JAIST/TRUST - AIST/CVS joint workshop on verification technology
3rd VERITE
2006 / 11/ 28（Tue.）

構成管理モデルFMCのAlloyによる検証

Hiroaki Tanizaki, Takuya Katayama
School of Information Science
JAIST

Agenda

- Background
- Our approach
- FMC model
- Verification of FMC model by Alloy
- Example：Ｗｅｂ Application
- Future Work

Background

- The requirement to a software system changes.
  – ex) functionality expansion, execution environment change
- The configuration of software continues change.
  – This is software evolution
  – Configuration management is important to software evolution.
- Then, Systematic and formal configuration management is required.

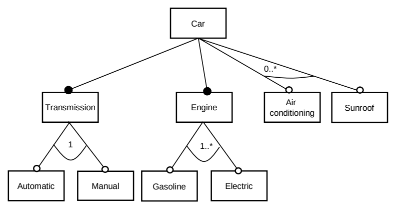
Our approach (1)

- We propose the configuration management model.
  – management target
    - Configuration of application and execution environment of the software system.
  – verification of configuration
    - relation of features
- Model
  – application layer ･･･ Feature
  – environment ･･･ Module
  – relation between application and environment ･･･ Connection

Our approach (2)

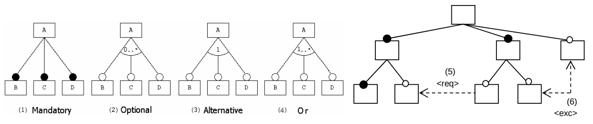
- Modeling
  – Our model based on FODA-model
    - Because we deal with configuration management and change, variable point should be described in model.
  – FODA Feature-Oriented Domain Analysis
    - Domain Analysis method for Software Product Line
    - Focus on commonality and variability of a domain.
    - Model notation
      – mandatory, optional, alternative，or
      – requires, excludes

FODA model



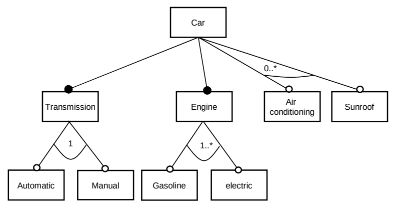
- Simple Car
  – (Car, Transmission, Automatic, Manual, Engine, Gasoline, Electric, Air conditioning, Sunroof)

Notation



(1)Mandatory
The feature must be included in an instance.
(2)Optional Features
The feature may or may not be included in an instance.
(3)Alternative Features
Exactly one feature can be included in an instance.
(4)Or
One or more features can be included in an instance.
(5)req : Requires
A feature requires some other features included in an instance.
(6)exc : Excludes
Both of features are not included in an instance.

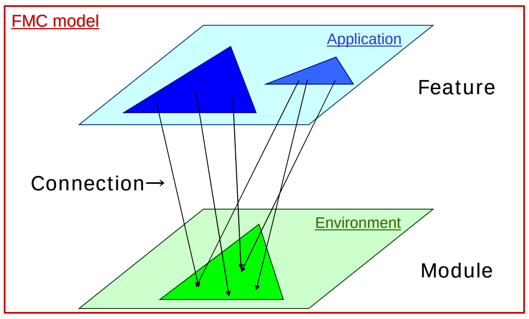
FODA model



- Simple Car
  - Possible configuration
    - (Car, Transmission, Automatic, Engine, Gasoline)
    - (Car, Transmission, Automatic, Engine, Gasoline, Air conditioning)
    - (Car, Transmission, Automatic, Engine, Electric)
    - (Car, Transmission, Automatic, Engine, Gasoline, Electric, Air conditioning) …

FMC model

Relationship Feature and Module



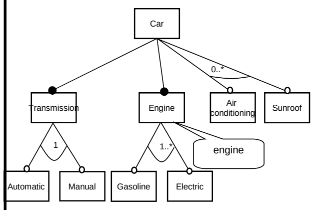
FMC model

- FMC = (Fm, Mm, Cm)
  - Fm：Feature model
    - Configuration of application
  - Mm：Module model
    - Configuration of environment
  - Cm：Connection model
    - Feature → Module

Syntax of Fm

- Fm =（F, fr, Rman, Ropt, Ralt, Ror, Rreq, Rexc Spec, Mfs）
  - F ・・・ Set of Feature name
  - froot ・・・ root node, froot ∈ F
  - Rman ・・・ Subset of F×F, child node is Mandatory
  - Ropt ・・・ Subset of F×F, child node is Optional
  - Ralt ・・・ Subset of F×F, child node is Alternative
  - Ror ・・・ Subset of F×F, child node is Or
  - Rreq ・・・ Subset of F×F, Requires relation
  - Rexc ・・・ Subset of F×F, Excludes relation
  - Spec ・・・ Set of specification name
  - Mfs ・・・ $F \to 2^{Spec}$

