О построении формальной модели управления доступом QP ОС

НТП «Криптософт», г. Пенза
М.А. Алехина,
Васин А.В.,
Афонин А.Ю.,
Егоров В.Ю.

Основные элементы модели

Множество субъектов системы: $Subjects$

Множество пользователей: $Users \subset Subjects$

Множество групп пользователей: $Groups \subset Subjects$

Отношение членства в группах: $GroupUser: Users \leftrightarrow Groups$

Множество дескрипторов (идентификаторов) безопасности: $Sids$

Отношение принадлежности дескриптора субъекту $SubjectSid: Sublects \rightarrowtail Sids$

Среди множества предопределенных групп выделим группу администраторов безопасности: $Admins \in Groups,\ GroupUser \rhd \{Admins\} \neq \emptyset$

Множество объектов: $Objects$

Множество объектов-контейнеров: $Containers$

$$Entities = Objects \cup Containers$$

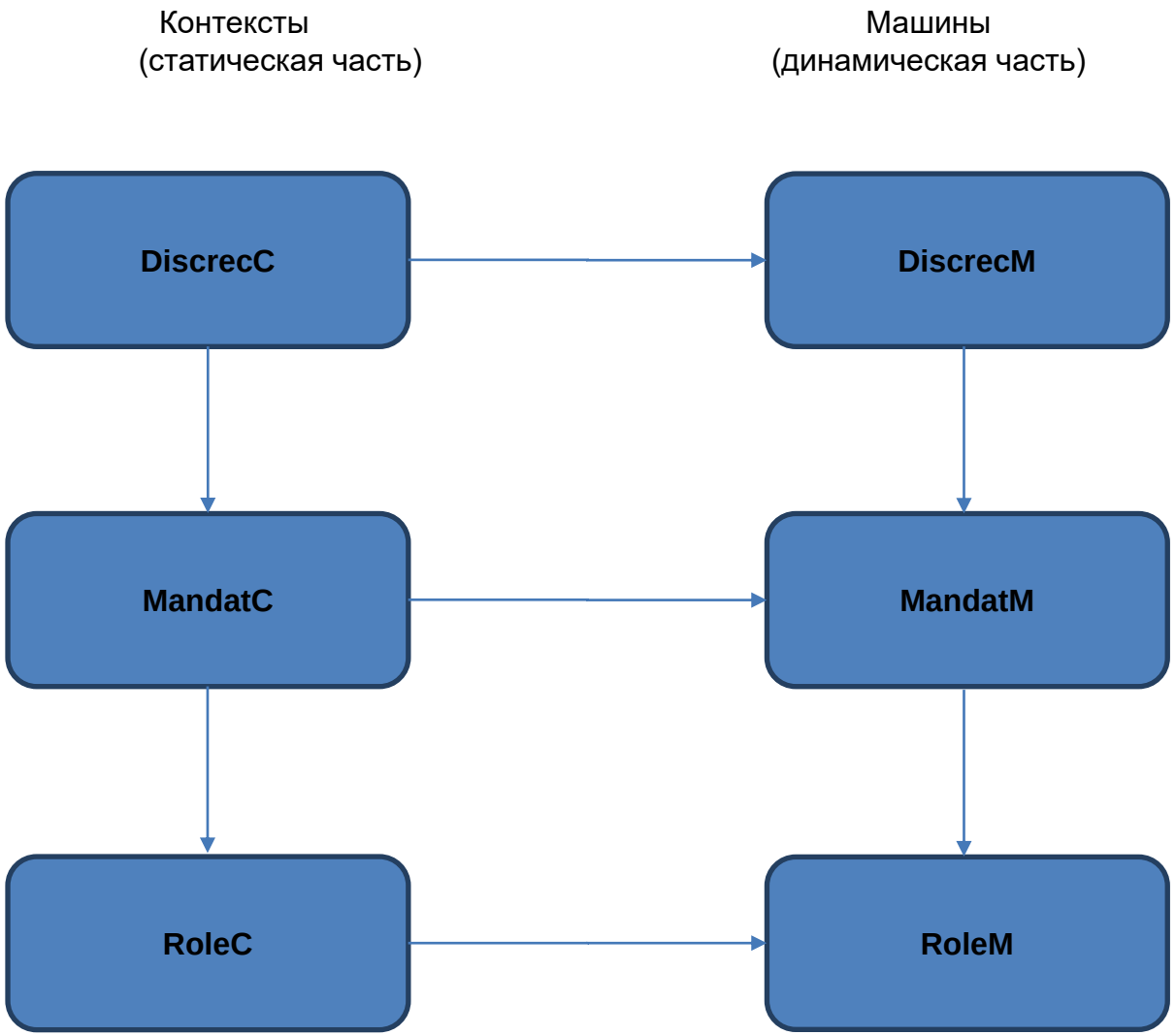
Множество типов: $Types$, каждый объект имеет тип: $Type: Entities \longrightarrow Types$

