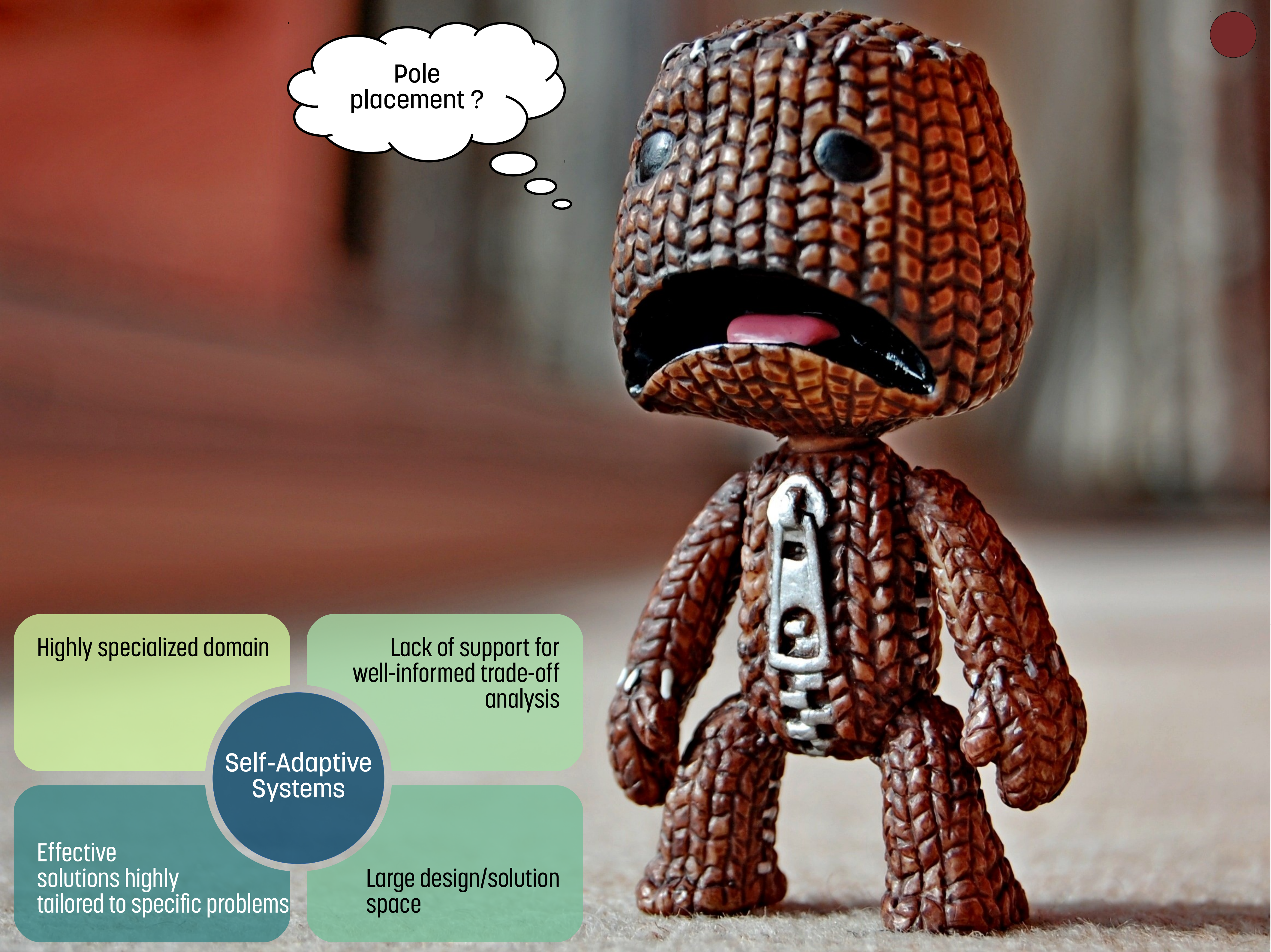
Toward Systematic Conveying of Architecture Design Knowledge for Self-Adaptive Systems

