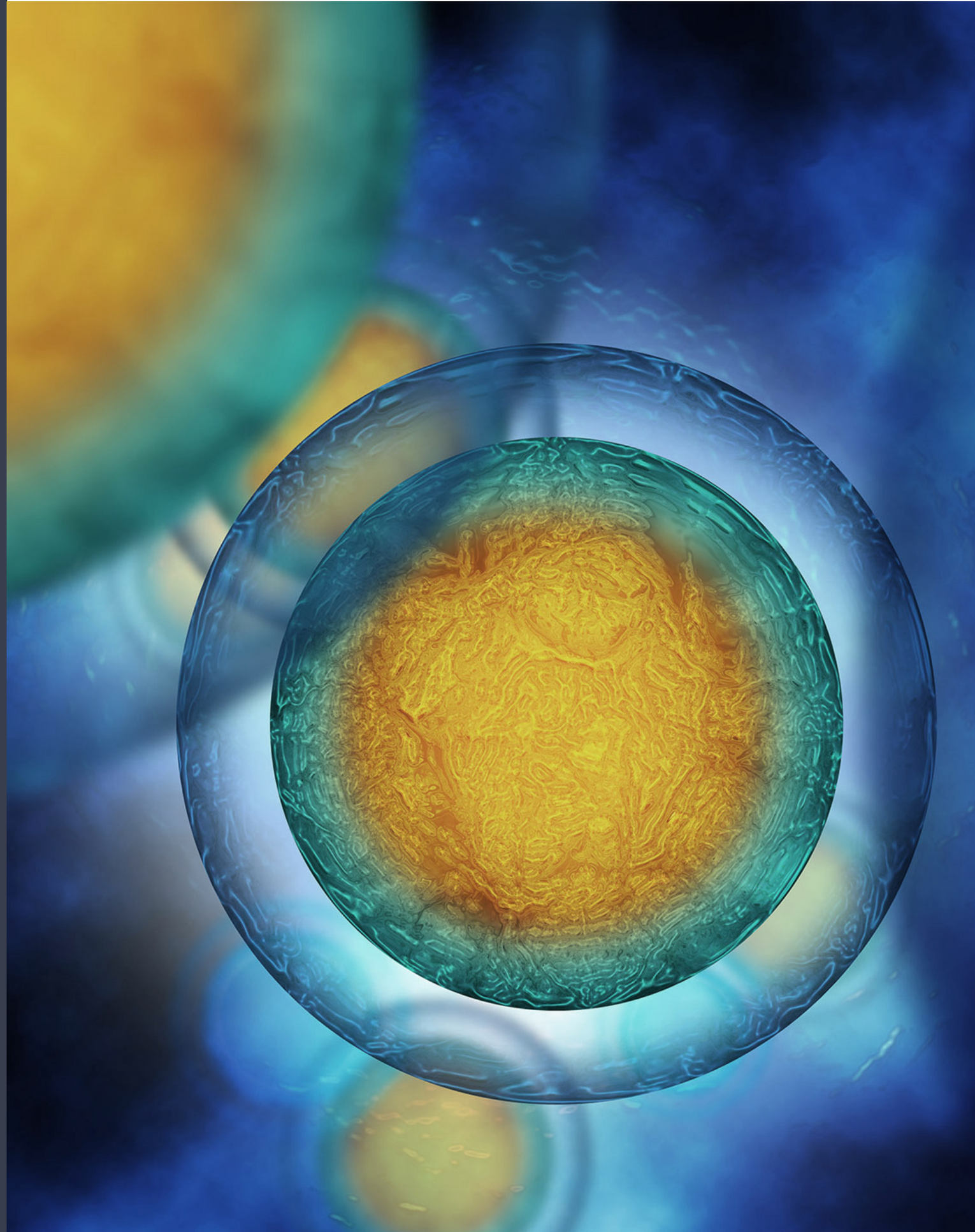
FE3CWS

MATERIAIS DE TREINO DOS PROFESSORES EM NIJMEGEN

Produção intelectual 3 do projeto ERASMUS + 2017-1-SK01-KA203-035402



Algumas palavras sobre o

CONTEÚDO

- **Um tópico muito poderoso relacionado à composição, compreensão e correção do software**
- **Disponível em 7 idiomas: inglês, húngaro, eslovaco, croata, romeno, búlgaro e português**

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

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

Segundo Material de Treinamento para Professores - "Programação Funcional no Laboratório de Novos Dispositivos".

