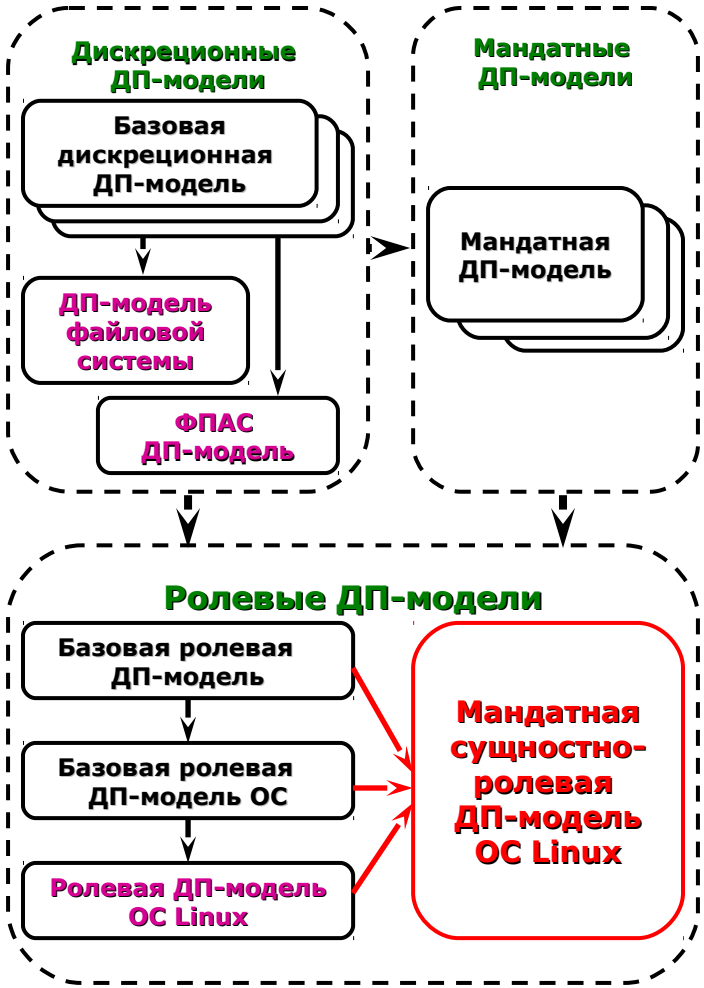
О представлении МРОСЛ ДП-модели в формализованной нотации Event-B (Rodin Platform)

д.т.н., доцент Девянин П.Н. (УМО ИБ),
к.ф.-м.н., доцент В.В. Кулямин,
д.ф.-м.н., профессор А.К. Петренко,
к.ф.-м.н. А.В. Хорошилов,
И.В. Щепетков (ИСП РАН)