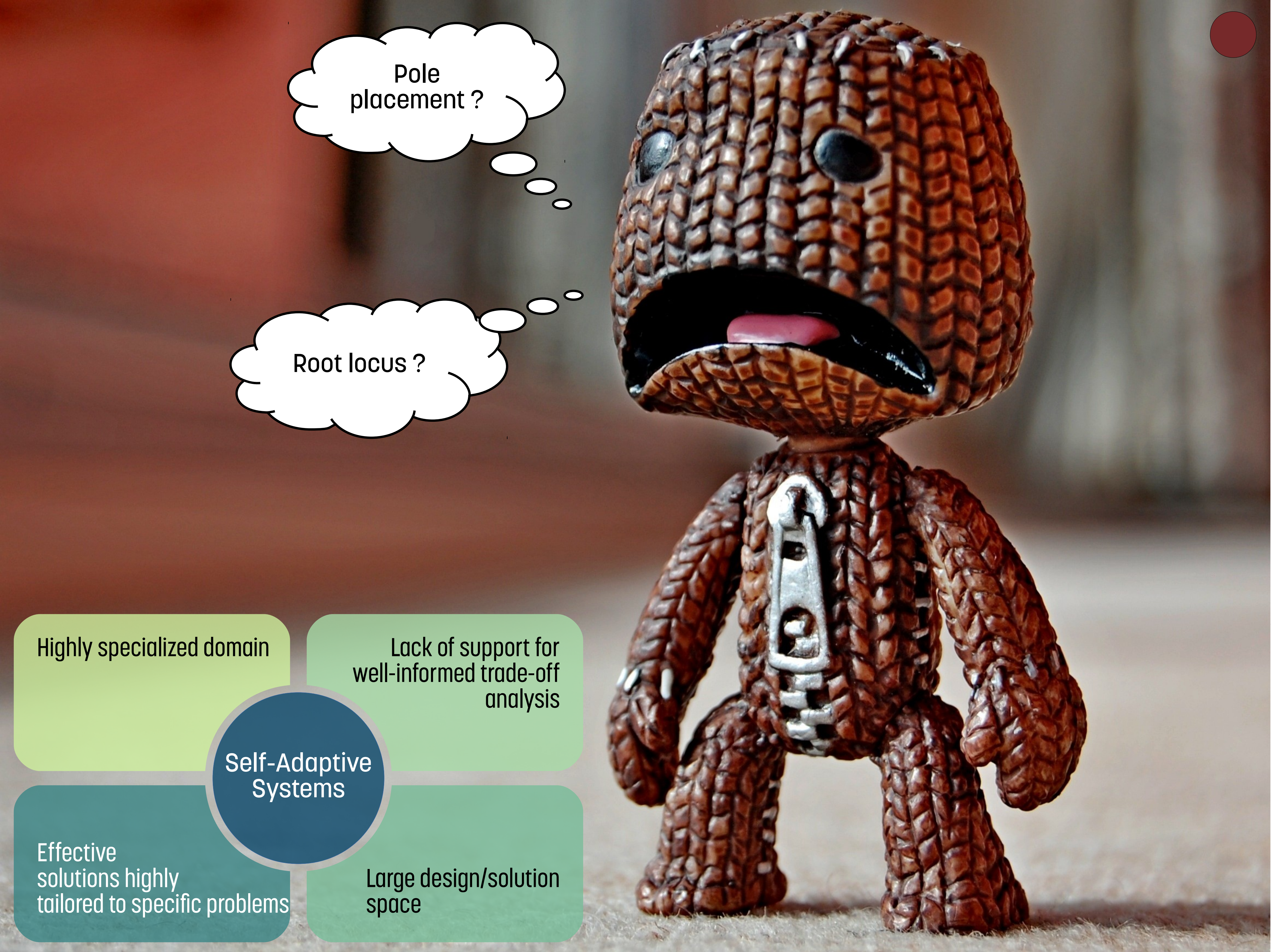
Sandro S. Andrade

GSORT Distributed Systems Group
Federal Institute of Education, Science and Technology of Bahia (IFBa)
Salvador - BA - Brazil
sandros@ufba.br

