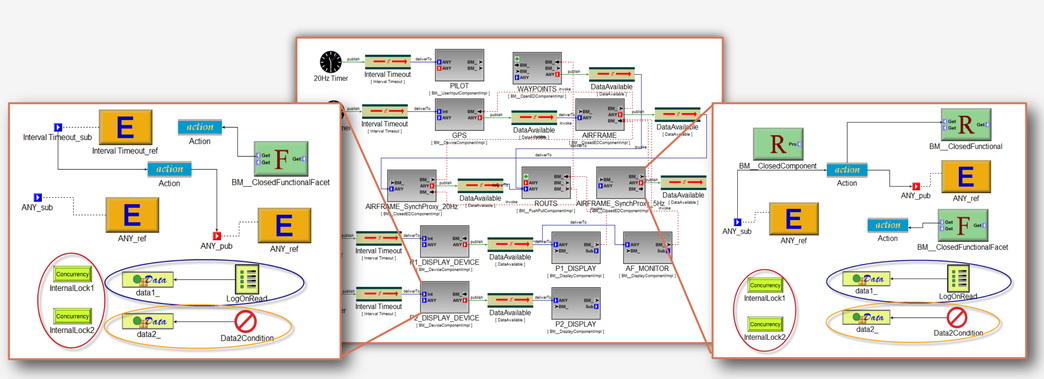
An End-User Demonstration Approach to Support Aspect-Oriented Modeling

Yu Sun
Software Composition and Modeling Laboratory
Department of Computer and Information Sciences
University of Alabama at Birmingham, USA
yusun@cis.uab.edu

BACKGROUND

- A typical Aspect-Oriented Modeling (AOM) process weaves aspect models (i.e., the crosscutting concerns that are scattered across a model) into the base model (i.e., the main model sans crosscutting behaviors).
- The model weaving process is accomplished by locating specific locations in the base model according to some pattern of model properties, and composing the necessary aspect models at these locations.

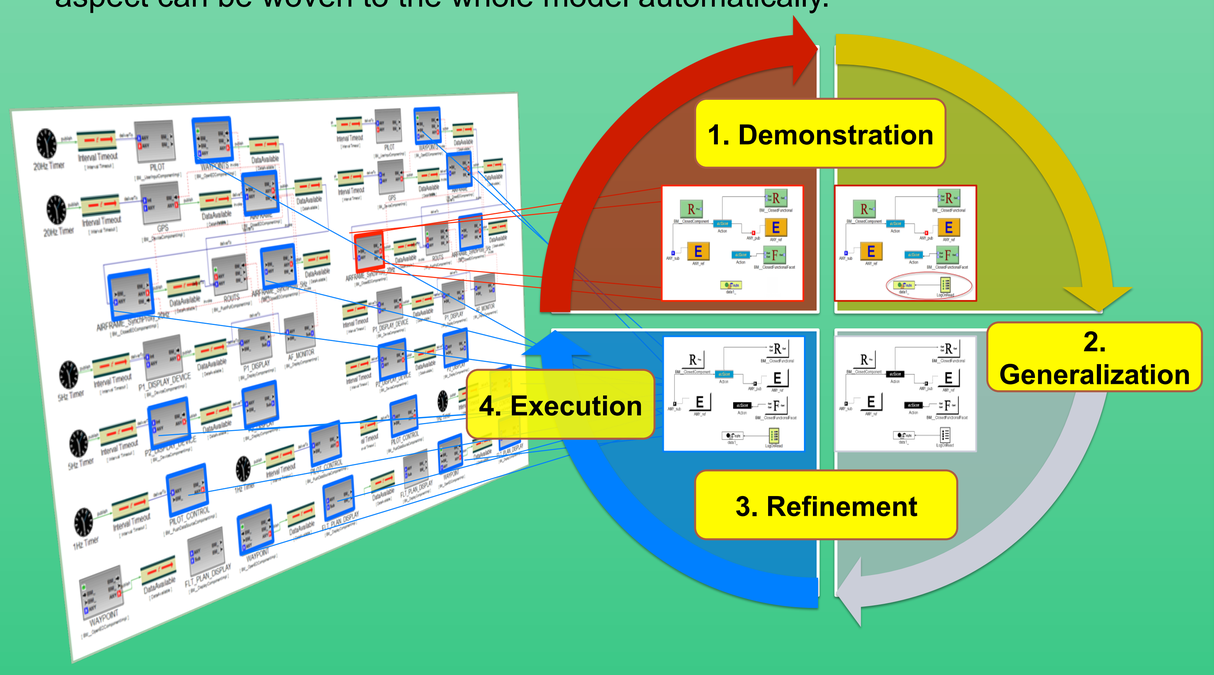


RESEARCH GOAL

- Design and implement a new approach to simplify the implementation of aspect-oriented modeling, so that general end-users are enabled to realize aspect model weaving tasks in an automated manner, **without knowing any model transformation languages or metamodel definitions**.

CASE STUDY

- The **Embedded Systems Modeling Language (ESML)** is a DSML used to graphically model real-time mission computing embedded avionics systems, which allows users to model the system from several different aspects such as *Interfaces*, *Events*, *Components*, *Interactions*, and *Configurations*.
- The AOM task to be accomplished is: in each *Implementation Component* (i.e., the name of the component ends with "impl"), if an *Action* exists in the *Component*, a *LogOnRead* logging element should be attached to every *Data* element in the *Component*.
- We select a *Component*, and demonstrate how to weave an *Action* in it. Then, by refining the generalized pattern and executing the pattern, the aspect can be woven to the whole model automatically.