Постановка задачи исследования

Семейство ДП-моделей



Отечественная защищенная ОССН Astra Linux Special Edition

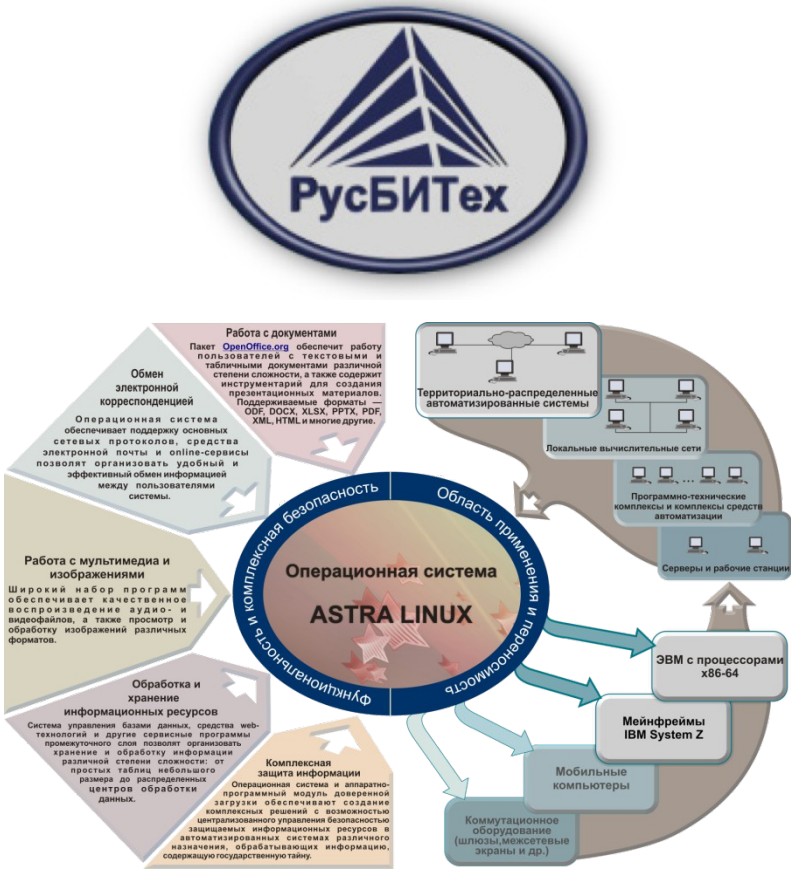
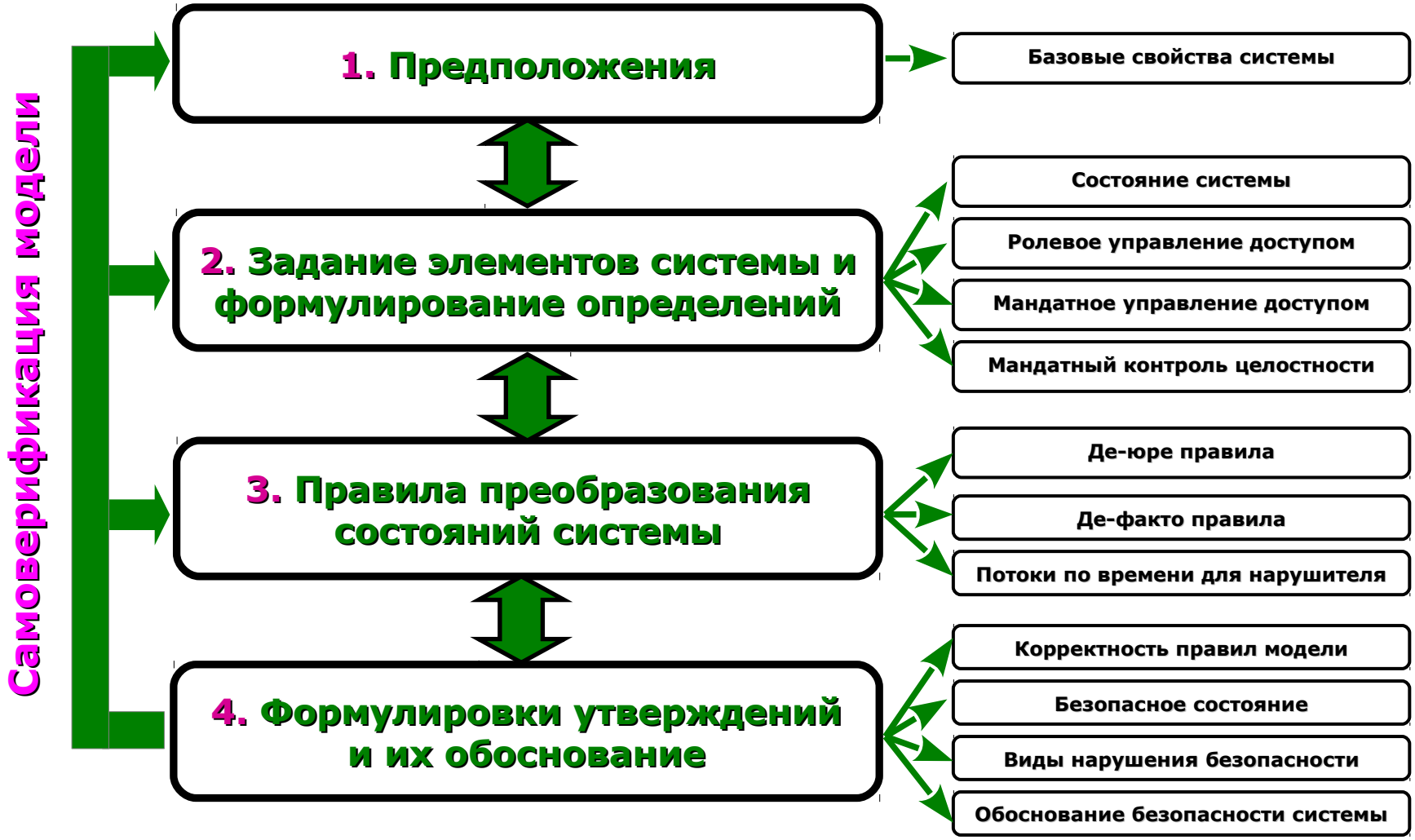


Схема решения задач исследования

Схема формирования МРОСЛ ДП-модели



Де-юре и де-факто правила преобразования состояний

de_facto_own: $S \to 2^S$ – функция фактического владения субъект-сессий субъект-сессиями.

de_facto_accesses: $S \to 2^{(E \cup R \cup AR) \times Ra}$ – функция де-факто доступов субъект-сессий, при этом по определению в каждом состоянии системы G для каждой субъект-сессии $s \in S$ верно равенство:

de_facto_accesses$(s) = \{(e, \alpha_a)$: существует $s' \in$ ***de_facto_own***(s) и [либо $e \in E$ и $(s', e, \alpha_a) \in A$], [либо $e \in R \cup AR$ и $(s', e, \alpha_a) \in AA$]$\}$.