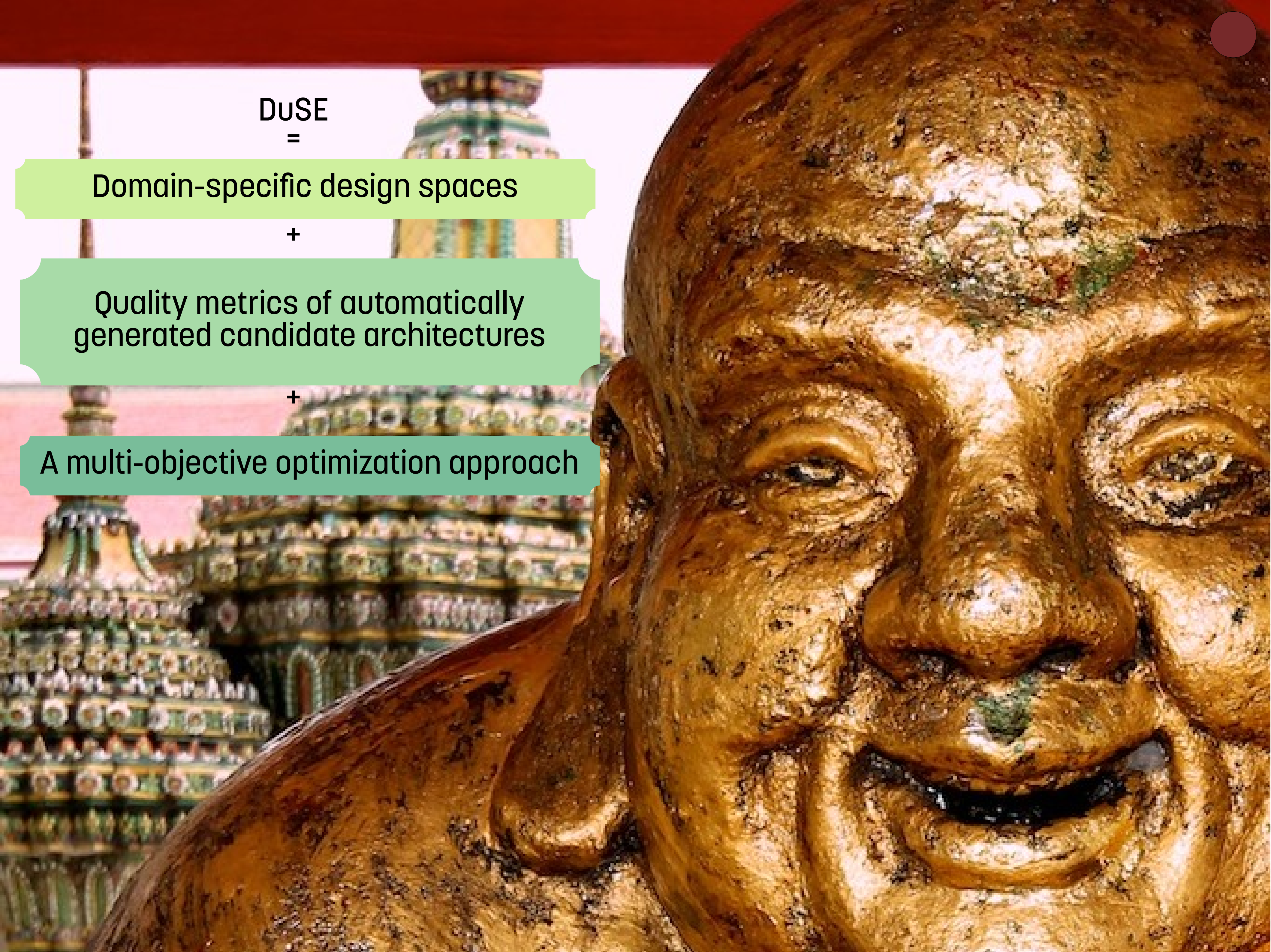
Raimundo José de A. Macêdo

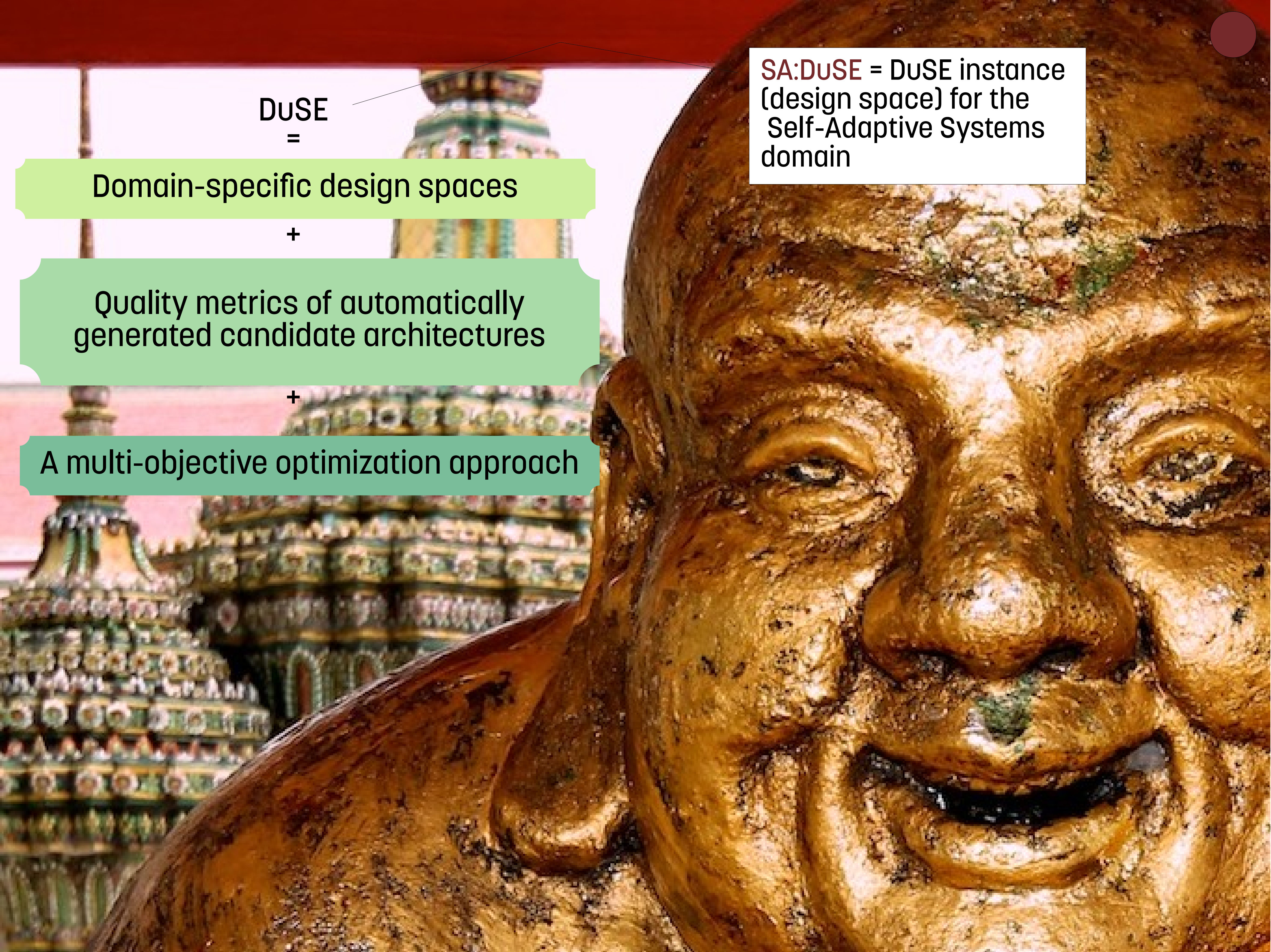
Distributed Systems Laboratory (LaSiD)
Department of Computer Science
Federal University of Bahia
Salvador - BA - Brazil
macedo@ufba.br

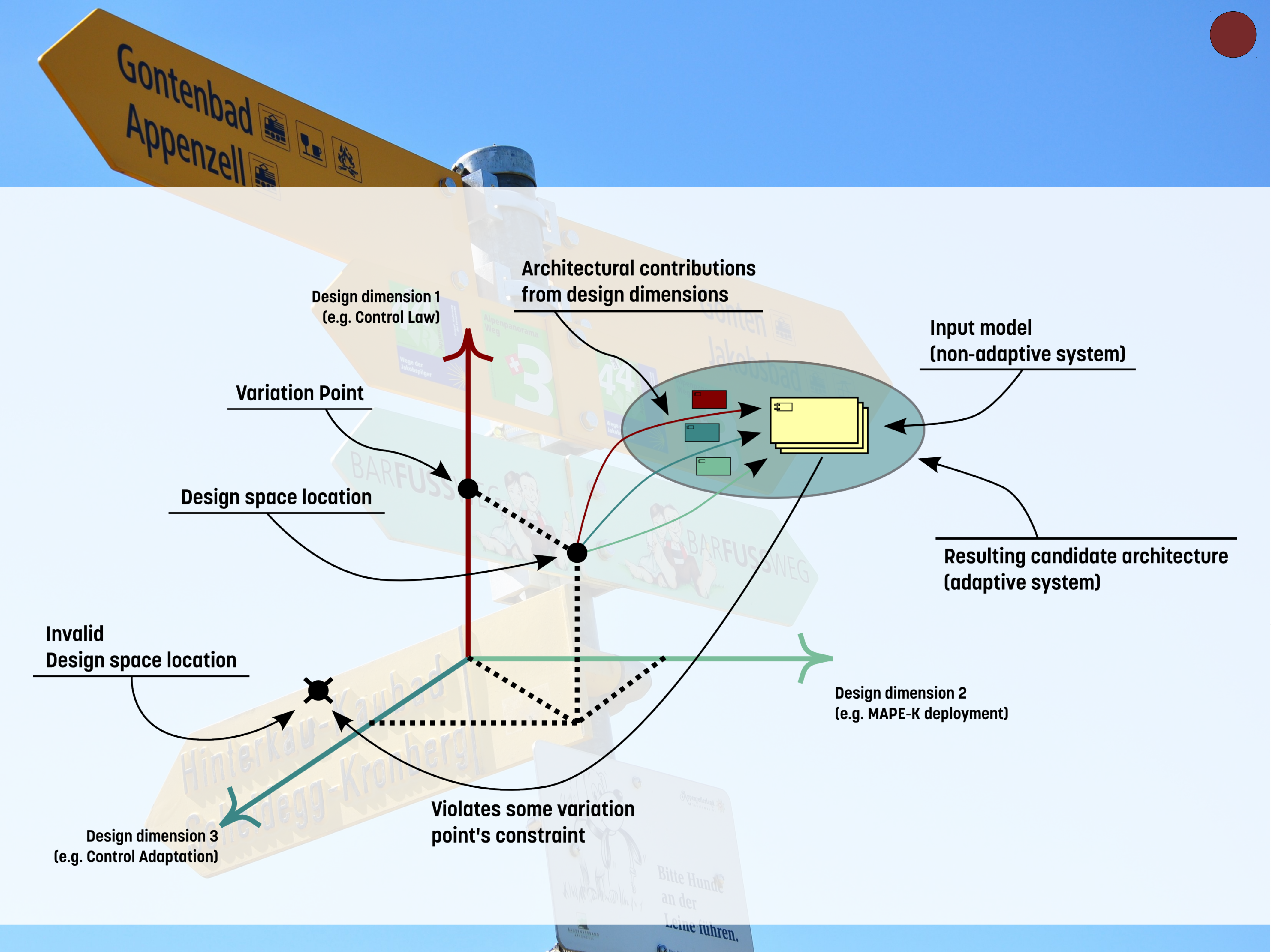

Pole placement ?
Highly specialized domain
Lack of support for well-informed trade-off analysis
Self-Adaptive Systems
Effective solutions highly tailored to specific problems
Large design/solution space


