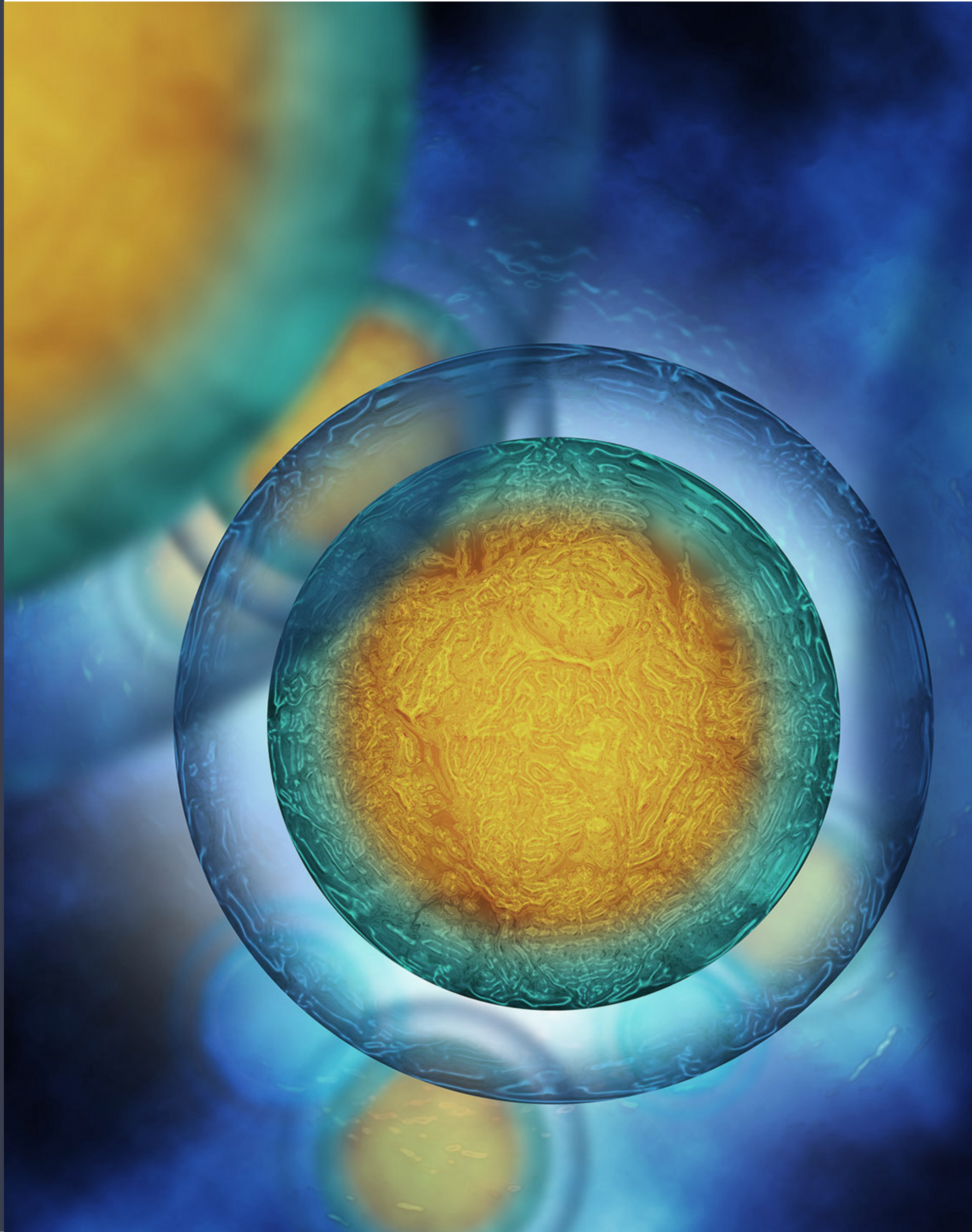
FE3CWS

NIJMEGEN TEACHER TRAINING MATERIAL

Intellectual output 3 of the ERASMUS+ project 2017-1-SK01-KA203-035402



Some words about the

CONTENTS

- A single but powerful topic related to software composition, comprehension and correctness
- Available in 7 languages: English, Hungarian, Slovak, Croatian, Romanian, Bulgarian and Portuguese