Примеры де-юре правил

access_read (x, x', y)	$x, x' \in S, y \in E \cup R \cup AR$, существует $r \in R \cup AR$: $(x, r, read_a) \in AA$, [если $y \in E$, то $(y, read_i) \in PA(r)$ и либо $(execute\_container(x, y) = true$ и $f_e(y) \le f_e(x))$, либо $(x, downgrade\_admin\_role, read_a) \in AA$], [если $y \in R \cup AR$, то $(y, read_i) \in APA(r)$, $i_i(y) \le i_s(x)$, $Constraint_{AA}(AA') = true$, (для $e \in]y[$ либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$), (либо $f_i(y) \le f_s(x)$, либо $(x, downgrade\_admin\_role, read_a) \in AA$)], [если $y \in R \cup AR$ и $i_i(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]	$S' = S, E' = E, APA' = APA, PA' = PA, user' = user,$ $H_E' = H_E, F' = F,$ если $y \in E$, то $A' = A \cup \{(x, y, read_a)\}, AA' = AA,$ если $y \in R \cup AR$, то $AA' = AA \cup \{(x, y, read_a)\}, A' = A$
create_hard_link $(x, x', y, name, z)$	$x, x' \in S, y \in O, z \in C, execute\_container(x, y) = true, (x, z, write_a) \in A,$ $name \in NAME \setminus \{\text{“”}\}$, [либо $(f_e(y) = f_e(z) = f_e(x)$, если $CCR(y) = false$ или $CCRI(y) = false$, то $(x, entities\_admin\_role, read_a) \in AA)$, либо $(f_e(y) \le f_e(z)$ и $(x, downgrade\_admin\_role, read_a) \in AA)$], $i_e(y) \le \min(i_e(z), i_e(x))$ [если $i_e(z) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]	$S' = S, E' = E, APA' = APA, PA' = PA, A' = A, AA' = AA,$ $user' = user, F' = F,$ $entity\_name(z, y) = name,$ $H_E'(z) = H_E(z) \cup \{y\}$, для $e \in E \setminus \{z\}$ выполняется $H_E'(e) = H_E(e)$
create_first_session (x, x', u, y, z, zc, zi)	$x, x' \in S, u \in U, y \in E, z \notin S$, существует $r \in R \cup AR$ такая, что $(x, r, read_a) \in AA$ и $(y, execute_i) \in PA(r)$, $execute\_container(x, y) = true$, $f_e(y) \le \min(f_e(x), f_e(u))$, $zc \le f_s(u)$, $zi \le \min(i_i(u), i_e(y))$, [либо $f_e(x) \le zc$, либо $(x, downgrade\_admin\_role, read_a) \in AA$], [для $e \in]u[$ либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$], [если $zi = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]	$S' = S \cup \{z\}, E' = E, APA' = APA, A' = A, F' = F,$ $f_e'(z) = zc, i_s'(z) = zi, user'(z) = u,$ для $s \in S$ $user'(s) = user(s), f_s'(s) = f_s(s), i_s'(s) = i_s(s),$ $AA' = AA \cup \{(z, u\_admin\_i\_zi, read_a), (z, u\_c\_zc\_zi, write_a)\}$ $\cup \{(z, u\_c\_l\_li, read_a)$, где $l \le zc$ и $li \le zi\}$, $PA'(u\_c\_zc\_zi) = PA(u\_c\_zc\_zi) \cup \{(r, own_i)\}$, и для $r' \in R \setminus \{u\_c\_zc\_zi\}$ выполняется $PA'(r') = PA(r')$, $[z] = fa(u, y)$, $]z[= fp(u, y)$, $H_S'(z) = \varnothing$, для $s \in S$ выполняется $H_S'(s) = H_S(s)$

Примеры де-факто правил

de_facto_op $(s, op(x, x', \ldots))$	$s \in NS, x, x' \in de\_facto\_own(s)$, выполняются условия применения де-юре правила преобразования состояний $op(x, x', \ldots)$	Соответствуют результатам применения правила $op(x, x', \ldots)$
control(x, y, z)	$x \in NS, y \in S, z \in E, x \ne y, z \in [y]$ и $(x, z, write_m) \in F$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_E' = H_E,$ $de\_facto\_own'(x) = de\_facto\_own(x) \cup \{y\},$ $F' = F \cup \{(x, y, write_i), (y, x, write_i)\}$
take_access_own (x, y, z)	$x \in NS, y, z \in S, y \in de\_facto\_own(x),$ $z \in de\_facto\_own(y)$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_E' = H_E,$ $de\_facto\_own'(x) = de\_facto\_own(x) \cup \{z\},$ $F' = F \cup \{(x, z, write_i), (z, x, write_i)\}$
flow_memory_access (x, y, α_a)	$x \in S, y \in E \setminus E\_HOLE,$ $(y, \alpha_a) \in de\_facto\_accesses(x)$, где $\alpha_a \in \{read_a, write_a\}$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_E' = H_E,$ если $\alpha_a = read_a$, то $F' = F \cup \{(y, x, write_m)\}$, если $\alpha_a = write_a$, то $F' = F \cup \{(x, y, write_m)\}$
flow_time_access(x, y)	$x \in NS \cup NFS$, или $[y \in E \cup R \cup AR$, $(y, \alpha_a) \in de\_facto\_accesses(x)$, где $\alpha_a \in R_a]$, или $[y \in S$ и $y \in de\_facto\_own(x)]$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_E' = H_E,$ если $y \in E \cup R \cup AR$ и $\alpha_a = write_a$, то $F' = F \cup \{(y, x, write_i)\} \cup \{(x, e, write_i)$: $e \in E \cup R \cup AR, x \ne e$ и или $y \le e$, или $e \le y\}$; если $y \in E \cup R \cup AR$ и $\alpha_a = read_a$, то $F' = F \cup \{(y, x, write_i)\}$; если $y \in S$, то $F' = F \cup \{(y, x, write_i)\} \cup \{(x, e, write_i)$: $e \in S, x \ne e$ и или $y \le e$, или $e \le y\}$
post(x, y, z)	$x, z \in S, y \in E \cup S \cup R \cup AR, x \ne z,$ $(x, y, \alpha_f) \in F$, где $\alpha_f \in \{write_m, write_i\}$, $[(y \in E \cup R \cup AR$ и $(y, \beta_a) \in de\_facto\_accesses(z)$, где $\beta_a \in R_a)$, или $(y \in S$ и $y \in de\_facto\_own(z))]$	$S' = S, E' = E, PA' = PA, A' = A, AA' = AA, user' = user, H_E' = H_E,$ если $\alpha_f = write_m, \beta_a = read_a$ и $y \in E \setminus E\_HOLE$, то $F' = F \cup \{(x, z, write_m)\}$, иначе если $x, z \in NS \cup NFS$, то $F' = F \cup \{(x, z, write_i)\}$, иначе $F' = F$