Pole placement ?
Root locus ?
Highly specialized domain
Lack of support for well-informed trade-off analysis
Self-Adaptive Systems
Effective solutions highly tailored to specific problems
Large design/solution space

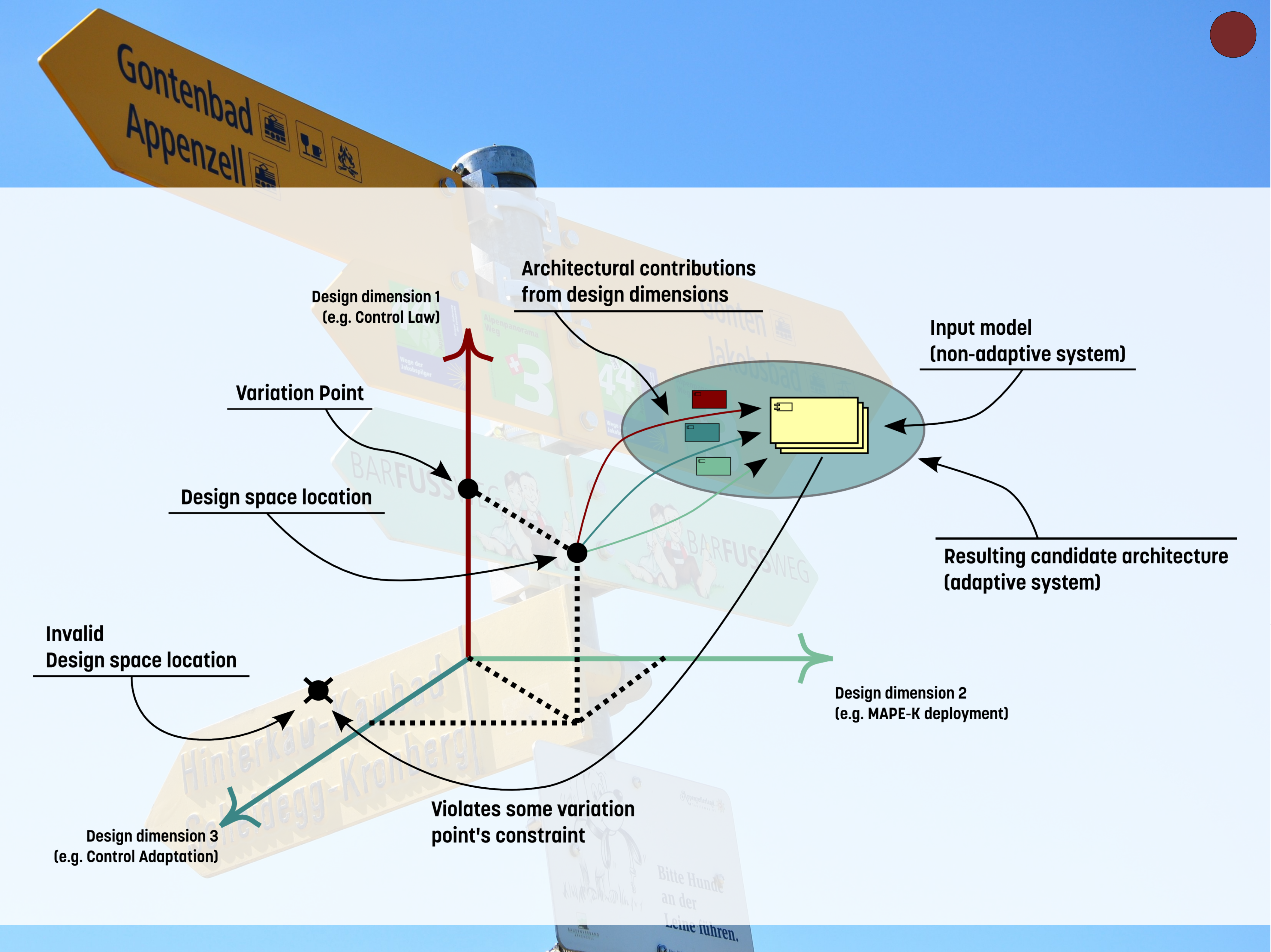

What am I supposed to do with all these stuff ?
Model-Based Software Design & Development
Formal Reference Models / Frameworks
Designing Self-Adaptive Systems
Architecture Design Handbooks
Reference Architectures Architectural Styles


DuSE
=
Domain-specific design spaces
+
Quality metrics of automatically generated candidate architectures
+
A multi-objective optimization approach

