Initial Report of the Impact of Using Overture/VDM in a Development Process in an Architecture-Oriented Approach

- (Work-In-Progress Report) -

Shigeru KUSAKABE, Yoichi OMORI, Keijiro ARAKI

Kyushu University, Japan

Outline

- Background
- New Project in Japan: Architecture Oriented one
- Sub-Topic: Software Process with Formal Method
- Preliminary Evaluation of Initial Trial
- Concluding Remarks

Outline

- Background
- New Project in Japan: Architecture Oriented one
- Sub-Topic: Software Process with Formal Method
- Preliminary Evaluation of Initial Trial
- Concluding Remarks

Background

(In Japan)

- Projects using formal methods are very limited. However, engineers and managers seem interested in formal methods.
- In order to break this situation, ...
  - Several organizations/groups, such as SEC[*1] of IPA[*2], are trying to establish guidelines to introduce formal methods in real projects.

*1 SEC: Software Engineering Center

*2 IPA: Information-technology Promotion Agency

Outline

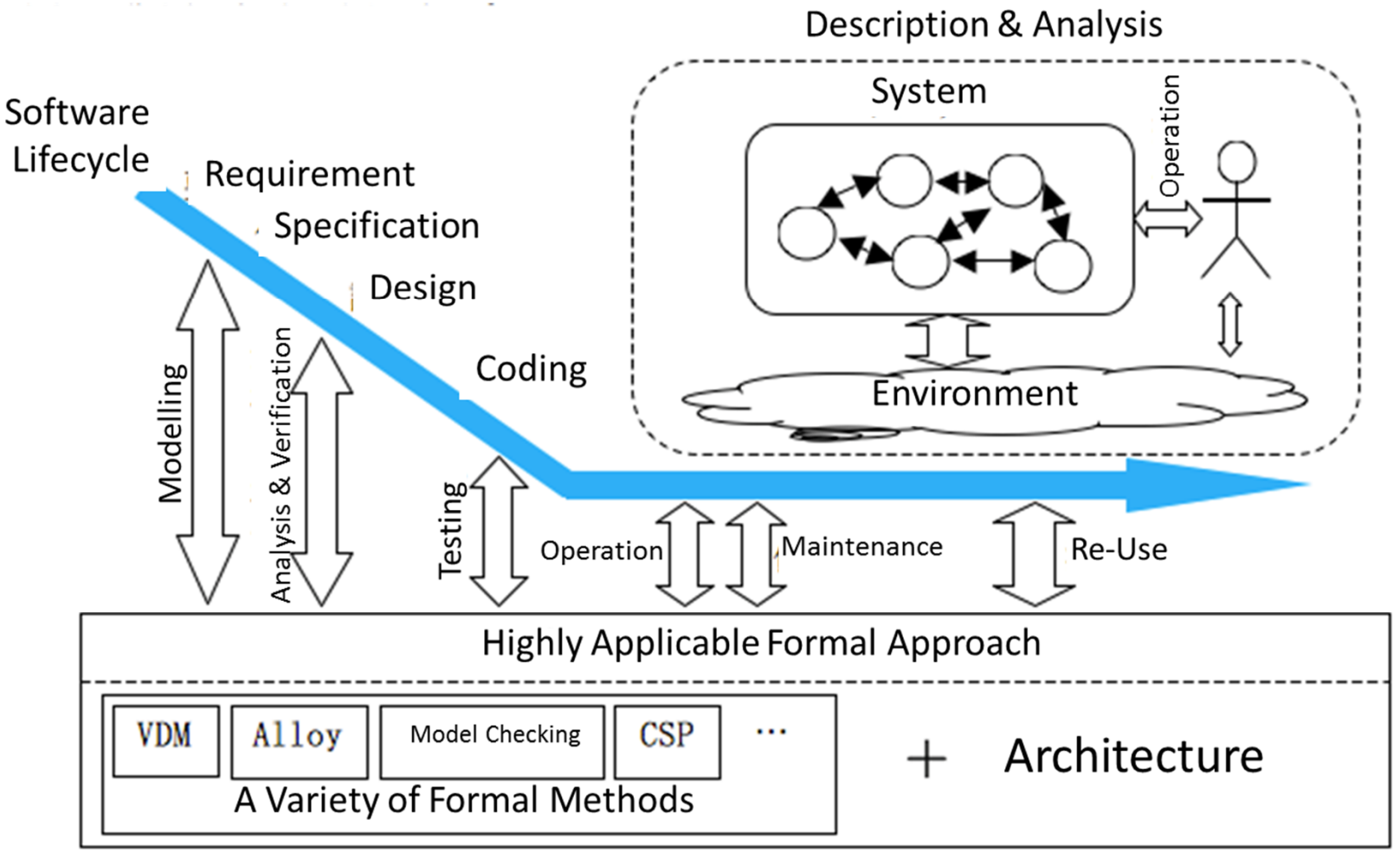
- Background
- New Project in Japan: Architecture Oriented one
- Sub-Topic: Software Process with Formal Method
- Preliminary Evaluation of Initial Trial
- Concluding Remarks