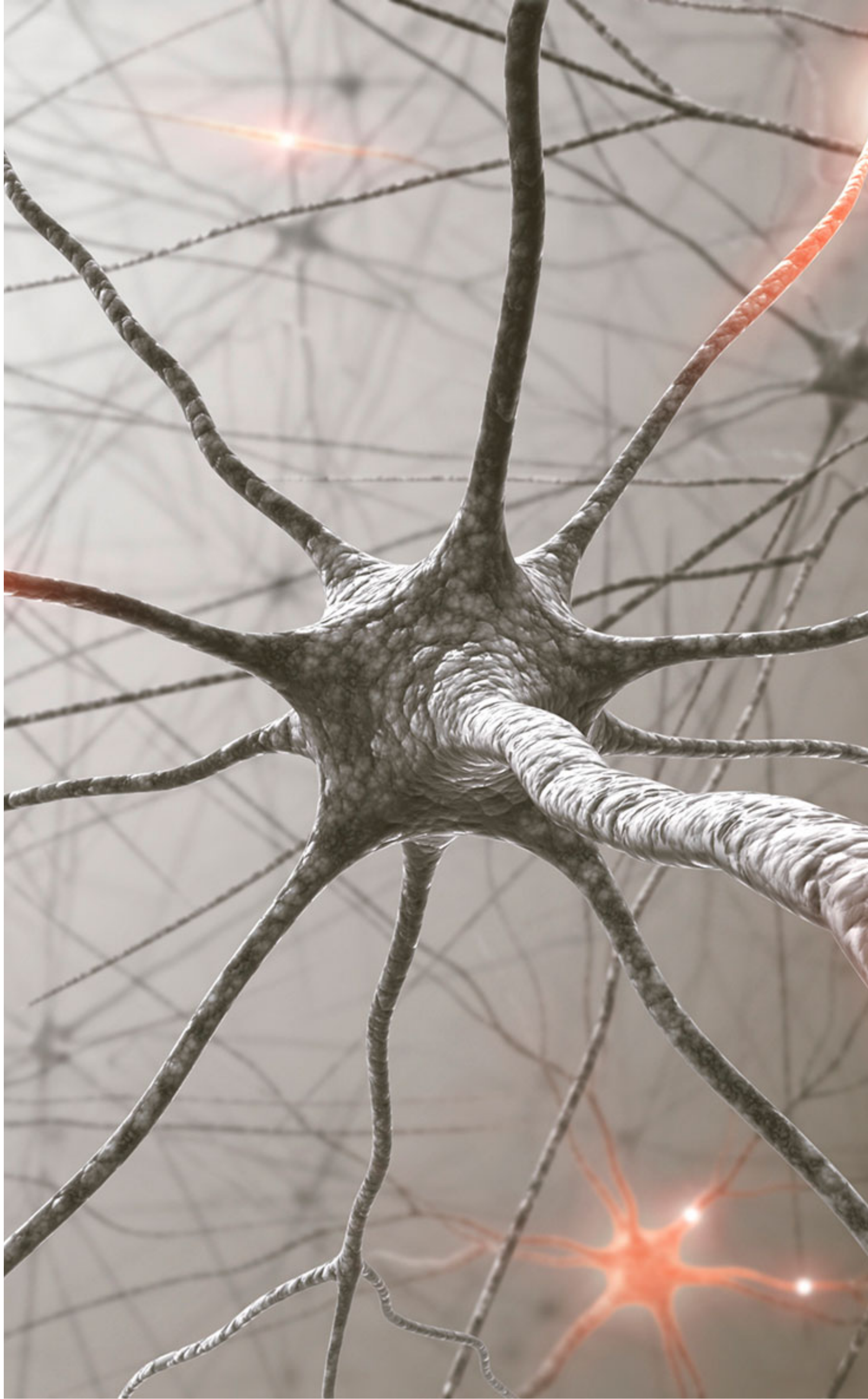
Co-funded by the Erasmus+ Programme of the European Union

