Università degli Studi di Trento
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Ingegneria e Scienza dell'Informazione

Requirements-Driven Qualitative Adaptation

Vítor E. Silva Souza

vitorsouza@disi.unitn.it
http://disi.unitn.it/~vitorsouza

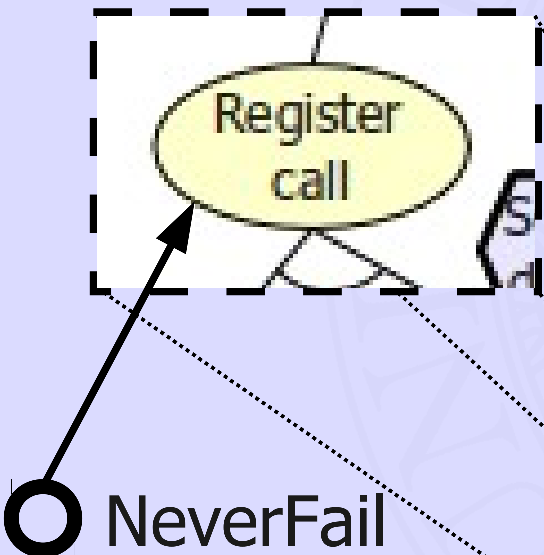
License to use, adapt and distribute

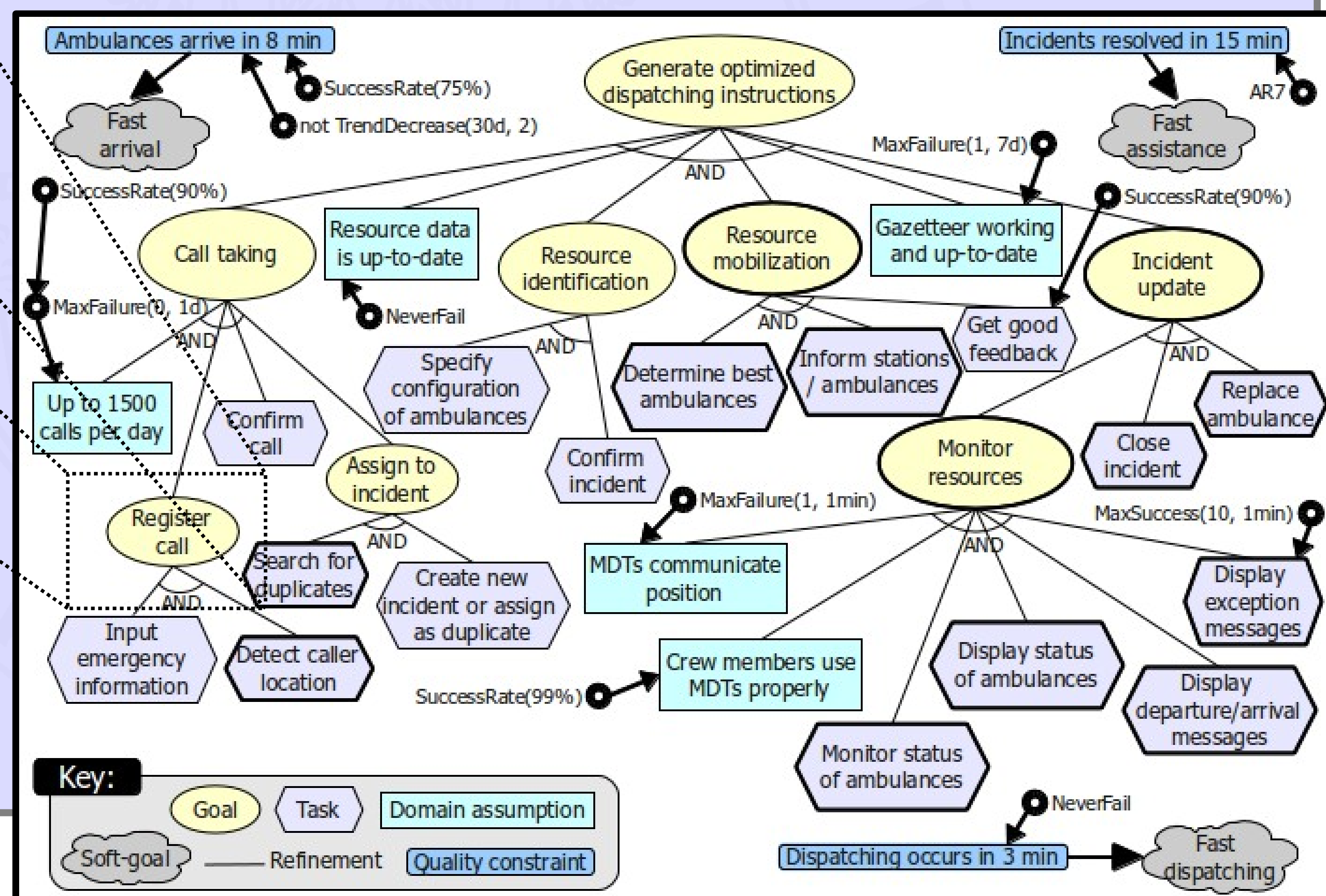
What I will present today...

