Constraint-based Approach for an Early Inspection of the Feasibility of Cyber Physical Systems

Benny Höckner & Peter Sauer & Thilo Vörtler & Petra Hofstedt & Thomas Hinze

12. September 2013

Contents

Motivation (I)

The importance of Embedded and Cyber Physical Systems (ES/CPS) is increasing more and more:

- digital camera
- cell / smart phones
- cars
- control system
- health care
- assisted living
- ...
- ...

http://www.stw.nl/en/node/4709

Motivation (II)

Designing them can be a tough task:

- lot of different hardware components with different configuration modes are connected to each other
- components heavily influence each other
- trade-off between rich functionality, lifetime and correct design

The following questions may arise:

- Is the system constructable regarding to the requirements?
- How much power will consume the system in use or how long will last the energy source?
- Is it possible to add further components and using which possible configurations?
- ...

Idea and Advantages

In the early stage, normally only partial information is available.

Idea

Constraints are well suited to deal with incomplete information.
Thus we use them to model systems and to obtain valid systems.

Idea and Advantages

In the early stage, normally only partial information is available.

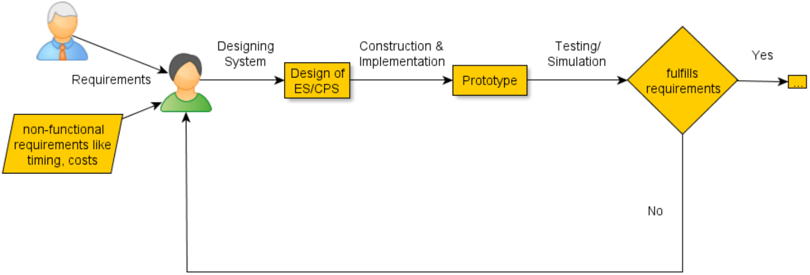
Idea

Constraints are well suited to deal with incomplete information.
Thus we use them to model systems and to obtain valid systems.

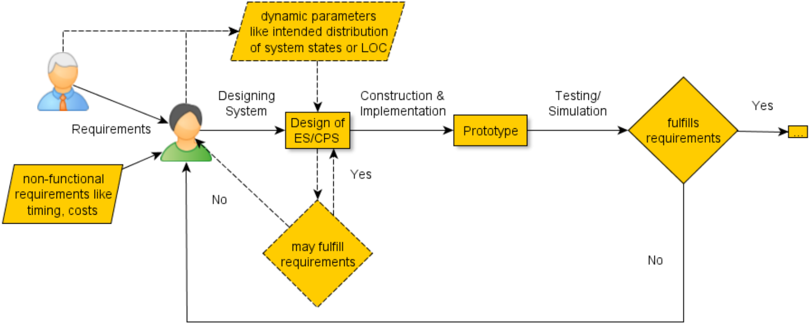
Advantages

- System has *not* to be specified completely, but we nevertheless obtain some answers and also retrieve remaining configuration possibilities.
- Easy to model, since every requirement can be encoded by one or more constraints independent from other requirements.
- Reduce prototyping effort

Possibility to make constructing a ES/CPS more efficient



Possibility to make constructing a ES/CPS more efficient



Considered Systems

- For now, we focused on systems consisting of:
  - a single cpu/micro controller,
  - sensors/actuators,
  - buses
- Reason: Unknown whether the solver would be able to handle large systems.

. . . but the method is not limited to these systems.

How is a system defined?

- $CPU = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI_1, \ldots, PI_m)$
- $I = (\mathtt{I}_{\mathtt{low}}, \mathtt{I}_{\mathtt{high}})$
- $PI_i = \{Prot_1, \ldots, Prot_n\}$

How is a system defined?

- $CPU = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI_1, \ldots, PI_m)$
- $I = (\mathtt{I}_{\mathtt{low}}, \mathtt{I}_{\mathtt{high}})$
- $PI_i = \{Prot_1, \ldots, Prot_n\}$

- $BUS = (\{C_1^{BUS}, \ldots, C_p^{BUS}\}, overhead)$
- $C_i^{BUS} = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI)$

How is a system defined?