Множество прав доступа: $AccessRights$

Каждому типу прав применяемых к объекту, $Rd: Types \leftrightarrow AccessRights$

Владелец объекта определен отношением $EntityOwner: Entities \rightarrow Sids$

Схема модели в системе Rodin



Модель дискреционного контроля доступа

Сессии пользователей, выполнивших процедуру входа: $Sessions$
Множество маркеров доступа: $Tokens$
Пользователь, сессия и маркер связаны отношениями
$SessionUser: Users \to Sessions$ и $SessionToken: Sessions \rightarrowtail Tokens$
Каждому процессу/нити соответствует маркер: $GetToken: Entities \twoheadrightarrow Tokens$.
Каждый маркер связан с дескриптором безопасности:
$TokenUser$: $TokenUser: Tokens \twoheadrightarrow Sids$
Маркер безопасности содержит идентификаторы разрешенных/запрещенных групп:
$TokenRules$: $TokenRules: Tokens \leftrightarrow Sids \times BOOL$
Маркер ограниченным и олицетворенным:
$TokenIsRestricted: Tokens \to BOOL$ и $TokenIsImpersonated: Tokens \to BOOL$.
Множество списков контроля доступа: $DACLs$
Отношение сущностей и списков доступа объектов: $EntityDACL: Entities \twoheadrightarrow DACLs$
Отношение правил доступа $ACEs$:

$$ACEs: DACLs \twoheadrightarrow (Sids \times (AccessRights \times Types)) \twoheadrightarrow \mathbb{P}(RFS)$$

Описатели нити: $HandlesOfThread: Containers \leftrightarrow \mathbb{N}1$
Права, полученные нитью для объекта:

$$Handle: Containers \twoheadrightarrow (\mathbb{N}1 \times Entities \leftrightarrow AccessRights).$$

Модель дискреционного контроля доступа

Пусть $E \in Entities, token \in Tokens, rd \subseteq AccessRights)$

В операционной системе контроль доступа осуществляет функция $access\_check\ (E, token, rd)$:

Определение разрешенных прав:

1. $t = Type(E),\ r = Rd[t]$
2. $sid = TokenUser(token)$ и $G^0_{sid} = dom(ran(\{token\} \lhd TokenRules) \rhd \{TRUE\})$.
3. $S_o = \{sid\} \cup G^0_{sid}$
4. $dacl = EntityDACL(E)$ и $Rules_0 = S_o \times (r \times \{t\}) \lhd ACEs(dacl)$
5. $Allowed = dom(ran(dom(Rules \rhd FlagSetsWithTRUE)))$
6. $G^1_{sid} = dom(ran(\{token\} \lhd TokenRules))$.
7. $Rules_1 = G^1_{sid} \times (r \times \{t\}) \lhd ACEs(dacl)$.
8. $Denied = dom(ran(dom(Rules \rhd FlagSetsWithFALSE)))$

Множество прав доступа субъекта к объекту: $Allowed \backslash Denied$.

Тогда имеем:

$$access_{check(E,token,rd)} = \begin{cases} true, \text{при } EntityOwner(E) = TokenUser(token) \\ true, \text{при } rd \subseteq Allowed \backslash Denied \\ false, \text{в остальных случаях} \end{cases}$$

Модель мандатного контроля доступа

Пусть $(B = \{0,1,\dots,l\}, \leq)$, $l \leq 15$ - вполне упорядоченное множество

Иерархические категории: $B_k \subseteq B$, $k \leq 8$

Неиерархические категории: $C_m \subseteq \{0,1\}$, $m \leq 16$

Пусть $\Pi = B_1 \times B_2 \times \cdots \times B_k \times \{0,1\}^m$, $\alpha, \beta \in \Pi$ - классификационные уровни (КУ).