Our New Project

- Leader: Prof. Keijiro Araki
  - 5-year project of Scientific Research (S) of Grant-in-Aid for Scientific Research in Japan (MEXT*), with 122.2 million Yen budget and 7 researchers.

    *Ministry of Education, Culture, Sports, Science & Technology
- Title: Architecture Oriented Formal Approaches to High Quality Software Development
  - to propose effective and usable formal methods to cover the whole software lifecycle including operation and maintenance phases.

Architecture Oriented Formal Approaches to High Quality Software Development



Formal Approach Effective at Each Stage in Software Lifecycle

Research Topics

1. Effective formal techniques for modeling and analyzing complicated IT systems and case studies of their applications.
2. **Reference models of software development processes with formal methods.**
3. Architecture oriented formal approaches to treat complicated systems of systems including environment and operation phases
4. Development of support tools

Outline

- Background
- New Project in Japan: Architecture Oriented one
- Sub-Topic: Software Process with Formal Method
- Preliminary Evaluation of Initial Trial
- Concluding Remarks

Process / Formal Methods in Architecture

- Architecture is both the **process** and product of planning, designing and construction. (Wikipedia)
- "Software Architecture in Practice" (L. Bass, et.al., 2003)
  - Architecture is the vehicle for stakeholder **communication**. Software architecture represents a common **abstraction** of a system that stakeholders can use as a basis for **mutual understanding** and communication.
  - Architecture manifests the earliest set of design decisions. These early decisions are the most difficult to get correct and the hardest to change later in the development process.
  - Architecture can be a transferable and reusable model, abstraction of a system. While code reuse is beneficial, reuse at the architecture level provides tremendous leverage for systems with similar requirements.

Software Process

- We accumulate case studies of applying formal methods into conventional/standard software development processes,
- Then, we present as process models effective to real development projects.
- Our first trial uses VDM/Overture and PSP (Personal Software Process) *
  - well-defined -> easy to analyze
  - customizable process -> easy to extend with formal methods

* Service Mark of Carnegie Mellon University, Software Engineering Institution

A Standard Process, PSP

Intended as a course for personal skills to implement CMMI, providing an improvement framework that helps us to control, manage, and improve the way we work.

- **Phases**: plan, detailed design, detailed design review, code, code review, compile, unit test, and post mortem, with a set of associated **scripts, forms, and templates**.
- **Data**: time and defects injected and removed for each phase, size, size and time estimating error, cost-performance index, defects injected and removed per hour, personal yield, appraisal and failure cost of quality, and the appraisal to failure ratio.

Outline

- Background
- New Project in Japan: Architecture Oriented one
- Sub-Topic: Software Process with Formal Method
- Preliminary Evaluation of Initial Trial
- Concluding Remarks

Preliminary Experiment

Concerns

- Can we (non-expert) figure out how to use VDM in a guided manner?
- Does this approach of sub-topic fit to our main project?

Setting

- A graduate student : not familiar with formal methods
  - Basic experience of programming and software development project such as PBL (Project-Based Learning)
- Defect prevention based on personal historical data

Introduce VDM in Establishing Process

PSP course structure

- PSP0*: measurement (2 exercises)
- PSP1*: estimate (2)
- PSP2*: quality (2 -)
  - very simple formal notation by default

Process extension

1. Collect baseline data: process data of PSP0* and PSP1*
   - Time, defect (type, fix time, ..)
2. Analyze baseline data and consider how to improve
3. Start using FM from PSP2