- $CPU = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI_1, \ldots, PI_m)$
- $I = (\mathtt{I}_{\mathtt{low}}, \mathtt{I}_{\mathtt{high}})$
- $PI_i = \{Prot_1, \ldots, Prot_n\}$
- $BUS = (\{C_1^{BUS}, \ldots, C_p^{BUS}\}, overhead)$
- $C_i^{BUS} = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI)$
- $Sensor = (S, \{C_1^{Sensor}, \ldots, C_q^{Sensor}\}, size)$
- $C_i^{Sensor} = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, \{Prot_1, \ldots, Prot_{k1}\})$
- $S = \{\mathtt{StateId}_1, \ldots, \mathtt{StateId}_r\}$

How is a system defined?

- $CPU = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI_1, \ldots, PI_m)$
- $I = (\mathtt{I_{low}}, \mathtt{I_{high}})$
- $PI_i = \{Prot_1, \ldots, Prot_n\}$

- $BUS = (\{C_1^{BUS}, \ldots, C_p^{BUS}\}, overhead)$
- $C_i^{BUS} = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, PI)$

- $Sensor = (S, \{C_1^{Sensor}, \ldots, C_q^{Sensor}\}, size)$
- $C_i^{Sensor} = (\mathtt{Vcc}, \mathtt{f}, I_{active}, I_{inactive}, \{Prot_1, \ldots, Prot_{k1}\})$
- $S = \{\mathtt{StateId_1}, \ldots, \mathtt{StateId_r}\}$

- $Module_i = (S, \{C_1^{Module}, \ldots, C_s^{Module}\})$
- $C_i^{Module} = (\mathtt{Vcc}, I_{active}, I_{inactive}, \{Prot_1, \ldots, Prot_{k2}\})$

Some constraints in such a system (I)

- design requirements
  - Modules can only be connected to interfaces, if they have at least one protocol in common.
  - All sensors on the same bus have to use the same protocol, which is also supported by the interface.

Some constraints in such a system (I)

- design requirements
  - Modules can only be connected to interfaces, if they have at least one protocol in common.
  - All sensors on the same bus have to use the same protocol, which is also supported by the interface.
- functional requirements
  - CPU should be able to process the data of all connected sensors.

$$t_{cpu_{active}} = (f_{sensor_1} * LOC_1 + \ldots + f_{sensor_n} * LOC_n)/f_{cpu}$$

$$0.0 \leq t_{cpu_{active}} \leq 1.0$$

Some constraints in such a system (I)

- design requirements
  - Modules can only be connected to interfaces, if they have at least one protocol in common.
  - All sensors on the same bus have to use the same protocol, which is also supported by the interface.
- functional requirements
  - CPU should be able to process the data of all connected sensors.

$$t_{cpu_{active}} = (f_{sensor_1} * LOC_1 + \ldots + f_{sensor_n} * LOC_n)/f_{cpu}$$

$$0.0 \leq t_{cpu_{active}} \leq 1.0$$

  - bus should be able to transport all data of modules/sensors

$$f_{bus} \geq f_{sensor_1} * (size_1 + overhead_{bus}) + \ldots + f_{sensor_n} * (size_n + overhead_{bus})$$

Some constraints in such a system (II)

- non-functional requirements
  - Overall power consumption should be less than or equal to c

$$P_{cpu} = t_{cpu_{active}} * Vcc_{cpu} * I_{cpu_{active}} + (1 - t_{cpu_{active}}) * Vcc_{cpu} * I_{cpu_{inactive}}$$

$$P_{system} = P_{cpu} + P_{bus} + \dots$$

$$P_{system} \leq c$$

Some constraints in such a system (II)

- non-functional requirements
  - Overall power consumption should be less than or equal to c

$$P_{cpu} = t_{cpu_{active}} * Vcc_{cpu} * I_{cpu_{active}} + (1 - t_{cpu_{active}}) * Vcc_{cpu} * I_{cpu_{inactive}}$$

$$P_{system} = P_{cpu} + P_{bus} + \dots$$

$$P_{system} \leq c$$

  - the battery should last at least d hours/days.

$$t_{life} = x \cdot V * y \cdot Ah / P_{system}$$

$$t_{life} \geq d$$

Implementation details

Solver: ECL^{i}PS^e Prolog with IC library (integer and real interval arithmetic constraints)

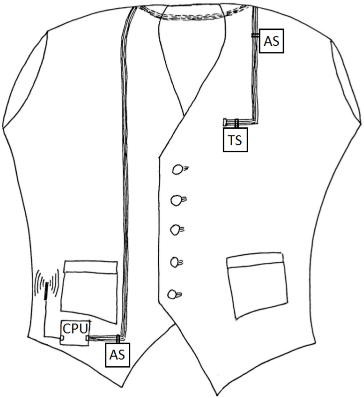
Advantages:

- Easy combination of different constraint types (finite domain and interval arithmetic)
- Especially, real interval constraints make encoding of arithmetic constraints very straight forward.