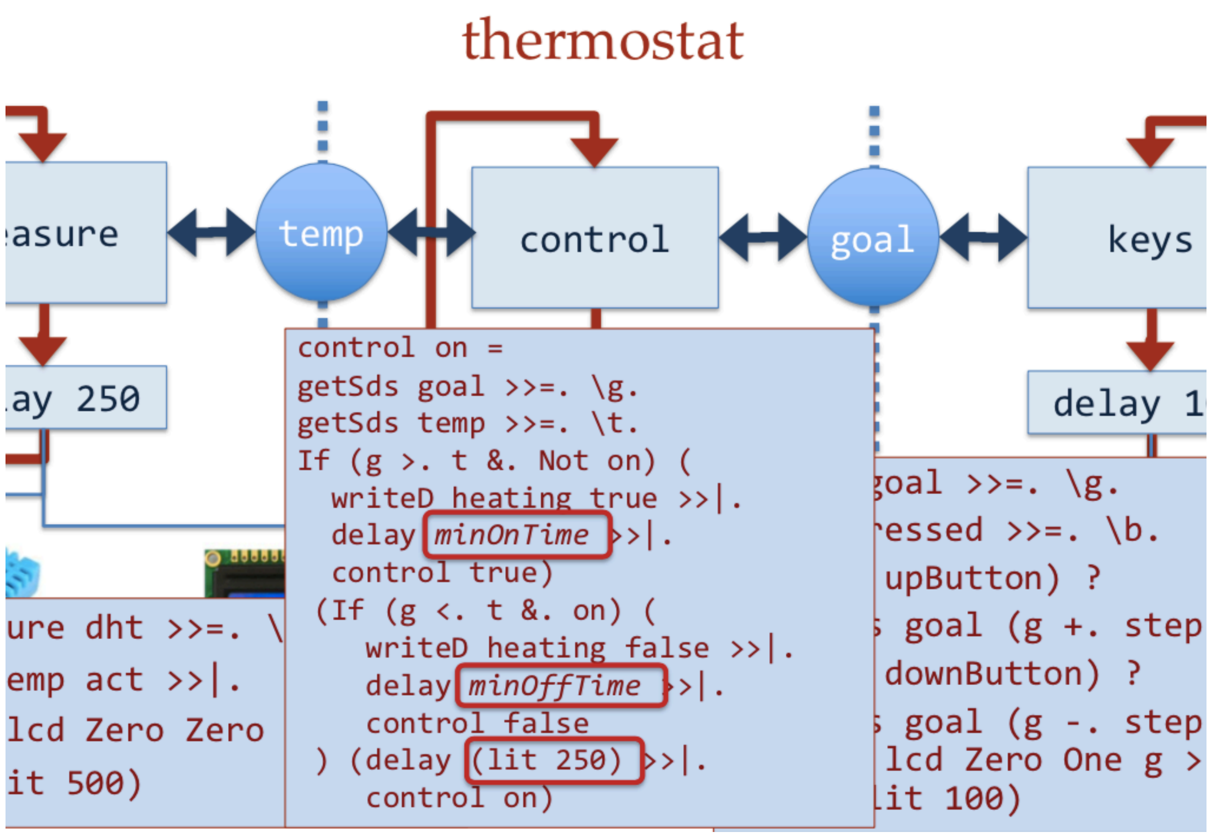
Second Teacher Training Material - "Functional Programming in the New Devices Lab".