- New idea in the context of my PhD research;
  - Awareness Requirements (AwReqs)...
  - System Identification...
  - Evolution Requirements (EvoReqs)...
  - Qualitative Adaptation!
- Working with John and Alexei;
- Paper to be submitted to RE 2012 (deadline March 5th).

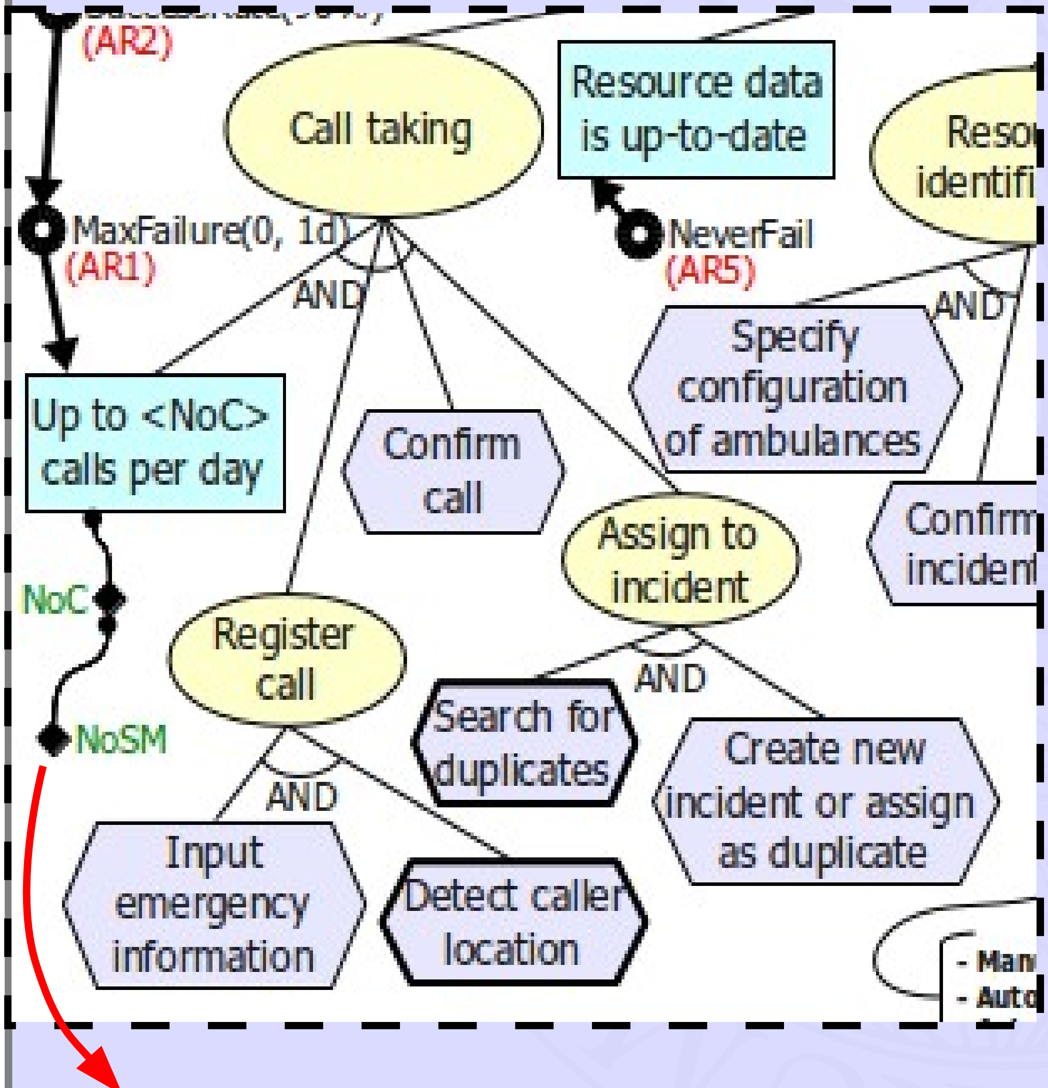
Background: AwReqs

- AwReqs determine the critical requirements. We should be aware of their success/failure;
  - But: any indicator can be used with this proposal...





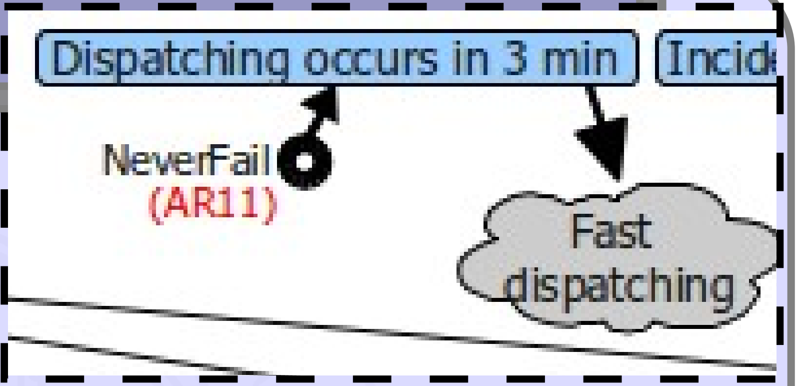
Background: SysId



$\Delta(AR1 / NoSM)[0, MaxSM] > 0$

- System Identification:
  - Indicator/parameter qualitative relations;
  - Information to reason over during reconfig.
- Extensions:
  - New attributes for parameters and indicators;
  - Prioritized indicators.

Example



- Ambulance Dispatch System
  - Softgoal: *Fast dispatching*
  - QC: *Dispatching occurs in 3 min*
  - AwReq *AR11*: **`NeverFail(Q_Dispatch3min)`**
  - Parameters:
    - **`NoSM`**: *Number of Staff Members working* ↑
    - **`LoA`**: *Level of Automation* ↑
    - **`MST`**: *Minimum Search Time* ↓
    - **`VP1`** @ goal *Provide route assistance* ↓
    - **`VP2`** @ goal *Map retrieval* ↓
    - **`VP3`** @ goal *Obtain map info manually* ↓
    - **`VP4`** @ goal *Update position of engaged ambulances* ↓

SISO, MISO, MIMO, ...

- Control System inputs/outputs in our case:
  - Inputs = parameter values;
  - Outputs = state of indicators (AwReqs).
- Levels of complexity:
  - Single Input, Single Output;
  - Multiple Inputs, Single Output;
  - Multiple Inputs, Multiple Outputs.

Simple control systems, e.g., temperature control

Vast majority of information systems that we would like to make adaptive!

In Control Theory...