Каждому маркеру и каждому объекту приписан КУ.

Определение: α из П предшествует набору β, если каждая компонента α не больше соответствующей компоненты β (обозначение α ≤ β).

Определение: Два набора из множества Π сравнимы, если один из них предшествует другому. В противном случае наборы α и β несравнимыми.

Правила доступа формулируются следующим образом:

- если КУ субъекта равен КУ объекта защиты, то субъект может и «читать» и «писать» данные этого объекта;
- субъект может «читать» данные объекта, если КУ субъекта больше КУ этого объекта;
- субъект может «писать» данные объекта, если КУ субъекта меньше КУ объекта;
- если КУ субъекта и КУ объекта защиты несравнимы, то доступ запрещен.

Модель мандатного контроля доступа

Пусть $Rm = \{write, read\}$.

Пользователю присваивается множество КУ субъекта:

$UserMandatRule: Users \nrightarrow \mathbb{P}1(\Pi)$.

При создании маркера для $U \in Users$ маркеру ставится в соответствие КУ из множества $UserMandatRule(U)$: $TokenMandat: Users \nrightarrow \Pi$

Каждому объекту присваивается КУ: $EntityXACLMandat: Entities \nrightarrow \Pi$.

Проверка прав доступа $maccess\_check(E, U, Rm)$ - функция трёх переменных (объекта, пользователя и запрашиваемого права мандатного доступа), значения которой задаются следующим образом:

$$maccess\_check(E, U, \{write\}) = TokenMandat(U) \leq EntityXACLMandat(E)$$

$$maccess\_check(E, U, \{read\}) = TokenMandat(U) \geq EntityXACLMandat(E)$$

$$maccess\_check(E, U, Rm) = maccess\_check(E, U, \{write\}) \& maccess\_check(E, U, \{read\})$$

Модель ролевого контроля доступа

Ролевое управление доступом реализуется на основе понятий «пользователь», «группа пользователей» и механизма привилегий.

Набор привилегии - конечное определённое множество

$$Priveleges = \{p_1, p_2, \dots, p_n\}.$$

Для каждого субъекта $s \in Subjects$ задаётся множество привилегий при помощи отношения $SubjectPrivileges$: $Subjects \leftrightarrow Privileges$

Отношение доступных привилегий маркера:

$TokenPrivileges$: $Tokens \leftrightarrow Privileges \times BOOL$.

При создании маркера $token$ отношение $TokenPrivileges$ меняется следующим образом:

$$TokenPrivileges = TokenPrivileges \cup \{token\} \times (privileges \times \{FALSE\}),$$

где $privileges = ran((\{user\} \cup ran(\{user\} \lhd GroupUser)) \lhd SubjectPrivileges)$

Для использования привилегии p требуется ее «включение», т.е. отношение $TokenPrivileges$ должно содержать тройку $token \mapsto (p \mapsto TRUE)$.

Модель ролевого контроля доступа

Проверка привилегий субъекта в системе выполняется на основе функции $privilege\_check$, принимающей значение «истина» в случае, если требуемое множество привилегий принадлежит набору привилегий пользователя:

$$privilege\_check(token, p) = \\ = \begin{cases} true, \text{при } \{token\} \times (p \times \{TRUE\}) \subseteq TokenPrivileges \\ false, \text{при } \{token\} \times (p \times \{TRUE\}) \nsubseteq TokenPrivileges \end{cases}$$

Пусть $A \in \{a_1, a_2, \dots, a_n\}$ - множество системных вызовов, требующих от пользователя наличия особых прав.

Тогда отношение требования наличия привилегии:

$ActionPriveleges: A \leftrightarrow Privileges$.

Выполнение системного $system\_call(a, token)$ вызова разрешается при истинном значении функции $privilege\_check$:

$system\_call(a, token): privilege\_check(token, ActionPriveleges[a])\&action(a)$.

