Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

ADAPTATION MECHANISMS FOR COMPLEX SOFTWARE SYSTEMS

PhD Student: Konstantinos Angelopoulos
Advisor: John Mylopoulos

Qualifying Exam Presentation 08/02/2013

Outline

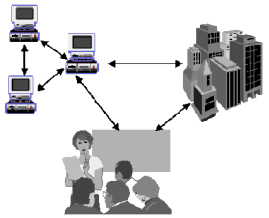
- Introduction
- State of the art
- Research problem
- Baseline
- Research approach
- Evaluation plan
- Conclusions

Motivation

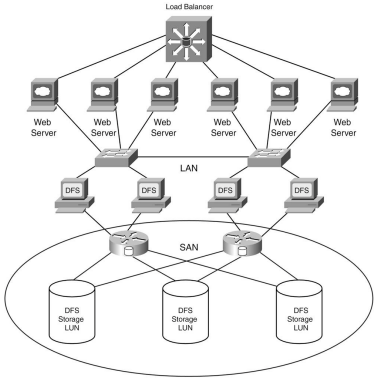
- Increasing complexity
- Multiple objectives
- Requirements change
- Maintenance and administration are expensive $$$

1+1=2

$$
\begin{aligned}
|a(w,\tilde{v}_h) - a_h(w^*,v_h)| &\le Ch^n\|A\|_n\|w\|_{1,\Omega}|v_h|_{1,\Omega_h} \\
&+ \left|\int_{\Omega_h}\sum_{i,j=1}^{3} a_{ij}^h(x)\left(\nabla w^*(D\Phi_h)^{-1}\cdot\underline{e}_i\right)\right. \\
&\left.\times\left(\nabla v_h(D\Phi_h)^{-1}\cdot\underline{e}_j\right)(J(\Phi_h-1)\mathrm{d}x\right| \\
&+ \left|\int_{\Omega_h}\sum_{i,j=1}^{3} a_{ij}^h(x)\left[\left(\nabla w^*(D\Phi_h)^{-1}\cdot\underline{e}_i\right)\right.\right. \\
&\left.\left.\times\left(\nabla v_h(D\Phi_h)^{-1}\cdot\underline{e}_j\right) - (\nabla w^*\cdot\underline{e}_i)(\nabla v_h\cdot\underline{e}_j)\right]\mathrm{d}x\right| \\
&\le C\|A\|_n\|w\|_{1,\Omega}|v_h|_{1,\Omega_h}(h^n + \|J(\Phi_h)-1\|_{0,\infty,\Omega_h}) \\
&+ \left|\int_{\Omega_h}\sum_{i,j=1}^{3} a_{ij}^h(x)((\nabla w^*)t(D\Phi_h)^{-1}\underline{e}_i)((\nabla v_h)^t[(D\Phi_h - I_3)\underline{e}_j]\mathrm{d}x\right. \\
&+ \left|\int_{\Omega_h}\sum_{i,j=1}^{3} a_{ij}^h(x)[(\nabla w^*(D\Phi_h)^{-1}\underline{e}_i)(\nabla v_h\cdot\underline{e}_j]\right. \\
&\left.-(\nabla w^*\cdot\underline{e}_i)(\nabla v_h\cdot\underline{e}_j)]\mathrm{d}x\right|
\end{aligned}
$$



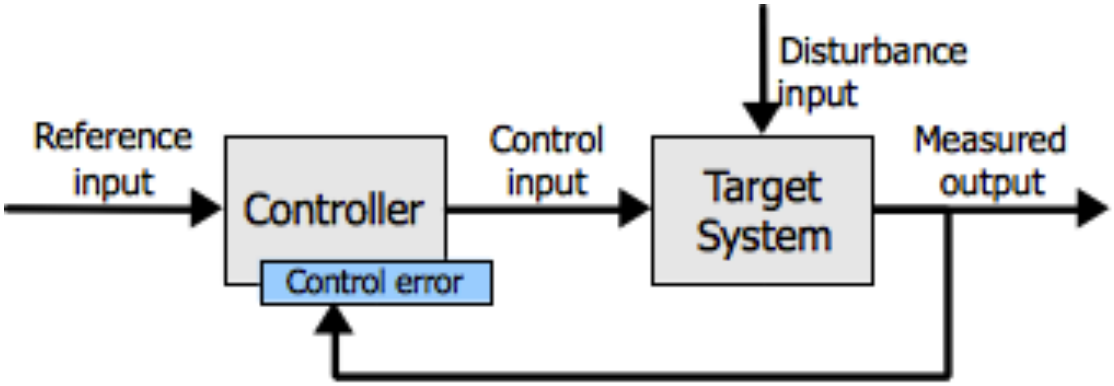
Software systems must self-adapt in order to “survive”



What is an adaptive system?