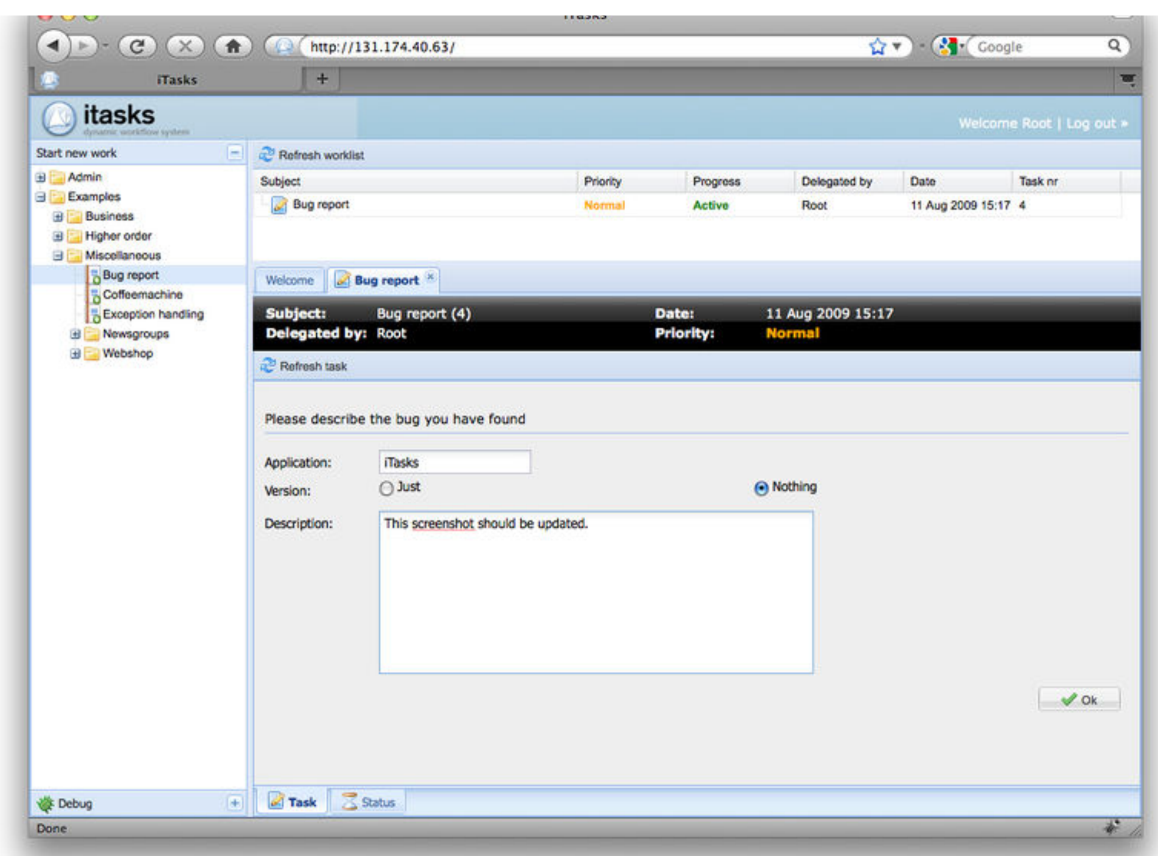
ABOUT

The Teacher training in Nijmegen the Netherlands covers five days of intensive schooling and cooperation at the Radboud University Nijmegen. The goals of this Teacher Training within the FE3CWS project was to bring the participants on par with the current state-of-art of functional programming and Task Oriented Programming, TOP. The topics treaded covers:

- Recap of pure functional programming in Clean. See https://clean.cs.ru.nl/Clean.
- Generic programming: how to define manipulations that work for any first order data type, even for types that are not yet defined.
- TOP to specify the cooperation of humans and automated tasks to achieve goals. The iTask system supports and monitors the execution of tasks. See https://clean.cs.ru.nl/ITasks. Interaction with users is mainly done via web-based editors that are generated from the types used in the high-level task definitions. The TOP formalism contains a small number of flexible basic tasks (like web-editors and control of IoT peripherals) and a set of combinators for the parallel and sequential composition of tasks.







TOP to control the Internet of Things, IoT.

TOP is very suited to control the IoT and frees the developers from the burden of many programming languages, protocols, interfaces and their interoperability. To cope with the limited computing power and energy constraints of the IoT devices a special Domain Specific Language, DSL, called mTasks is made to program the IoT devices. The iTask and mTask systems work seamlessly together to connect the world of web-based tasks with the tiny tasks running on the IoT.

The teaching of these topics was done by a combination of interactive tutorials and hands-on programming with the participants. The topics of this teacher training are in the heart of the project. The concise programs at a high level of abstraction directly contributes to their comprehensibility. The task combinators are a very useful illustration of composability in software. The static type system of the pure functional host language Clean ensures that runtime type errors cannot occur. Together with the concise notation in these DSLs this contributes to the correctness of TOP programs.