Example



Fm = (
F = { Car, Transmission, Automatic, Manual, Engine, Gasoline, Electric, Air conditioning, Sunroof}
floot : Car
Mman = { (Car, Transmission), (Car, Engine) }
Malt = { (Transmission, Automatic), (Transmission, Manual) }
Mor = { (Engine, Gasoline), (Engine, Electric) }
Mopt= { (Car, Air conditioning), (Car, Sunroof) }
Spec={mission, auto, manual, engine …}
Mfs(Engine) = engine …
)

13

Syntax of Mm

- Mm = （M, mr, Mman, Mopt, Malt, Mor, Mreq, Mexc, Conf, Mmc）
  - M ・・・ Set of Module name
  - mroot ・・・ root node, mr ∈ M
  - Mman ・・・ Subset of M×M, child node is Mandatory
  - Mopt ・・・ Subset of M×M, child node is Optional
  - Malt ・・・ Subset of M×M, child node is Alternative
  - Mor ・・・ Subset of M×M, child node is Or
  - Mreq ・・・ Subset of M×M, Requires relation
  - Mexc ・・・ Subset of M×M, Excludes relation
  - Conf ・・・ Set of Configuration name
  - Mmc ・・・ $M \rightarrow 2^{Conf}$

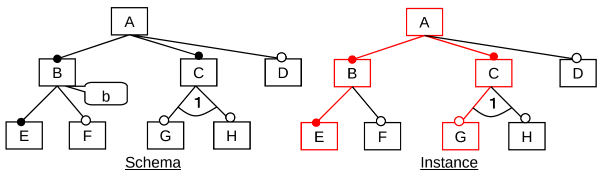
14

FMC : Schema and Instance

- Schema
  - Fm，Mm，Cm
  - All the structure members of the system are expressed.
- Instance
  - Subset of Fm and Subset of Mm
    - The root node must be included.
    - F_{ins} ・・・ Instance of Fm $F_{ins} \subseteq F$
      - froot ∈ F_{ins}
    - M_{ins} ・・・ Instance of Mm $M_{ins} \subseteq M$
      - mroot ∈ M_{ins}
  - The composition of a system is expressed.
  - The instance is generated by requirement.

15

FMC : Schema and Instance



F = { A,B,C,D,E,F,G,H } ⇨ F_{ins} = { A,B,C,E,G }
Mfs(B) = {b}
Requirement = { A,B,C,E,G}

- The relation (Rman, Ropt ...) of a node expresses the restrictions when generating an instance.

16

Verification of FMC model

verification target

- Consistency check
  - Does instance generate without inconsistency?
  - Does instance satisfy the specification? （specification : Spec, Conf）
    - An invalid requirement or inconsistent schema can be discovered.

⇩

We use the Alloy Analyzer for verification of FMC model.
Alloy can generate instances and check specification of a model.

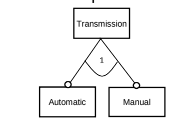
Alloy Homepage http://alloy.mit.edu/

17

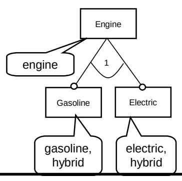
Consistency

- What's inconsistency?

<example>



Feature:REQ = { Transmission, Automatic, Manual }
⇒ REQ is invalid
Automatic and Manual are Alternative
⇨ correct the requirement



∗Both “Gasoline" and “Electric" are needed for "hybrid"
Spec:REQ = { engine, hybrid }
⇒ Engine, Gasoline ∈ *Instance*
The instance does not satisfy the requirement.
⇨ correct the schema
ex)”Alternative” → “Or”

18

Verification of FMC-model by Alloy

FMC model to Alloy model

- Purpose
  - Check the schema and requirement
  - Find an instance
- In order to use FMC model by Alloy
  - FMC meta model
    - Describe the definition of Fm, Mm , Cm and R-restriction
  - FMC schema
    - Describe Fm, Mm and Cm

FMC meta model（1/2）

- sig Spec, Conf {}
  - Define Spec and Conf
- sig Feature { parent : lone Feature, func : set Spec }
  - Define Feature
    - Feature has “parent” and “func (set of Spec)”
      - parent : Feature × Feature
      - func : Feature × 2^{Spec} (Mfs)
- sig Module { layer : lone Module, config : set Conf }
  - Define Module
    - Module has “layer” and “config (set of Conf)”
      - layer : Module × Module
      - config : Module × 2^{Conf} (Mmc)
- sig Fins { include : set Feature, function : set Spec }
  - Instance of Feature
- sig Mins { construction : set Module, configuration : set Conf }
  - Instance of Module