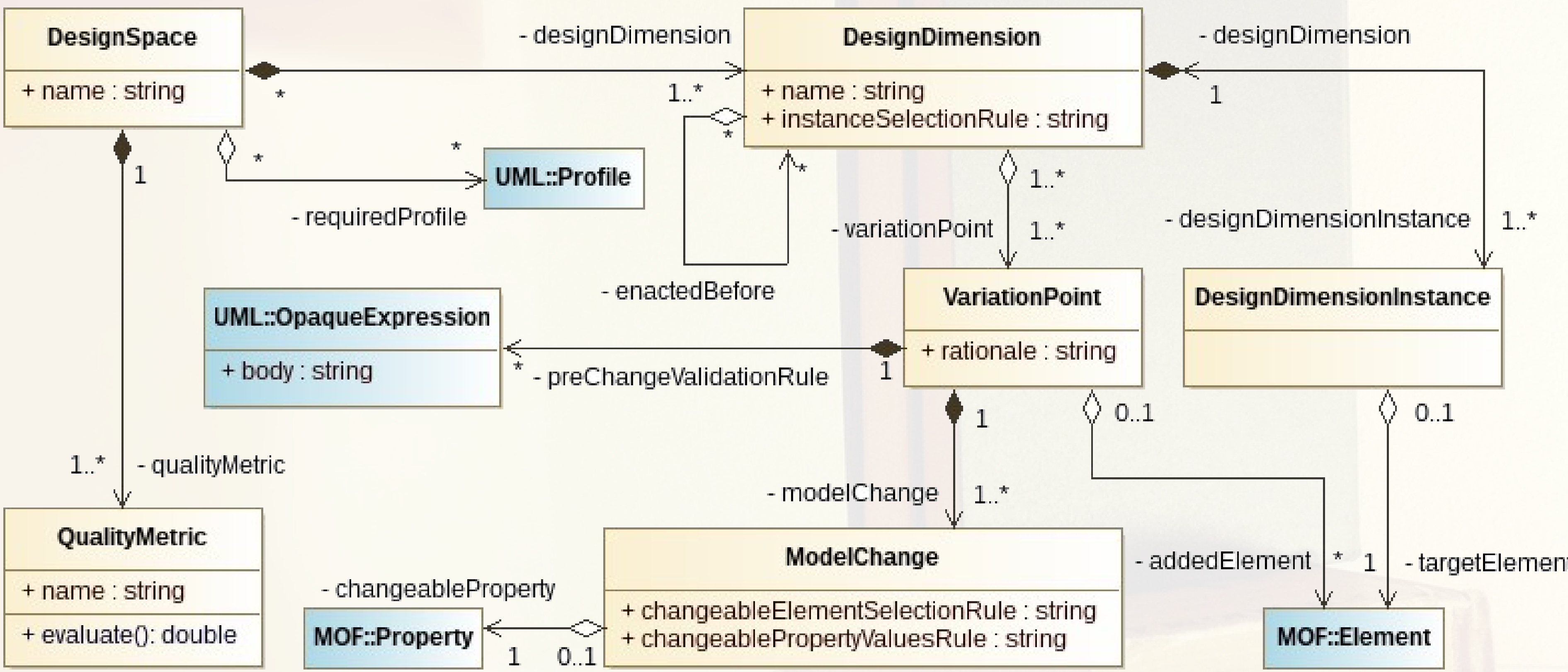

DuSE
=
Domain-specific design spaces
+
Quality metrics of automatically generated candidate architectures
+
A multi-objective optimization approach
SA:DuSE = DuSE instance (design space) for the Self-Adaptive Systems domain



Architectural contributions from design dimensions
Design dimension 1 (e.g. Control Law)
Input model (non-adaptive system)
Variation Point
Design space location
Resulting candidate architecture (adaptive system)
Invalid Design space location
Design dimension 2 (e.g. MAPE-K deployment)
Design dimension 3 (e.g. Control Adaptation)
Violates some variation point's constraint
Gontenbad
Appenzell


Design dimension 1
(e.g. Control Law)
Architectural contributions
from design dimensions
Input model
(non-adaptive system)
Variation Point
Design space location
Resulting candidate architecture
(adaptive system)
Invalid
Design space location
Design dimension 2
(e.g. MAPE-K deployment)
Violates some variation
point's constraint
Design dimension 3
(e.g. Control Adaptation)
Gontenbad
Appenzell

DuSE Metamodel



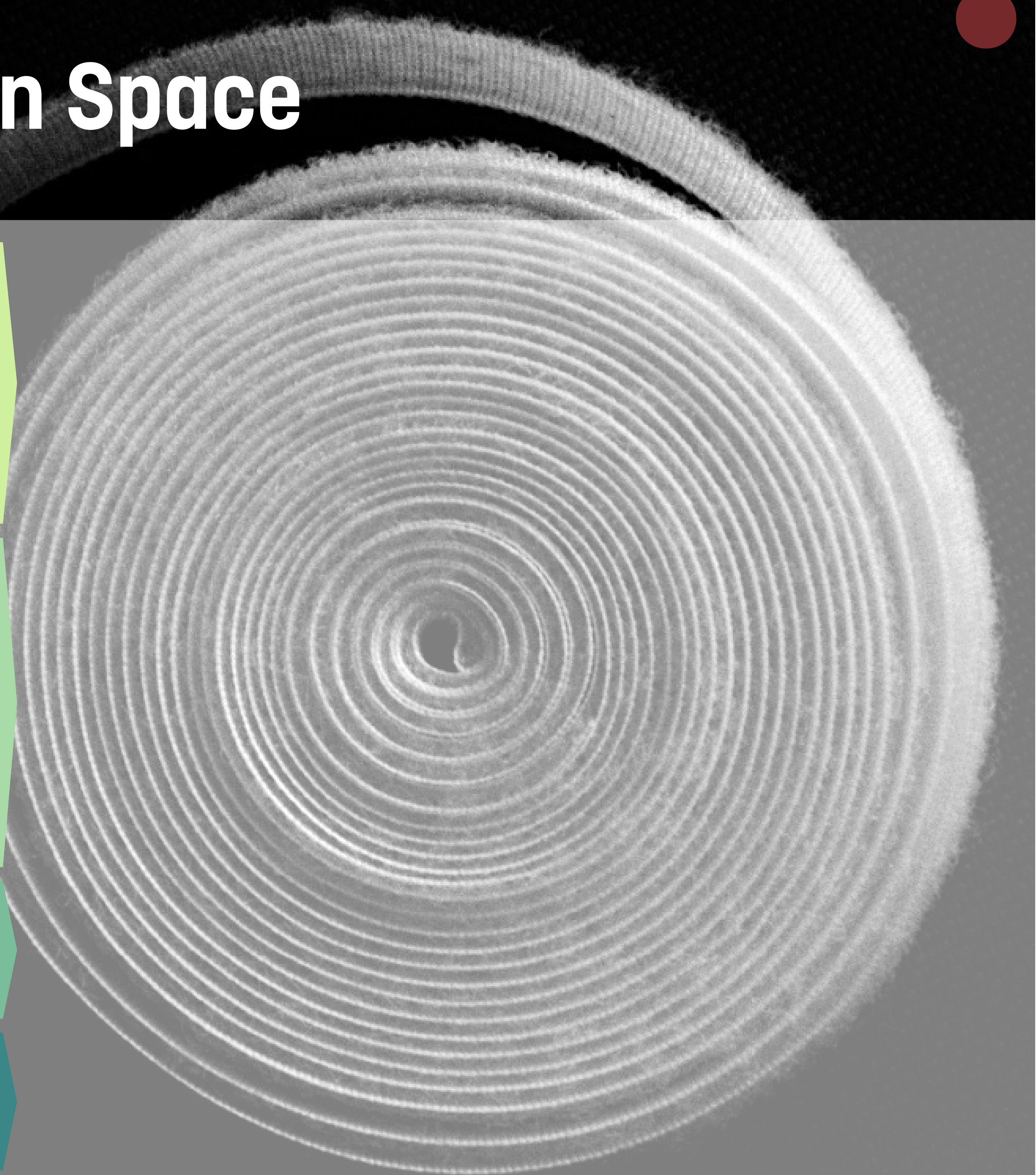
SA:DuSE Design Space

DD$_1$: Control Law

DD$_2$: Tuning Approach

DD$_3$: Control Adaptation

DD$_4$: MAPE Deployment



SA:DuSE Design Space

DD_1: Control Law
- VP11: Proportional
- VP12: Proportional-Integral
- VP13: Proportional-Integral-Derivative
- VP14: Static State Feedback
- VP15: Precompensated Static State Feedback
- VP16: Dynamic State Feedback

DD_2: Tuning Approach
- VP21: Chien-Hrones-Reswick, 0 OS, Dist. Rejection
- VP22: Chien-Hrones-Reswick, 0 OS, Ref. Tracking
- VP23: Chien-Hrones-Reswick, 20 OS, Dist. Rejection
- VP24: Chien-Hrones-Reswick, 20 OS, Ref. Tracking
- VP25: Ziegler-Nichols
- VP26: Cohen-Coon
- VP27: Linear Quadratic Regulator

