CUSTIS®

# Attribute-Based Authorization: how we stopped giving roles and got into politics

Антон Лапицкий

Архитектор приложений

Joker

20 октября 2018 года

## Антон Лапицкий

For over 8 years I have been engaged i
java development in banking
and educational areas as well
for the public sector

## { Let's talk about it }

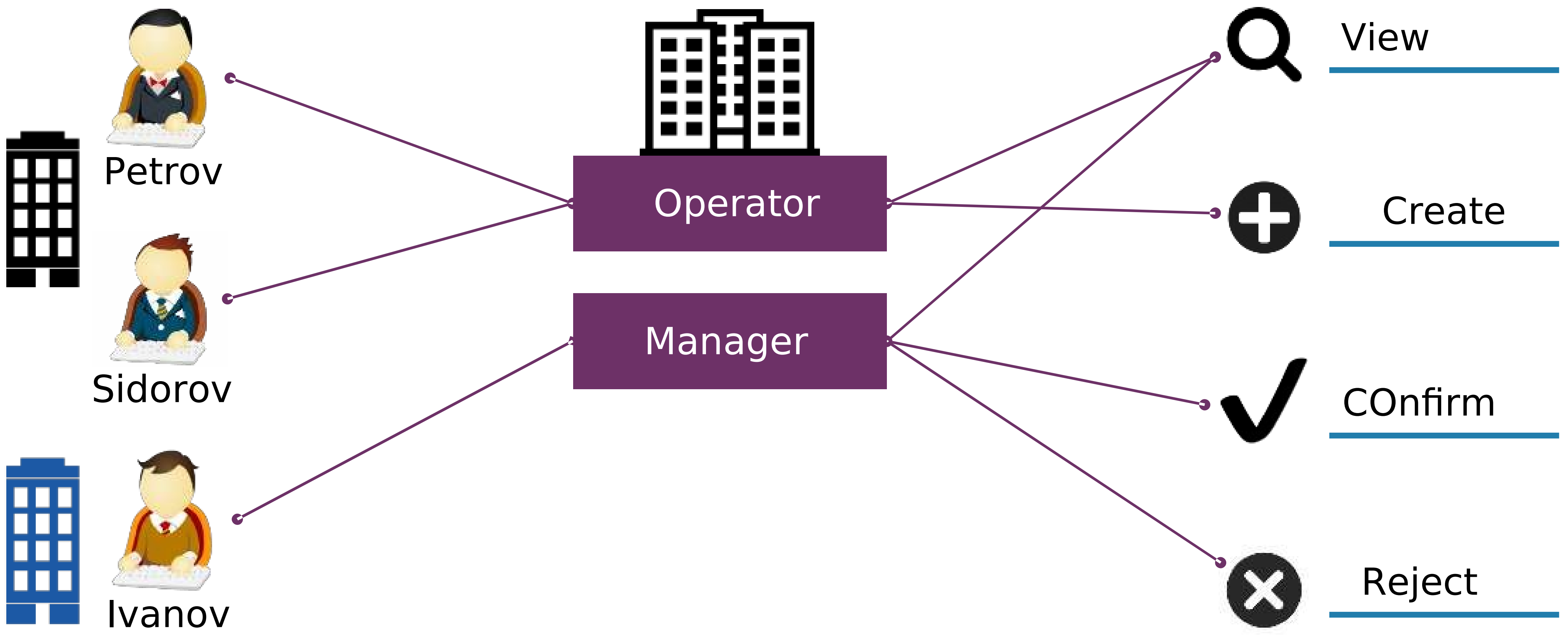
Role vs attribute model on a real example
XACML standard
EasyABAC Framework

- Utilities
- Device
- API
- Performance

## {Task}

The shop sells household appliances wholesale
The store has several branches planned expansion
Operator accepts orders
Manager confirms or rejects orders

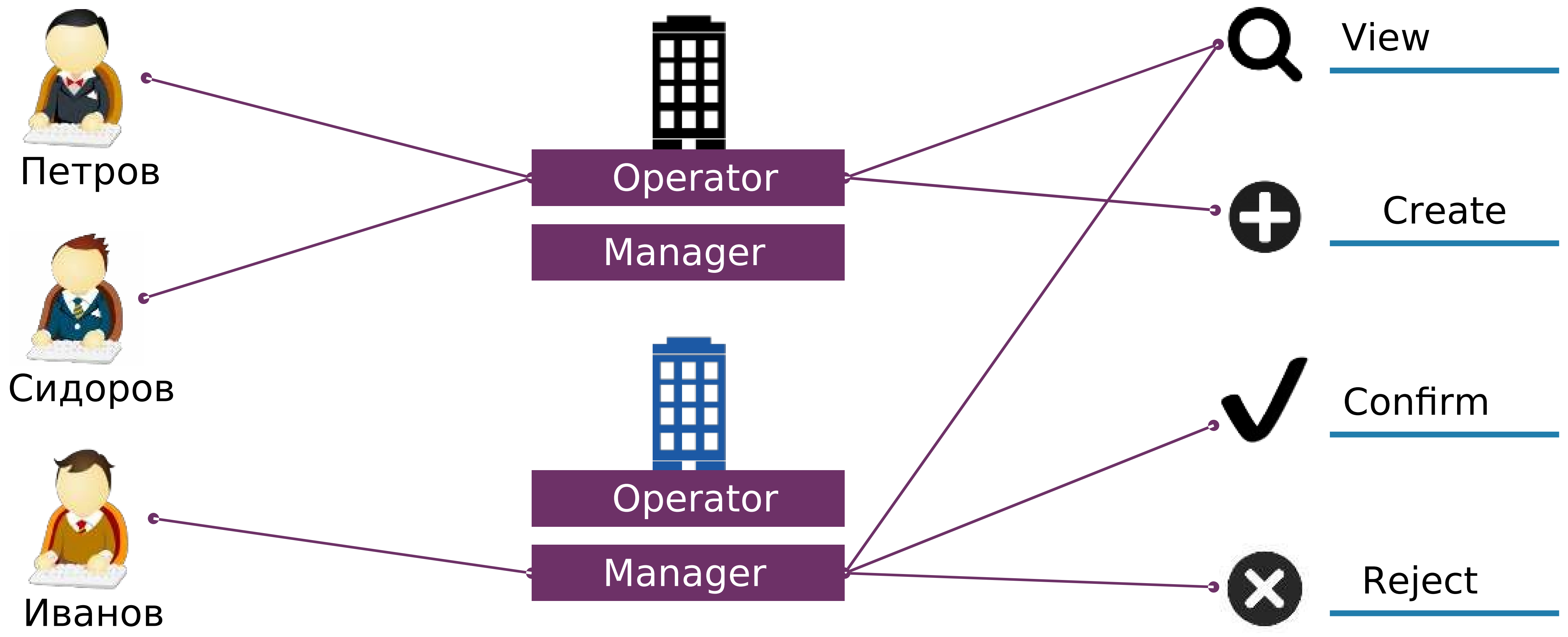
# { Access Model v. 1 (RBAC) }



## {Access Model v. 1 (RBAC)}

```
void checkView() {
        User user = AuthenticationContext.currentUser();

        if (user.hasRole(ROLE_MANAGER.name())
                || user.hasRole(ROLE_OPERATOR.name())) {
            return;
        }

        throw new NotPermittedException("not permitted");
}
```

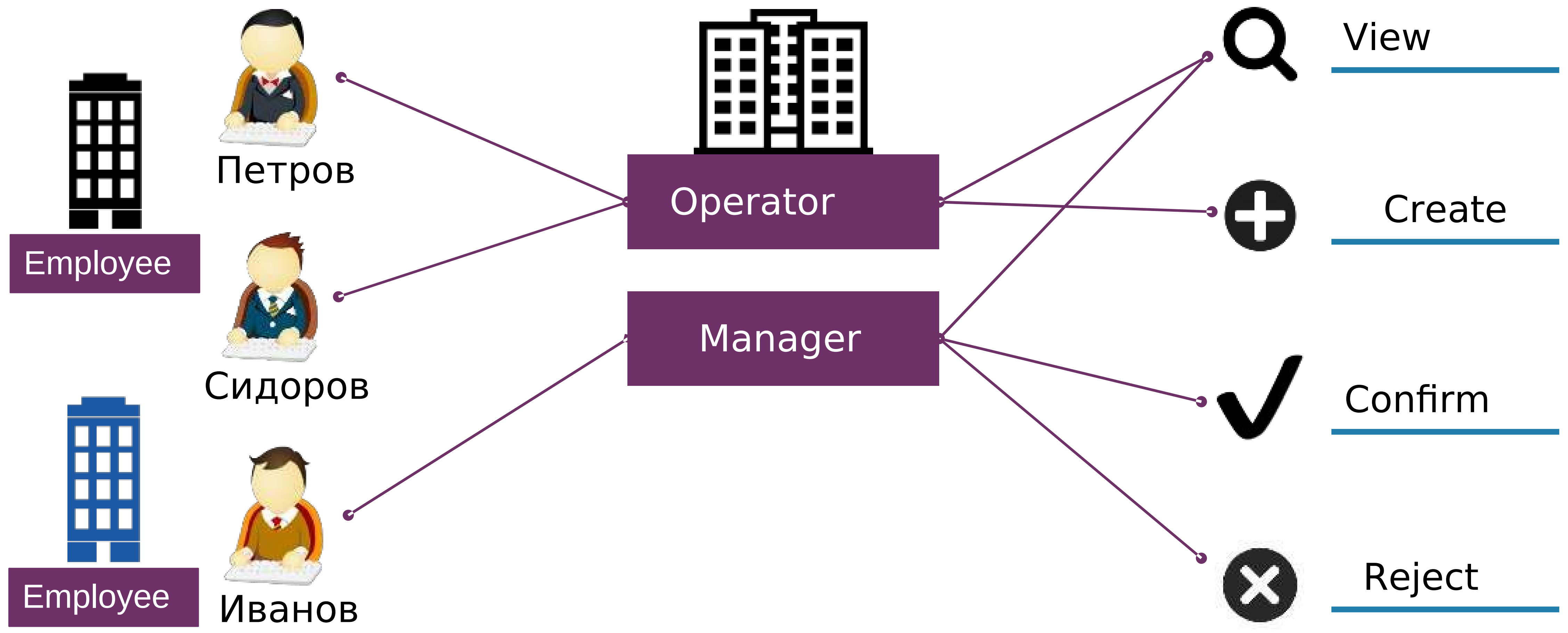
# {Access Model v.2 (RBAC)}



## {Access Model v.2 (RBAC)}

```
void checkView(Order order) {
        User user = AuthenticationContext.currentUser();

        if (user.hasRole(ROLE_MANAGER.ofBranch(order.getBranchId()))
                || user.hasRole(ROLE_OPERATOR.ofBranch(order.getBranchId()))) {
            return;
        }

        throw new NotPermittedException("not permitted");
}
```