FMC meta model（2/2）

```
F×F （1/2）
•   pred Man (i : Ins, sf : set Feature) {
            all f : sf | f.parent in i.include => all f : sf | f in i.include
            some f : sf | f.parent !in i.include => all f : sf | f !in i.include
    }

•   pred Opt (i : Ins, sf : set Feature) {
            all f : sf | f.parent !in i.include => all f : sf | f !in i.include
    }

•   pred Alt (i : Ins, sf : set Feature) {
            all f : sf | f.parent in i.include => one f : sf | f in i.include
            some f : sf | f.parent !in i.include => all f : sf | f !in i.include
    }
•   pred Or (i : Ins, sf : set Feature) {
            all f : sf | f.parent in i.include => some f : sf | f in i.include
            some f : sf | f.parent !in i.include => all f : sf | f !in i.include
    }

•   pred Req (i : Ins, f1 : Feature, sf : set Feature) {
            f1 !in sf
            f1 in i.include => all f : sf | f in i.include
    }

•   pred Exc (i : Ins, f1 : Feature, f2 : Feature) {
            f1 != f2
            f1 in i.include => f2 !in i.include
    }
M×M : same as F×F
```

FMC Schema

Describe Fm and Mm

- F*ins*, F, Spec
  - one sig “*name of instance*” extends Fins {}
  - one sig “*name of feature*” extends Feature {}{
    - parent = “*parent feature's name*” // F×F
    - func = “*Spec name*” // Mfs
  - }
    - Describe all Features
  - one sig “*name of spec*”extends Spec {}
    - Describe all Spec
- F×F
  - fact { Man(F*ins*, f1), Opt(F*ins*, f2+f3) … }
    - f1, f2, f3 ∈ F
- Mins, M, Conf
  - same as Fins, F, Spec

How to verification

- pred () {} , run
  - Describe the feature and module which should be contained in an example.
  - search for the instance of predicate
- assert{}, check
  - Describe the feature and module which must be contained in an example.
  - search for counterexample to assertion

Example : Web Application
Simple address book

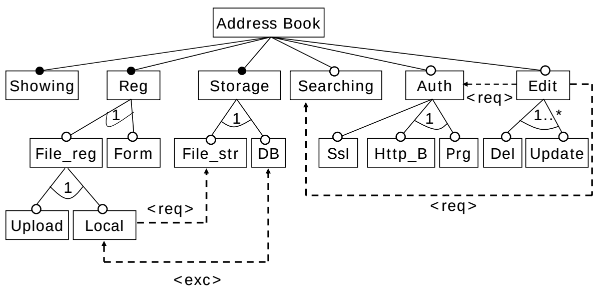
25

Simple Address book

- Assumed features
  - register, search, delete, update
  - two or more registration methods
    - local file, upload file, registration form
  - authentication of user
  - two or more storage methods
    - file, Database
- Assumed environment
  - Web Server : Apache
  - ServerSideProgram : PHP, Ruby, Perl, JSP …
  - DBMS : MySQL, PostgreSQL …

26

Feature model



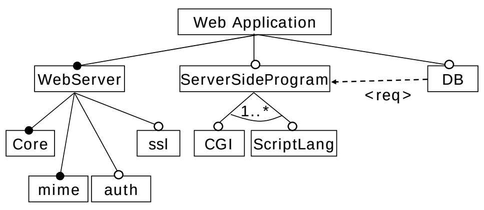
27

Feature schema description

```
Address Book
// Instance
one sig AB extends Ins {}
// Feature
one sig ab extends Feature {}{ no parent                no func    }
one sig Sw extends Feature {} {           parent = ab        func = show            }
one sig Rg extends Feature {} {           parent = ab        func = regest          }
                                              :
one sig Upd extends Feature {} {          parent = Ed        func = update          }
// Spec
one sig show, regest, file_r, form, upload, local, storage, file_s, db extends Spec {}
one sig search, authe, ssl, basic_a, prog_a, edit, delete, update extends Spec {}
// R
fact {
     Man(AB, Sw+Rg+Str)        Opt(AB, Search+Auth+Ed)
     Alt(AB, File_r+Form_r)    Alt(AB, Upl+Loc)           Alt(AB, File_s+Db)
     Opt(AB, Secure)           Alt(AB, Ba_au+Pr_au)       Or(AB, Del+Upd)
     Req(AB, Loc, File_s)      Req(AB, Ed, Search+Auth)
     Exc(AB, Loc, Db)
}
```