Мандатный контроль доверия

Уровни мандатного доверия:

$$MandatoryTrustLevels = \{normal, medium, high, full\}$$

Мандатный уровень доверия пользователя:

$$UserMandatoryTrustLevel: Users \rightarrow MandatoryTrustLevels$$

Уровни мандатного доверия привилегий:

$$PrivilegeMandatoryLevel: Privileges \rightarrow MandatoryTrustLevels$$

Мандатный контроль доверия

Базовый уровень доверия может быть повышен в результате интерактивного запроса:

$$rise\_confidence(user, privilege) =$$

$$\begin{cases} true, UserMandatoryTrustLevel(user) = \max(PrivilegeMandatoryLevel[privilege]), \\ \text{при положительном решении пользователя;} \\ false, \text{при отрицательном решении пользователя} \end{cases}$$

Функция проверки мандатного уровня доверия описывается следующим образом:

$$confidence\_check(user, privileges) =$$

$$\begin{cases} rise\_confidence(user, privilege), \\ \quad \text{при } \max(PrivilegeMandatoryLevel[privilege]) > UserMandatoryTrustLevel(user) \\ true, \\ \quad \text{при } \max(PrivilegeMandatoryLevel[privilege]) \leq UserMandatoryTrustLevel(user) \end{cases}$$

Системный вызов:

$$system\_call(a, token): confidence\_check(user, ActionPriveleges[a]) \& \\ privilege\_check(token, ActionPriveleges[a]) \,\&\, action(a)$$

Операции над объектами. Создание объекта

$$Entities' = Entities \cup \{object\}$$
$$Objects' = Objects \cup \{object\}$$
$$Type' = Type \cup object \mapsto object\_type$$
$$DACLs' = DACLs \cup \{dacl\}$$
$$ACEs'(dacl) = (Sids \times (ran(\{object\_type\} \lhd Rd) \times \{object\_type\})) \lhd ACEs(TokenDACL(token))$$
$$EntityDACL'(object) = dacl$$
$$EntityOwner'(object) = TokenUser(token)$$
$$Handle'(Parent(thread)) = Handle(Parent(thread)) \cup (\{n \mapsto object\} \times rights)$$
$$HandlesOfThread' := HandlesOfThread \cup \{thread \mapsto n\}$$
$$gettedRights' := gettedRights \cup (\{token \mapsto object\} \times rights)$$

Открытие объекта

$$Handle'(Parent(thread)) := Handle(Parent(thread)) \cup (\{n \mapsto object\} \times needed\_rights)$$
$$HandlesOfThread' := HandlesOfThread \cup \{thread \mapsto n\}$$
$$gettedRights' := gettedRights \cup (\{token \mapsto object\} \times needed\_rights)$$

Выполнение операции над объектом

Выполнение функции check_access при выполнении действий реализуется одноименным событием со следующим набором охранных условий:

$$thread \in Entities$$
$$entity \in Entities$$
$$type = Type(entity)$$
$$rights \subseteq Rd[\{type\}]$$
$$\exists n \cdot n \in ran(\{thread\} \lhd HandlesOfThread) \wedge (\{n \mapsto entity\} \times rights) \subseteq Handle\big(Parent(thread)\big)$$

Закрытие описателя

$$Handle'\big(Parent(thread)\big) = Handle\big(Parent(thread)\big) \setminus (\{n \mapsto entity\} \times Rd[\{type\}])$$

$$HandlesOfThread' = HandlesOfThread \setminus (\{thread\} \times dom(dom(Handle(Parent(thread))) \rhd \{entity\}))$$