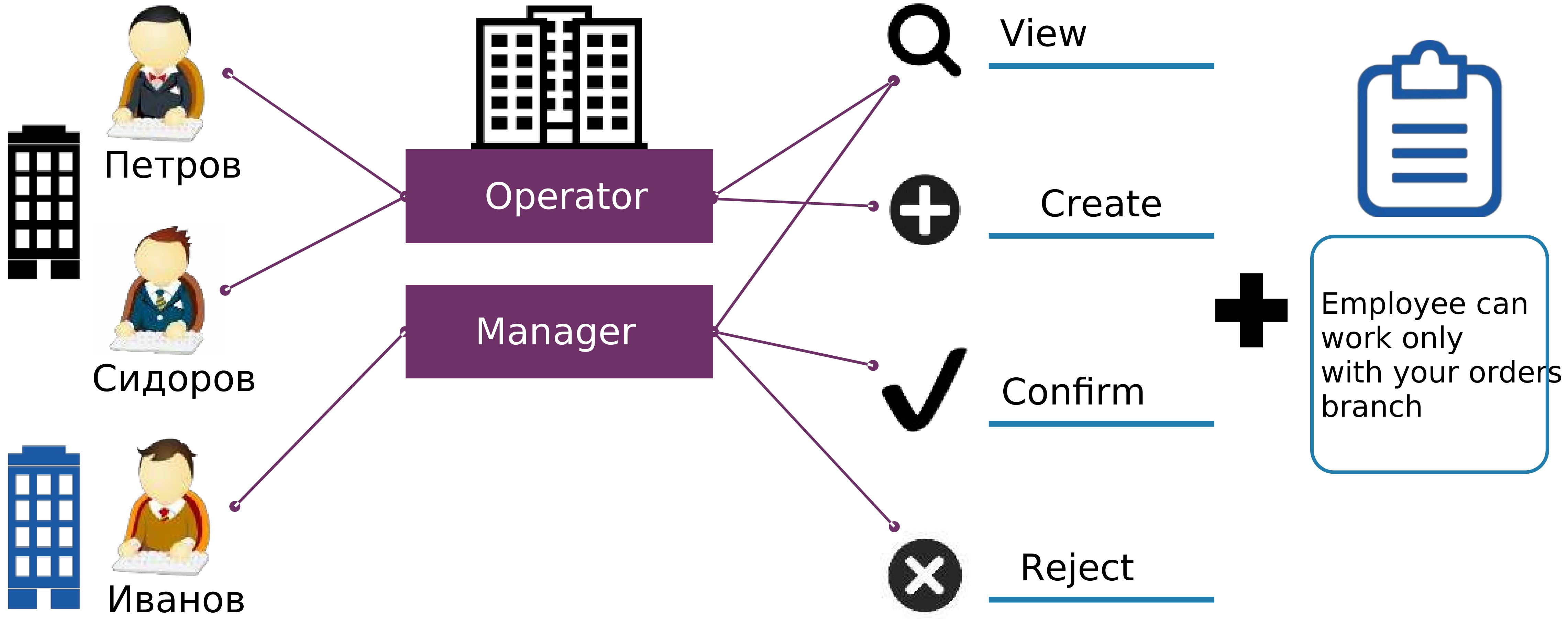
# {Access Model v.3 (RBAC)}



# {Access Model v.3 (RBAC)}

```
void checkView(Order order) {
        User user = AuthenticationContext.currentUser();
        checkBranchRole(user, order.getBranch());
        if (user.hasRole(ROLE_MANAGER.name())
                        || user.hasRole(ROLE_OPERATOR.name())) {
            return;
        }
        throw new NotPermittedException("not permitted");
}

void checkBranchRole(User user, Branch branch) {
        if (user.hasRole(ROLE_USER.ofBranch(branch.getId()))) return;
        throw new NotPermittedException("not permitted");
}
```

# {Access Model v.4 (RBAC)}



Employee can work only with your orders branch

## {Access Model v.4 (RBAC)}

```
void checkView(Order order) {
        User user = AuthenticationContext.currentUser();

        checkUserBranch(user, order.getBranch());
        if (user.hasRole(ROLE_MANAGER.name())
                                || user.hasRole(ROLE_OPERATOR.name())) {
            return;
        }
        throw new NotPermittedException("not permitted");
}

void checkUserBranch(User user, Branch branch) {
        if (user.getBranch().getId().equals(branch.getId())) return;
        throw new NotPermittedException("not permitted");
}
```

# {Comparing Access Models}

| | ✚ | ━ |
|---|---|---|
| 2 Roles | Simplicity | There is no separation between affiliates |
| 2 Roles for everyone branch | Simplicity | Scalablity<br>Artificial roles |
| 2 Business Rolls + Role Branch Employee | Correct business games | Scalablity<br>Artificial roles |
| 2 Business Rolls + Business Code | Scalablity | Hardcode business logics |