Definition: An adaptive system monitors its status and operation, performing actions to cope with changes when required

Example: Thermostat



Reference Input = T_I

Measured Output = T_O

Control Error $E = T_I - T_O$

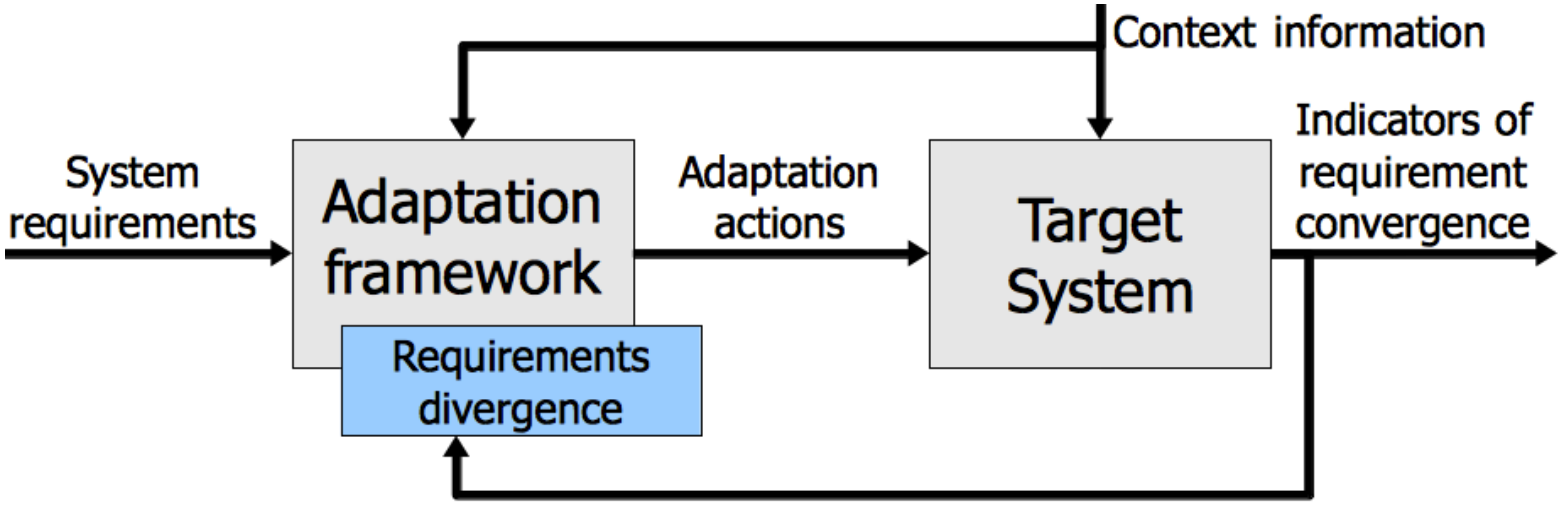
If $E > 0$ heat

If $E < 0$ cool

Other examples: homeostasis, PH maintenance (biology), amplifiers (electronics), stock market (economics)

Adaptive Systems in Software

- Requirements are the reference input
- Control is applied by frameworks
- Disturbance caused by the environment's context
- Error when requirements are not met



State of the Art

- Zanshin: Requirements based approach that exploits goal models and feedback loops. Adopts principles from Control Theory to apply adaptation [1]
- Rainbow: Architecture based approach also based on feedback mechanisms. Applies adaptation strategies that express administrative processes. Uses Utility Theory to select the best strategy [2]
- RELAX: Language to deal with uncertainty of the adaptive systems' environments using modal, temporal and ordinal operators [3]
- FLAGS: Based on KAOS uses fuzzy logic to relax "crisp" goals. Introduces the adaptive goals that involve adaptation countermeasures. Includes operationalization for service oriented systems [4]
- STARMX: Specialized framework for Java based systems. Follows the feedback loop model and applies external control using JMX technologies [5]