Defects

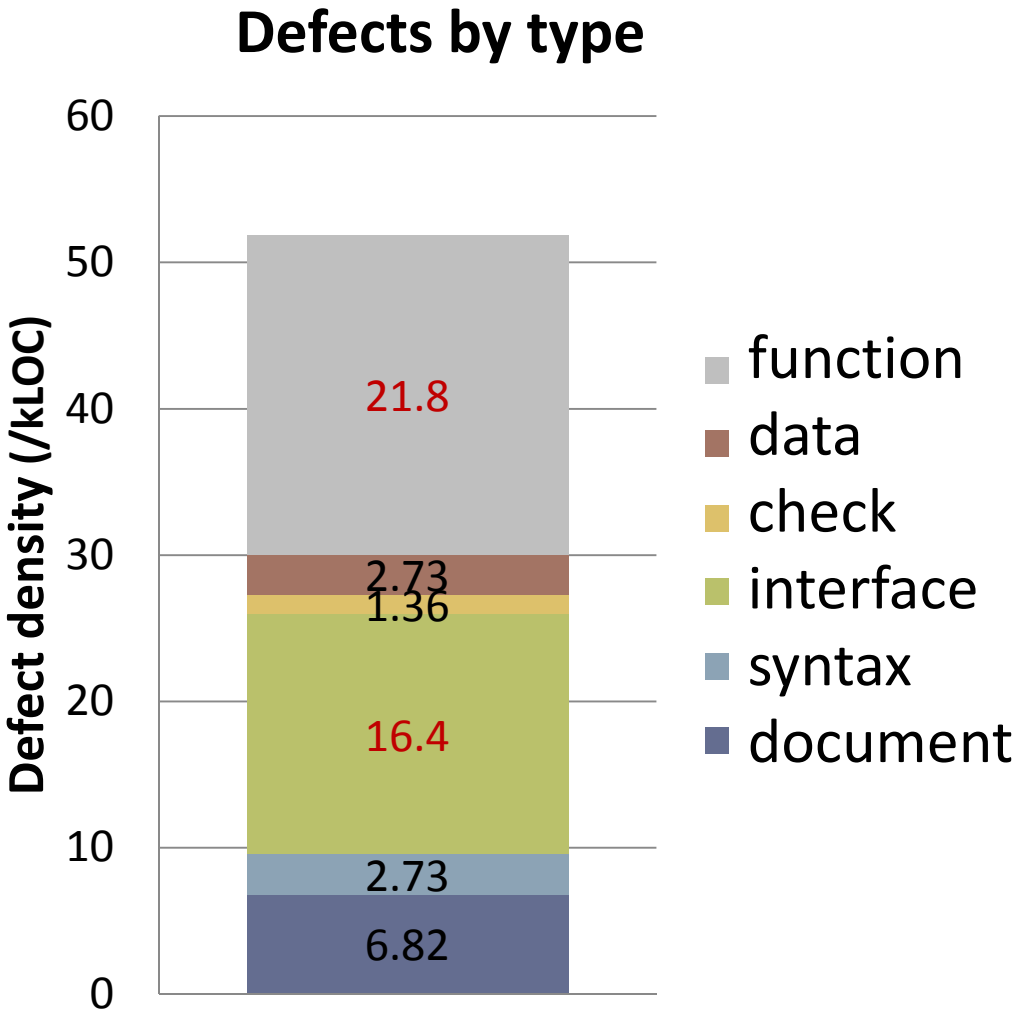
The defect data:

- defect type
- time to find & fix defect
- defect injection phase
- defect removal phase
- brief explanation

Defect type:

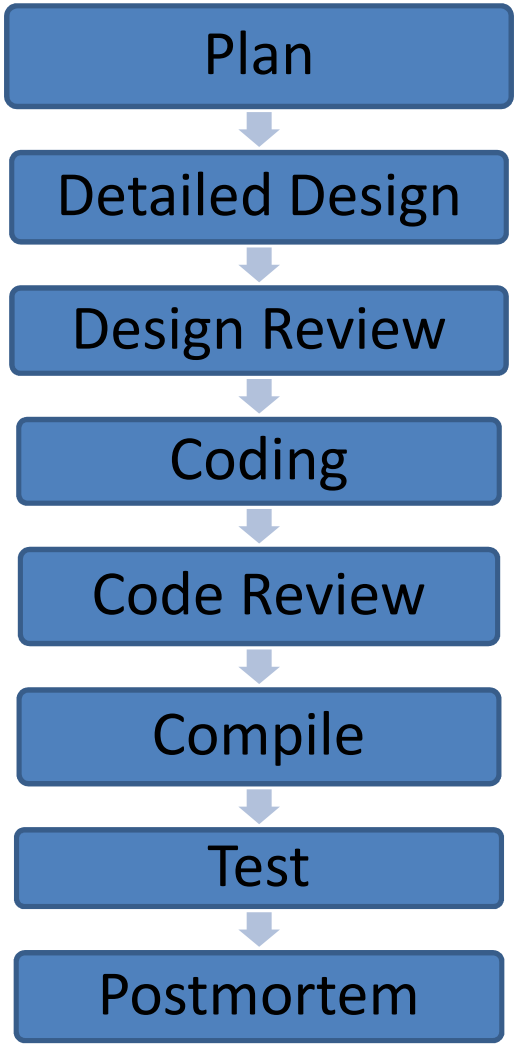
- Documentation
- Syntax
- Build, Package
- Assignment
- Interface
- Checking
- Data
- Function
- System
- Environment