# {How should work}

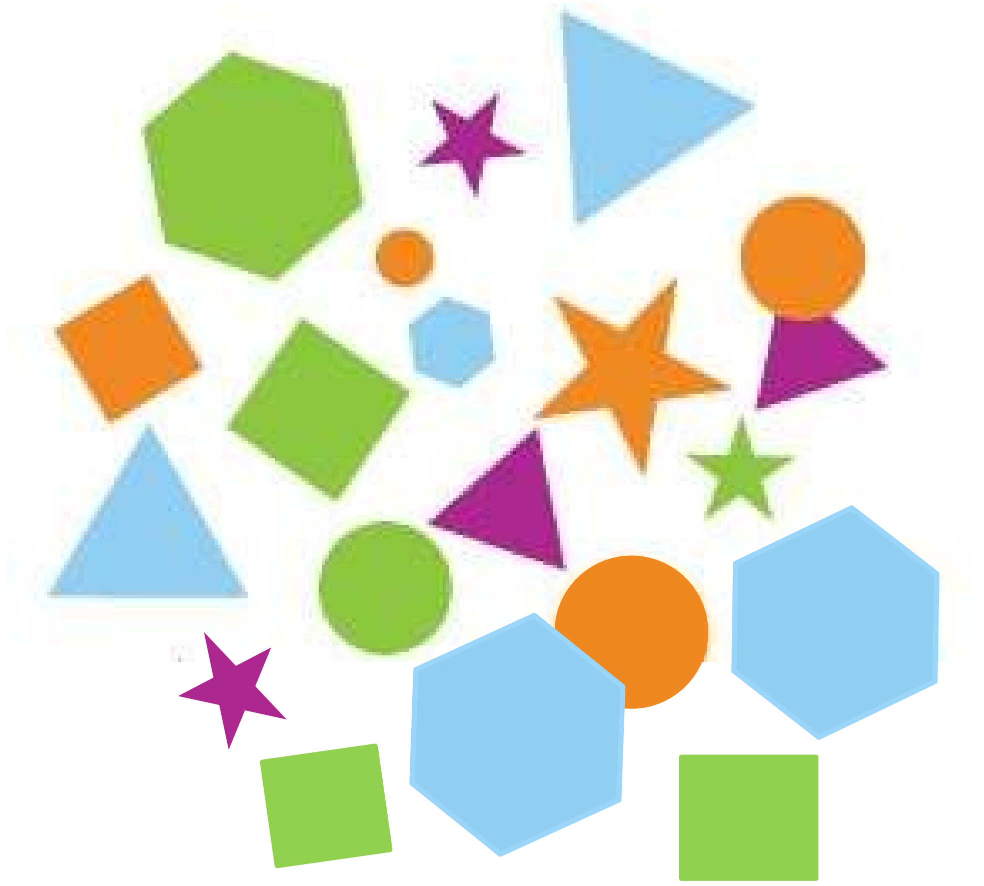
## Context

🟩 Subject

🟦 Act

🟧 An Object

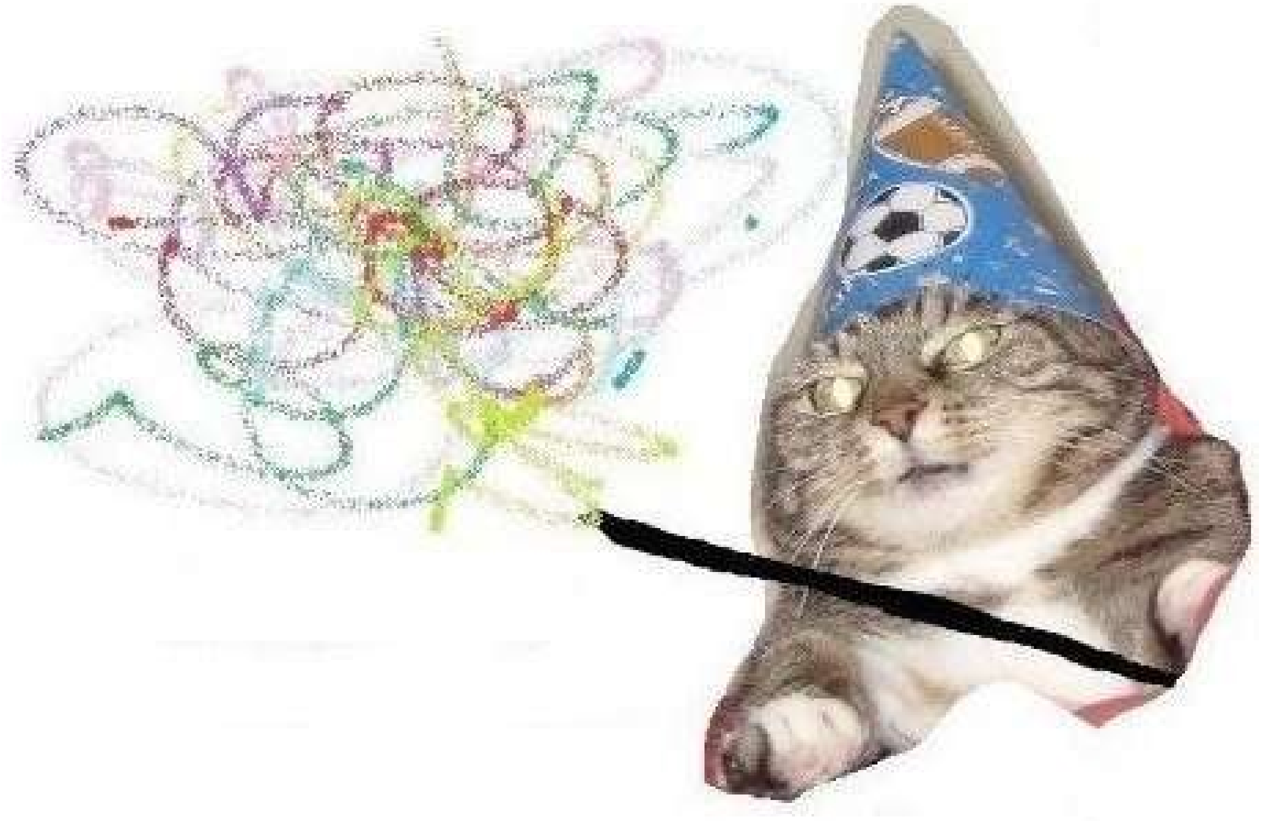
🟪 Enviroment



## {How should work}

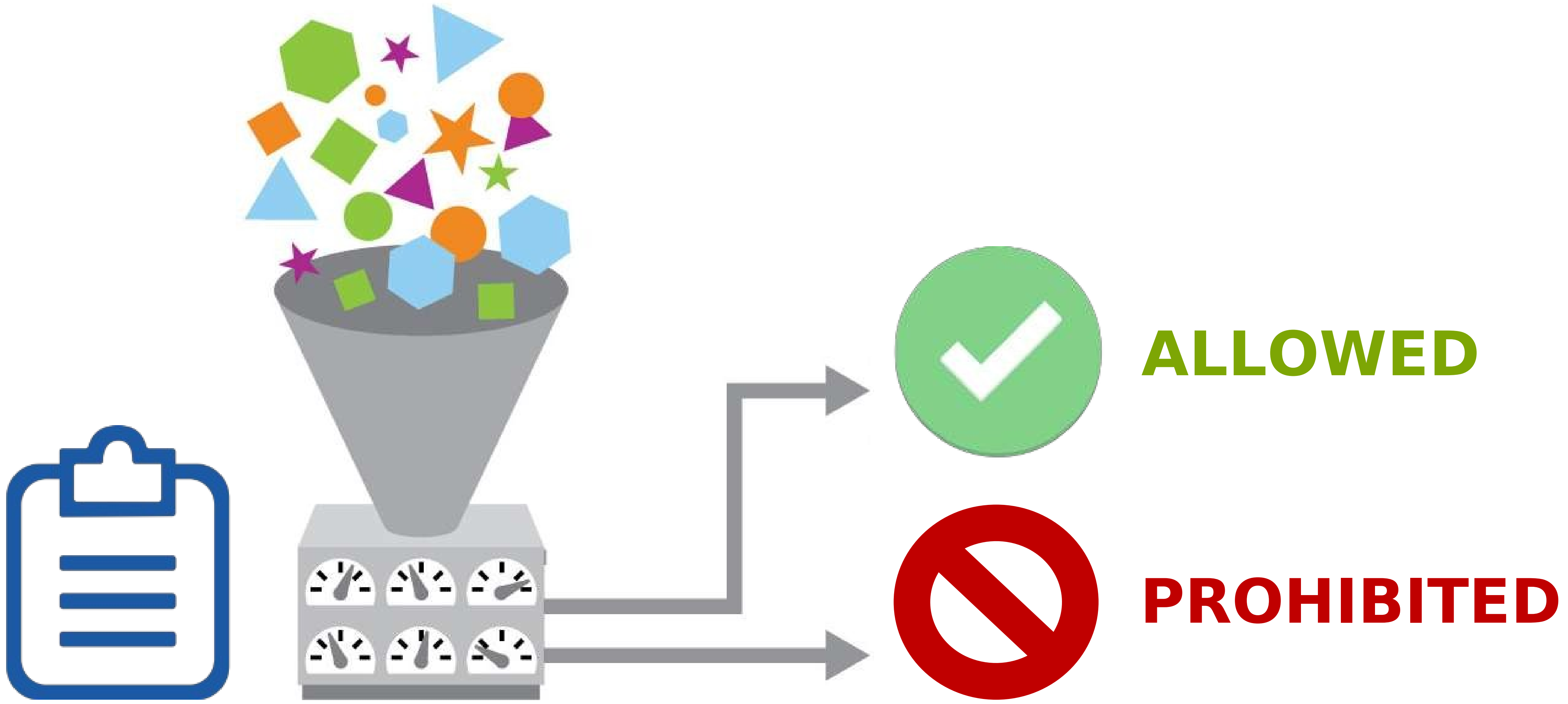
CUSTIS®

REGULATIONS



## {How should work}

CUSTIS®



CUSTIS®

## XACML

“eXtensible Access Control Markup Language”

## XACML

“eXtensible Access Control Markup Language”

**Allow order viewing from 09:00 to 17:00**

## XACML

“eXtensible Access Control Markup Language”

```
<Policy PolicyId="SamplePolicy"
      RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Rule RuleId="LoginRule" Effect="Permit">
    <Target>
      <Actions>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                            AttributeId="OrderAction"/>
        </ActionMatch>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
                             AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
                             AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

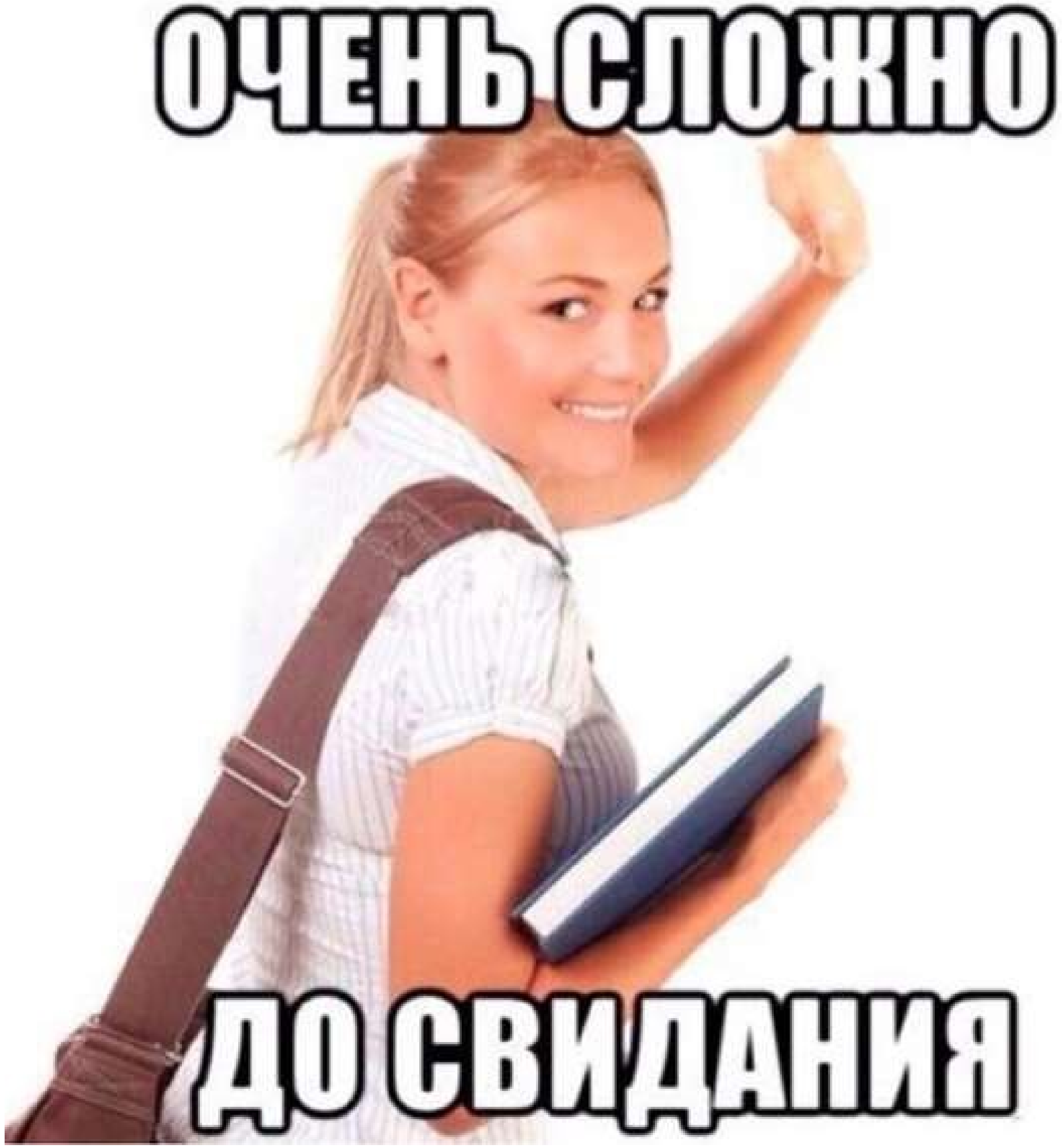
## XACML

“eXtensible Access Control Markup Language”

