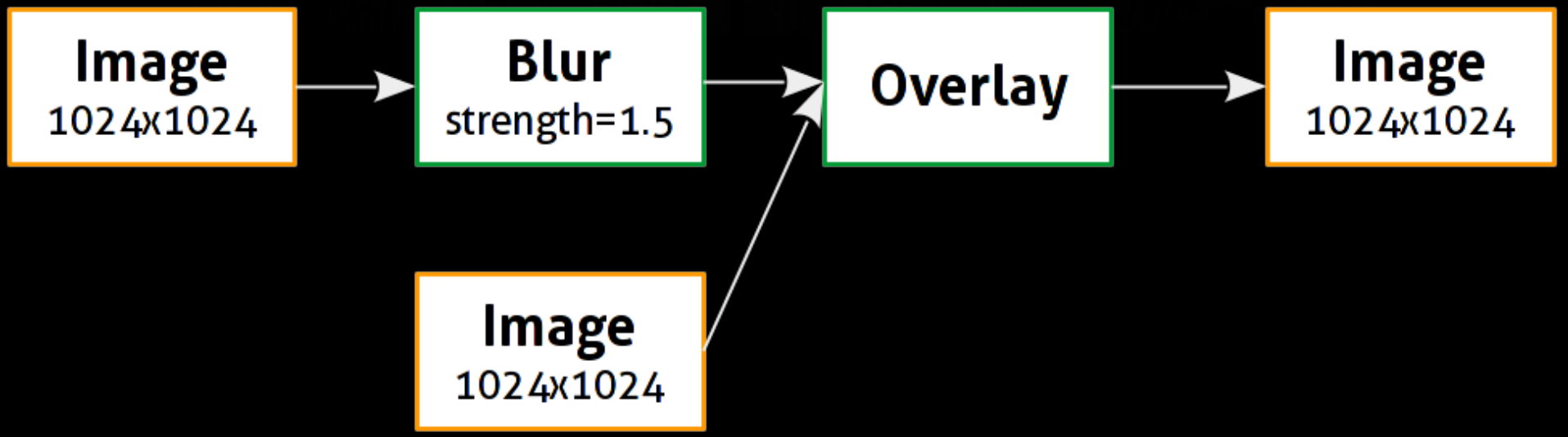
Extensions

State machine - Graphical



Case study

- Implement custom extension
- Image processing pipeline



Case study

Sample from the language:

```
1  Image in1;
2  Image in2;
3  BlurBlock blur(strength=1.5);
4  OverlayBlock overlay();
5  Image out;
6
7  in1.connect(blur);
8  in2.connect(overlay);
9  blur.connect(overlay);
10 overlay.connect(out);
11
12 out.run();
```

Comparison

vs.


Questions?

Resources

- mbeddr user guide: http://mbeddr.com/userguide/UserGuideExport.html
- Paper from the tooling perspective: http://mbeddr.com/files/voelteretal-mbeddr-final.pdf
- Paper from the language engineering perspective: http://mbeddr.com/files/wavefront-updatedsubmission2.pdf