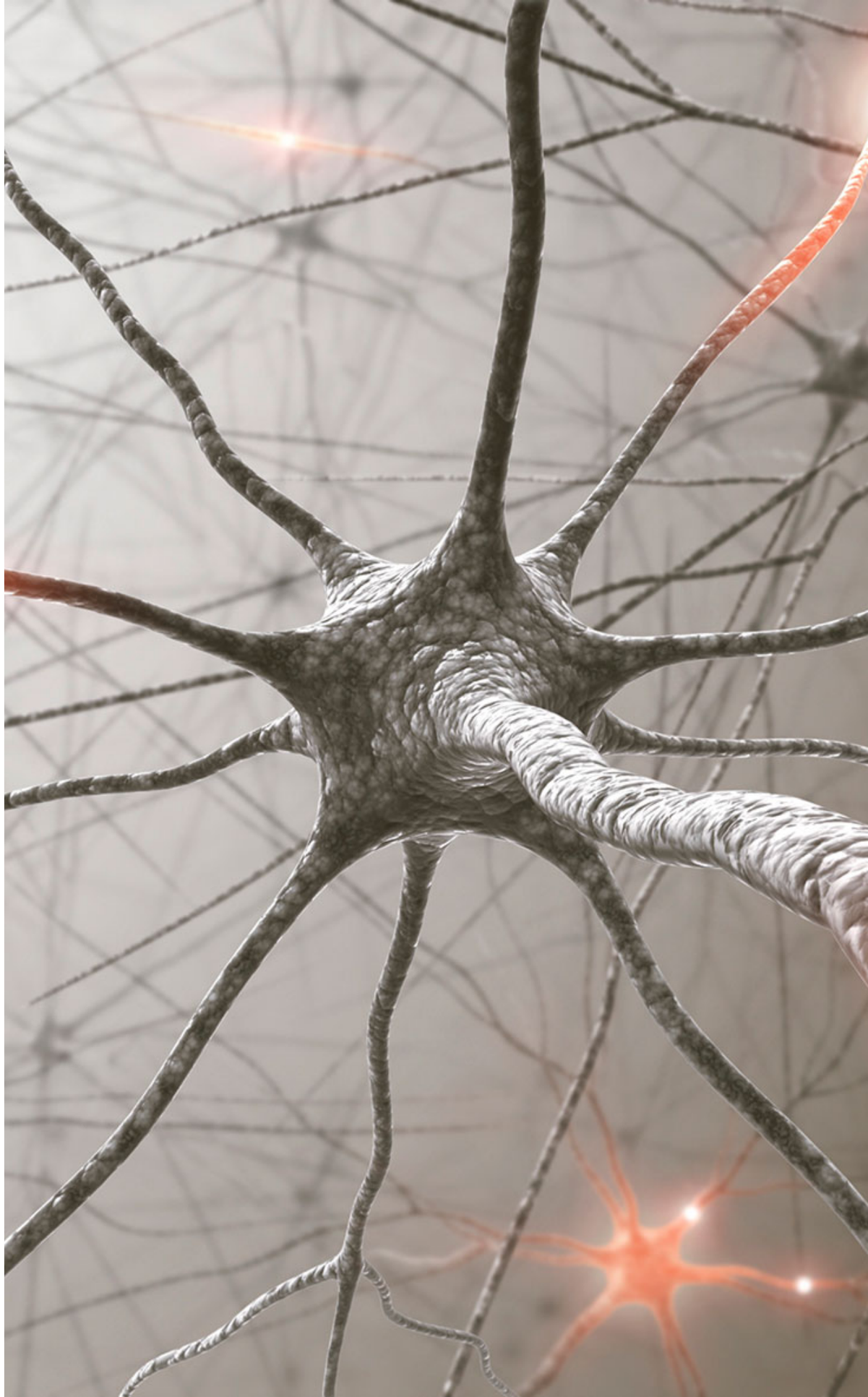
DESCRIÇÃO

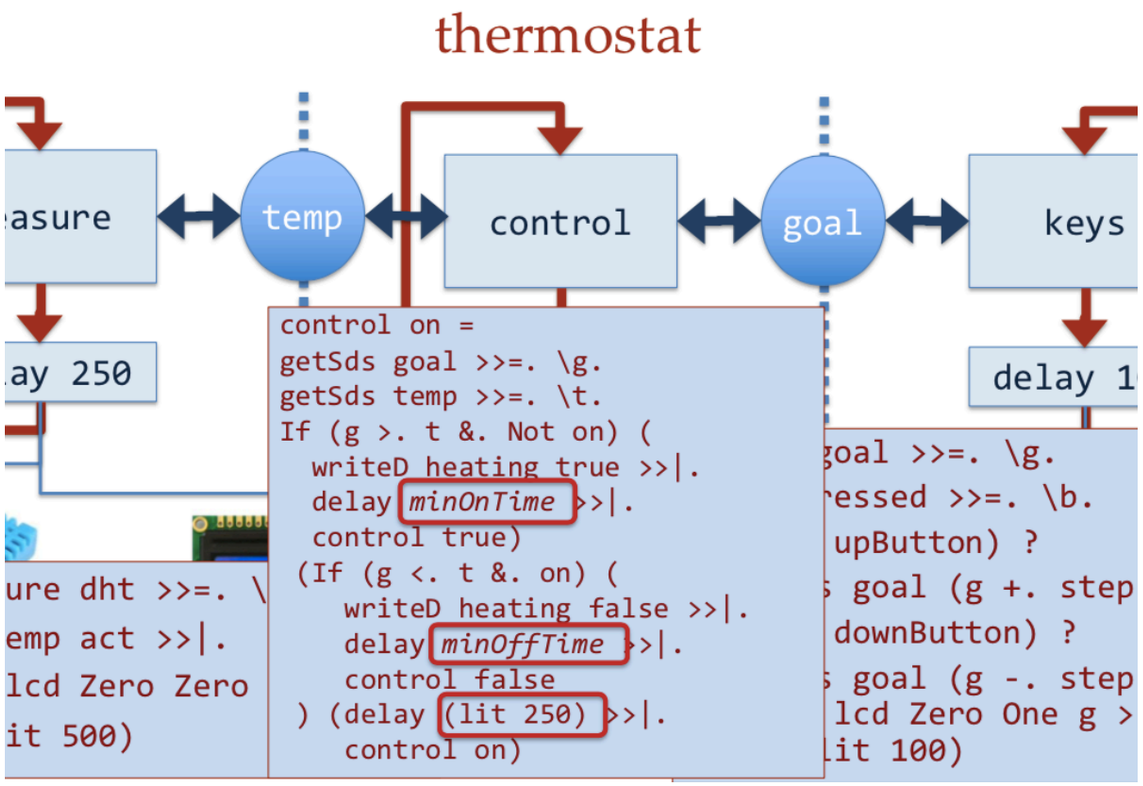
A formação de professores do 3COWS em Nijmegen, na Holanda, abrange cinco dias de ensino intensivo e cooperação na Universidade Radboud Nijmegen. Os objetivos desta formação de professores no projeto FE3CWS eram aproximar os participantes do estado da arte atual da programação funcional e da programação orientada a tarefas, TOP. Os tópicos abordados abordam:

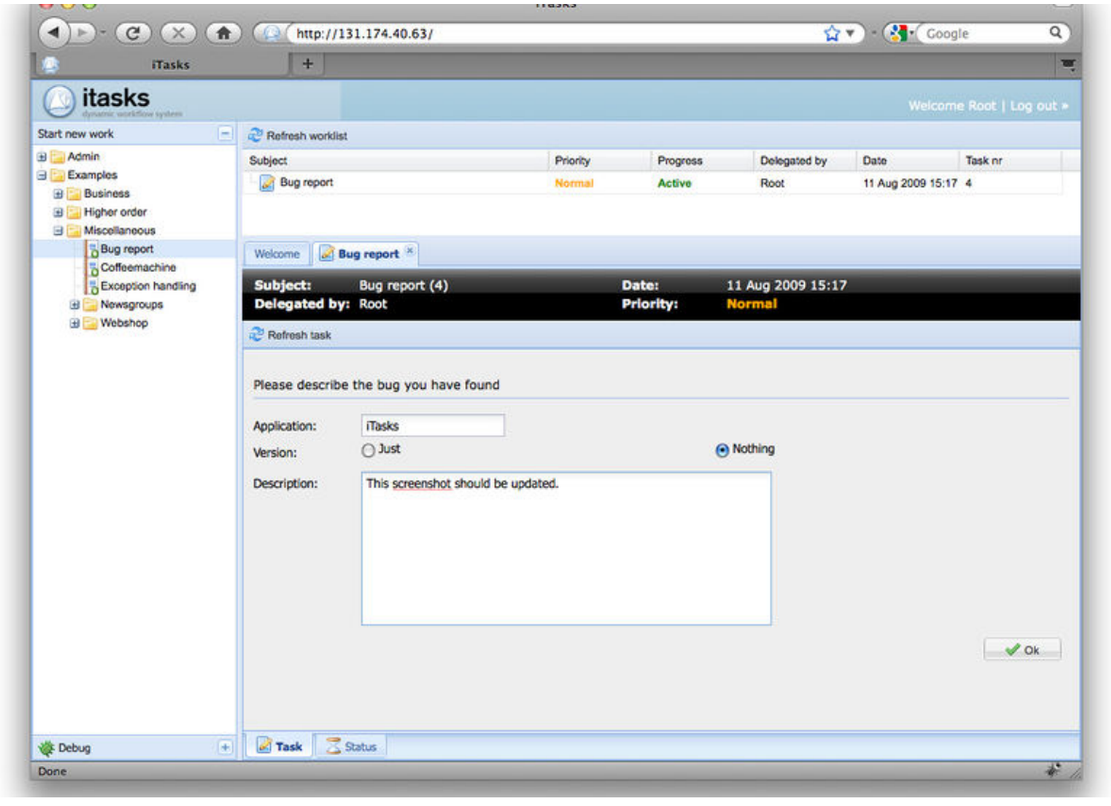
- Resumo da programação funcional pura no Clean. Consulte https://clean.cs.ru.nl/Clean.

- Programação genérica: como definir manipulações que funcionam para qualquer tipo de dados de primeira ordem, mesmo para tipos que ainda não estão definidos.

- TOP para especificar a cooperação de seres humanos e tarefas automatizadas para alcançar objetivos. O sistema iTask suporta e monitora a execução de tarefas. Consulte https://clean.cs.ru.nl/ITasks. A interação com os usuários é feita principalmente por meio de editores baseados na Web que são gerados a partir dos tipos usados nas definições de tarefas de alto nível.







O formalismo TOP contém um pequeno número de tarefas básicas flexíveis (como editores da Web e controle de periféricos de IoT) e um conjunto de combinadores para a composição paralela e seqüencial de tarefas.