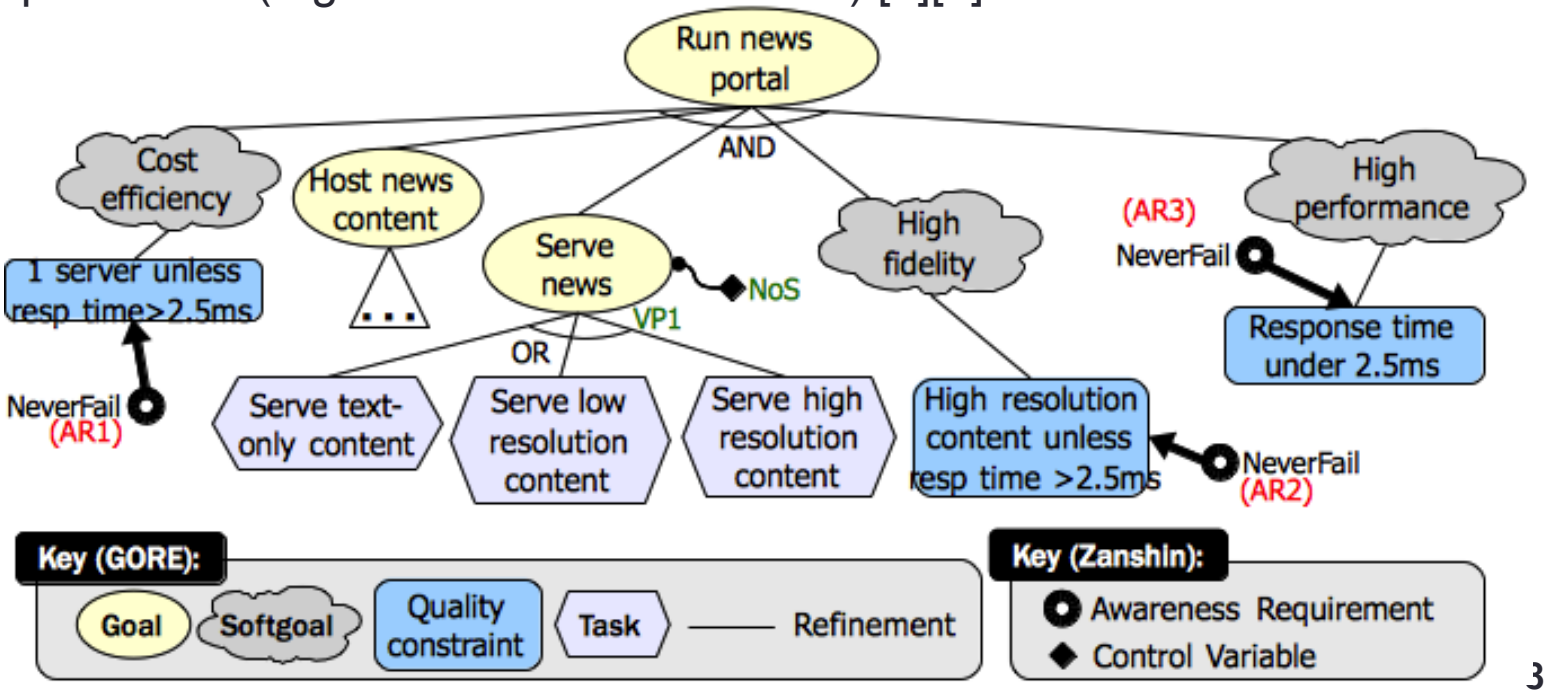
Research Problem: Overview

Current approaches:

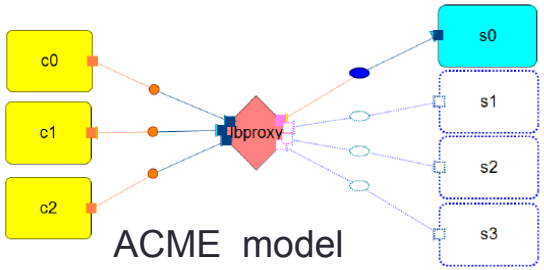
- Target specific kinds of systems
- Capture the variability either in requirements or architecture level (adaptation is based on variability)
- Requirements based approaches lack of technical details
- Architecture based approaches don’t deal with changes in requirements
- Most of them automate human administrative procedures (humans do mistakes) instead of applying reliable control mechanisms
- Those that apply control mechanisms lack precision (slow adaptation)
- Deal with multiple objectives in an empirical way prioritizing them intuitively

Research Baseline (Zanshin)

- Awareness requirements: indicators of the success failure of other requirements [6]
- System Identification: define the parameters of the system (CV and VP) and the impact over the indicators (e.g. Δ (AR3/NoS)[0,maxSrv] > 0 ➔ servers↑ then performance↑) [7]
- Adaptation: a)Reconfiguration by changing parameter values or b) Evolution requirements (e.g. relax constraint to 3ms) [8][9]



Requirements – Architecture Gap



ACME model

Q5:What is the role of the requirements?

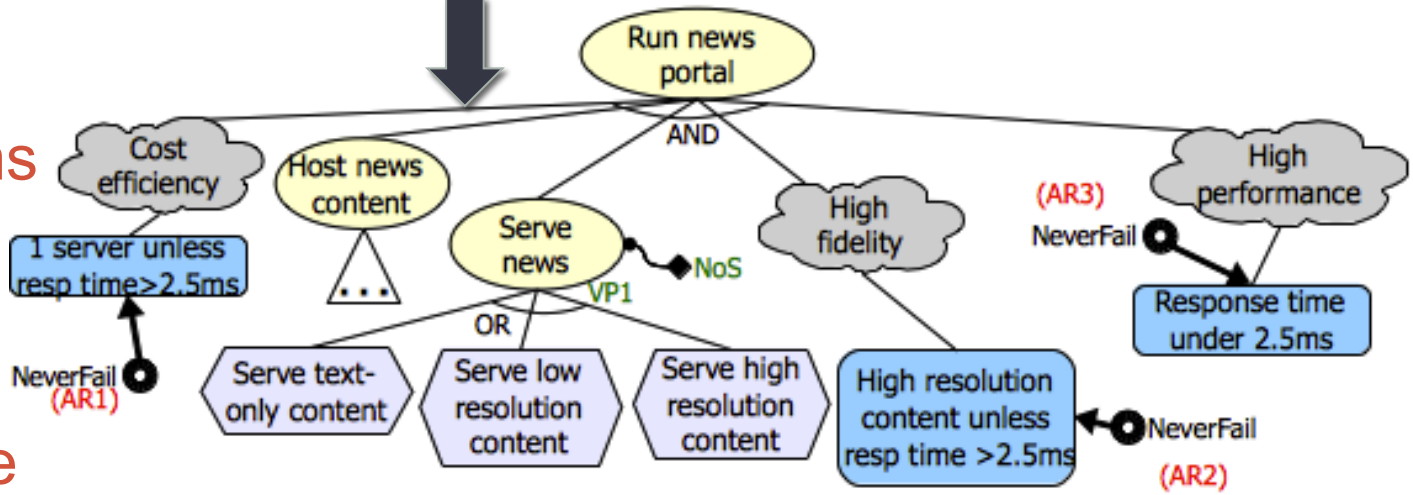
architectural model

Q6: What if requirements change?

requirements model

Q4: In what sequence the tasks are executed?