DOWNLOAD LINKS

https://fe3cws.kpi.fei.tuke.sk/O3ENG.html

BLINKING LEDS = HELLO WORLD

Arduino code

```
void setup () {
    pinMode(D4, OUTPUT);
}
void loop() {
    digitalWrite(D4, HIGH);
    delay(500);
    digitalWrite(D4, LOW);
    delay(500);
}
```

mTask (Clean) code

```
blink :: Main (MTask v ()) | mtask v
blink = { main=rpeat (
    writeD d4 (lit True)
>>|. delay (lit 500)
>>|. writeD d4 (lit False)
>>|. delay (lit 500)
)}
```

TWO FACTORIAL EXAMPLES

```
factorial :: Int → Main (MTask v Int) | mtask v
factorial x =
   fun λfac=(λi →
      If (i ==. lit zero)
         (lit one)
         (i *. fac (i -. lit one))) In
   {main=rtrn (fac (lit i))}

//Tail call optimized factorial
factorial‘ :: Int → Main (MTask v Int) | mtask v
factorial‘ x =
   fun λfacacc=(λ(n,a) →
      If (n ==. lit zero)
         a
         (facacc (n -. lit one, n*.a))) In
   fun λfac=(λi →
      facacc (i, lit one)) In
   {main=rtrn (fac (lit i))}
```

BLINKING LEDS THE FUNCTIONAL WAY IN MTASK

```
module blink

import StdEnv, iTasks

import Interpret
import Interpret.Device.TCP

Start :: *World → *World
Start w = doTasks main w

main :: Task Bool
main = enterDevice >>= λspec → withDevice spec
   λdev → liftmTask blink dev -|| viewDevice dev
where
   blink :: Main (MTask v Bool) | mtask v
   blink
      = fun λblink=(λx →
               delay (lit 500)
         >>|. writeD d4 x
         >>=. blink o Not)
      In {main=blink (lit True)}
```

INTERACTIVE BLINKING

```
main :: Task Bool
main = enterDevice >>= λspec → withDevice spec
   λdev → withShared 500 λdelayShare →
          liftmTask (blink delayShare) dev
      -|| updateSharedInformation "Interval" [updater] delayShare
where
   updater :: UpdateOption Int Int
   updater = UpdateUsing (λx → (x, x)) (const fst)
      (panel2
         (slider <<@ minAttr 5 <<@ maxAttr 10000)
         (integerField <<@ enabledAttr False))

   blink :: (Shared s Int) → Main (MTask v Bool) | mtask, liftsds v & RWShared s
   blink delayShare = liftsds λdelaysh=delayShare
      In fun λblink=(λx →
              writeD d4 x
         >>|. getSds delaysh
         >>~. delay
         >>=. λ_ → blink (Not x))
      In {main=blink (lit True)}
```

AN INTERACTIVE MTASK PROGRAM FOR INTERACTING WITH THE LED MATRIX

```
:: Ledstatus = {x :: Int, y :: Int, status :: Bool}
derive class iTask Ledstatus

main = enterDevice >>= λspec → withDevice spec
   λdev →  viewDevice dev >^*
       [OnAction (Action "Toggle") (always (
          enterInformation () [] >>= λs → liftmTask (toggle s) dev
             >>~ viewInformation "done" []))
       ,OnAction (Action "Clear") (always (
          liftmTask clear dev
          >>~ viewInformation "done" []))
       ] @! ()
where
   dot lm s = LMDot lm (lit s.x) (lit s.y) (lit s.status)

   toggle :: Ledstatus  →  Main (MTask v ()) | mtask, LEDMatrix v
   toggle s = ledmatrix D5 D7 λlm → {main=dot lm s >>|. LMDisplay lm}

   clear :: Main (MTask v ()) | mtask, LEDMatrix v
   clear = ledmatrix D5 D7 λlm → {main=LMClear lm >>|. LMDisplay lm}
```

MEASURE TEMPERATURE AND HUMIDITY

```
main = enterDevice >>= λspec → withDevice spec
   λdev → liftmTask temp dev >&> viewSharedInformation () [ViewAs templens]
where
   templens = maybe (0.0, 0.0) λ(t, h) → (toReal t / 10.0, toReal h / 10.0)

   temp :: Main (MTask v (Int, Int)) | mtask, dht v
   temp = DHT D4 DHT22 λdht → {main=temperature dht .&&. humidity dht}
```

Literature

Peter Achten. Clean for Haskell98 Programmers. 13th July 2007.