```
<Policy PolicyId="SamplePolicy"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Rule RuleId="LoginRule" Effect="Permit">
    <Target>
      <Actions>
        <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">view</AttributeValue>
          <ActionAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#string"
                              AttributeId="“OrderAction"/>
        </ActionMatch>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
                              AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
      </Apply>
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal">
        <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
          <EnvironmentAttributeSelector DataType="http://www.w3.org/2001/XMLSchema#time"
                              AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"/>
        </Apply>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#time">17:00:00</AttributeValue>
      </Apply>
    </Condition>
  </Rule>
  <Rule RuleId="FinalRule" Effect="Deny"/>
</Policy>
```

## XACML

“eXtensible Access Control Markup Language”



# {Implementation XACML}

CUSTIS®

WSO2 Identity Server

AXIOMATICS

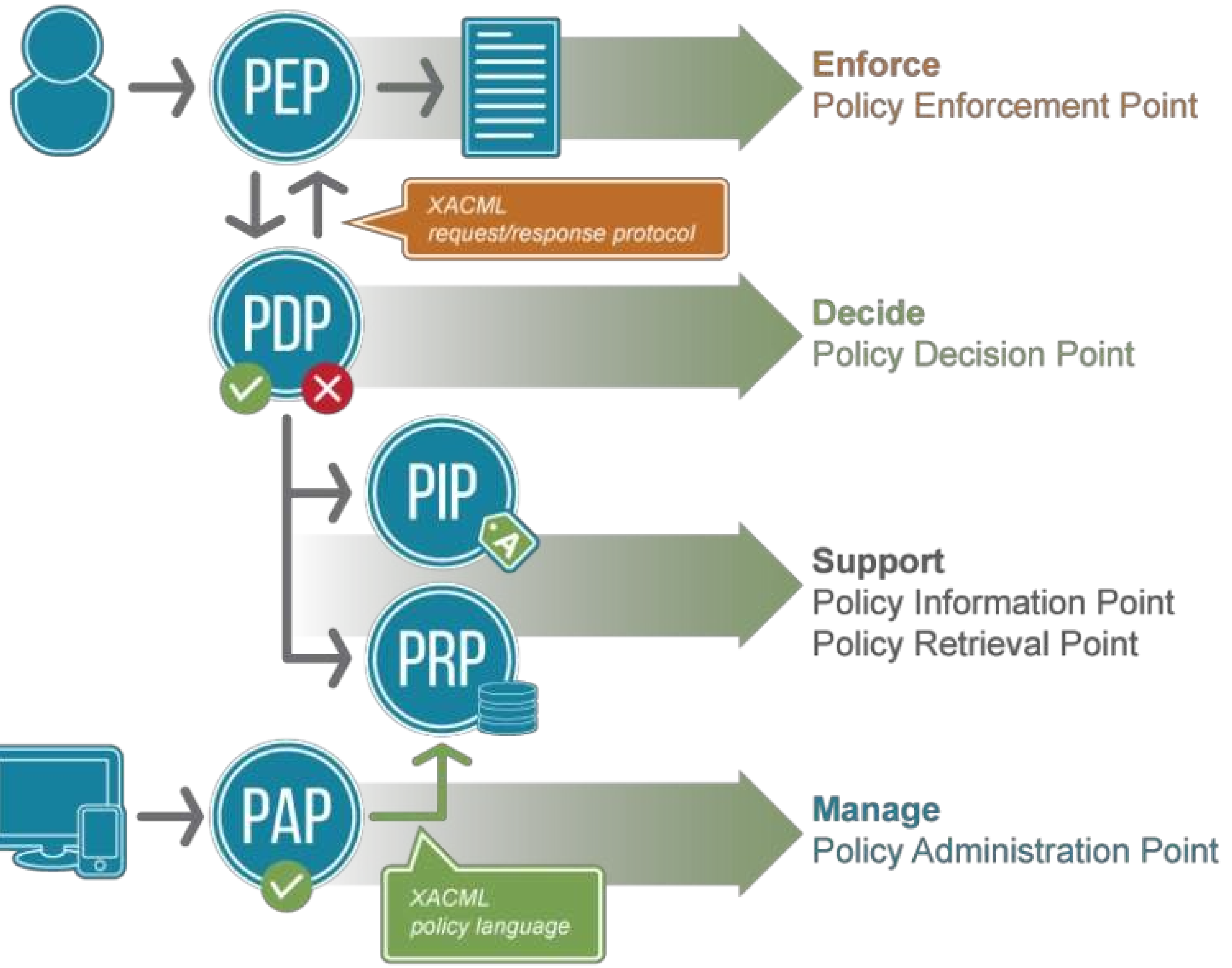
AUTHZ FORCE

WSO2

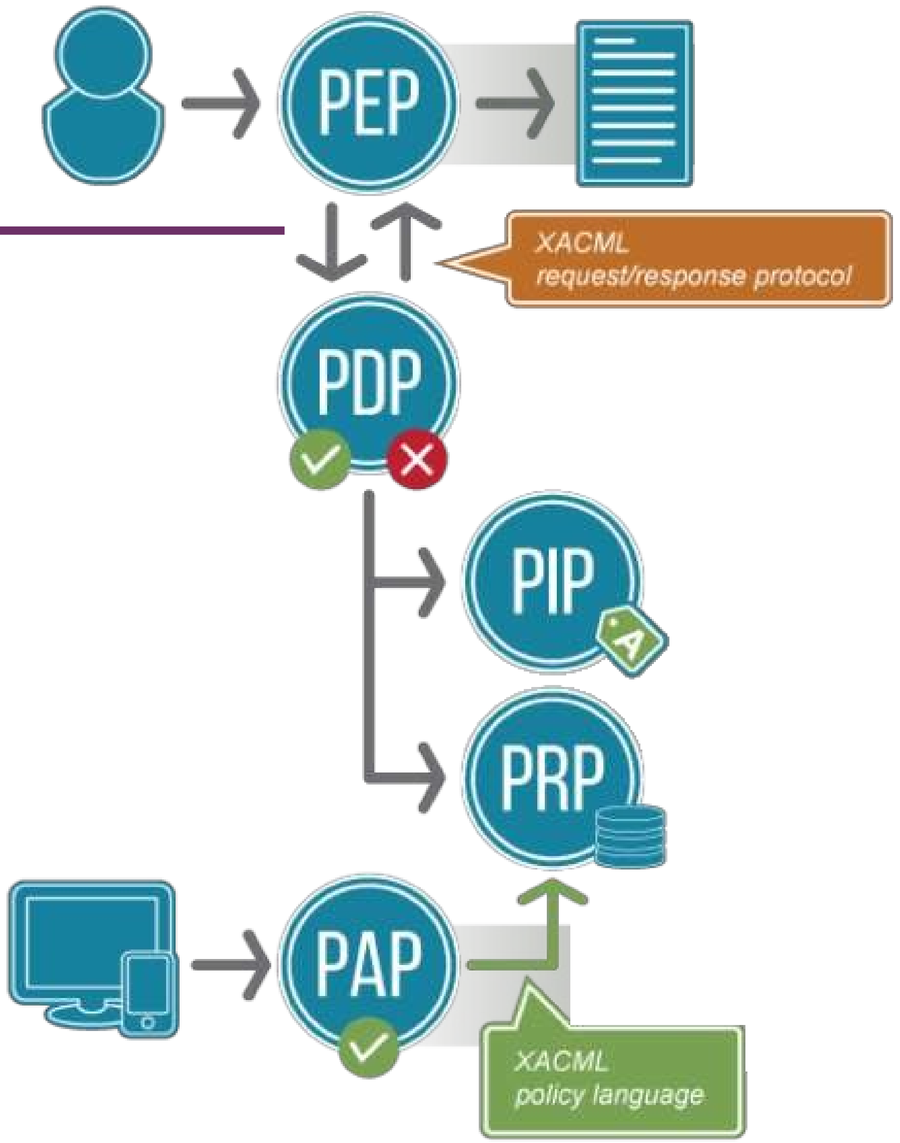
BALANA

POLICY

# {architecture XACML}



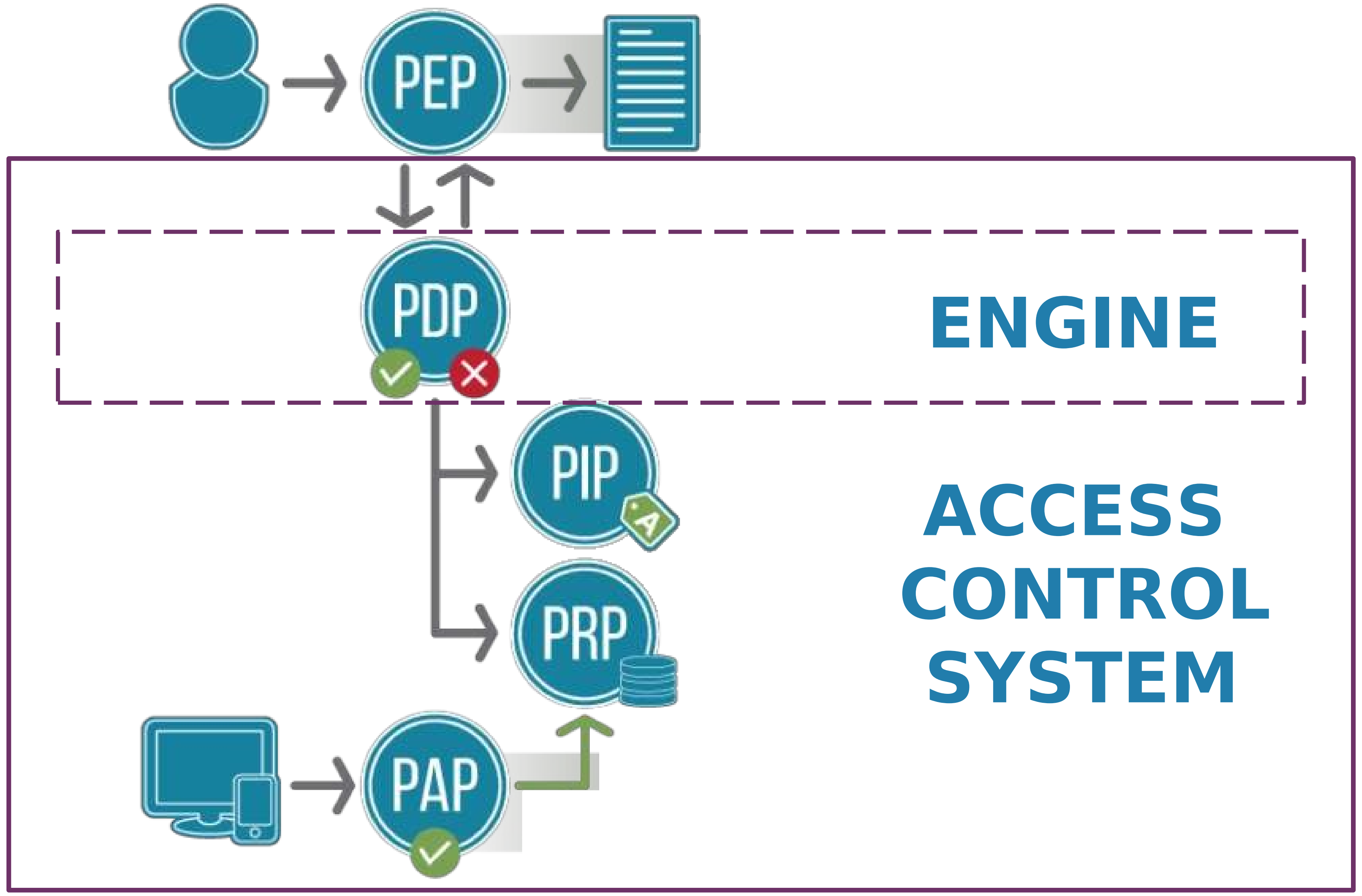
## {architecture XACML}



**APPLICATION**

**ACCESS CONTROL SYSTEM**

## {architecture XACML}



## { Problems of existing solutions }

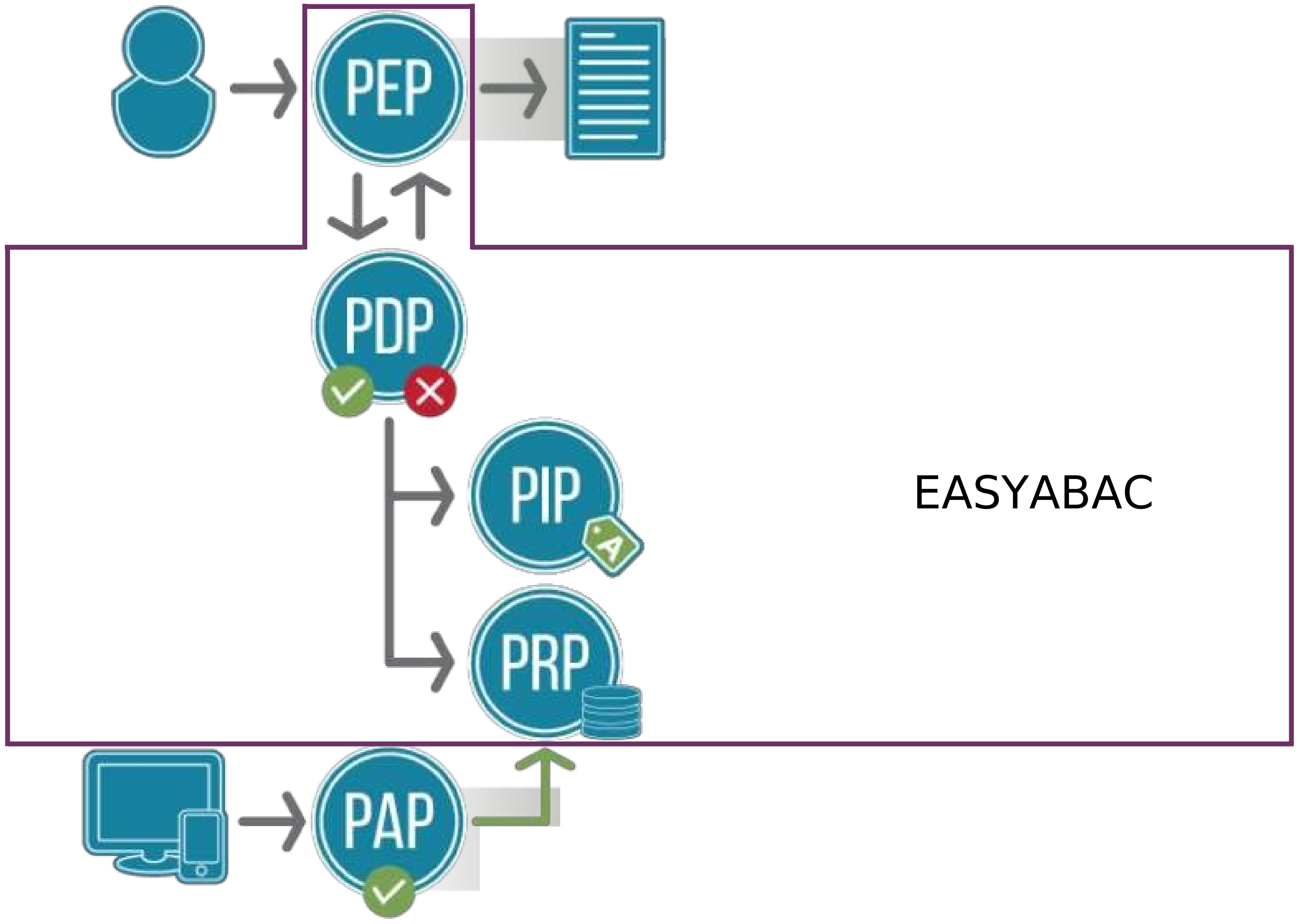
Bad Access Policy Testing Tools
It is not clear why the policy gave Permit or Deny
Processing speed depends heavily on the policy structure
No query optimization tools

## {Easyabac}

## {Easyabac:goals}

Simplify policy creation
Simplify policy testing
Make a convenient Java API
Optimize queries

## {Easyabac and XACML}



## {Easyabac:model}