28

Module model



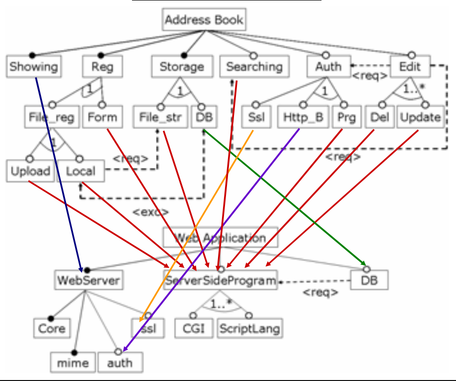
29

Module schema description

```
Web Application Environment
// Instance
one sig WA extends Ins {}
// Module
one sig wa extends Feature  {}{           no parent          no func    }
one sig WServer extends Feature {} {      parent = wa        func = webserver    }
one sig M_core extends Feature {} {       parent = WServer   func = core_m       }
                                              :
one sig DBMS extends Feature {} {         parent = wa        func = dbms_m       }
// Conf
one sig webserver, core_m, mime_m, auth_m, ssl_m extends Spec {}
one sig ssprogram, cgi_m, slang_m, dbms_m extends Spec {}
// R
fact {
     Man(WA, WServer)                Opt(WA, SSP+DBMS)
     Man(WA, M_core+M_mime)          Opt(WA, M_auth+M_ssl)
     Alt(WA, Cgi+Slang)              Req(WA, DBMS, SSP)
}
```

30

Connection



31

Connection

```
// Connection
fact {
    Cfm(Sw, WServer)    Cfm(Upl, SSP)
    Cfm(Loc, SSP)       Cfm(Form_r, SSP)
    Cfm(File_s, SSP)    Cfm(Db, DBMS)
    Cfm(Search, SSP)    Cfm(Secure, M_ssl)
    Cfm(Ba_au, M_auth)  Cfm(Pr_au, SSP)
    Cfm(Del, SSP)       Cfm(Upd, SSP)
}
```

32

verification (1)

- Find valid instance
  pred ABinstance0() {}
  run ABinstance0 for 1 REQ, 1 Ins, 1 Env, 18 Feature, 10 Module,
  17 Spec, 9 Conf

```
– include : set fmc_adbook/Feature =
–   {AB_0 -> {Auth_0, Db_0, Del_0, Ed_0, Form_r_0, Pr_au_0, Rg_0, Search_0, Str_0, Sw_0, Upd_0, ab_0}}
– function : set fmc_adbook/Spec =
–   {AB_0 -> {authe_0, db_0, delete_0, edit_0, form_0, prog_a_0, regist_0, search_0, show_0, storage_0, update_0}}
– sig Env extends univ = {WS_0}
– construction : set fmc_adbook/Module =
–   {WS_0 -> {Cgi_0, DBMS_0, M_auth_0, M_core_0, M_mime_0, SSP_0, WServer_0, ws_0}}
– configuration : set fmc_adbook/Conf =
–   {WS_0 -> {auth_m_0, cgi_m_0, core_m_0, dbms_m_0, mime_m_0, ssprogram_0, webserver_0}}
```

33

verification (2)

- Find selected valid instances

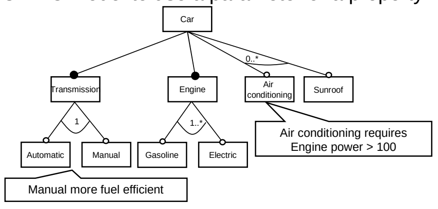
```
pred ABinstance1() {
        AB.include = ab+Sw+Rg+Form_r+Str+Db+Search
        WS.construction =ws+WServer+M_core+M_mime+SSP+Slang+DBMS
}
run ABinstance1 for 1 REQ, 1 Ins, 1 Env, 18 Feature, 10 Module,17 Spec, 9 Conf
```

```
• sig Ins extends univ = {AB_0}
•   include : set fmc_adbook/Feature =
•     {AB_0 -> {Db_0, Form_r_0, Rg_0, Search_0, Str_0, Sw_0, ab_0}}
•   function : set fmc_adbook/Spec =
•     {AB_0 -> {db_0, form_0, regist_0, search_0, show_0, storage_0}}
• sig Env extends univ = {WS_0}
•   construction : set fmc_adbook/Module =
•     {WS_0 -> {DBMS_0, M_core_0, M_mime_0, SSP_0, Slang_0, WServer_0, ws_0}}
•   configuration : set fmc_adbook/Conf =
•     {WS_0 -> {core_m_0, dbms_m_0, mime_m_0, slang_m_0, ssprogram_0, webserver_0}}
```

34

Future works

- Extends FMC model to use a parameter or a property.



- Dynamic reconfiguration
  - by REQ, Connection, Parameter and Property

35