Удаление объекта

$$Entities' = Entities \setminus \{entity\}$$

$$Objects' = Objects \setminus \{entity\}$$

$$DACLs' = DACLs \setminus \{dacl\}$$

$$ACEs' = \{dacl\} \lhd ACEs$$

$$EntityDACL' = \{entity\} \lhd EntityDACL$$

$$Type' = \{entity\} \lhd Type$$

$$GetToken' = \{entity\} \lhd GetToken$$

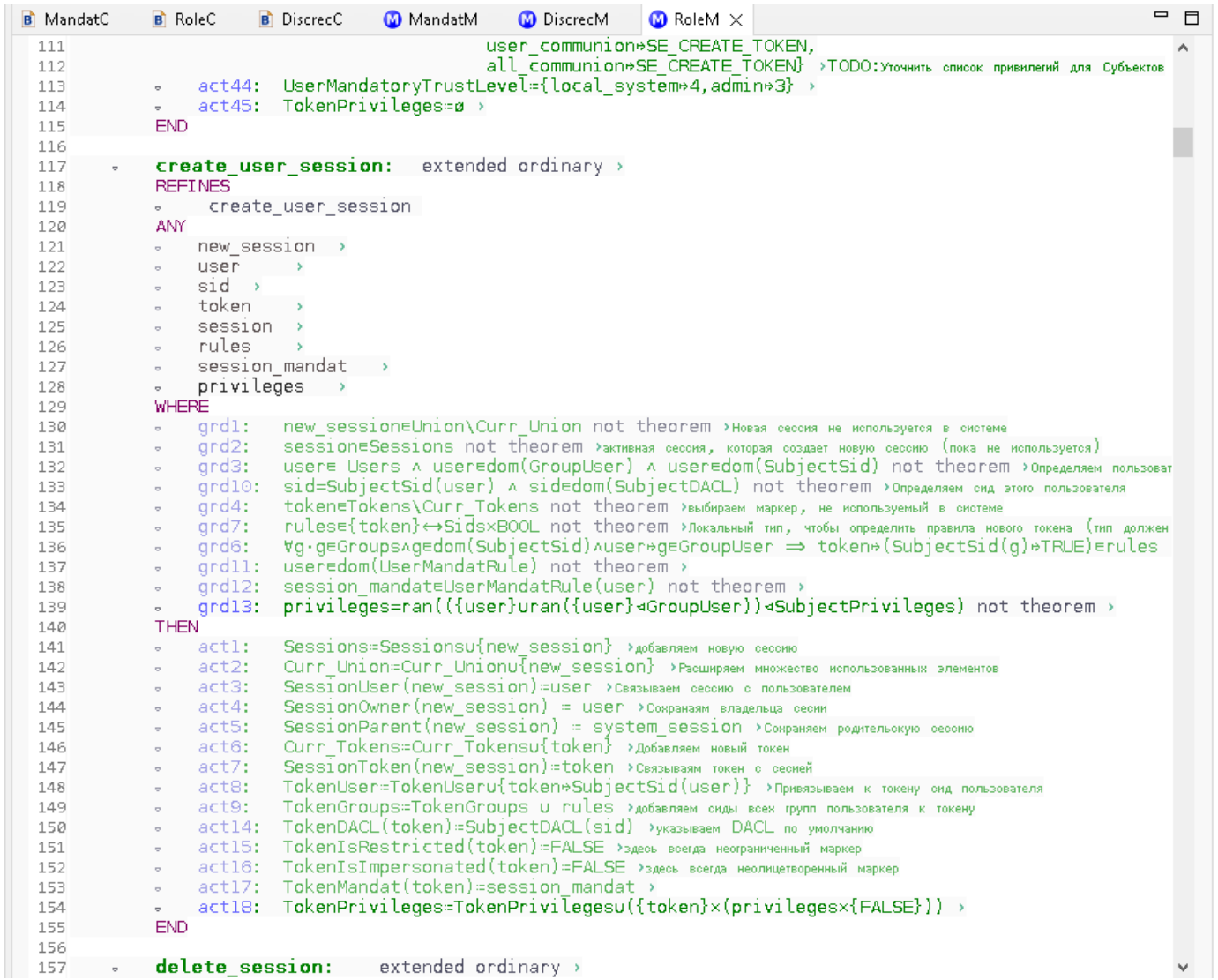
$$EntityOwner' = \{entity\} \lhd EntityOwner$$

$$Handle'(Parent(thread)) = (dom(Handle(Parent(thread))) \rhd \{entity\}) \lhd Handle(Parent(thread))$$