Peter Achten, Pieter Koopman and Rinus Plasmeijer. 'An Introduction to Task Oriented Programming'. In: Central European Functional Programming School. Springer, 2015, pp. 187– 245.

Douglas Adams. The Hitchhiker's Guide to the Galaxy Omnibus: A Trilogy in Four Parts. Vol. 6. Pan Macmillan, 2017.

Matheus Amazonas Cabral De Andrade. 'Developing Real Life, Task Oriented Applications for the Internet of Things'. Master's Thesis. Nijmegen: Radboud University, 2018. 60 pp.

Tom Brus et al. 'Clean – a language for functional graph rewriting'. In: Conference on Functional Programming Languages and Computer Architecture. Springer, 1987, pp. 364– 384.

Jacques Carette, Oleg Kiselyov and Chung-Chieh Shan. 'Finally tagless, partially evaluated: Tagless staged interpreters for simpler typed languages'. In: Journal of Functional Programming 19.5 (Sept. 2009), p. 509. issn: 0956-7968, 1469-7653. doi: 10.1017/S0956796809007205.

James Cheney and Ralf Hinze. First-class phantom types. Cornell University, 2003.

Li Da Xu, Wu He and Shancang Li. 'Internet of things in industries: a survey'. In: Industrial Informatics, IEEE Transactions on 10.4 (2014), pp. 2233–2243.

L. M. G. Feijs. 'Multi-tasking and Arduino : why and how?' In: Design and semantics of form and movement. 8th International Conference on Design and Semantics of Form and Movement (DeSForM 2013). Ed. by L. L. Chen et al. Wuxi, China, 2013, pp. 119–127. isbn: 978-90-386-3462-3.

Patrick C. Hickey et al. 'Building embedded systems with embedded DSLs'. In: ACM Press, 2014, pp. 3–9. isbn: 978-1-4503-2873-9. doi: 10.1145/2628136.2628146.

Pieter Koopman, Mart Lubbers and Rinus Plasmeijer. 'A Task-Based DSL for Microcomputers'. In: Proceedings of the Real World Domain Specific Languages Workshop 2018 on - RWDSL2018. Vienna, Austria: ACM Press, 2018, pp. 1–11. isbn: 978-1-4503-6355-6. doi: 10.1145/3183895.3183902.

Mart Lubbers. 'Task Oriented Programming and the Internet of Things'. Master's Thesis. Nijmegen: Radboud University, 2017. 69 pp.

Mart Lubbers, Pieter Koopman and Rinus Plasmeijer. 'Multitasking on Microcontrollers using Task Oriented Programming'. In: 2019 42st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). COnference on COmposability, COmprehensibility and COrrectness of Working Software. Opatija, Croatia: IEEE, 2019, pp. 1842–1846.

Mart Lubbers, Pieter Koopman and Rinus Plasmeijer. 'Task Oriented Programming and the Internet of Things'. In: Proceedings of the 30th Symposium on the Implementation and Application of Functional Programming Languages. International Symposium on Implementation and Application of Functional Languages. Lowell, MA: ACM, 2018, p. 12. isbn: 978-1-4503-7143-8. doi: 10.1145/3310232.3310239.

Rinus Plasmeijer, Peter Achten and Pieter Koopman. 'iTasks: executable specifications of interactive work flow systems for the web'. In: ACM SIGPLAN Notices 42.9 (2007), pp. 141–152.

Rinus Plasmeijer and Pieter Koopman. 'A Shallow Embedded Type Safe Extendable DSL for the Arduino'. In: Trends in Functional Programming. Vol. 9547. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016. isbn: 978-3-319-39109-0 978-3-319-39110-6. doi: 10.1007/978-3-319-39110-6.

Camil Staps and Mart Lubbers. The Clean language search engine. 2017. url: https: //cloogle.org.

Jurrien Stutterheim, Peter Achten and Rinus Plasmeijer. 'Maintaining Separation of Concerns Through Task Oriented Software Development'. In: Trends in Functional Programming. Ed. by Meng Wang and Scott Owens. Vol. 10788. Cham: Springer International Publishing, 2018, pp. 19–38. isbn: 978-3-319-89718-9 978-3-319-89719-6. doi: 10.1007/978- 3-319-89719-6_2.