Q1: Who performs the tasks?

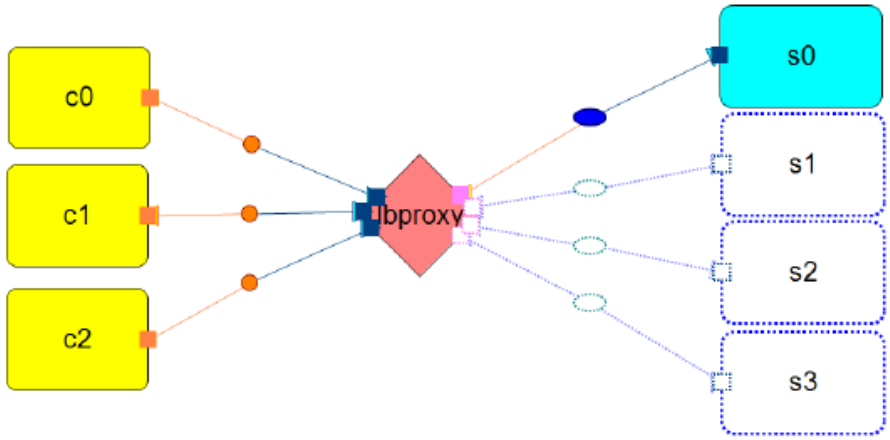


Q2: To whom the parameters belong to?

Q3: What do I monitor?

Conflicting Multiple Objectives

Znn Case Study: An news portal
With multimedia content.



Objectives:

- Low cost
- High fidelity
- High Performance

Q1: Can we have them all at the same time continuously?

Q2: What if more than one objective fails?

When the load is high there are 2 possible actions:

- Add more servers
- Switch to textual mode

When balance is achieved reverse adaptation processes take place to reduce the operational cost and increase the fidelity.

Precision Issues

Differential Equations:

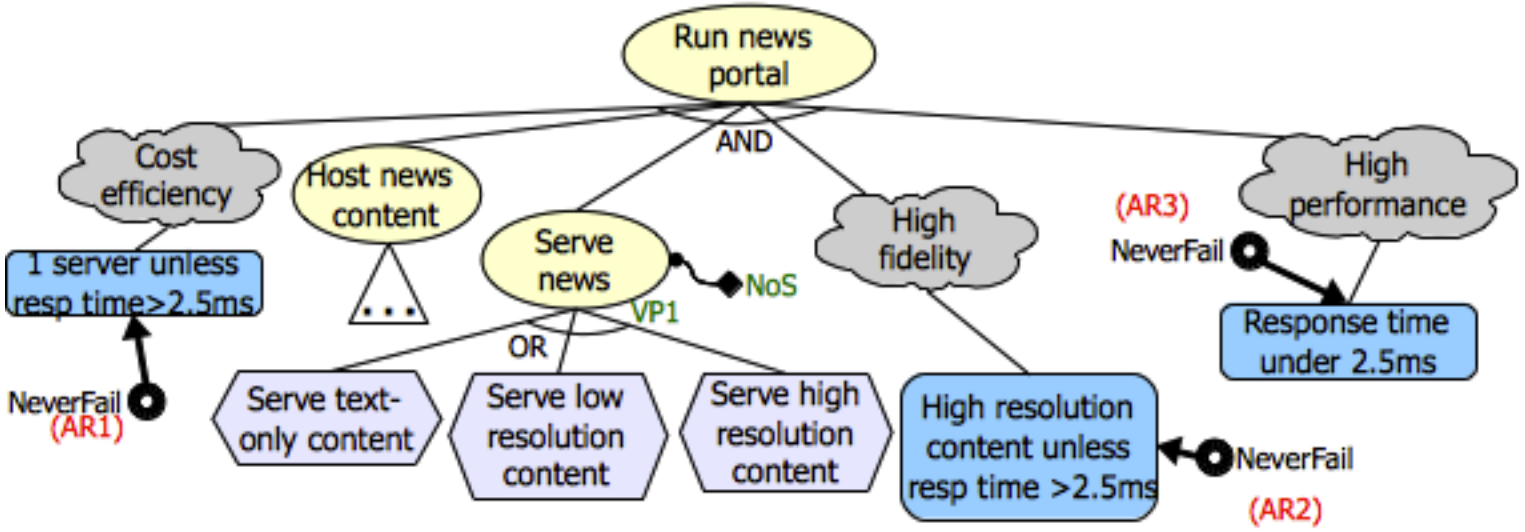
- Δ (AR1/NoS)[0,maxSrv] < 0
- Δ (AR3/NoS)[0,maxSrv] > 0
- Δ(AR2/VP1)[text➔ low ➔ high] > 0
- Δ (AR3/VP1)[text➔ low ➔ high] < 0

|Δ (AR3/NoS)| > |Δ (AR3/VP1)|

Q1: How many servers I have to add to achieve the highest performance with minimum cost?

Lack of quantitative system identification

Q2: How do I validate this?