Application to an example system - The Smart Vest

Could be used for fall detection and consists of the following components:

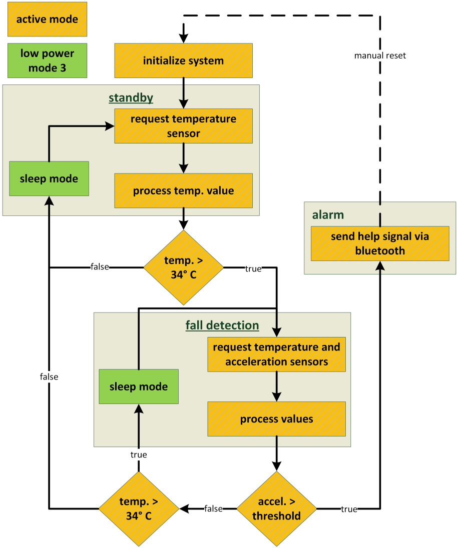
- 1 micro controller
- 1 bus
- 2 acceleration sensors
- 1 temperature sensor
- 1 Bluetooth module



Program flow

Program consists of 3 states

- vest not worn:
  - acceleration sensors inactive
  - temperature sensor active
  - Bluetooth module inactive
- vest worn:
  - all sensors active
  - Bluetooth inactive
- fall detected:
  - Bluetooth active
  - all sensors inactive



Setting of Components - Static parameters

- MSP430: (3.3 V, f, (500 μA, 600 μA), (2.6 μA, 3 μA), {UART, SPI, I^2C}, {UART, SPI})

Setting of Components - Static parameters

- MSP430: (3.3 V, f, (500 μA, 600 μA), (2.6 μA, 3 μA), {UART, SPI, I^2C}, {UART, SPI})
- 2 ADXL345: {(2.5 V, 6.25 Hz, 40 μA, 100 nA, {I^2C, SPI}), (2.5 V, 1600 Hz, 100 μA, 100 nA, {SPI}), ...}

Setting of Components - Static parameters

- MSP430: (3.3 V, f, (500 μA, 600 μA), (2.6 μA, 3 μA), {UART, SPI, I^2C}, {UART, SPI})
- 2 ADXL345: {(2.5 V, 6.25 Hz, 40 μA, 100 nA, {I^2C, SPI}), (2.5 V, 1600 Hz, 100 μA, 100 nA, {SPI}), ...}
- I^2C-bus: {(...), (...)}
- 1 SHT21: {(..., {I^2C})}
- 1 RN41: {(..., {UART})}

Setting of Components - Static parameters

- MSP430: (3.3 V, f, (500 μA, 600 μA), (2.6 μA, 3 μA), {UART, SPI, I^2C}, {UART, SPI})
- 2 ADXL345: {(2.5 V, 6.25 Hz, 40 μA, 100 nA, {I^2C, SPI}), (2.5 V, 1600 Hz, 100 μA, 100 nA, {SPI}), ...}
- I^2C-bus: {(...), (...)}
- 1 SHT21: {(..., {I^2C})}
- 1 RN41: {(..., {UART})}
- 1 3.6 V - 2 Ah battery

Setting of Components - Static parameters

- MSP430: (3.3 V, f, (500 μA, 600 μA), (2.6 μA, 3 μA), {UART, SPI, I^2C}, {UART, SPI})
- 2 ADXL345: {(2.5 V, 6.25 Hz, 40 μA, 100 nA, {I^2C, SPI}), (2.5 V, 1600 Hz, 100 μA, 100 nA, {SPI}), ...}
- I^2C-bus: {(...), (...)}
- 1 SHT21: {(..., {I^2C})}
- 1 RN41: {(..., {UART})}
- 1 3.6 V - 2 Ah battery

Additional assumptions:

- $f < 10$ MHz (cpu limit)
- ADXL345 sensors sample rate ≥ 25 Hz (to make fall detection useful)

Setting of Components - Dynamic parameters

Dynamic parameters to estimate the later system behavior.