Baseline Process Data (defects)



	target type		ave. fix time (mi)
I-1	Interface	insufficient refinement	15.8
F-1	Function	loop control	10.3
F-2	Function	logic	6.8

Process Extension



Base process: PSP 2*

- design
  - UML
  - VDM++
- design review
  - check list (manual check), and
  - tool

Added Steps

[Step 1:] Prevention of I-1 type defects

- Write signature of methods in VDM++ in detailed design phase
- use VDMTools for syntax and type check

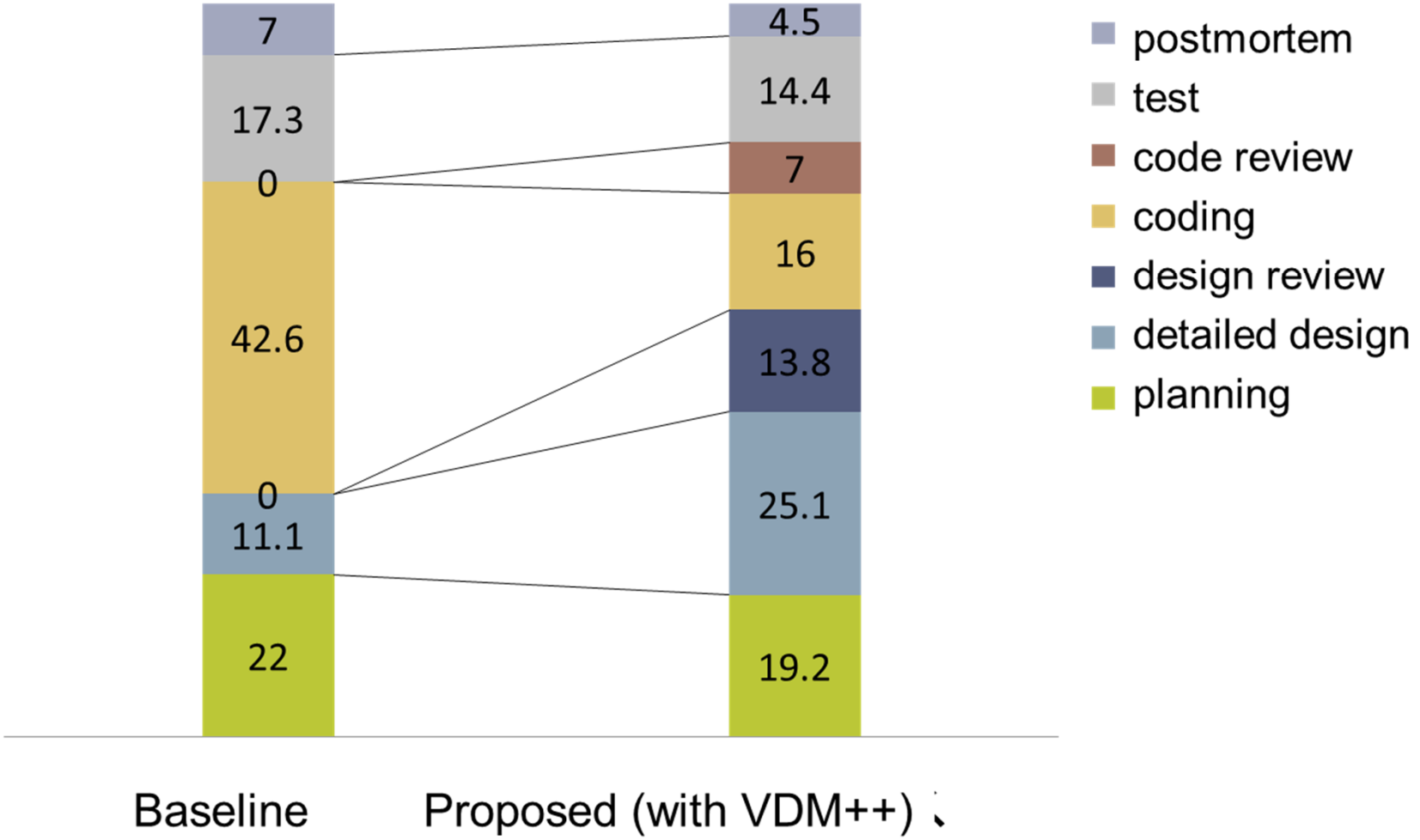
[Step 2:] Prevention of F-1 type defects