Безопасность в смысле Белла-ЛаПадулы и мандатного контроля целостности

Определение. Пусть G_0 – безопасное начальное состояние системы $\Sigma(G^*, OP, G_0)$, и существует траектория без кооперации доверенных и недоверенных субъект-сессий $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$, где $N \geq 1$. В состоянии G_N произошло нарушение безопасности системы, когда в нем выполняется одно из условий, при этом они не выполняются в G_i траектории, где $0 \leq i < N$.

Условие 1 (нарушение безопасности в смысле мандатного контроля целостности). Существуют недоверенная субъект-сессия $x \in N_{SN}$ и доверенная субъект-сессия $y \in de\_facto\_own_N(x) \cap L_{SN}$ такие, что $i_{sN}(y) = i\_high$.

Условие 2 (нарушение безопасности в смысле Белла-ЛаПадулы). Существует информационный поток по памяти $(x, y, write_m) \in F_N : x, y \in E_N$ и не верно неравенство $f_{eN}(x) \leq f_{eN}(y)$.

Условие 3 (нарушение безопасности в смысле контроля информационных потоков по времени). Существует информационный поток по времени $(x, y, write_t) \in F_N$ такой, что $x, y \in E_N$ и не верно $f_{eN}(x) \leq f_{eN}(y)$.

Теорема (базовая теорема безопасности – БТБ-ДП). Пусть G_0 – безопасное начальное состояние системы $\Sigma(G^*, OP, G_0)$. Пусть на всех траекториях системы без кооперации доверенных или недоверенных субъект-сессий $G_0 \vdash_{op1} G_1 \vdash_{op2} \dots \vdash_{opN} G_N$, где $N \geq 1$, и в каждом G_N для каждой $s \in S_N$ и сущности $e \in E_N$ выполняются следующие условия.

Условие 1 (корректность уровней конфиденциальности и целостности сущностей, функционально ассоциированных с субъект-сессиями). Если $e \in [s]$, то выполняются условия $i_{sN}(s) \leq i_{eN}(e)$ и ($f_{sN}(s) = f_{eN}(e)$ или $i_{eN}(e) = i\_high$).

Условие 2 (корректность уровней конфиденциальности и целостности, а также прав доступа на чтение к сущностям, параметрически ассоциированным с субъект-сессиями). Если $e \in]s[$, то $f_{sN}(s) = f_{eN}(e)$ и для каждой роли или административной роли $r \in R_N \cup AR_N$ такой, что $(e, read_r) \in PA_N(r)$, выполняются условия $i_{sN}(s) \leq i_{eN}(e) \leq i_{rN}(r)$.

Условие 3 (функциональная и параметрическая корректность всех доверенных субъект-сессий относительно всех доверенных субъект-сессий и сущностей). Для всех доверенных $s \in L_{SN}$ верно $f\_correct_N(s) = p\_correct_N(s) = L_{SN} \times (E_N \cup S_N)$.

Условие 4 (абсолютная функциональная и параметрическая корректность субъект-сессии относительно всех сущностей и субъект-сессий с совпадающим уровнем конфиденциальности). Для всех $s \in S_N$ верно $\{s' \in S_N: f_{sN}(s') = f_{sN}(s)\} \times (E_N \cup S_N) \subset af\_correct_N(s) = ap\_correct_N(s)$.

Тогда система $\Sigma(G^*, OP, G_0)$ безопасна в смысле условий 1, 2 определения безопасности системы.