DD_3: Control Adaptation
- VP31: Fixed Gain (no adaptation)
- VP32: Gain Scheduling
- VP33: Model Identification Adaptation Control

DD_4: MAPE Deployment
- VP41: Global Control
- VP42: Local Control + Shared Reference
- VP43: Local Control + Shared Error

SA:DuSE Quality Metrics

M_1: Control Overhead

M_2: Average Settling Time

M_3: Average Maximum Overshoot

M_4: Control Robustness

SA:DuSE Quality Metrics

M_1: Control Overhead

$$ME_1 = \frac{allOwnedElements() \rightarrow selectAsType(QController) \rightarrow collect(overhead()) \rightarrow sum()}{allOwnedElements() \rightarrow selectAsType(QController) \rightarrow size()}$$

$; QController :: overhead()\ increasingly\ penalizes\ VP_{32}, VP_{33}, VP_{41}$ and VP_{43}

M_2: Average Settling Time

$$ME_2 = \frac{allOwnedElements() \rightarrow selectAsType(QParametricController) \rightarrow sum(stime())}{allOwnedElements() \rightarrow selectAsType(QParametricController) \rightarrow size()}$$

$$; where\ QParametricController :: stime() = \frac{-4}{\log(max_i |p_i|)}$$

; and $max_i |p_i|$ is the magnitude of the largest closed-loop pole

M_3: Average Maximum Overshoot

$$ME_3 = \frac{allOwnedElements() \rightarrow selectAsType(QParametricController) \rightarrow sum(maxOS())}{allOwnedElements() \rightarrow selectAsType(QParametricController) \rightarrow size()}$$

$$; where\ QParametricController :: maxOS() = \begin{cases} 0 ; real\ dominant\ pole\ p_1 \geq 0 \\ |p_1| ; real\ dominant\ pole\ p_1 < 0 \\ r^{\pi/|\theta|} ; dominant\ poles\ p_1, p_2 = r.e^{\pm j.\theta} \end{cases}$$

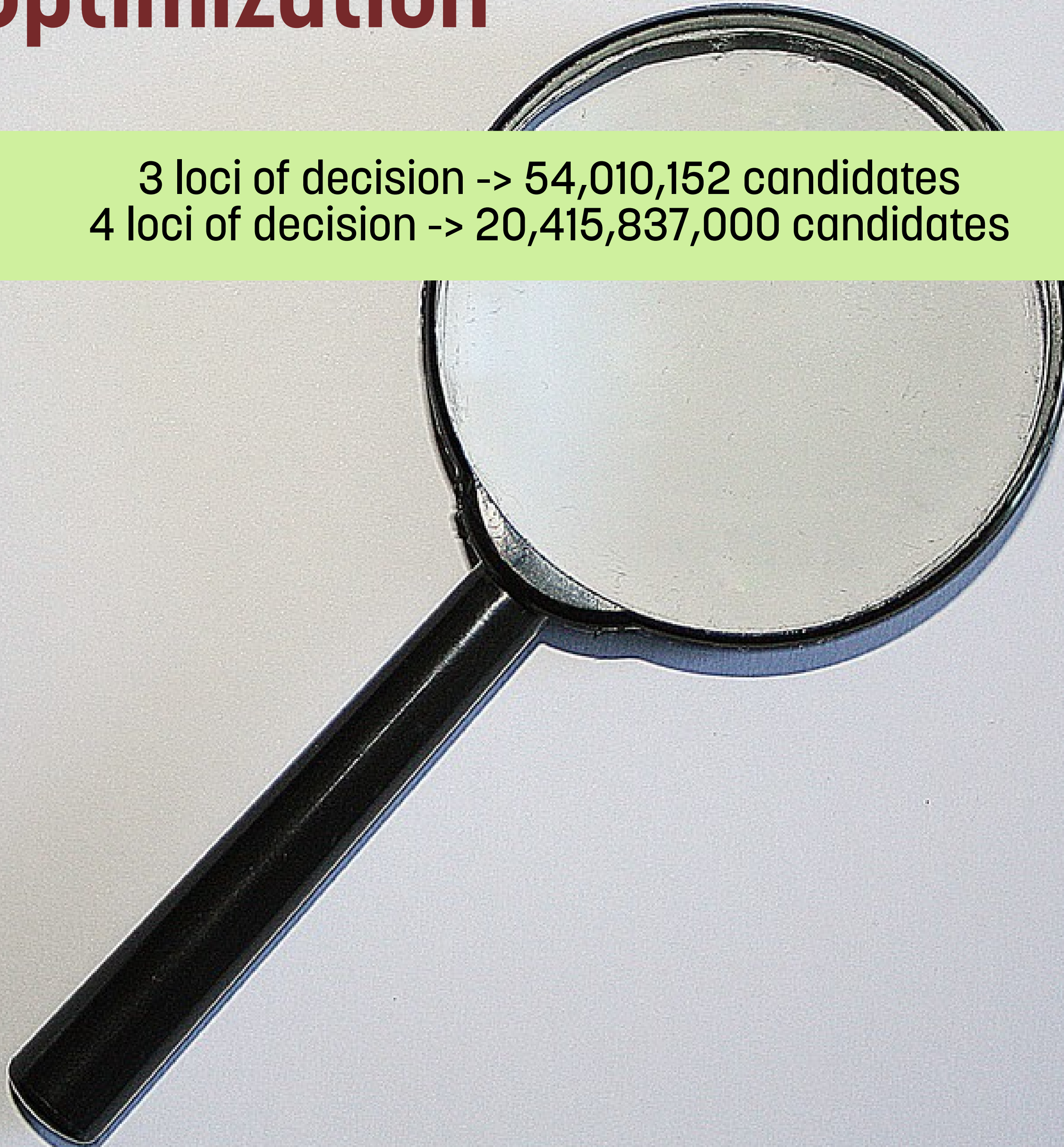
M_4: Control Robustness

$$ME_4 = \frac{allOwnedElements() \rightarrow selectAsType(QController) \rightarrow collect(robustness()) \rightarrow sum()}{allOwnedElements() \rightarrow selectAsType(QController) \rightarrow size()}$$

$; QController :: robustness()\ increasingly\ penalizes\ VP_{31}$ and VP_{32}

DuSE Optimization

3 loci of decision -> 54,010,152 candidates
4 loci of decision -> 20,415,837,000 candidates