- Describe sequence handling part in VDM++

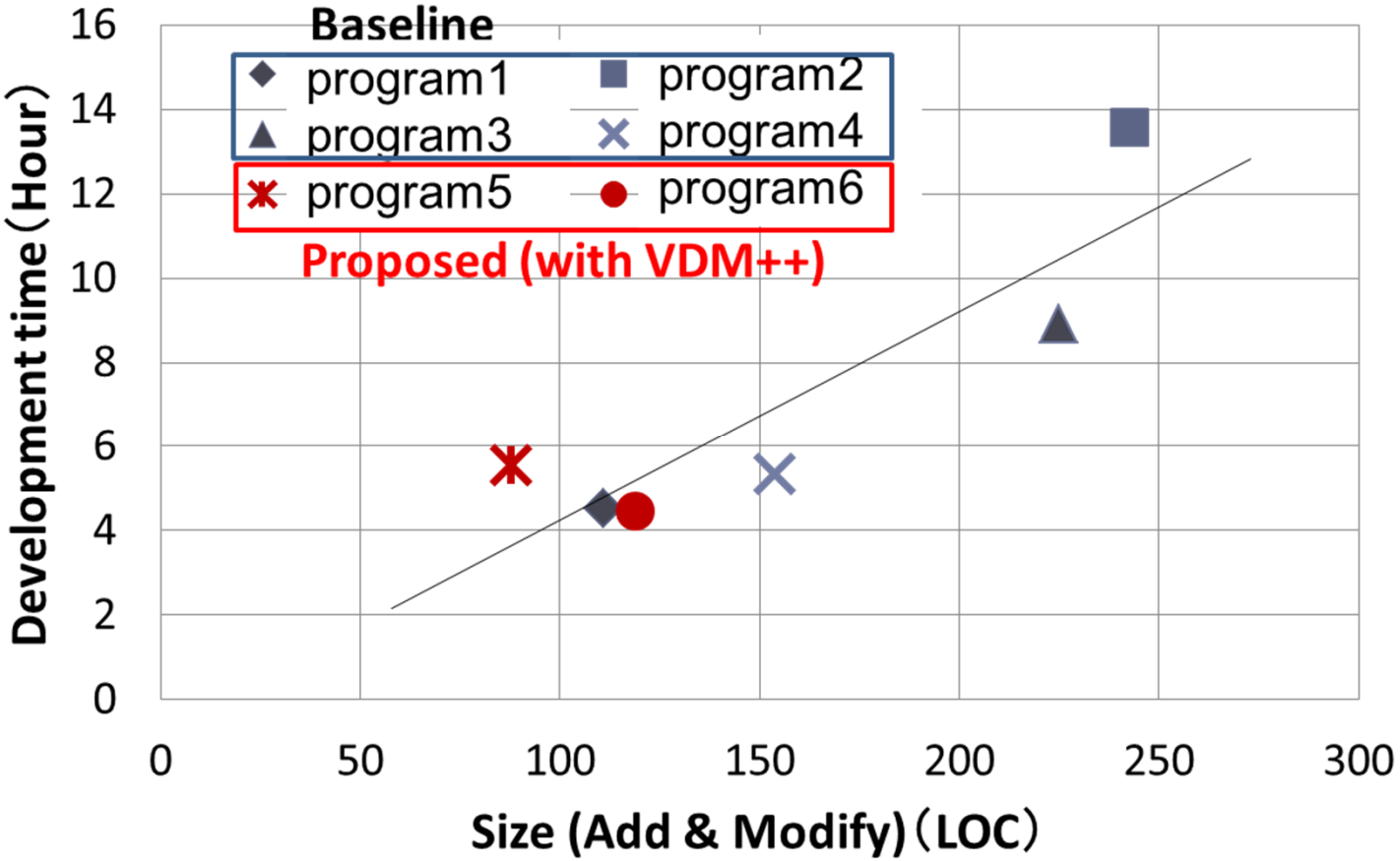
[Step 3:] Prevention of F-2 type defects

- Write explicit VDM++ specification for selected part
- Use animation of VDMTools.