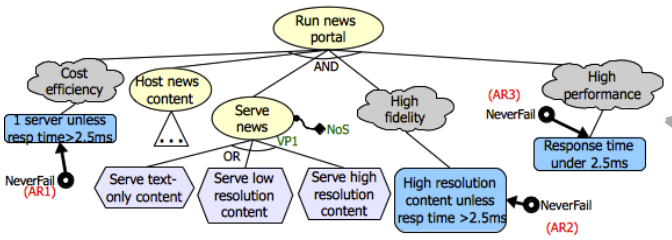


Research Approach: Overview

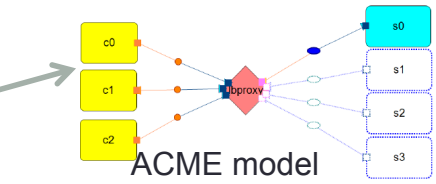
- Exploit both requirements and architectural models
- Apply optimization methods to deal with multiple objectives
- Propose advanced system identification methods that guarantee precision
- Integrate all the above with Zanshin
- Experiment with various kinds of systems to evaluate the generality and performence of our approach

Bridging the Gap

- Adaptation relies on variability
- Apply model transformations:
  - goal models ➔ statecharts (behavioral variability) [submitted RE'13]
  - goal models ➔ ACME (structural variability)
    by tailoring STREAM-A approach [10] [ongoing work]



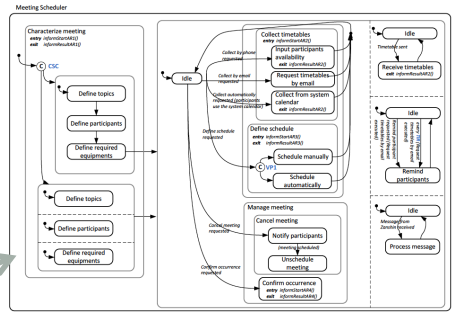
goal model



ACME model

$(A\,B\,(C\,|\,D)\,E\,F^*\,G)\,\circ\,(H^*)$

flow expression



statechart

- Benefits:
  - A specification of all the system's alternatives to operate
  - Deal with requirements changes
  - Deal with behavioral changes
  - System's technical details are available

Quantitative System Identification

[ongoing work]

Proposal 1: Use machine learning techniques to derive the weight of each parameter to the indicators (we may discover relations that we haven't thought before the implementation)

e.g. $I_1(P_1, P_2, P_3) = w_1P_1 + w_2P_2 + w_3P_3$

Supporting theories : Markov networks, Bayesian networks etc

Proposal 2: Given a dataset of inputs and outputs of the system perform a **regression** analysis to derive the quantitative relations among the parameters and the indicators

Supporting tools: Matlab, Mathematica etc

Warning: The system should be built and operate first!

Dealing With Multiple Objectives

[ongoing work]

Proposal 1 (intuitive) : Prioritize the indicators, dealing first with the failed indicators with higher priority and put locks on parameters that would harm other failing objectives

Proposal 2 (conservative) : Apply multiple objective optimization methods based on Control Theory (linear quadratic regulator) or decision making

Evaluation Plan

- Embed our proposed mechanisms into Zanshin
- Experiment on various kind of systems (socio-technical, software oriented, robotic etc) to validate the generality of our approach
- Experiment on case studies where precision is critical to evaluate the contribution of quantitative adaptation
- Experiment on case studies with multiple objectives to evaluate the contribution of optimization
- Carry out comparative studies with similar approaches to extract information about the advantages and the disadvantages of our proposal

Conclusions

We propose an approach that:

- combines both requirement and architectural models to apply adaptation mechanisms
- explores variability in a)requirements b)behavior and structure revealing all the alternative ways of execution
- increases the precision of the adaptation process
- deals with multiple objectives

and we plan to:

- integrate it with Zanshin
- evaluate its generality
- evaluate its performance on precision and optimization problems in adaptation

References

[1] V. E. S. Souza, “Requirements-based Software System Adaptation,” PhD Thesis, University of Trento, Italy, 2012.

[2] D. Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste, “Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure,” Computer, vol. 37, no. 10, pp. 46–54, 2004

[3] J. Whittle, P. Sawyer, N. Bencomo, B. Cheng, and J.-M. Bruel, “RELAX: a language to address uncertainty in self-adaptive systems requirement,” Requirements Engineering, vol. 15, no. 2, pp. 177–196, 2010.

[4] L. Baresi, L. Pasquale, and P. Spoletini, “Fuzzy Goals for Requirements driven Adaptation,” in Proc. of the 18th IEEE International Requirements Engineering Conference. IEEE, 2010, pp. 125–134.

[5] R. Asadollahi, M. Salehie, and L. Tahvildari. Starmx: A framework for developing self-managing java-based systems. Software Engineering for Adaptive and Self-Managing Systems, International Workshop on, 0:58{67, 2009.

[6] V. E. S. Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos, “Awareness Requirements for Adaptive Systems,” in Proc. of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. ACM, 2011, pp. 60–69.

[7] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, “System Identification for Adaptive Software Systems: a Requirements Engineering Perspective,” in Conceptual Modeling – ER 2011, ser. Lecture Notes in Computer Science, M. Jeusfeld, L. Delcambre, and T.-W. Ling, Eds. Springer, 2011, vol. 6998, pp. 346–361.

[8] V. Souza, A. Lapouchnian, K. Angelopoulos, and J. Mylopoulos, “Requirements-driven software evolution (online first),” Computer Science - Research and Development, pp. 1–19, 2012.