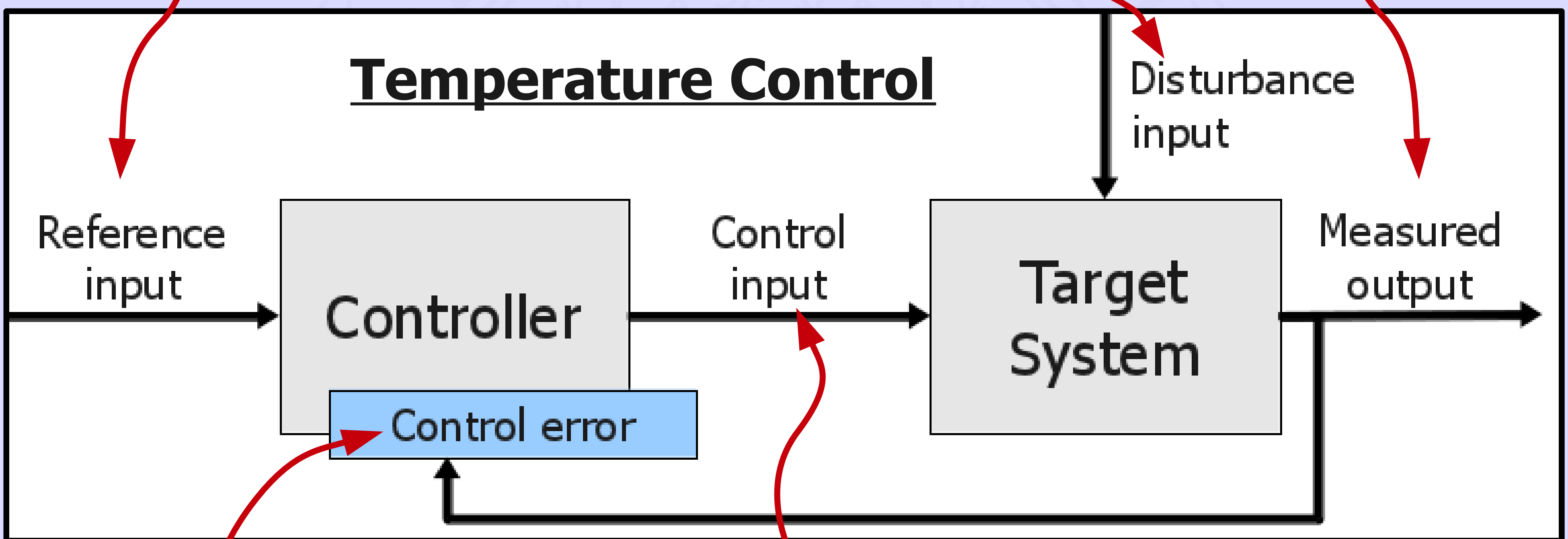
- MIMO systems:
  - State Space Representation;
  - Linear Quadratic Regulator;
  - Full State Feedback;
- SISO systems:
  - PID controller (much simpler!);
- Our research:
  - Avoid excessive complexity for the req. engineer;