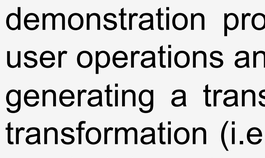
RESULTS

- No model transformation languages are used and the generated transformation patterns are invisible to users. Therefore, users are completely isolated from knowing a model transformation language and the metamodel definition.
- To evaluate the approach, the following provides a comparison of an AOM effort that was performed using a model transformation engine (in this, case C-SAW), to that using MTBD.

AOM Example	MTBD	C-SAW Language
ESML	3 editing operations 2 refinement operations	23 SLOC
QoSAML	4 editing operations 8 refinement operations	40 SLOC

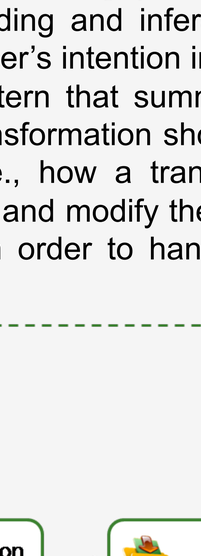
MOTIVATION

- The traditional approaches to weave aspect models are:
  - Manual editing



Tedious
Error-prone
Time-consuming

  - Writing model transformation rules

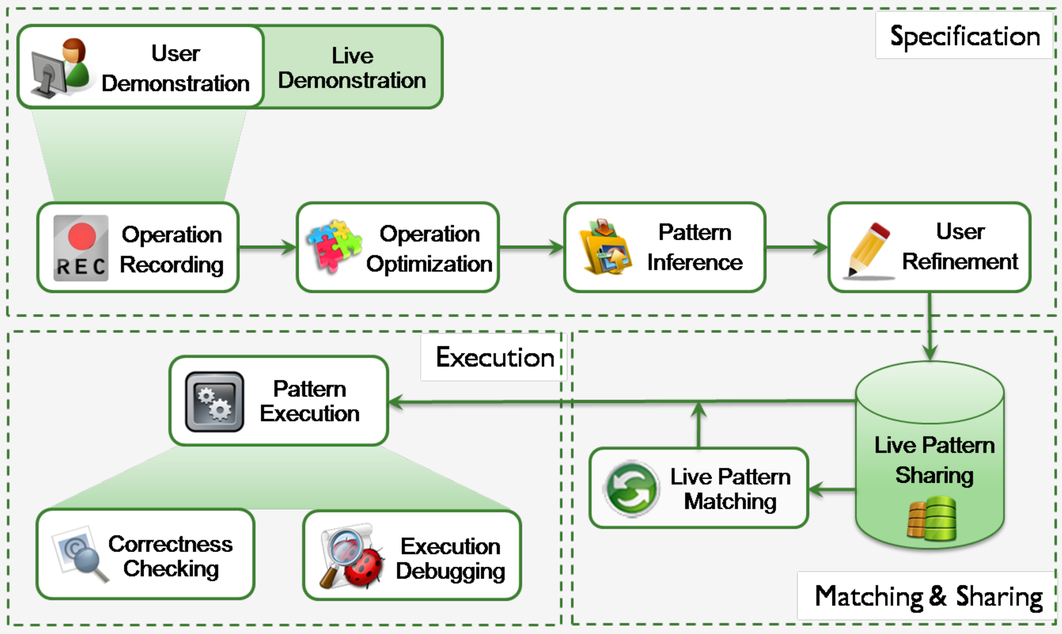


Steep language learning curve &
The challenge to understand domain definitions

- Open problem: How to enable general end-users (e.g., domain-experts, non-programmers) to weave aspect models into a base model?

OUR APPROACH

- **Model Transformation By Demonstration (MTBD)** enables users to demonstrate how an aspect model should be woven by directly editing the source model to simulate the weaving process step-by-step. During the demonstration process, a recording and inference engine captures all the user operations and infers the user's intention in a model transformation task, generating a transformation pattern that summarizes the precondition of a transformation (i.e., where a transformation should be done) and the actions needed in a transformation (i.e., how a transformation should be done). Users are also enabled to refine and modify the generated pattern to provide additional specific constraints in order to handle more complicated aspect model weaving requirements.



- Some new features have been added to MTBD:
  1) Live Demonstration provides a more general demonstration environment that allows users to specify editing activities based their editing history flexibly.
  2) Live Sharing is a centralized model transformation pattern repository has been built so that transformation patterns can be reused across different editors more efficiently.
  3) Live Matching has been developed to automatically match the saved transformation patterns at runtime, and provide editing suggestion and guidance to users during the editing process.

CONCLUSION

- We have applied our approach to successfully implement several practical AOM tasks in different domains, without writing any transformation rules or codes, showing improvement in the efficiency and simplicity.
- As future work, we will investigate how to ensure and check whether a demonstration truly reflects the desired AOM tasks, as well as how to debug the generated transformation.
- More examples and demos can be found at:
  - http://www.cis.uab.edu/softcom/mtbd

This material is based upon work supported by the National Science Foundation under Grant No. CCF-1052616 (CAREER).