Обоснование адекватности. Пример: правило *access_write*(*x*, *x'*, *y*)

$x, x' \in S$,

$y \in E \cup R \cup AR$,

существует $r \in R \cup AR$: $(x, r, read_a) \in AA$,

[если $y \in E$, то

$i_e(y) \le i_s(x)$

и (либо ($execute\_container(x, y) = true$

и, если $y \in E\_HOLE$, то $f_s(x) \le f_e(y)$,
иначе $f_e(y) = f_s(x)$),
либо (x, $downgrade\_admin\_role$, $read_a$) $\in AA$),

и $(y, write_r) \in PA(r)$],
[если $y \in R \cup AR$, то $(y, write_r) \in APA(r)$,
$i_r(y) \le i_s(x)$, $Constraint_{AA}(AA') = true$,
(для $e \in]y[$ либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$), (либо $f_r(y) = f_s(x)$,
либо (x, $downgrade\_admin\_role$, $read_a$) $\in AA$)],
[если ($y \in E$ и $i_e(y) = i\_high$) или
($y \in R \cup AR$ и $i_r(y) = i\_high$),
то (x', $f_s(x)\_i\_entity$, $write_a$) $\in A$]

```
int pdp_permission(const PDP_0* s, const PDP_0* o, int mode)
{
    if( o->type & PDP_TYPE_EHOLE ) return 0;

    if (mode & R_OK) {
            if(  (s->lev < o->lev)  ||  ( (s->cat & o->cat) != o->cat )  ) return -1;
    }

    if (mode & W_OK) {
        if(  (s->lev > o->lev)  ||  (s->ilev < o->ilev)  ||  ( (s->cat & o->cat) != s->cat )  ) return -1;
    }

    if (mode & X_OK) {
            if(  (s->lev < o->lev)  ||  ( (s->cat & o->cat) != o->cat )  ) return -1;
    }

    return 0;
}
```

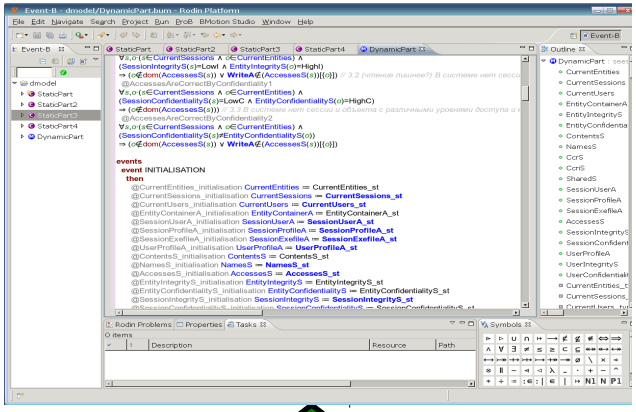
```
    mask &= (MAY_READ|MAY_WRITE|MAY_EXEC|MAY_APPEND);
    task_role = list_entry(next_task_role, struct role, list);
    inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
    while(next_task_role != task_roles_list)
    {
        while(next_inode_role != inode_roles_list)
        {
            if(inode_role->role_seed > task_role->role_seed)
            {
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(task_role->role_seed == inode_role->role_seed)
            {
                ret = rback_may_access(inode_role->role_access, mask);
                if(0 == ret)
                    return ret;
                next_inode_role = next_inode_role->next;
                inode_role = (struct inode_rback*) list_entry(next_inode_role, struct role, list);
                continue;
            }
            if(inode_role->role_seed < task_role->role_seed)
                break;
            return ret;
        }
        next_task_role = next_task_role->next;
        task_role = list_entry(next_task_role, struct role, list);
    }
    return ret;
```

Обоснование адекватности реализации

Правила МРОСЛ ДП-модели

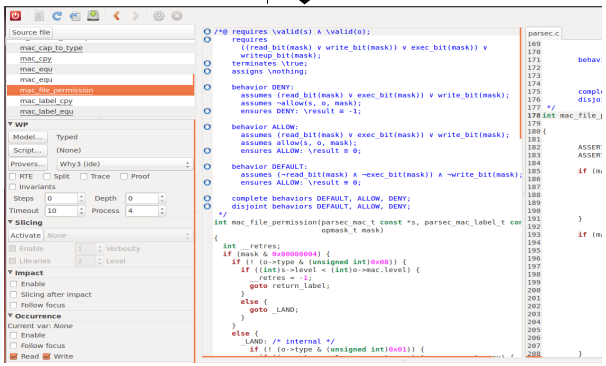
access_read(x, x', y)	
$x, x' \in S, y \in E \cup R \cup AR$, существует $r \in R \cup AR$: $(x, r, read_a) \in AA$, [если $y \in E$, то $(y, read_r) \in PA(r)$ и либо $(execute\_container(x, y) = true$ и $f_e(y) \le f_s(x))$, либо $(x, downgrade\_admin\_role, read_a) \in AA$], [если $y \in R \cup AR$, то $(y, read_r) \in APA(r)$, $i_r(y) \le i_s(x)$, $Constraint_{AA}(AA') = true$, (для $e \in]y[$ либо $(x, e, read_a) \in A$, либо $(x, e, write_a) \in A$), (либо $f_r(y) \le f_s(x)$, либо $(x, downgrade\_admin\_role, read_a) \in AA$)], [если $y \in R \cup AR$ и $i_r(y) = i\_high$, то $(x', f_s(x)\_i\_entity, write_a) \in A$]	$S' = S, E' = E, APA' = APA, PA' = PA, user' = user, H_E' = H_E, F' = F$, если $y \in E$, то $[A' = A \cup \{(x, y, read_a)\}, AA' = AA]$, если $y \in R \cup AR$, то $[AA' = AA \cup \{(x, y, read_a)\}, A' = A]$