```
subject:
  attributes:
  – id: id
  – id: role
    title: Role Employeea
    allowableValues:
      – OPERATOR
      – MANAGER
 – id: branchId
    title: Branch Id
 – id: maxOrderAmount
    title: Maximum Order
    type: int
```

```
customer:
  title: Client
  attributes:
  – id: id
  – id: branchId
     title: Branch Id
```

```
order:
  title: Заказ
  actions:
  – view
  – create
  – approve
  – reject
  attributes:
  – id: id
     title: Oder Id
  – id: amount
     title: Oder ammount
     type: int
  – id: branchId
     title: Branch Id
  – id: customerId
     title: Customer ID
```

# {Easyabac:model}

```
subject:
  attributes:
  – id: id
  – id: role
    title: Role Employeea
    allowableValues:
      – OPERATOR
      – MANAGER
 – id: branchId
   title: Branch Id
 – id: maxOrderAmount
   title: Maximum Order
   type: int
```

```
customer:
  title: Client
  attributes:
  – id: id
  – id: branchId
    title: Branch Id
```

```
order:
  title: Заказ
  actions:
  – view
  – create
  – approve
  – reject
  attributes:
  – id: id
    title: Oder Id
  – id: amount
    title: Oder ammount
    type: int
  – id: branchId
    title: Branch Id
  – id: customerId
    title: Customer ID
```

# {Easyabac:model}

```
subject:
  attributes:
  – id: id
  – id: role
    title: Role Employeea
    allowableValues:
      – OPERATOR
      – MANAGER
 – id: branchId
    title: Branch Id
 – id: maxOrderAmount
    title: Maximum Order
    type: int
```

```
customer:
  title: Client
  attributes:
  – id: id
  – id: branchId
    title: Branch Id
```

```
order:
  title: Заказ
  actions:
  – view
  – create
  – approve
  – reject
  attributes:
  – id: id
    title: Oder Id
  – id: amount
    title: Oder ammount
    type: int
  – id: branchId
    title: Branch Id
  – id: customerId
    title: Customer ID
```

## {Easyabac:policy}

CUSTIS®

```
order:
  title: Заказ
  actions:
  – view
  – create
  – approve
  – reject
  attributes:
  – id: id
    title: Oder Id
  – id: amount
    title: Oder ammount
    type: int
  – id: branchId
    title: Branch Id
  – id: customerId
    title: Customer ID
```

```
– title: Restriction on order changes
  accessToActions: [order.create, order.approve, order.reject]
  rules:
   – title: Only during office hours for your branch
     operation: AND
     conditions:
      – env.time >= 09:00
      – env.time <= 17:00
      – order.branchId == subject.branchId
```

```
– title: Manager
  accessToActions: [order.approve, order.reject]
  rules:
   – title: Manager Access
     operation: AND
     conditions:
     – subject.role in 'MANAGER'
     – subject.maxOrderAmount > order.amount
```

# {Easyabac:policy}

CUSTIS®

```
order:
  title: Заказ
  actions:
  – view
  – create
  – approve
  – reject
  attributes:
  – id: id
    title: Oder Id
  – id: amount
    title: Oder ammount
    type: int
  – id: branchId
    title: Branch Id
  – id: customerId
    title: Customer ID
```

```
– title: Restriction on order change
  accessToActions: [order.create, order.approve, order.reject]
  rules:
   – title: Only during working hours for your branch
     operation: AND
     conditions:
      – env.time >= 09:00
      – env.time <= 17:00
      – order.branchId == subject.branchId
```

```
– title: Manager
  accessToActions: [order.approve, order.reject]
  rules:
   – title: Manager Access
     operation: AND
     conditions:
     – subject.role in 'MANAGER'
     – subject.maxOrderAmount > order.amount
```

# {Easyabac:policy}

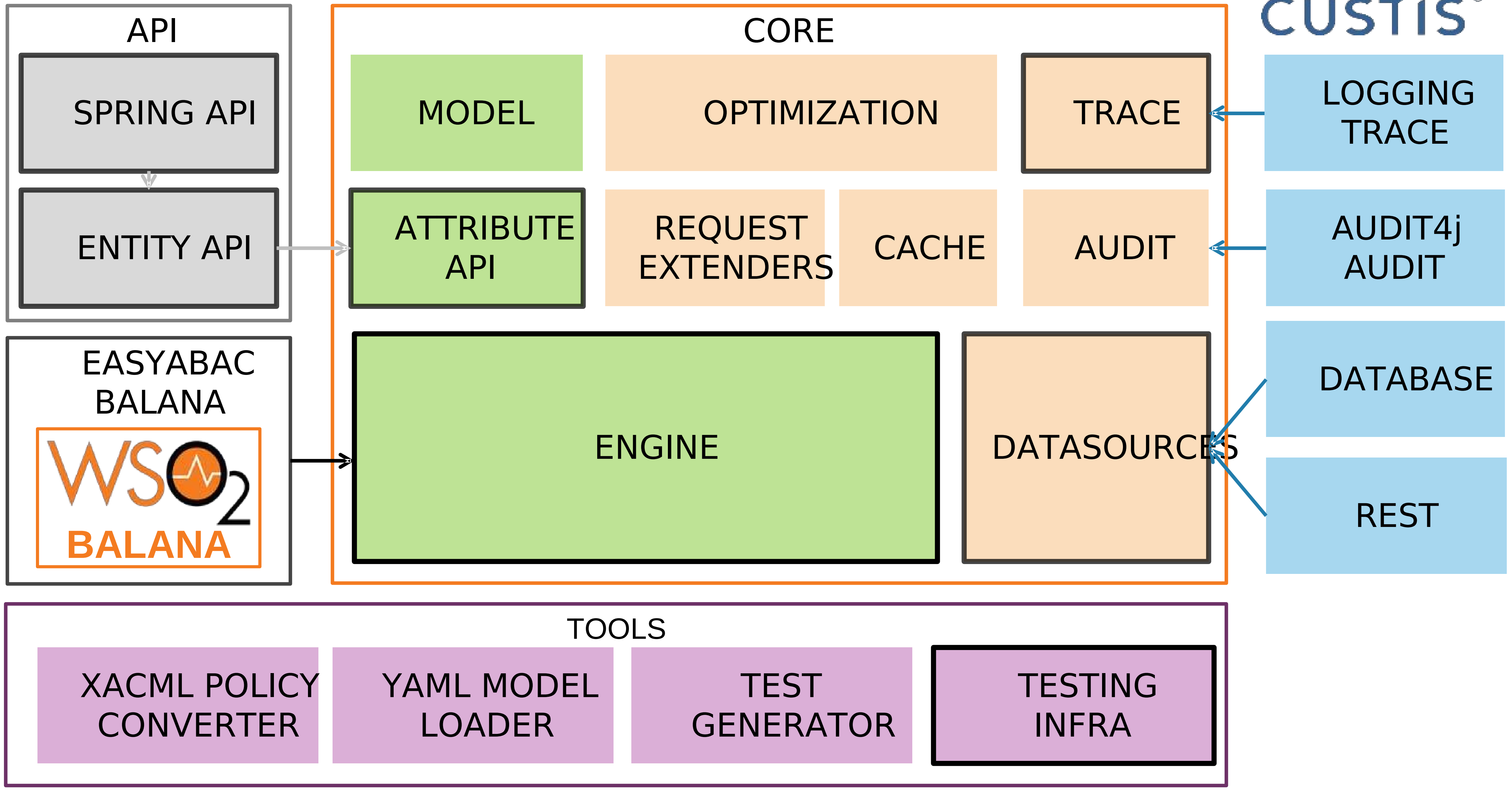
```
order:
  title: Заказ
  actions:
  – view
  – create
  – approve
  – reject
  attributes:
  – id: id
    title: Oder Id
  – id: amount
    title: Oder ammount
    type: int
  – id: branchId
    title: Branch Id
  – id: customerId
    title: Customer ID
```