TOP para controlar a Internet das Coisas, IoT.

O TOP é muito adequado para controlar a IoT e libera os desenvolvedores do fardo de muitas linguagens de programação, protocolos, interfaces e sua interoperabilidade. Para lidar com as limitações limitadas de energia e energia de computação dos dispositivos IoT, é criada uma linguagem específica de domínio especial, DSL, chamada mTasks, para programar os dispositivos IoT. Os sistemas iTask e mTask trabalham perfeitamente juntos para conectar o mundo das tarefas baseadas na Web às pequenas tarefas executadas na IoT.

O ensino desses tópicos foi realizado por uma combinação de tutoriais interativos e programação prática com os participantes. Os tópicos desta formação de professores estão no coração do projeto FE3CWS.

Os programas concisos em um alto nível de abstração contribuem diretamente para sua compreensibilidade. Os combinadores de tarefas são uma ilustração muito útil da composibilidade no software. O sistema de tipo estático da linguagem pura e funcional do host Clean garante que erros do tipo em tempo de execução não possam ocorrer. Juntamente com a notação concisa nessas DSLs, isso contribui para a correção dos programas TOP.

LINKS PARA DOWNLOAD

https://fe3cws.kpi.fei.tuke.sk/O3PT.html

LEDS PISCANDO = OLÁ MUNDO

Código Arduino

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

Código mTask (Clean)

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

DOIS EXEMPLOS FATORIAIS

```
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))}
```

LEDS PISCANDO DA MANEIRA FUNCIONAL NO MTASK

```
1  module blink
2
3  import StdEnv, iTasks
4
5  import Interpret
6  import Interpret.Device.TCP
7
8  Start :: *World → *World
9  Start w = doTasks main w
10
11 main :: Task Bool
12 main = enterDevice >>= λspec→withDevice spec
13    λdev→liftmTask blink dev -|| viewDevice dev
14 where
15    blink :: Main (MTask v Bool) | mtask v
16    blink
17       = fun λblink=(λx→
18                delay (lit 500)
19          >>|. writeD d4 x
20          >>=. blink o Not)
21       In {main=blink (lit True)}
```

INTERATIVO PISCANDO

```
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)}
```

UM PROGRAMA MTASK INTERATIVO PARA INTERAGIR COM A MATRIZ DE LED

```
1  :: Ledstatus = {x :: Int, y :: Int, status :: Bool}
2  derive class iTask Ledstatus
3
4  main = enterDevice >>= λspec→withDevice spec
5     λdev→ viewDevice dev >^*
6        [OnAction (Action "Toggle") (always (
7           enterInformation () [] >>= λs→liftmTask (toggle s) dev
8              >>~ viewInformation "done" []))
9        ,OnAction (Action "Clear") (always (
10          liftmTask clear dev
11          >>~ viewInformation "done" []))
12       ] @! ()
13 where
14    dot lm s = LMDot lm (lit s.x) (lit s.y) (lit s.status)
15
16    toggle :: Ledstatus → Main (MTask v ()) | mtask, LEDMatrix v
17    toggle s = ledmatrix D5 D7 λlm→{main=dot lm s >>|. LMDisplay lm}
18
19    clear :: Main (MTask v ()) | mtask, LEDMatrix v
20    clear = ledmatrix D5 D7 λlm→{main=LMClear lm >>|. LMDisplay lm}
```

MEDINDO TEMPERATURA E UMIDADE

```
1 main = enterDevice >>= λspec → withDevice spec
2    λdev → liftmTask temp dev >&> viewSharedInformation () [ViewAs templens]
3 where
4    templens = maybe (0.0, 0.0) λ(t, h) → (toReal t / 10.0, toReal h / 10.0)
5
6    temp :: Main (MTask v (Int, Int)) | mtask, dht v
7    temp = DHT D4 DHT22 λdht → {main=temperature dht .&&. humidity dht}
```

Bibliografia

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.