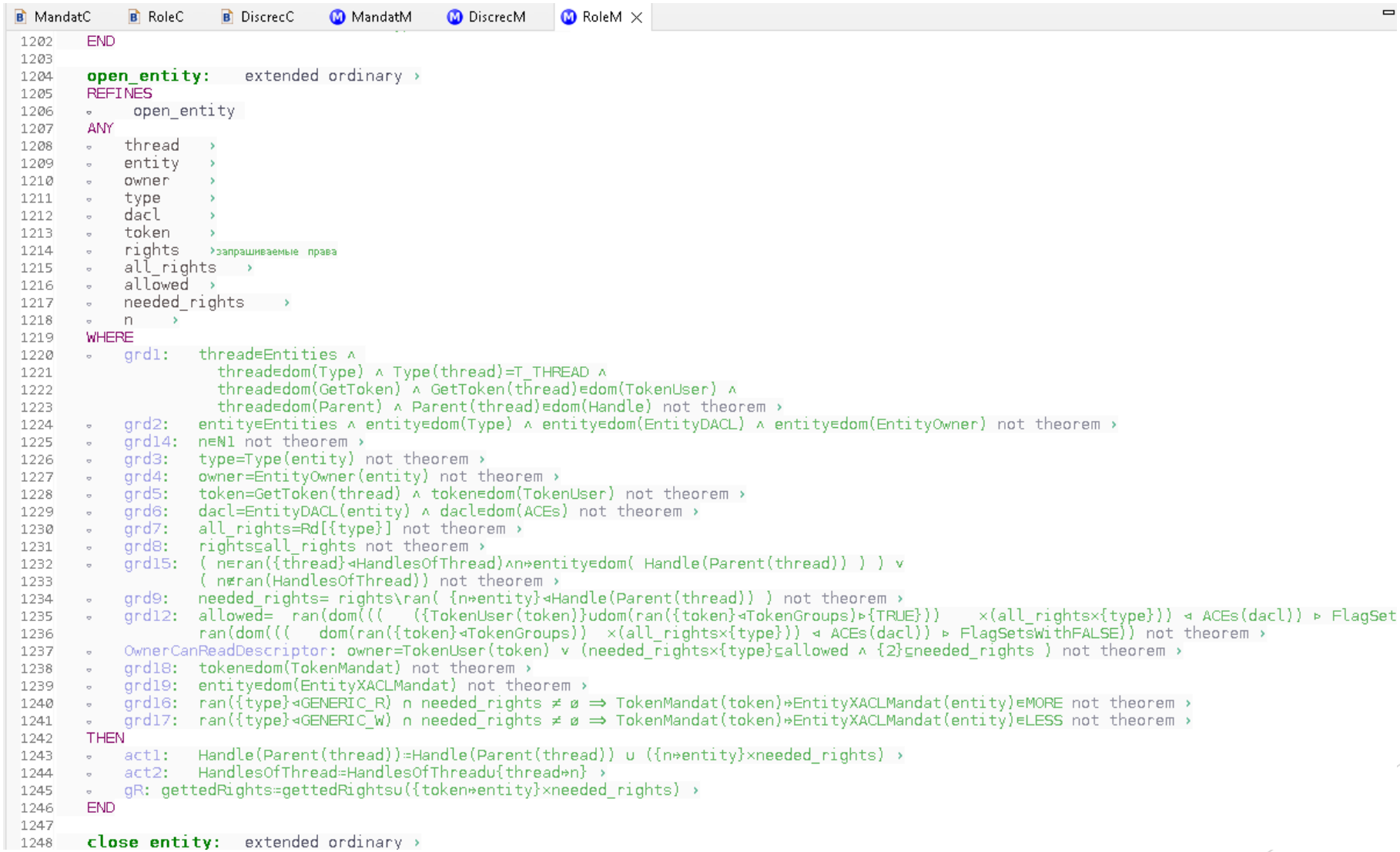
$$HandlesOfThread' = HandlesOfThread \setminus (\{thread\} \times dom(dom(Handle(Parent(thread))) \rhd \{entity\}))$$