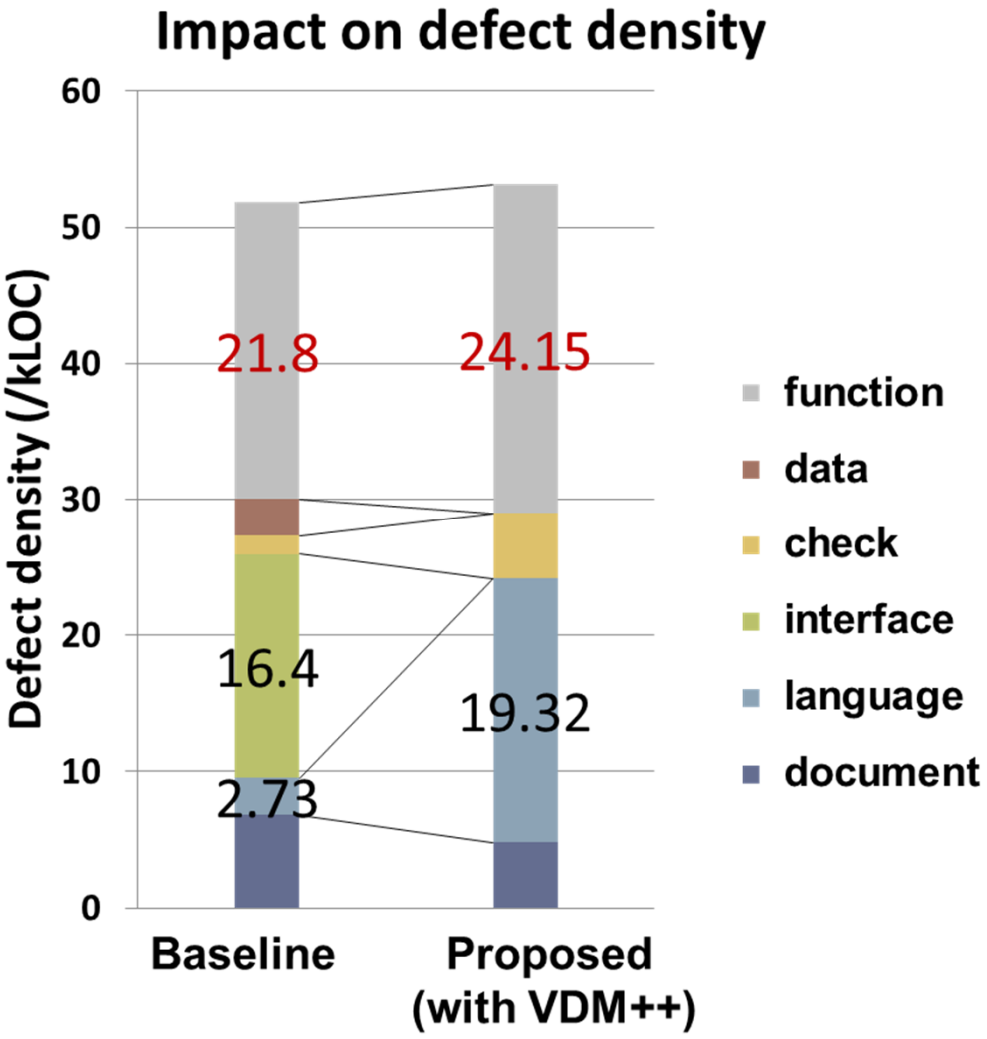
Time Distribution



Productivity



Defect Density



Interface type

- none

Function type

- baseline
  - mainly (87.5%) removed in Test
- proposed
  - mainly remove in design review
  - only 20% in Test

Total

- no reduction
  - language proficiency?

Outline

- Background
- New Project in Japan: Architecture Oriented FM
- Sub-Topic: Software Process with Formal Method
- Preliminary Evaluation of Initial Trial
- Concluding Remarks

Concluding Remarks

- Introducing VDM in a guided manner
  - We can use our baseline data in extending the base process
  - We can more easily produce rigorous detailed design if an architecture oriented approach uses formal methods. (if documents are already written in a formal notation before starting a development process at a personal level.)
- Fitting to our main project
  - Rigorously describing interfaces is effective in reducing defects, and we expect describing interfaces rigorously in early stages is facilitated in an architecture-oriented formal approach.
- Future work: Other processes, formal methods, topics, ...