References

[9] V. E. S. Souza, A. Lapouchnian, and J. Mylopoulos, “Requirements driven Qualitative Adaptation,” in Proc. of the 20th International Conference on Cooperative Information Systems (to appear). Springer,

2012.

[10] J. a. Pimentel, M. Lucena, J. Castro, C. Silva, E. Santos, and F. Alencar, “Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach,” Requirements Engineering, vol. 17, no. 4, pp. 259–281, 2012.

Thank you!

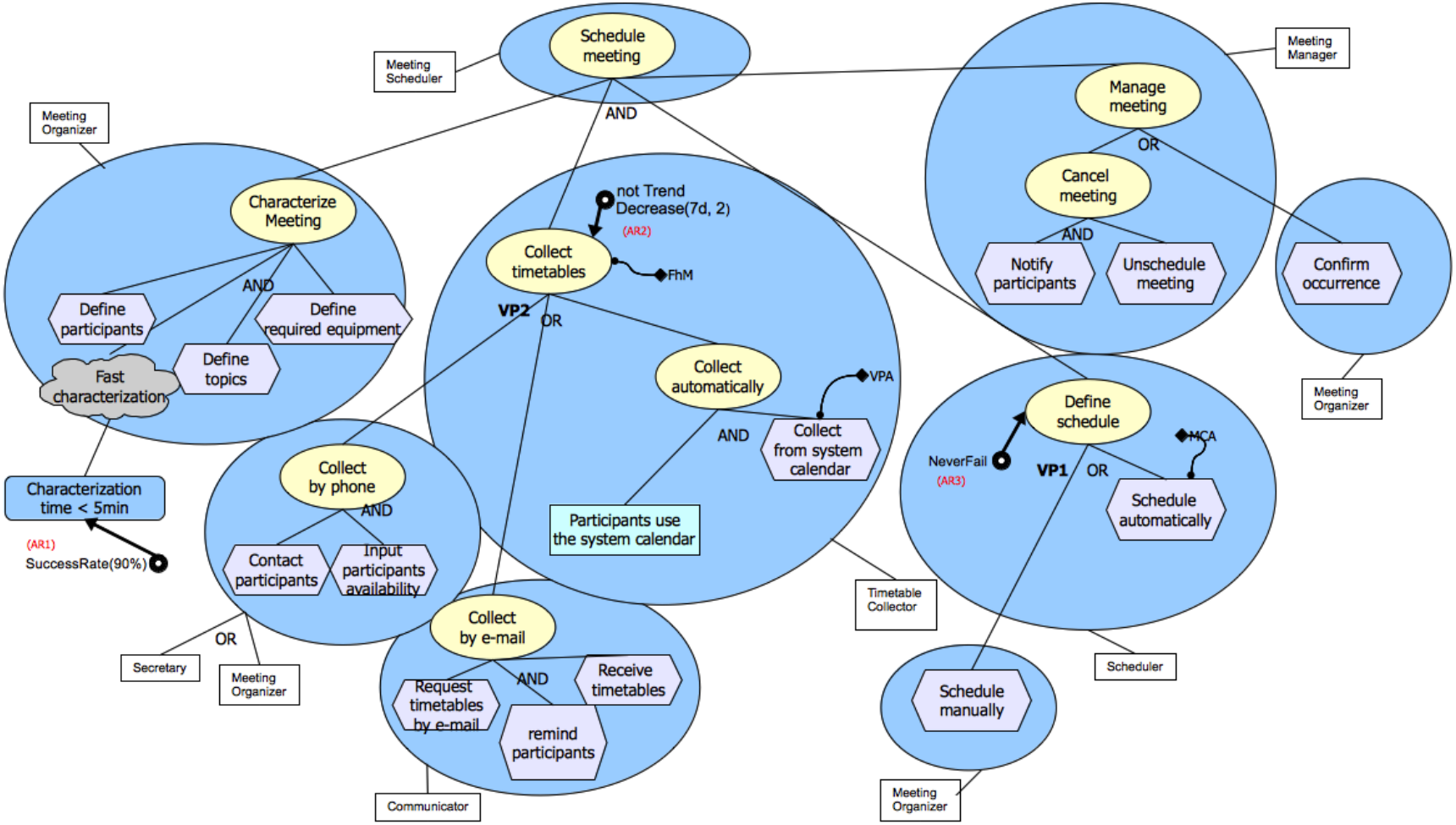
Deriving Architecture From Goals

Derivation process adopted from STREAM-A:

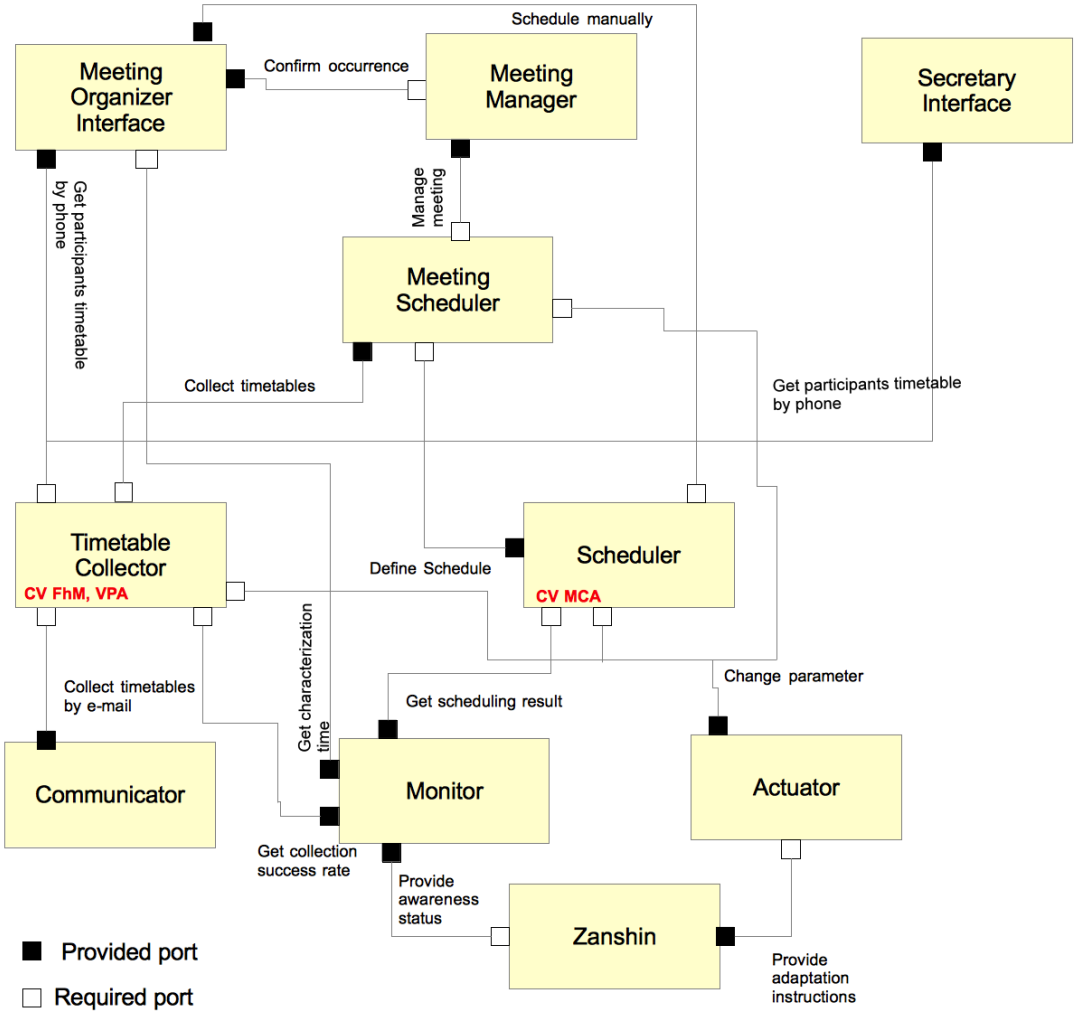
1. Assign goals and tasks to actors manually instead of the guiding heuristics in STREAM-A
2. Decide where the monitoring points related to the awareness requirements belong
3. Decide where the parameters belong
4. The actors turn to components and the interactions inferred from the goal model refinements are turned to connectors generating the architectural model
5. Add a monitor component that is related to every monitoring point
6. Add an actuator component that applies adaptation operations
7. Attach Zanshin to the monitor and the actuator

Part of the ongoing work: Define how evolution requirements are related to the architecture

Deriving Architecture From Goals



Deriving Architecture From Goals



Deriving Behavior From Goals

Similar to the previous approach:

1. Delegate tasks
2. Define basic flow
3. Generate base statechart
4. Specify transitions
5. Specify adaptive behavior
6. Perform further refinements

Submitted to RE' 13

Deriving Behavior From Goals

Step 1: We delegate the tasks that are neither performed nor assisted by the software system under development

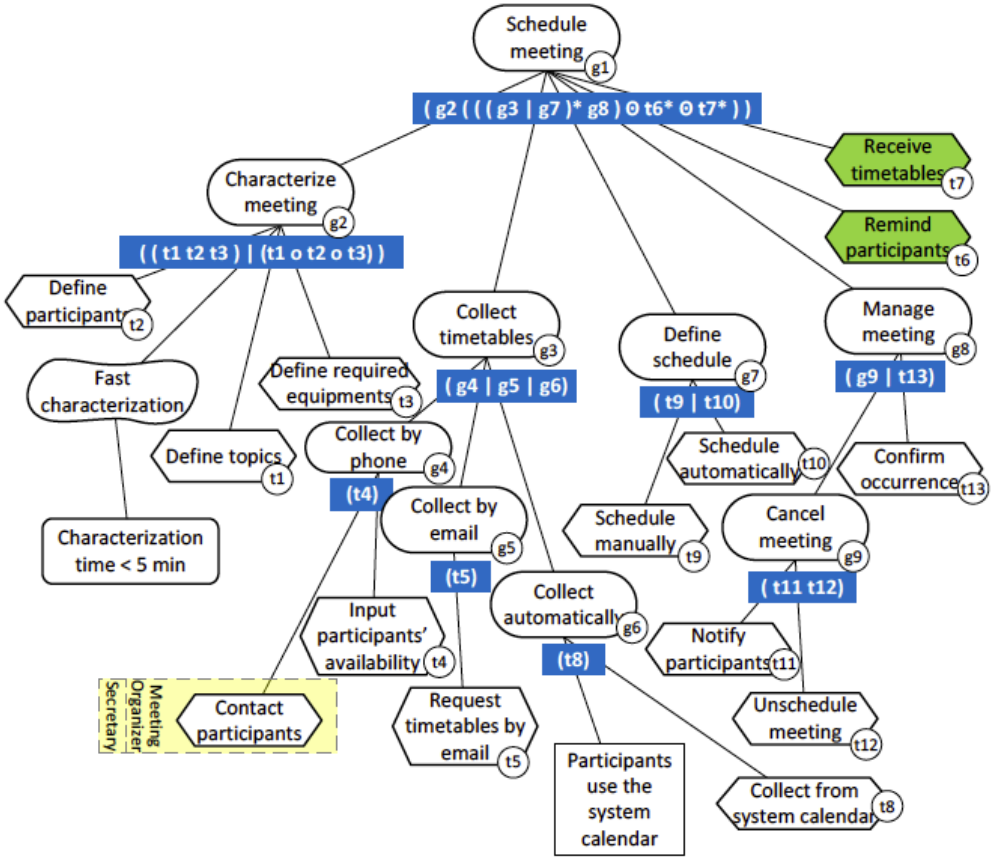
Step 2: We define the order of goal fulfillment and task execution with flow expressions. Then add intermediate states as a point where the system is waiting for some input, e.g., waiting for a selection by the user