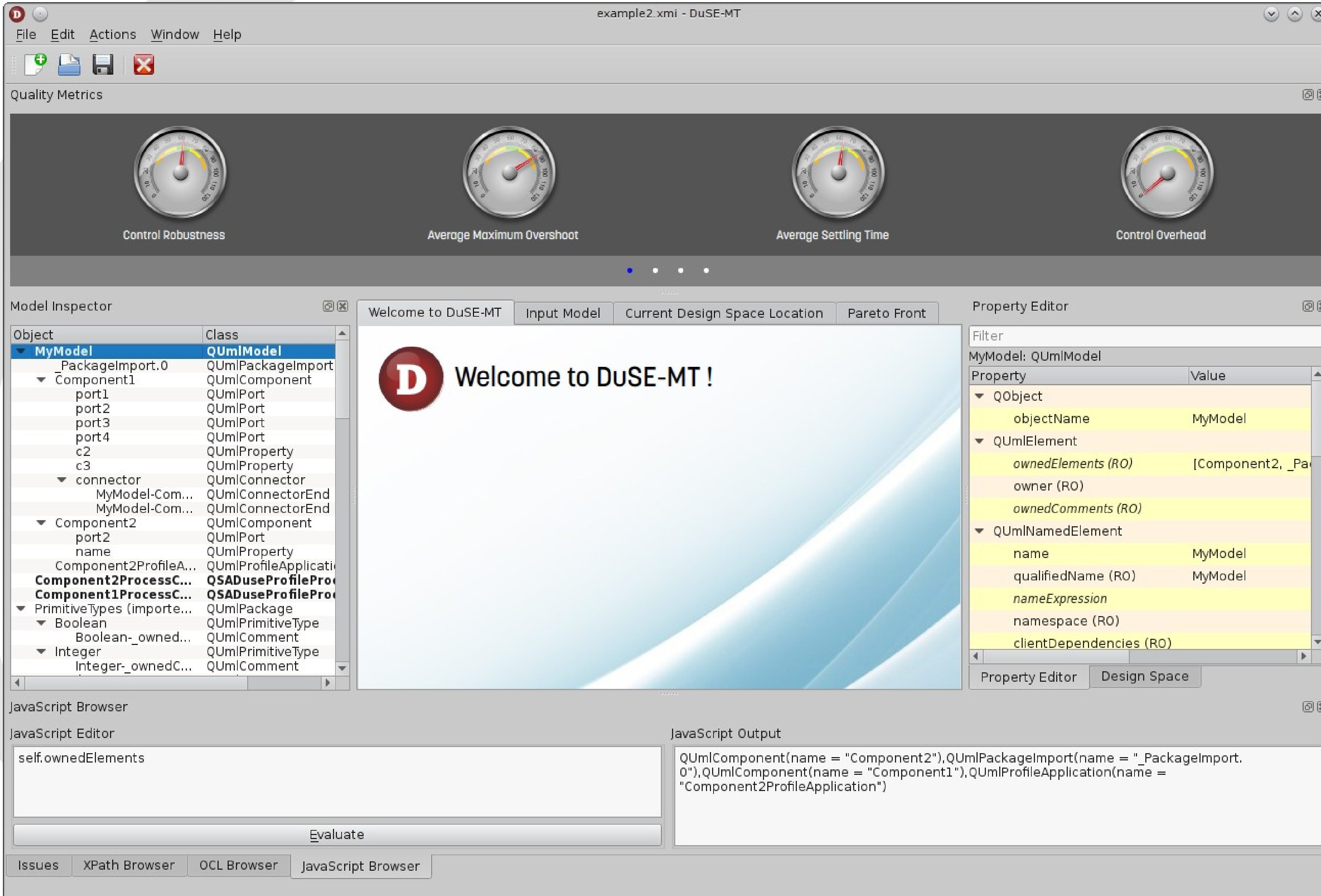
DuSE Optimization

3 loci of decision -> 54,010,152 candidates
4 loci of decision -> 20,415,837,000 candidates

A search-based approach enables effective design space exploration and helps preventing false intuition and technology bias

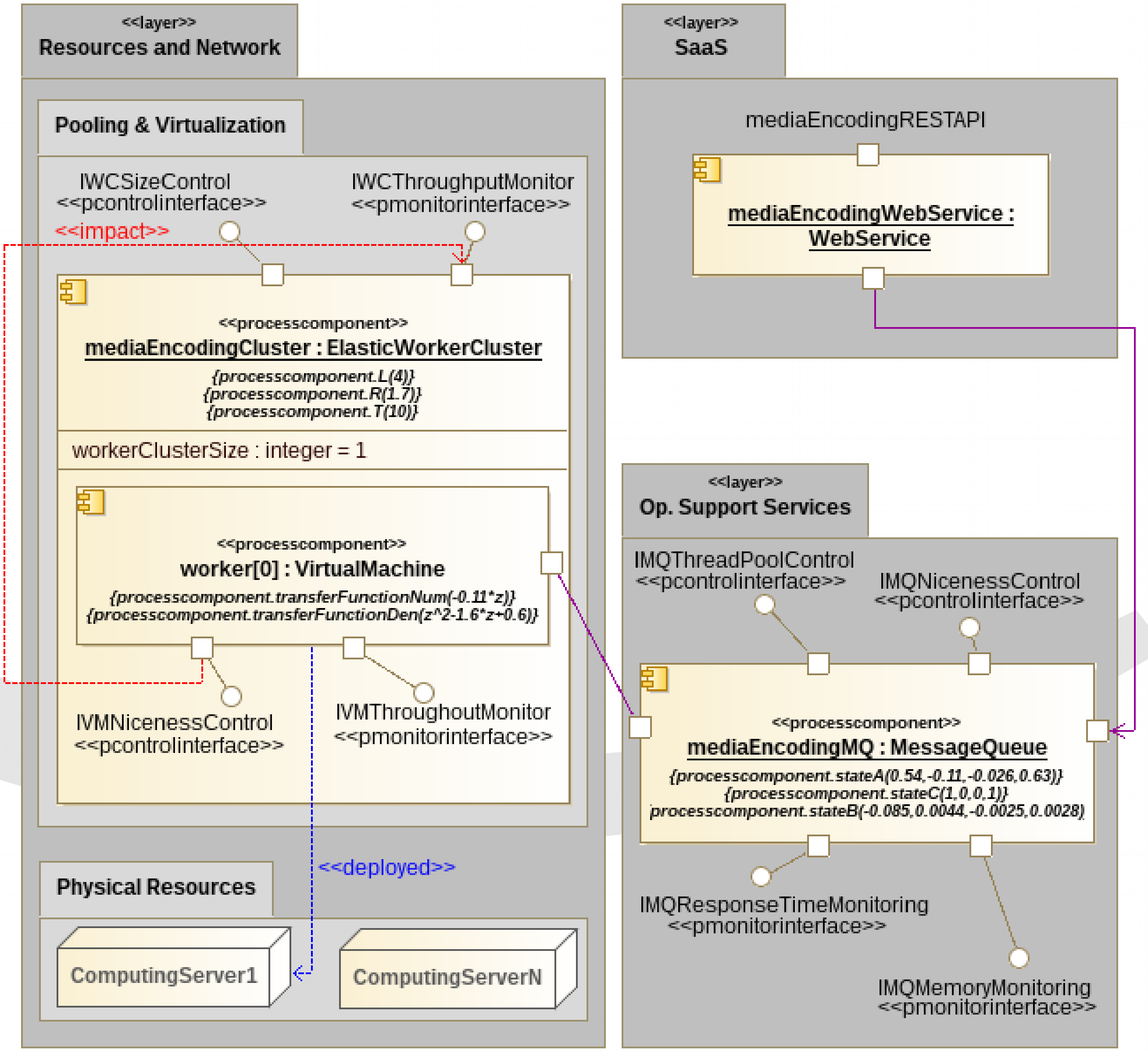
procedure DUSE_OPT(i, di)
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\triangleright$ i = input model; di = DuSE instance
$\quad s \leftarrow globalSettings()$
$\quad dii \leftarrow createAppDSInstance(i, di)$
$\quad p \leftarrow randomPopulation(dii, s.populationSize)$
$\quad curIteration \leftarrow 0$
$\quad$ **while** $curIteration \leq s.maxIterations$ **do**
$\qquad p \leftarrow nsga2Rank(p, dii, s.populationSize)$
$\qquad p \leftarrow p \cup nsga2Mate(p, s.offspringSize)$
$\qquad curIteration \leftarrow curIteration + 1$
$\quad$ **end while**
$\quad$ **return** $nsga2highestParetoRank(p)$
end procedure