```
– title: Restriction on order change
  accessToActions: [order.create, order.approve, order.reject]
  rules:
   – title: Only during working hours for your branch
     operation: AND
     conditions:
      – env.time >= 09:00
      – env.time <= 17:00
      – order.branchId == subject.branchId
```

```
– title: Manager
  accessToActions: [order.approve, order.reject]
  rules:
   – title: Manager Access
     operation: AND
     conditions:
     – subject.role in 'MANAGER'
     – subject.maxOrderAmount > order.amount
```

## {Easyabac:policy:restrictions}

No policy groups
Simple terms
Only basic functions (==,>, <, in, etc.)
Only basic data types (no support for rfc822, uri, etc.).
Strings - the main data type


API
SPRING API
ENTITY API
CORE
MODEL
OPTIMIZATION
TRACE
ATTRIBUTE API
REQUEST EXTENDERS
CACHE
AUDIT
ENGINE
DATASOURCES
EASYABAC BALANA
WSO2 BALANA
LOGGING TRACE
AUDIT4j AUDIT
DATABASE
REST
TOOLS
XACML POLICY CONVERTER
YAML MODEL LOADER
TEST GENERATOR
TESTING INFRA

# {Easyabac:tools}

## {Easyabac:tools:test}

```
public class OrderAuthTest extends EasyAbacBaseTestClass {

    public OrderAuthTest () {
            super(model); // model for testing
    }

    @Parameters(name = "{index}: resource({0}) and action({1}). Expected = ({2})")
    public static List<Object[]> data() throws Exception {
            // test data
    }
}
```

## {Easyabac:tools:test}

```
@RunWith(Parameterized.class)
public abstract class EasyAbacBaseTestClass {

    @Parameterized.Parameter
    public Object resource;

    @Parameterized.Parameter(value = 1)
    public Object action;

    @Parameterized.Parameter(value = 2)
    public boolean expectedPermit;

    @Parameterized.Parameter(value = 3)
    public TestDescription testDescription;
….
}
```

## {Easyabac:tools:test}

CUSTIS®

**order_0.yaml**

```yaml
expectedResult: PERMIT
action:
   id: order.action
   value: order.approve
attributes:
  order:
    amount: 500
    branchId: branchId_0
  subject:
    role: MANAGER
    branchId: branchId_0
    maxOrderAmount: 600
```

## {Easyabac:core}

# {Easyabac:core:datasources}

```
public abstract class Datasource {
    private final Set<Param> params;
    private final String returnAttributeId;
    private final long expire;

    private Attribute returnAttribute;


    abstract public List<String> find()
                            throws EasyAbacDatasourceException;

}

public class Param {
    private final String name;
    private final String attributeParamId;
    private String value;
    private Attribute attributeParam;
}
```

# {Easyabac:core:datasources}

```
public abstract class Datasource {
    private final Set<Param> params;
    private final String returnAttributeId;
    private final long expire;

    private Attribute returnAttribute;


    abstract public List<String> find()
                          throws EasyAbacDatasourceException;

}
```

There are already implementations for
DB - DatabaseDatasource
REST - RESTDatasour

## {Easyabac:core:trace}

CUSTIS®

```
– title:  Restriction on order changes
  accessToActions: [order.create, order.approve, order.reject]
  rules:
   – title: Only during office hours for your branch
     operation: AND
     conditions:
      – env.time >= 09:00
      – env.time <= 17:00
      – order.branchId == subject.branchId
```

```
– title: Manager
  accessToActions: [order.approve, order.reject]
  rules:
   – title: Manager Access
     operation: AND
     conditions:
     – subject.role in 'MANAGER'
     – subject.maxOrderAmount > order.amount
```

# {Easyabac:core:trace}

```
– title: Restriction on order changes
  accessToActions: [order.create, order.approve, order.reject] MATCH=TRUE, ACTION=order.approve
  rules:
   – title: Only during working hours for your branch
     operation: AND                                            RESULT=TRUE
     conditions:
      – env.time >= 09:00                                      RESULT=TRUE, env.time = 15:43
      – env.time <= 17:00                                      RESULT=TRUE, env.time = 15:43
      – order.branchId == subject.branchId                      RESULT=TRUE,
                                                                     order.branchId = Moscow
                                                                     subject.branchId = Moscow
```

```
– title: Manager
  accessToActions: [order.approve, order.reject]                MATCH=TRUE, ACTION=order.approve
  rules:
   – title: Manager Access
     operation: AND                                            RESULT=FALSE
     conditions:
     – subject.role in 'MANAGER‘                               RESULT=FALSE, subject.role = OPERATOR
     – subject.maxOrderAmount > order.amount                   RESULT=N/A
```

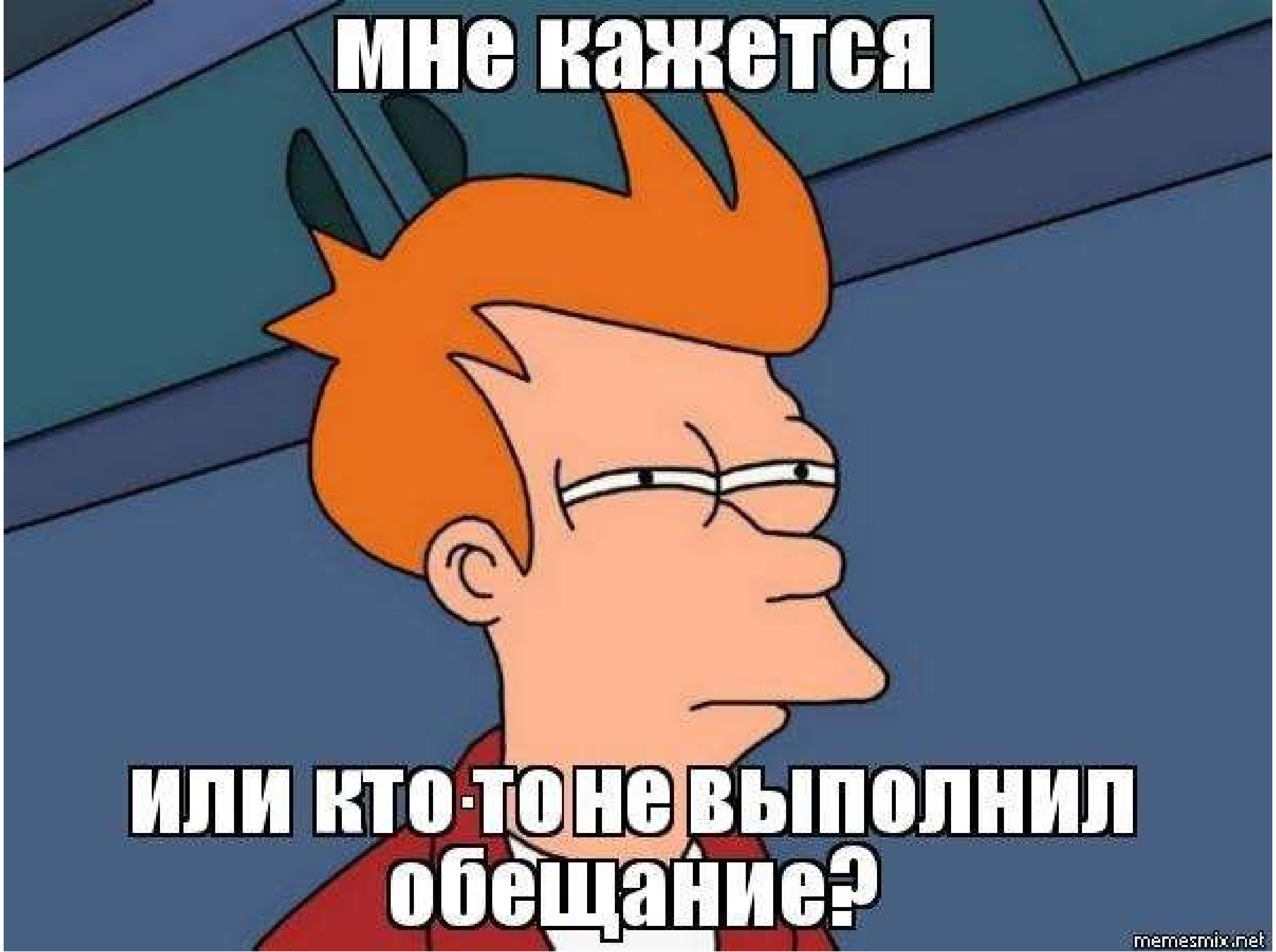
## {Easyabac:api}

## {Example}

```
public interface AuthService {

    AuthResponse authorize(List<AuthAttribute> attributes);