Автоматизированная верификация МРОСЛ ДП-модели



Rodin (Event-B), Alloy

```
1  static int access_read(struct task_struct *subject,struct inode *entity)
2  {
3      struct task_security *subject_security;
4      struct inode_security *entity_security;
5      int ret = -EACCES;
6
7      subject_security = get_task_security(subject);
8      entity_security = get_entity_security(entity);
9
10     if(!subject_security || !entity_security)
11         return ret;
12
13     if(is_role(entity)) //проверяем возможность получения доступа на чтение к сущности
14     {
15         ret = can_access(&subject_security->roles, &entity_security->list, MAY_READ, 0);
16         if(ret != 0)
17             ret = can_access(&subject_security->admin_roles, &entity_security->list, MAY_WRITE, 0);
18         if(ret == 0)
19         {
20             ret = execute_container(subject, entity);
21             if(ret != 0)
22                 ret = is_downgrade_admin_role(&subject_security->admin_roles, MAY_READ);
23         }
24     }
25     else //проверяем возможность получения доступа на чтение к роли или административной роли
26     {
27         ret = can_admin_access(&subject_security->admin_roles, &entity_security->list, MAY_READ);
28     }
29     return ret;
```

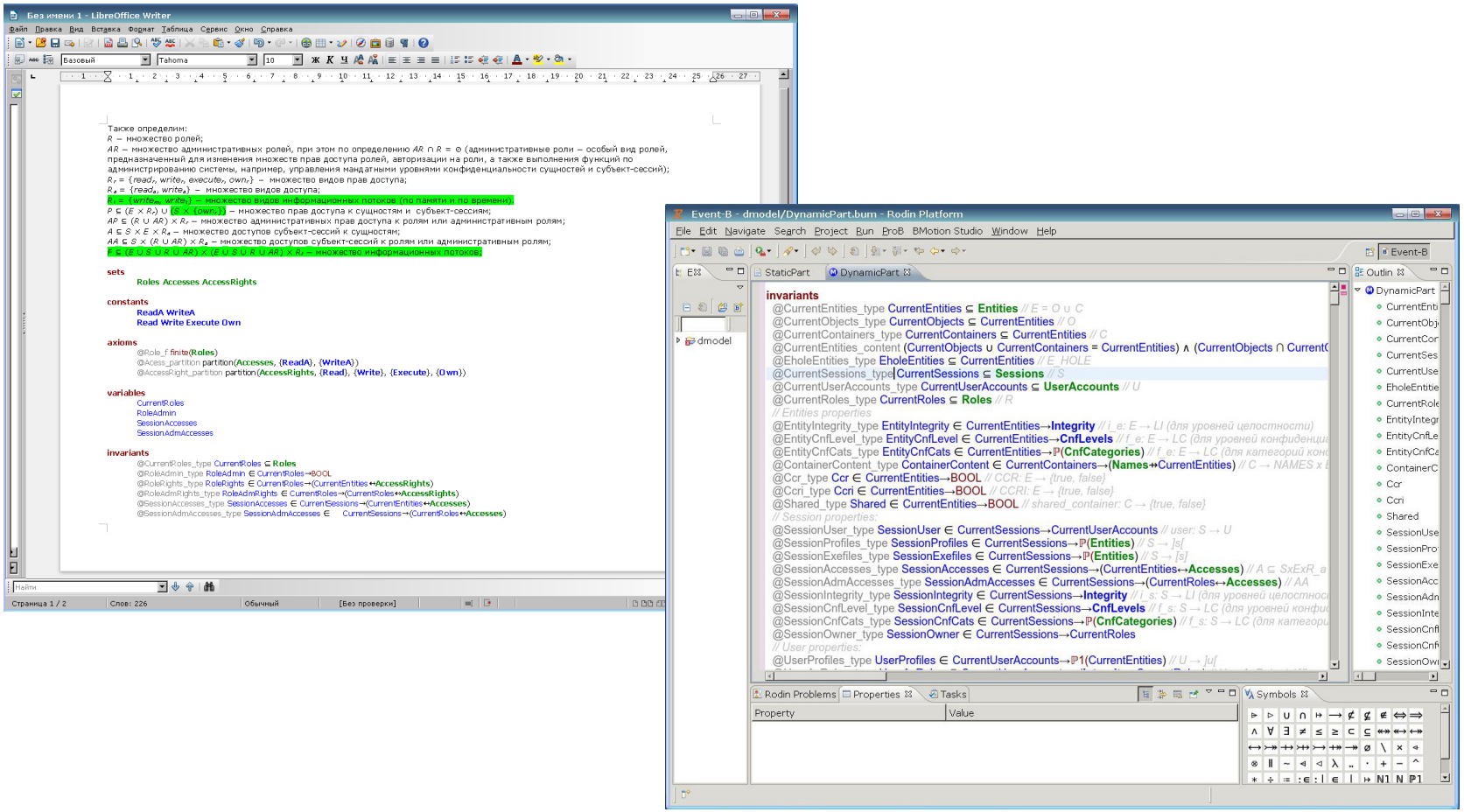


Frama-C, Why3

Программный код механизма управления доступом ОССН

Верификации кода на соответствие спецификациям, заданным на основе правил преобразования состояний