Tool Support (DuSE-MT)

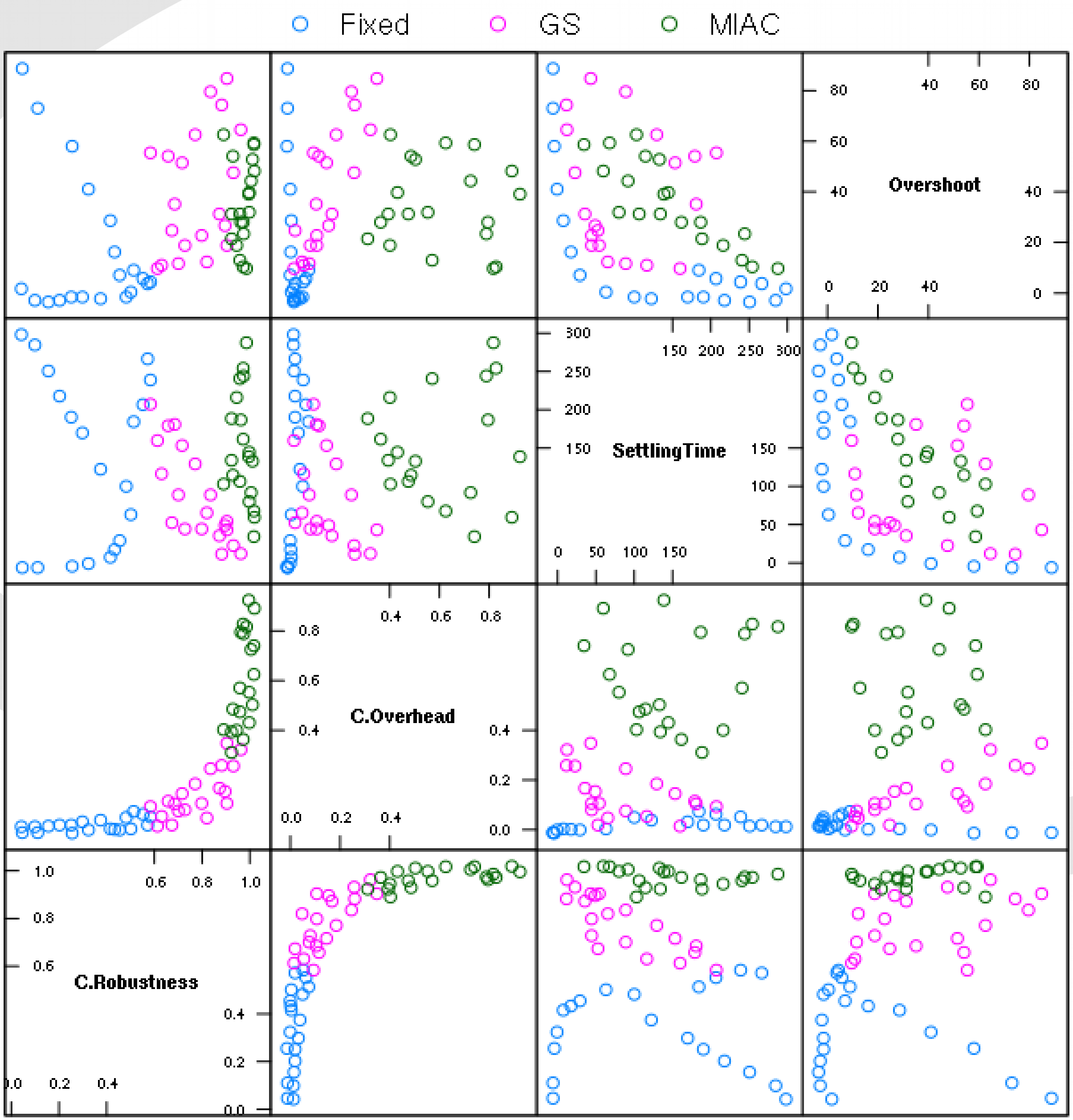


Case Study

- Cloud-based media encoding service
- Three loci of decision (controllable components)
- Annotations from Qemu + Hadoop + CloudStack experiments
- Control goal: enforce encoding throughput



Findings



Fixed
GS
MIAC
Overshoot
SettlingTime
C.Overhead
C.Robustness

Conclusion

Contributions

- Systematic gathering of architecture design knowledge in the field of Self-Adaptive Systems
- A search-based approach for endowing architectures with self-adaptative behaviour and explicit support for well-informed design trade-off analysis
- A supporting tool (DuSE-MT)

Conclusion

Contributions

- Systematic gathering of architecture design knowledge in the field of Self-Adaptive Systems
- A search-based approach for endowing architectures with self-adaptative behaviour and explicit support for well-informed design trade-off analysis
- A supporting tool (DuSE-MT)

Limitations

- Requires an initial annotated architectural model
- No guaranteed optimality (local-optimal Pareto fronts)
- Still requires *a posteriori* preference articulation

Current & Future Work

- Second case study
- Resulting Parent front evaluation (indicators)
- From Design Spaces to Design Theories