- LOC the CPU needs for requesting and processing one sensor package:

Component	request LOC	process LOC
SHT21	1500	2000
ADXL345	2000	5000

- Intended distribution of the system states:

State	Proportion
Standby	75.00 %
Fall detection	24.99 %
Alarm	0.01 %

Constraints

- Modules can only be connected to interfaces, if they have at least one protocol in common.

Constraints

- Modules can only be connected to interfaces, if they have at least one protocol in common.
  $\Longrightarrow$ I^2C bus, as well as, SHT21 sensor should be connected to the first peripheral interface
  $\Longrightarrow$ RN41 module has to be connected to the second interface

Constraints

- Modules can only be connected to interfaces, if they have at least one protocol in common.
  $\Longrightarrow$ I^2C bus, as well as, SHT21 sensor should be connected to the first peripheral interface
  $\Longrightarrow$ RN41 module has to be connected to the second interface
- All sensors on the same bus have to use the same protocol.

Constraints

- Modules can only be connected to interfaces, if they have at least one protocol in common.
  $\Longrightarrow$ I^2C bus, as well as, SHT21 sensor should be connected to the first peripheral interface
  $\Longrightarrow$ RN41 module has to be connected to the second interface
- All sensors on the same bus have to use the same protocol.
  $\Longrightarrow$ ADXL sensors have to use I^2C protocol i.e. configurations only supporting the SPI protocol are not allowed.

Constraints

- Modules can only be connected to interfaces, if they have at least one protocol in common.
  $\Longrightarrow$ I^2C bus, as well as, SHT21 sensor should be connected to the first peripheral interface
  $\Longrightarrow$ RN41 module has to be connected to the second interface
- All sensors on the same bus have to use the same protocol.
  $\Longrightarrow$ ADXL sensors have to use I^2C protocol i.e. configurations only supporting the SPI protocol are not allowed.
- Cpu should be able to process the data of all connected sensors; State: vest worn (all three sensors active):

$$\frac{f^1_{adxl} 7{,}000 LOC + f^2_{adxl} * 7{,}000 LOC + f_{sht} * 3{,}500 LOC}{f_{cpu}}$$

$$\Longrightarrow f_{cpu} > 0.36 MHz$$

Scenario (I)

User knows:

nearly everything (has a clear idea about the system):

- $f_{cpu} = 2.4576MHz$
- both acceleration sensors are clocked to $50Hz$
- bus is clocked to $100kHz$

User does not know:

the expected system power consumption and/or system lifetime

Scenario (I)

User knows:

nearly everything (has a clear idea about the system):

- $f_{cpu} = 2.4576 MHz$
- both acceleration sensors are clocked to $50 Hz$
- bus is clocked to $100 kHz$

User does not know:

the expected system power consumption and/or system lifetime

Answer:

power consumption: 9.455 mW ... 9.516 mW

lifetime: around 31 days

Scenario (II)

User does not know:
If it is possible to add an additional acceleration sensor (behaving like the others), but also to ensure that the system has an expected lifetime around 30 days.

Scenario (II)

User does not know:
If it is possible to add an additional acceleration sensor (behaving like the others), but also to ensure that the system has an expected lifetime around 30 days.
Answer:
power consumption: nearly 10 mW
sample rate: up to 200 Hz

Scenario (III)

User does not know:
what are the highest possible sample rates for the 2 acceleration sensors while ensuring a system lifetime of about 2 weeks.

Scenario (III)

User does not know:
what are the highest possible sample rates for the 2 acceleration sensors while ensuring a system lifetime of about 2 weeks.
Answer:
sample rates: 800 Hz and 400 Hz
lifetime: around 20 days (nearly 3 weeks)

Summary

- constraints are well suited to model conjunctions between components in an intuitive way, e.g. to . . .
  - predict power consumption
  - predict system lifetime
  - check feasibility
- performance of the calculations are very promising
- system optimization

Future work

- Extend the area of application
  - Prolog code in general is hard to read/develop/maintain, and
  - current implementation is very special purpose

  $\Longrightarrow$ Model-Driven-Development approach, where the user can specify his personal constraints for his concrete model, and the corresponding Prolog code is generated.
- Modelling the components through agents $\Longrightarrow$ running first simulations only using the design

Thank you!

Please do not hesitate to ask any question!

- Benny Höckner
- Peter Sauer
- Thilo Vörtler
- Petra Hofstedt
- Thomas Hinze

E-Mail:
benny.hoeckner@tu-cottbus.de