Переход от математической нотации к формализованной (*Event-B*, *Rodin Platform*)



Пример задания имен и иерархии в сущностей

$entity\_name: C \times E \to 2^{NAMES}$ – функция имен сущностей в составе сущностей-контейнеров.

В исходной математической нотации:

$H_E: E \to 2^E$ – функцию иерархии сущностей (сопоставляющую каждой сущности $e \in E$ множество сущностей $H_E(e) \subset E$, непосредственно в ней содержащихся), удовлетворяющую условиям:

- если сущность $e \in H_E(c)$, то $e < c$, при этом, если $e \in C$, то не существует сущности-контейнера $d \in C$ такой, что $e < d < c$;
- для любых сущностей $e_1, e_2 \in E$, $e_1 \neq e_2$ выполняется равенство $H_E(e_1) \cap H_E(e_2) \cap C = \varnothing$;
- если $o \in O$, то справедливо равенство $H_E(o) = \varnothing$.

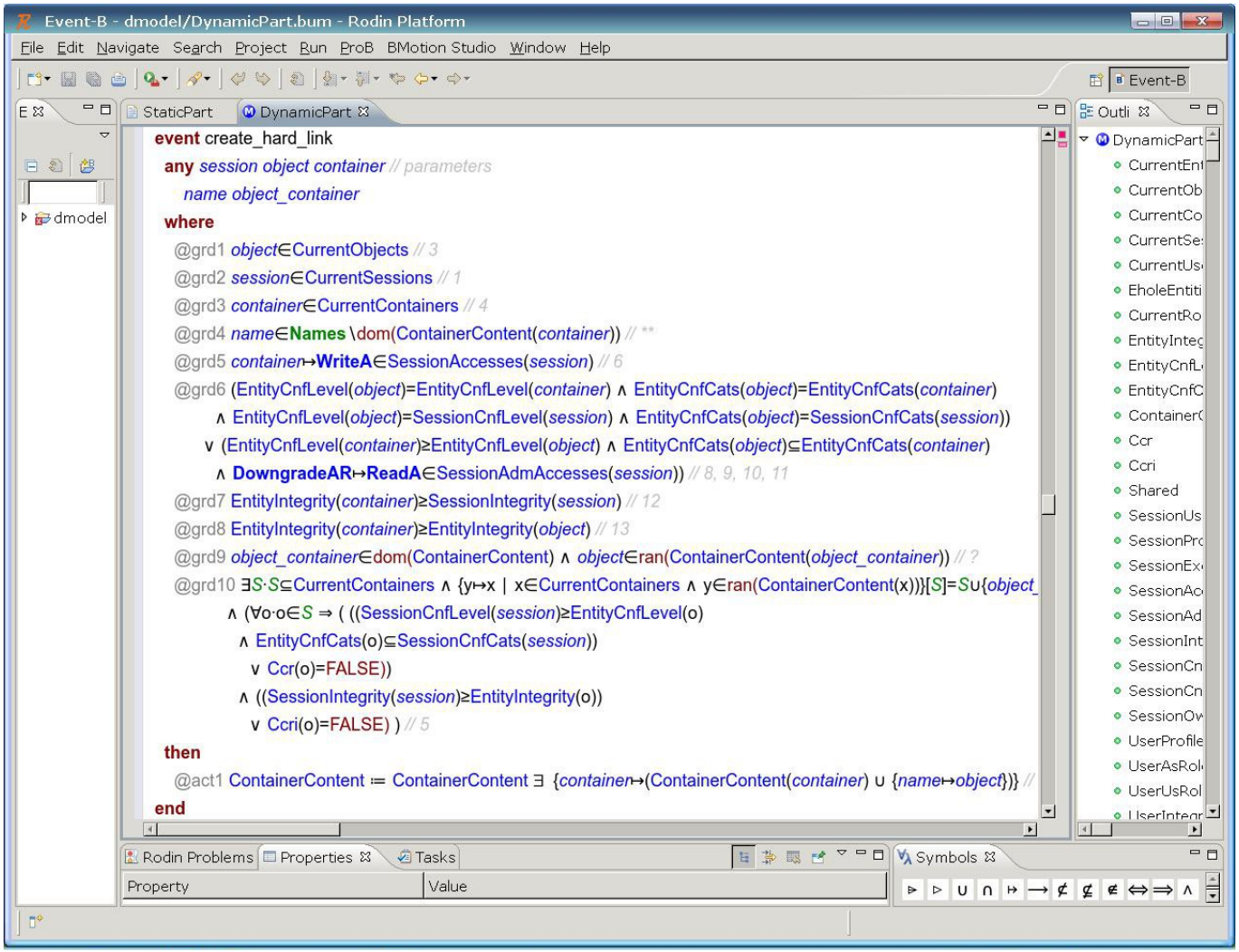
В адаптированной математической нотации:

для любых контейнеров $c, c' \in C$ по определению выполняются условия:

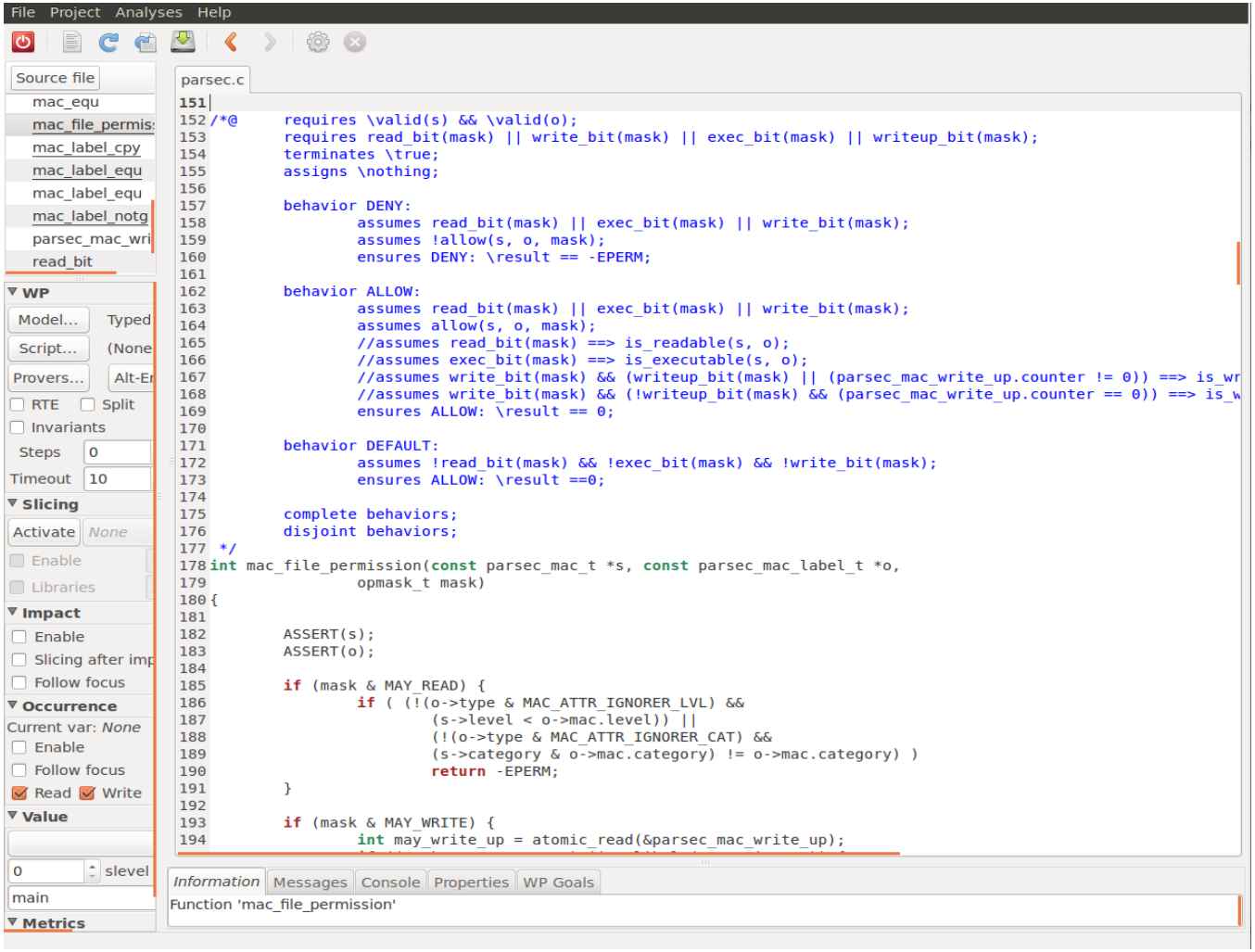
- $|entity\_name(c, c')| \leq 1$;
- если $c \neq c'$ и существует $c'' \in C$ такой, что $entity\_name(c, c'') \neq \varnothing$, то $entity\_name(c', c'') = \varnothing$;
- существует единственная сущность-«корневой контейнер» $ROOT \in C$ такая, что $entity\_name(c, ROOT) = \varnothing$ и, если $c \neq ROOT$, то существует единственная последовательность контейнеров $c_1 = c, c_2, \ldots, c_n = ROOT \in C$ такая, что $n \geq 2$ и $entity\_name(c_i, c_{i-1}) \neq \varnothing$, где $1 < i \leq n$.

$H_E: E \to 2^E$, где для $e \in E$ выполняется $H_E(e) = \{e' \in E \mid entity\_name(e, e') \neq \varnothing\}$.

Rodin Platform. Пример задания правила



Frama-C. Пример предусловия



Спасибо за внимание!