$$gettedRights' = (Tokens \times \{entity\}) \lhd gettedRights$$

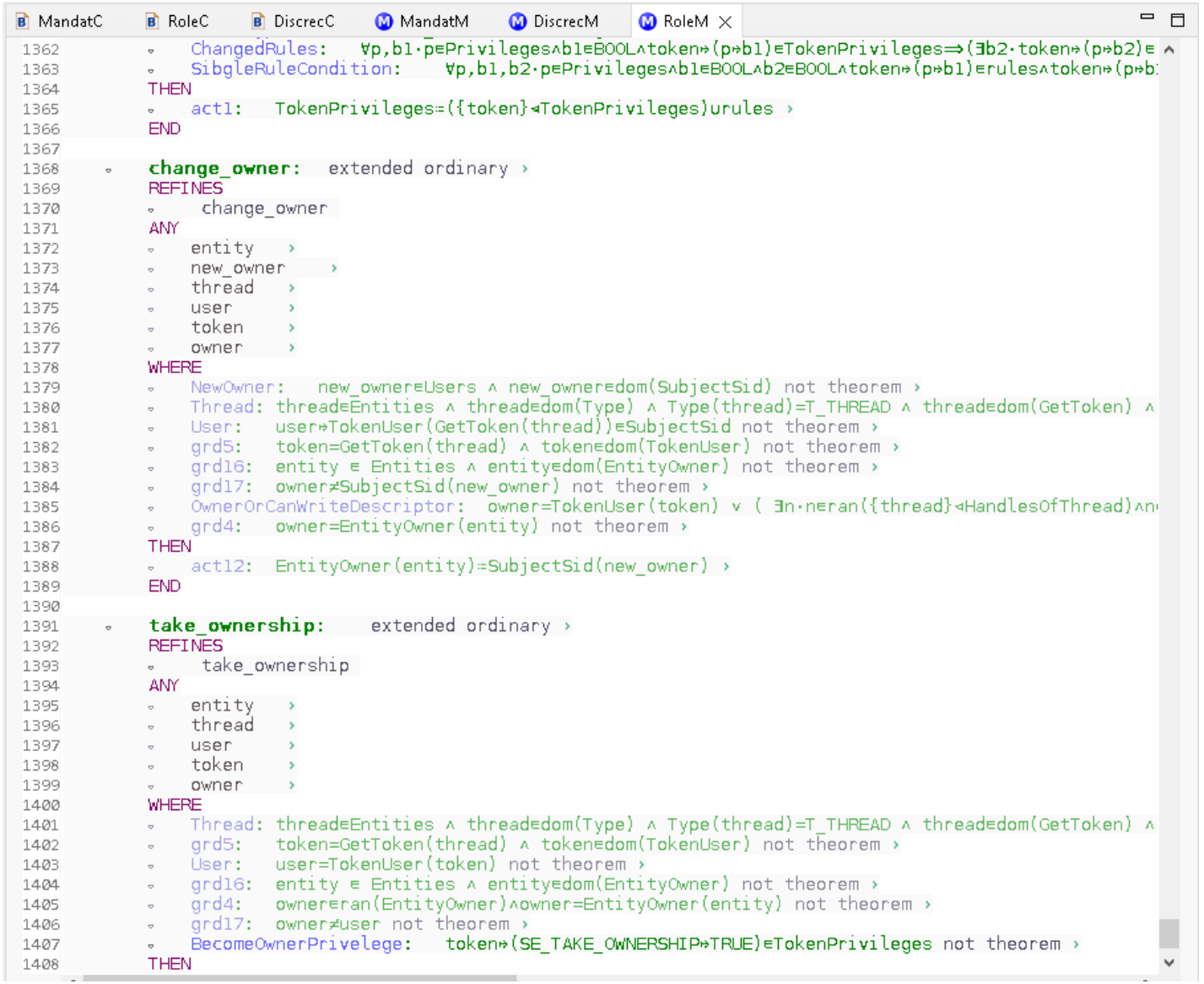
Модель в системе Rodin



Модель в системе Rodin

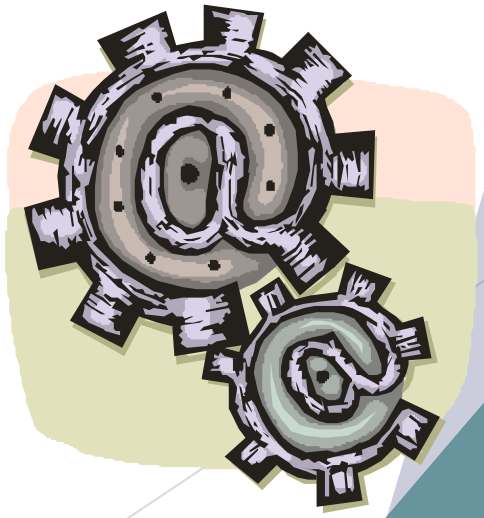


Модель в системе Rodin



Дальнейшее развитие

- Завершение верификации формальной модели в системе Rodin;
- уточнение модели;
- покрытие кода формальной моделью.



Спасибо за внимание!

qpos@cryptosoft.ru