    Map<RequestId, AuthResponse>
                                    authorizeMultiple(Map<RequestId, List<AuthAttribute>> attributes);
}



void checkOrder (Order order, List<OrderAction> operations) throws NotPermittedException {
              // TODO implement through EasyAbac
}
```

## {Example}

```
public void checkOrder (Order order, List<OrderAction> operations)
                                                              throws NotPermittedException {
  Map<RequestId, List<AuthAttribute>> attributes = operations.stream()
        .map(operation -> extract(order, operation))
        .collect(Collectors.toMap(o -> RequestId.newRandom(), o -> o));

  Map<RequestId, AuthResponse> results = authService.authorizeMultiple(attributes);

  for (AuthResponse result : results.values()) {
    if (result.getDecision() != AuthResponse.Decision.PERMIT) {
      throw new NotPermittedException("Not permitted");
    }
  }
}
```


мне кажется
или кто-то не выполнил обещание?
memesmix.net

## {Example}

```
public void checkOrder (Order order, List<OrderAction> operations)
                                                                      throws NotPermittedException {
  Map<RequestId, List<AuthAttribute>> attributes = operations.stream()
        .map(operation -> extract(order, operation))
        .collect(Collectors.toMap(o -> RequestId.newRandom(), o -> o));

  Map<RequestId, AuthResponse> results = authorizationService.authorizeMultiple(attributes);

  for (AuthResponse result : results.values()) {
    if (result.getDecision() != AuthResponse.Decision.PERMIT) {
      throw new NotPermittedException("Not permitted");
    }
  }
}
```

## {Easyabac:api:entity}

CUSTIS®

```
public interface EntityPermissionChecker<T, A> {

  void ensurePermitted(T entity, A operation) throws NotPermittedException;
  void ensurePermittedAny(T entity, List<A> operations) throws NotPermittedException;
  void ensurePermittedAll(T entity, List<A> operations) throws NotPermittedException ;
  void ensurePermittedAll(Map<T, A> operationsMap) throws NotPermittedException ;

  List<A> getPermittedActions(T entity, List<A> operations);
  Map<T, List<A>> getPermittedActions(Map<T, List<A>> operationsMap);
  Map<T, List<A>> getPermittedActions(List<T> entities, List<A> operations);
}
```

## {Flashback:rbac_v4}

```
void checkView(Order order) {
    User user = AuthenticationContext.currentUser();
    checkUserBranch(user, order.getBranch());
    if (user.hasRole(ROLE_MANAGER.name()) ||
                    user.hasRole(ROLE_OPERATOR.name())) {
        return;
    }
    throw new NotPermittedException("not permitted");
}

void checkUserBranch(User user, Branch branch) {
    if (user.getBranch().getId().equals(branch.getId())) return;
    throw new NotPermittedException("not permitted");
}
```

## {Easyabac:api:entity}

```
EasyAbac easyAbac = EasyAbacBuilderHelper.defaultBuilder(…).build();

EntityPermissionChecker<Order, OrderAction> permissionChecker =
EntityPermissionCheckerHelper.newPermissionChecker(easyAbac);


    public void checkView(Order order) {
        permissionChecker.ensurePermitted(order, OrderAction.VIEW);
    }
    …
}
```

## {Easyabac:api:entity:attr}

```
class Order implements AttributeAuthEntity {

    @Override
    public List<AuthAttribute> getAuthAttributes()  {
          // list of entity attributes
   }
}
```

```
enum OrderAction implements AttributiveAuthAction {

    @Override
    public AuthAttribute getAuthAttribute() {

          // action identifier

   }
}
```

## {Easyabac:api:entity:attr}

```
@AuthorizationEntity(name = "order")
public class Order {

    @AuthorizationAttribute
    private String id;

    @AuthorizationAttribute(id = "amount")
    private int orderAmount;

    @AuthorizationAttribute
    private String branchId;

    private String customerId;
    …
}
```

```
@AuthorizationAction(entity = "order")
public enum Action {

    VIEW("view"), CREATE("create"),
    APPROVE("approve"), REJECT("reject");

    @AuthorizationActionId
    private String id;
….
}
```

CUSTIS®

# {Easyabac:api}

+

## {Easyabac:api:spring}

```
@Indexed
public interface PermissionChecker<T, A> {

}


@Configuration
@EnablePermissionCheckers("custis.easyabac.demo.permissionchecker")
public class EasyABACConfiguration {

}
```

## {And what can …}

Check any results(permit, deny, …)

```
void ensureIndeterminate(Order order, OrderAction action)
                                                throws NotExpectedResultException;

void ensureDeniedAll(Order order, List<OrderAction> action)
                                                throws NotExpectedResultException;
```

## { And what can ...}

Make methods for a specific action.

```
void ensureDeniedRead(Order order) throws NotExpectedResultException;

void ensurePermittedReadOrApprove(Order order)
                                                    throws NotExpectedResultException;
```

## { And what can ...}

| Return **boolean**, not an exception

```
boolean isIndeterminate(Order order, OrderAction action);

boolean isDeniedAll(Order order, List<OrderAction> action);

boolean isDeniedRead(Order order);

boolean isPermittedReadOrApprove(Order order);
```

## { And what can ...}

### | Change input parameters

```
boolean isDeniedAll(List<Order> order, OrderAction action);

boolean isDeniedAll(OrderAction action, List<Order> order);

boolean isDeniedAll(Map<OrderAction Order> order);

boolean isDeniedAll(Map<OrderAction, List<Order>> order);

....
```

## { And what can ...}

CUSTIS®

### | Group results through get * methods

```
List<OrderAction> getDeniedActions(Order order, List<OrderAction> actions);

List<Order> getPermittedResources(List<Order> orders, OrderAction action);

Map<Order, List<OrderAction>> getDeniedActions(
                                        List<Order> orders, OrderAction action);