  - Current proposal inspired by PID, but extensible.

PID Controller 101

Desired temperature (T_I)

Actual temperature (T_O)

Outside weather



$E = T_I - T_O$

if (E < 0) then start heating
else start cooling

?

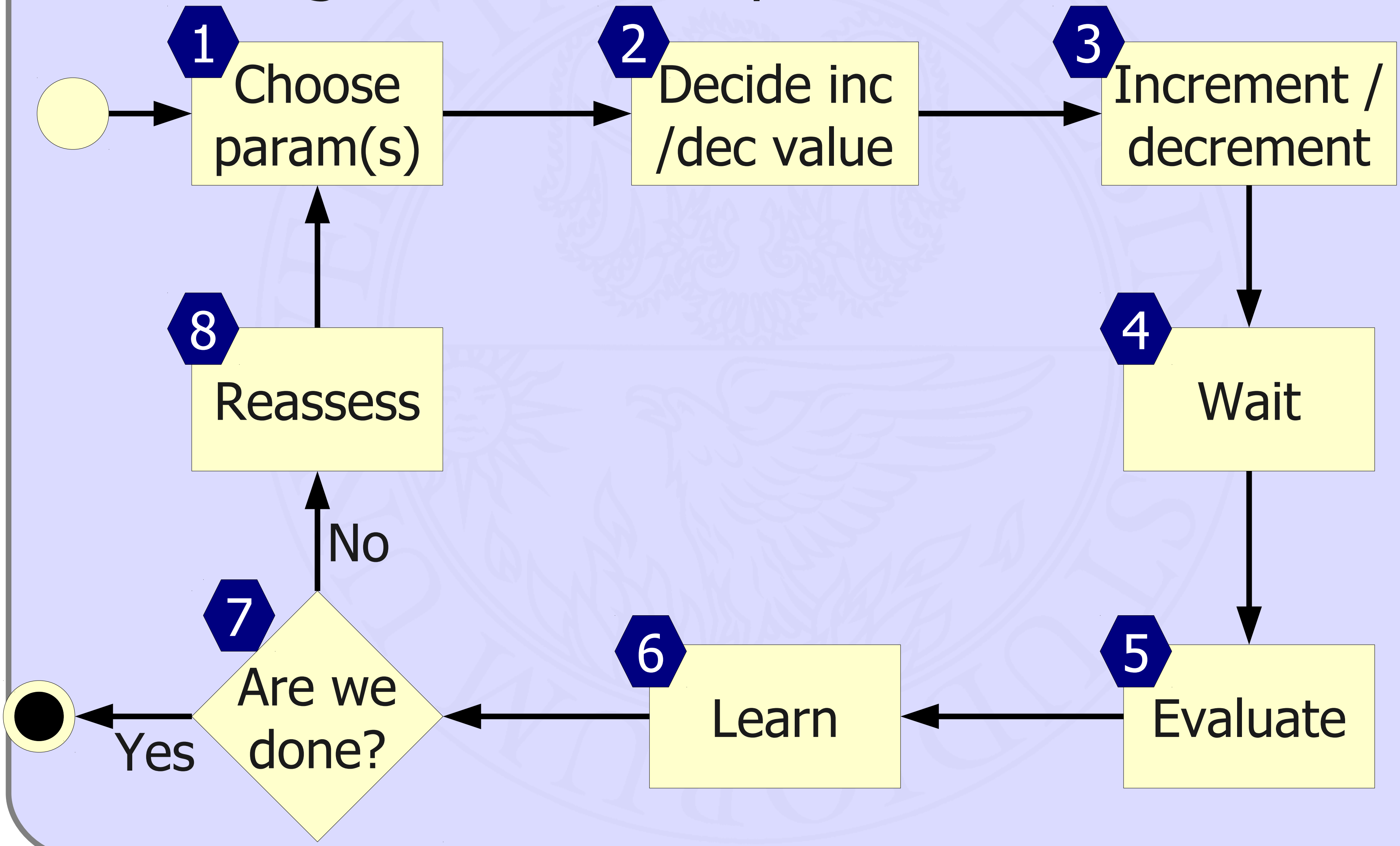
With what power?

