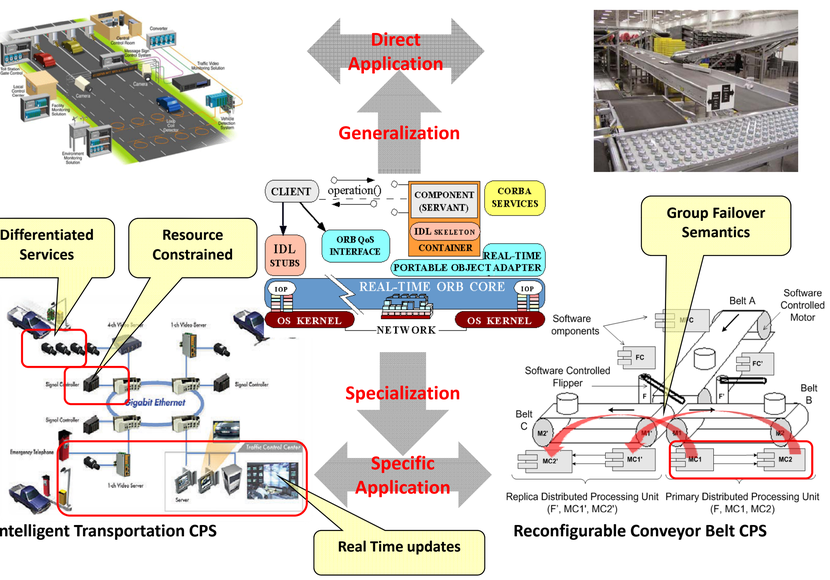
Systematic Specialization of General-Purpose Middleware for Cyber-Physical Systems

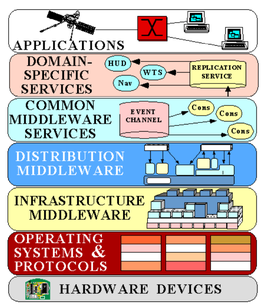
Akshay Dabholkar, Sumant Tambe and Aniruddha Gokhale
Dept. of EECS, Vanderbilt University, Nashville, TN, USA
{aky,sutambe,gokhale}@dre.vanderbilt.edu

1. Why Specialize Middleware for CPS?



2. What is Middleware Specialization?

Middleware Feature Layers



Principles of Middleware Specialization

1. Move away from the rigid layer processing
2. Prune away unnecessary features based on domain requirements , desired configurations, platform constraints and physics of operating environment.
3. Incorporate domain-specific semantics through AOSD and FOSD techniques
4. Automate construction of specialized middleware

➢ Aspect Oriented Software Development (AOSD)

- has a limited architectural model
- provides unbounded quantification
- good for fine-grained weaving/specialization

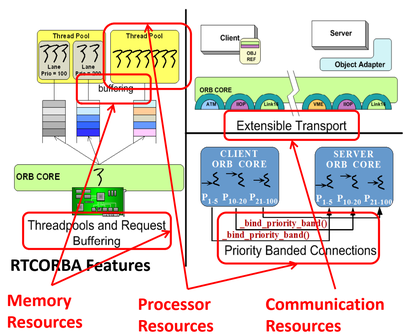
➢ Feature Oriented Software Development (FOSD)

- can represent single aspects or collections of aspects
- complements model-based development, preferred in CPS
- supports bounded (i.e., selective) quantification

➢ **Aspectual Feature Modules (AFMs) combines the best principles of AOSD & FOSD**

Goal is to enable ***specializations*** at ***all stages*** of development life cycle in an ***integrated*** manner.

3. AFM-driven Middleware Specialization



1. Map aspects onto feature modules to create Aspectual Feature Modules (AFMs)
2. Use Origami matrices to map middleware classes/components on to aspects
3. Use stepwise folding of matrix columns to synthesize specialized AFMs

Base \ RT	BasicRT	Priority	Conc	Synch
ORB	RTORB	PriMapper	TPReactor	
POA	RTPOA			
Xport	ExtXport	BandConn		
ReqHndl		CLI_PROP	TPLane	MUTEX

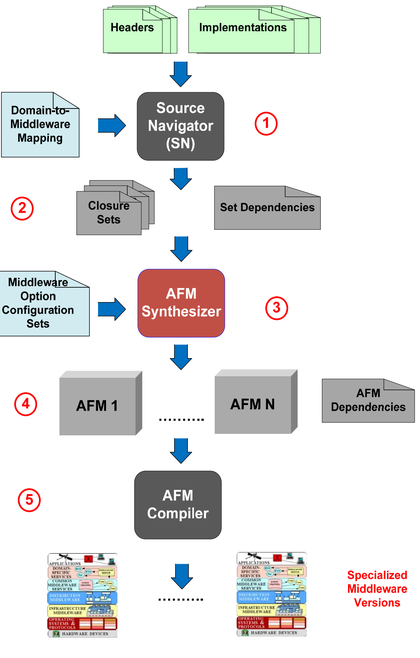
Feature Folding

Base \ RT	BasicRT	Priority • Conc	Synch
ORB	RTORB	PriMapper • TPReactor	
POA	RTPOA		
Xport	ExtXport	BandConn	
ReqHndl		CLI_PROP • TPLane	MUTEX

RTCORBA=MUTEX•TPLane•CLI_PROP•BandConn•ExtXport•TPReactor•PriMapper•RTPOA•RTORB

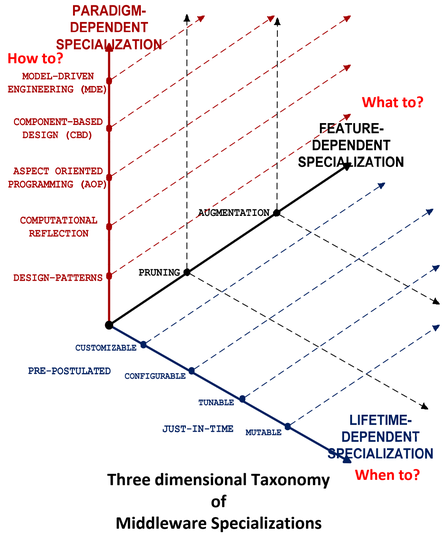
Research Contribution: Leverage Origami Folding for ***Build-time Middleware Specialization***

4. Preliminary Work: Specialization at Build-time



1. Navigate the source code dependencies of the selected middleware features from the folded origami ma
2. Create a closure set of source files and their dependencies to be compiled based upon the selected feature sets
3. Combine multiple closure sets and perform fine-grained pruning based on configuration sets
4. Synthesize AFMs and their dependencies
5. Compile the AFMs into specialized middleware versions

5. Future Work



Three dimensional Taxonomy of Middleware Specializations

- Overlapping dimensions share concepts, *e.g.* AOSD includes both *feature pruning* & *augmentation* and can be used for *customization* as well as *tuning*
- Can be combined to produce new variants of specialization
- Serves as a **guideline** for synthesis of tools for design, V&V, analysis of specializations

6. Conclusions & Open Research Areas

➢ AOSD and FOSD together is a promising approach to middleware specialization
➢ Current work focuses on specializing middleware a build-time

- How to efficiently annotate middleware source code for feature identification and management?
- How are features manipulated across different stages of application lifecycle?
- How do we handle feature and aspect interactions?
- Can we formalize the theory as a software engineering process?
- How should next-generation middleware be designed?
- Pattern Languages of Specialization?
- Safe specializations?
- V&V of specializations?

This work is sponsored by **NSF CAREER Award # 0845789 (2009-2014)**