Map<OrderAction, List<Order>> getDeniedResources(
                                  List<OrderAction> actions, List<Order> orders);
```

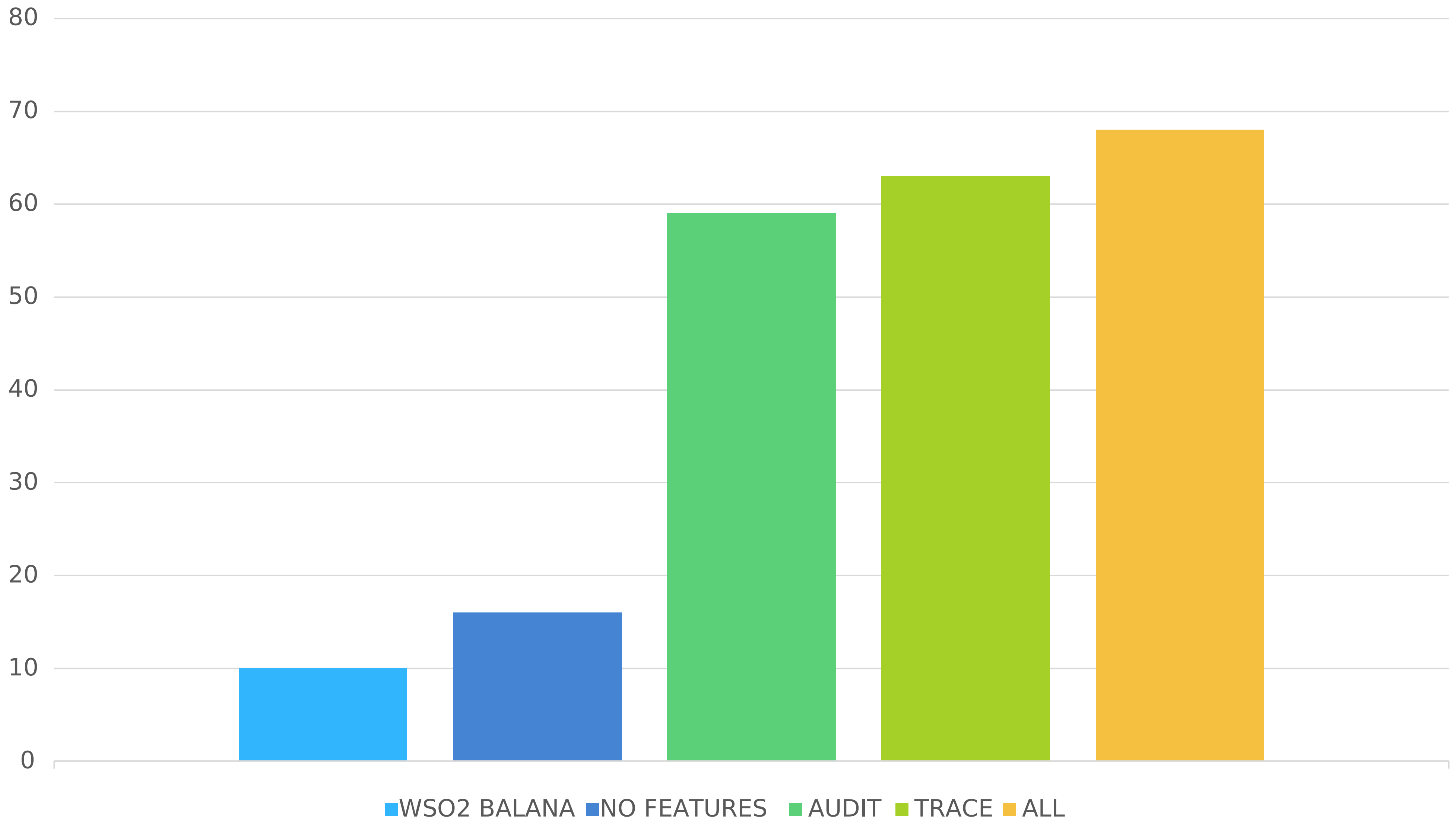
## {Easyabac:demo}

## {Easyabac:perf}

## {Easyabac core features}

- WSO2 BALANA
- NO FEATURES
- AUDIT
- TRACE
- ALL FEATURES

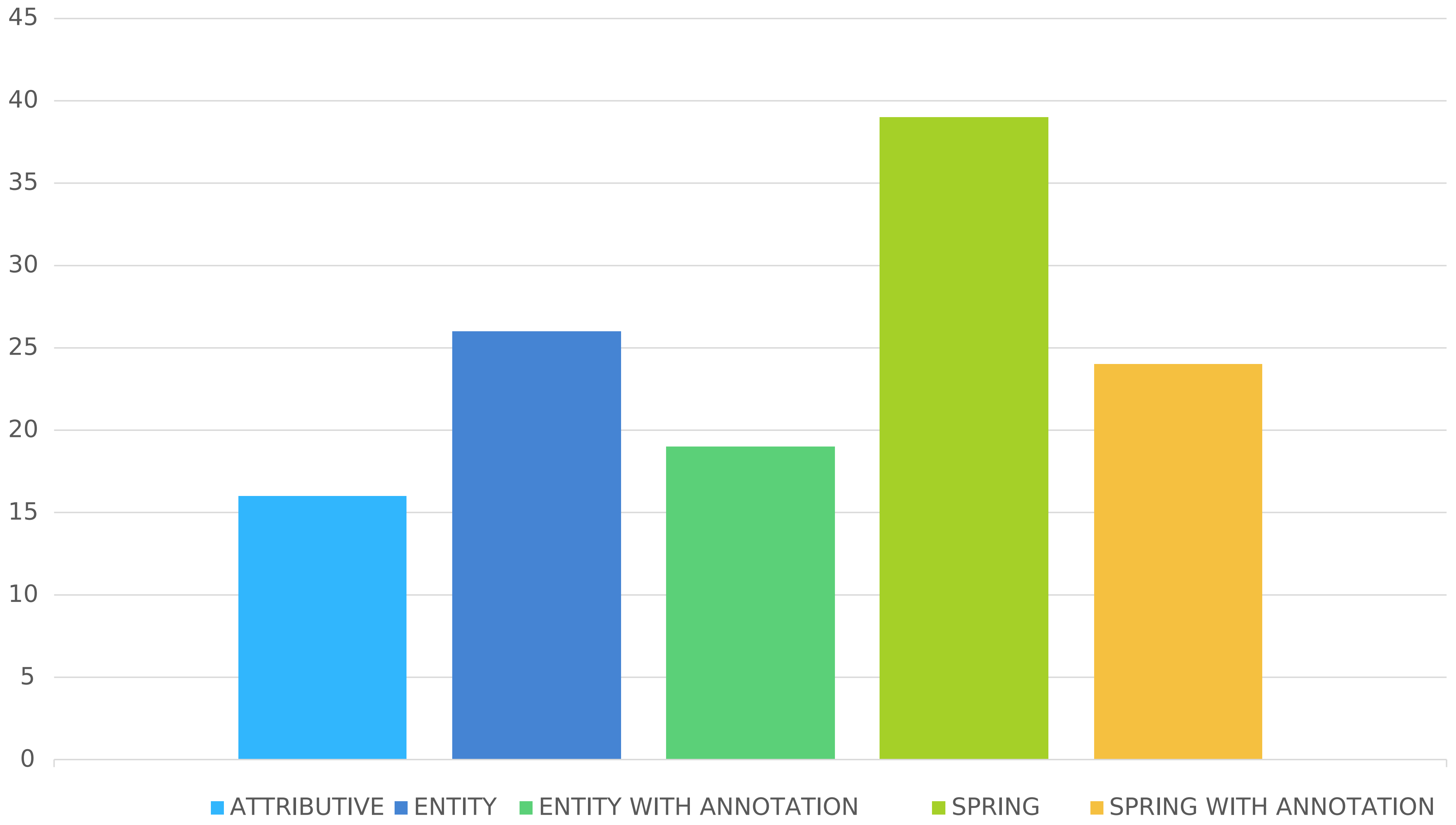
# Milliseconds per operation



## {Easyabac API}

- ATTRIBUTIVE
- ENTITY
- ENTITY WITH ANNOTATION
- SPRING
- SPRING WITH ANNOTATION

# Milliseconds per operation



# { Multi-query optimization }

**id:** 1
**amount:** 100 000
**branchId:** Moscow
**customerId:** 1234

APPROVE REJECT

**id:** 2
**amount:** 300 000
**branchId:** Moscow
**customerId:** 56789

APPROVE REJECT

**id:** 3
**amount:** 300 000
**branchId:** Moscow
**customerId:** 9090

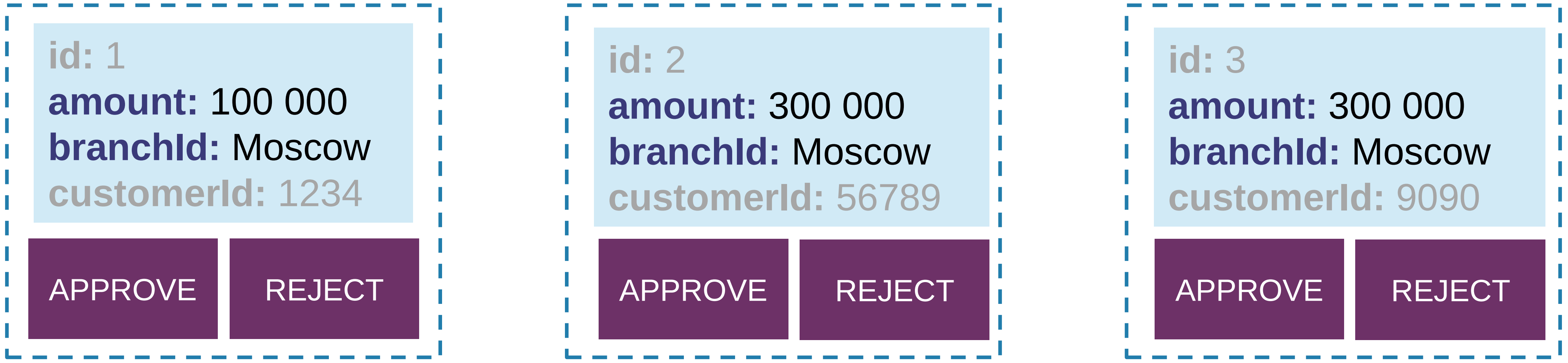
APPROVE REJECT

**About**

order.branchId == subject.branchId

subject.maxOrderAmount > order.amount

# { Multi-query optimization }



**id:** 1
**amount:** 100 000
**branchId:** Moscow
**customerId:** 1234

**id:** 2
**amount:** 300 000
**branchId:** Moscow
**customerId:** 56789

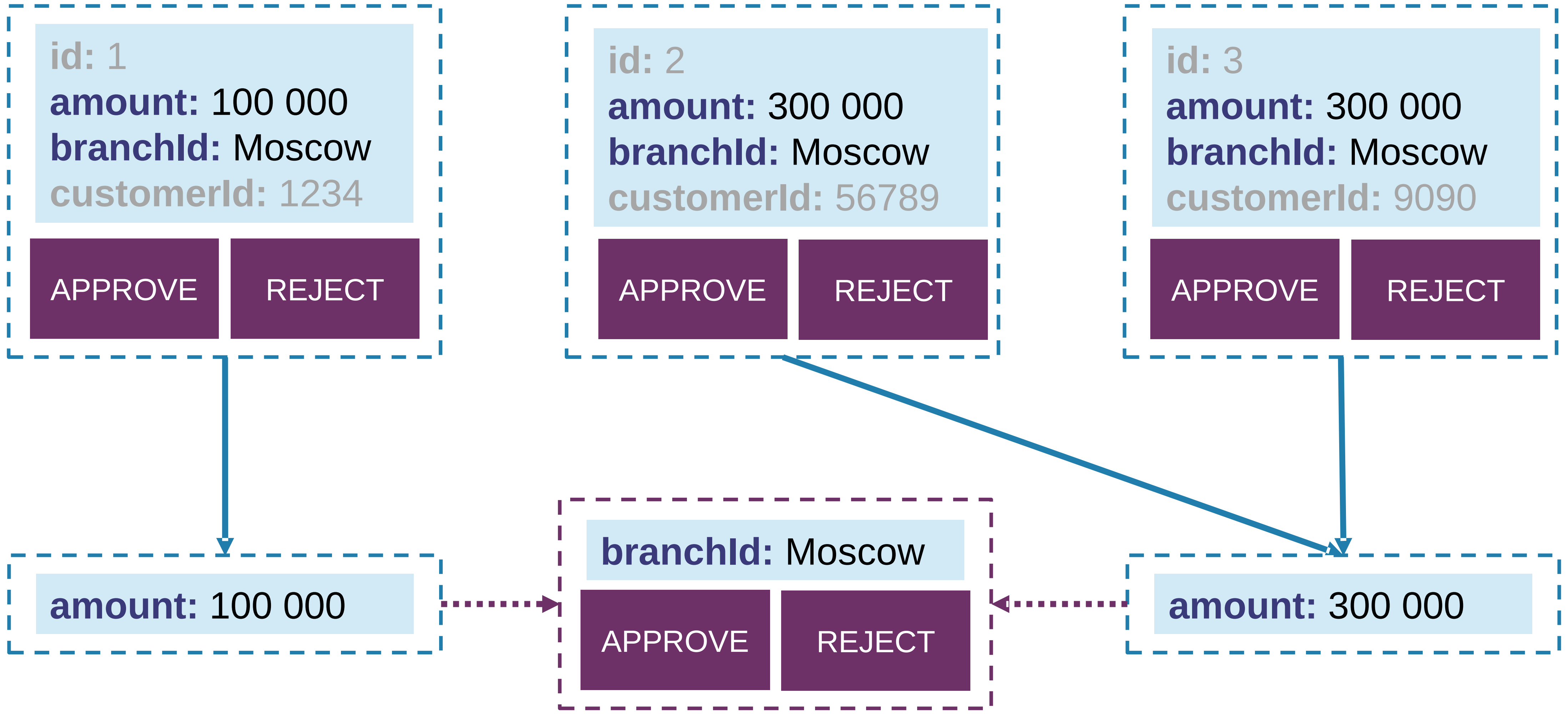
**id:** 3
**amount:** 300 000
**branchId:** Moscow
**customerId:** 9090

**About**

order.branchId == subject.branchId

subject.maxOrderAmount > order.amount

# { Multi-query optimization }



# { Multi-query optimization }

Depends on the specific task
Small requests are not required.
Customizable convolution threshold

## {Easyabac:roadmap}

Loading models from various sources
Standalone EasyABAC
Data Access Restriction
Simplified Policy Calculation
Additional extensions (Active Directory, ...)

## {Summary}

Attributive authorization approach
XACML standard and its implementation
Opensource easyacbac framework

## {Easyabac:source}

CUSTIS®

GitHub: CUSTIS-public/EasyABAC

Twitter: @easyabac

# СПАСИБО ЗА ВНИМАНИЕ!

Антон Лапицкий

Архитектор приложений

custis.ru

E-mail: alapitskiy@custis.ru

Twitter: @anton_lapitskiy