PID Controller 101

- Proportional: control input proportional to error;
  - E = -10 degrees → heating power = 10 * K units;
  - Tune K to avoid too conservative or too aggressive;
- Integral: proportional to sum of past errors;
  - Motivation: P can't handle disturbances;
  - Ex.: proportional heating canceled by window open;
- Differential: adjusts input given rate of change;
  - P+I is most common, D is for tuning (with care!);
  - Rate of change predicts if P+I will take too long or will overshoot too quickly.

Proposal: qualitative adaptation

- Reconfigure based on qualitative information:



1 – Choose [1 or more] parameter(s)

- Random / shuffle;
- Based on their effect (random/shuffle if unknown):
  - |Δ(AR11 / MST)| > |Δ(AR11 / VP2)| > |Δ(AR11 / LoA)| ...
- Aggregate more than 1 param (MISO):
  - Use all available;
  - Choose some (randomly, priority-based).
- Example: **`AR11`** failed (dispatch takes > 3 min), choose **`MST`** to fix it.

2 – Decide the inc/dec value

- Each param specifies basic unit of inc/dec `U`;
  - Difficult to guess, so this is mandatory!
- Each AwReq specifies coefficient of inc/dec `K`;
  - Not mandatory, default `K = 1`;
- Increment/decrement value: `V = K * U`;
- Example: $U_{MST} = 10s$, $K_{AR11} = 2 \rightarrow V = 20s$.

If we know:

$\Delta(I / P) > C1$
$\Delta(I / P) < C2$
$C1, C2 \neq 0$
$C1 < C2$

Then V could be estimated by how much the indicator needs to be changed (i.e., the control error – more on this later).

3 – Increment / decrement

- Change at class or instance level?
- Default is class level;
- Example:
  - Initially **`MST = 60s`**;
  - Decrement **`MST`** in 20s, now **`MST = 40s`**.

4 – Wait for some time

- Default is to wait until the next AwReq check:
  - Example: for AR11 it's the next dispatch;
- Some parameters can specify maturation time:
  - Time it takes for param change to affect indicator;
  - If specified, choose the higher between these two;
  - Example: for **NoSM**, it might take 2 weeks to hire and train a new staff member.