Note: The flow expressions describe the flow of system behavior in terms of extended regular expressions

Example :

$$(A\,B\,(C\,|\,D)\,E\,F^{*}\,G)\;\circ\;(H^{*})$$

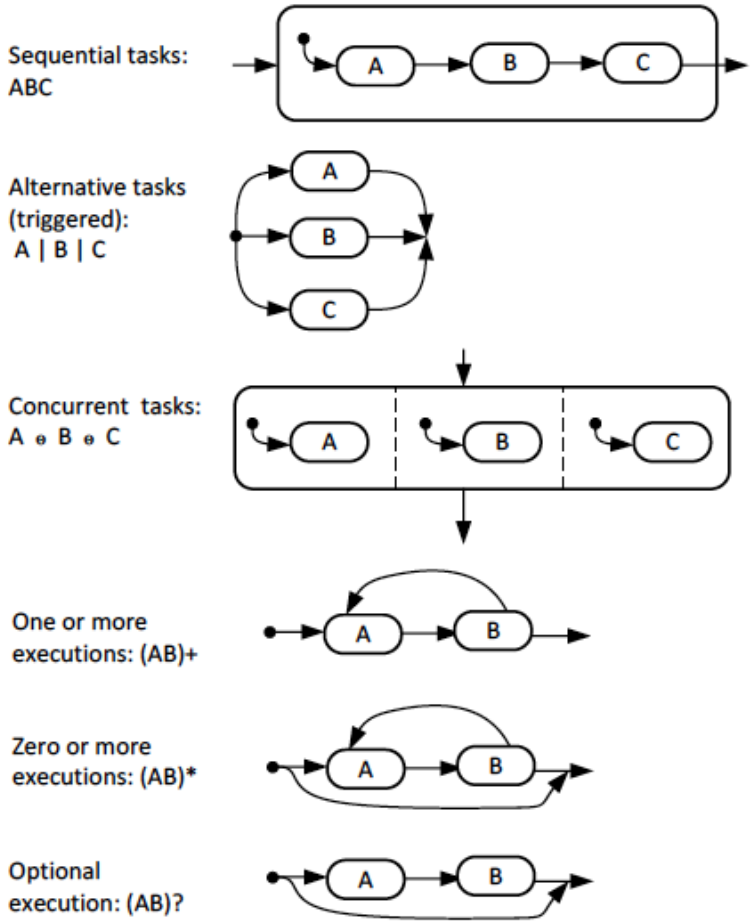
Deriving Behavior From Goals



$$((t1\ t2\ t3) \mid (t1 \circ t2 \circ t3)) (((i1 ((t4 \mid t5 \mid t8) \mid (t9 \mid t10))^* ((t11\ t12) \mid t13))) \circ (i2\ t6)^* \circ (i3\ t7)^*)$$

Deriving Behavior From Goals

Step 3: Generate the base statechart by using the flow expressions following specific derivation patterns:



Deriving Behavior From Goals

Step 4: The transitions are triggered by a) user requests b)timers c)request by another task d) request by another system or e) combination of the previous. Domain assumptions, quality constraints and parameters constitute possible pre-conditions.

#	From→To	Triggers
1	i1→t4	Collect by phone requested
2	i1→t5	Collect by email requested
3	i1→t8	Collect automatically requested
4	i1→(t9|t10)	Define schedule requested
5	i1→t11	Cancel meeting requested
6	i1→t13	Confirm occurrence requested
7	i2→t6	Remind participant requested OR Each day
8	i3→t7	Timetable sent

Deriving Behavior From Goals

Step 5: We add new parameters related to the behavior of the system

CSC – Characterize in Sequence or Concurrently: This parameter defines whether the Define topics, Define participants and Defined required equipments tasks should be performed sequentially or concurrently;

TIR – Time Interval between Reminders: This parameter expresses at which intervals the Remind participants task should be triggered by the system.

ScA – Scheduling Algorithm: There are different algorithms to perform the scheduling, with different tradeoffs between performance and number of conflicts.

Step 6: For our running no further refinements were necessary

Deriving Behavior From Goals