5 – Evaluate the indicator

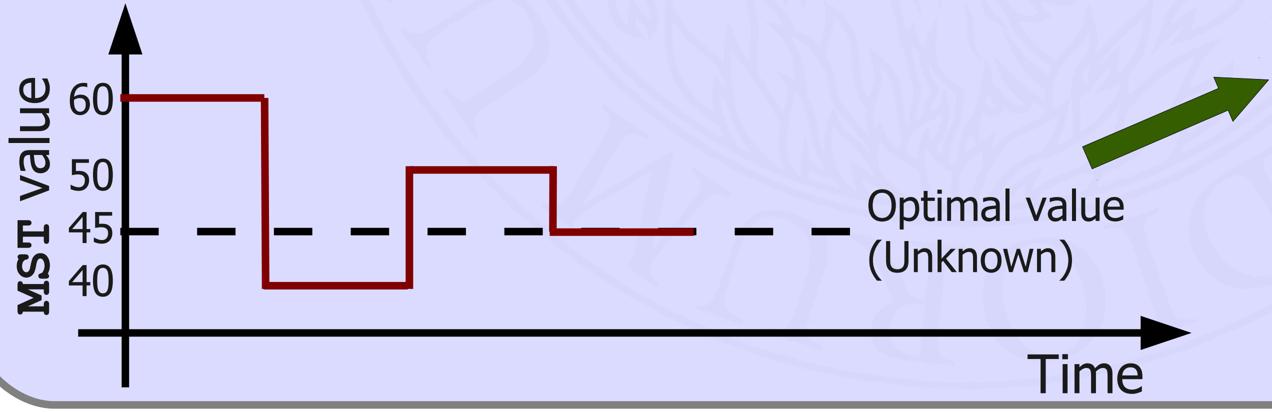
- Is the problem solved?
- Note that:
  - AwReqs are Boolean (they either failed or not);
  - Later we will consider calculating the AwReq error.
- Example:
  - With **`MST = 40s`**, did the next dispatch take less than 3 minutes? (i.e., did **`AR11`** succeed?)

6 – Learn from the outcome

- Evolution of the adaptation procedure based on history (adaptive control);
  - Better precision (qualitative → quantitative);
  - Fix domain expert mistakes (indicator/parameter relations, priorities, coefficients, etc.);
  - Find new relations;
  - Etc.
- "Long-term learning";
- Out of scope of this work (i.e., future work).

7 – Decide whether to stop or iterate

- Problem fixed, stop. Otherwise, iterate;
  - Example: `AR11` succeeded, leave `MST = 40s`;
  - However, lower `MST` values can make other AwReqs fail, so it might be incremented later...
- Look for optimal value;
  - Example: maybe 20s is overshooting and the problem would have been solved with `MST = 45s`;



Here, V is halved at each iteration. There should be a stop criteria.

Some stop criteria scenarios…

- Global or indicator-specific “maximum number of oscillations” attribute;
- Param-specific “minimum change value” (halve it at each oscillation until lower than minimum);
- Use all parameters:
  - Change params one by one until overshoot, go back to last non-overshot value;
  - For the last param (the one with least effect), leave the overshoot (in order to solve the problem).

8 – Reassess the chosen strategies

- Strategies were used to:
  - Choose one or more parameters;
  - Determine the value to increment/decrement them;
  - Decide whether to stop or iterate;
- Keep the strategies or change them?
- "Short-term learning";
- Example:
  - We used just one parameter, next use two;
  - We chose parameters randomly, now choose them based on their effect.

Can we calculate the AwReq error?

- The PID controller is based on the control error;
- AwReqs are Boolean, could they be numeric?
  - Delta AwReqs: calculate difference b/w desired and current value of domain property;
    - Example: **`AR7`** – Incidents resolved in 15 min;
  - Aggregate or "never fail" AwReqs: success rate;
    - Example: **`AR3`** – **`SuccessRate(AmbArv8min, 75%)`**.
  - Trend AwReqs: difference between last two rates;
    - Example: success rate should not decrease twice.
- With the AwReq error we can directly use a PID controller, if desired.

What about multiple outputs?

- Previous examples consider single output;
  - An AwReq fix might trigger another, it could go well or there could be infinite loops (instability)!
- Define new strategies that consider MOs:
  - Param choice: choose the ones that have least negative impact on other AwReqs;
  - Inc/dec value: also consider negative impact;
  - Evaluate: not only the failed AwReq, but others;
  - Stop/iterate: consider if other AwReqs were affected. E.g., backtrack if they were.
- Priorities among indicators is fundamental!

What strategy is best?

- Main contribution of this paper: the framework, which can be extended with new strategies;
- Such analysis is also future work:
  - Analyze SASO properties for control systems: stability, accuracy, settling time and overshoot;
  - Experiment with strategies to find out the best ones for particular problems;
  - Many other questions can be raised from this proposal...

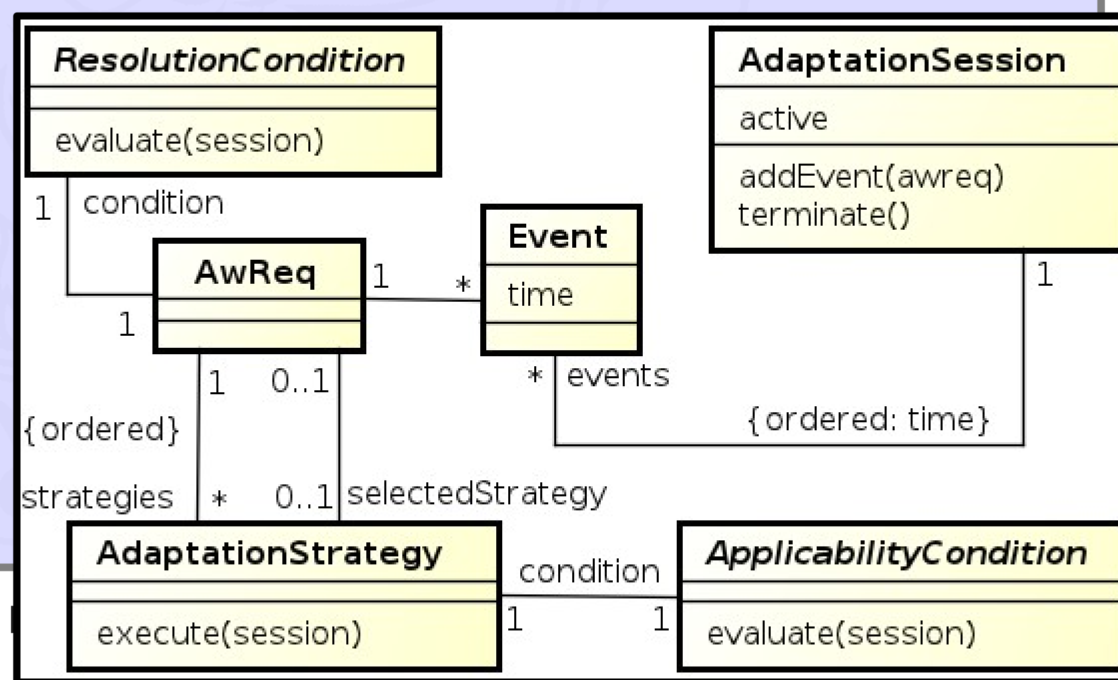
Contribution to requirements models

Element	Used by
Awareness Requirements as indicators	Monitoring framework
Control variables and variation points (parameters)	Parameter Choice
Differential relations between parameters and indicators	Parameter Choice, Parameter Change
Differential relations' refinements (comparison, cummulative effect)	Parameter Choice
Adaptation strategies (and their applicability/resolution conditions) associated to AwReqs	Adaptation framework (next slide)
AwReqs' increment coefficients	Value Calculation
Parameters' units of increment	Value Calculation
AwReqs' priorities	Parameter Choice
Relations' maturation times	Waiting
Global or AwReqs' max number of oscillations	Resolution Check
Parameters' minimum change values	Resolution Check
Parameters' halving factors	Parameter Change

Implementation

- Zanshin (JP: “a state of total awareness”)
  - Implemented the ECA-based adaptation process for the SEAMS 2012 paper;
  - The control process presented here is integrated as an adaptation strategy named Qualia;
  - https://github.com/vitorsouza/Zanshin

残心



Thank You! Questions? Feedback?

Acknowledgment:

The research reported in this presentation was partially funded by the ERC advanced grant 267856 "Lucretius: Foundations for Software Evolution", unfolding during the period of April 2011 - March 2016.

Università degli Studi di Trento
Facoltà di Scienze Matematiche, Fisiche e Naturali
Dipartimento di Ingegneria e Scienza dell'Informazione

Requirements-Driven Qualitative Adaptation

Vítor E. Silva Souza

vitorsouza@disi.unitn.it
http://disi.unitn.it/~vitorsouza