HUBCAP

Extending the Formal Security Analysis of the HUBCAP sandbox

Tomas Kulik, Prasad Talasila, Pietro Greco, Giuseppe Veneziano, Angelo Marguglio, Lorenzo Franco Sutton, Peter Gorm Larsen and Hugo Daniel Macedo

PROJECT PARTNERS
Aarhus University
Fortiss GmbH
Fundazione Bruno Kessler
University "Lucian Blaga" of Sibiu
Research Institutes of Sweden
Politecnico di Milano
Controllab Products
Verified Systems International
Technology Transfer Systems
Newcastle University
Virtual Vehicle Research
KTH Royal Institute of Technology
Engineering Ingegneria Informatica
F6S Network Limited
Unparallel Innovation
BEIA Consult
Validas

Horizon 2020 Programme
Grant Agreement #872698

Agenda

- Contribution
- HUBCAP project
- HUBCAP Sandbox Architecture
- Sandbox access control
- VDM Model
- Security analysis extensions
- Conclusion
- Future work

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Contribution

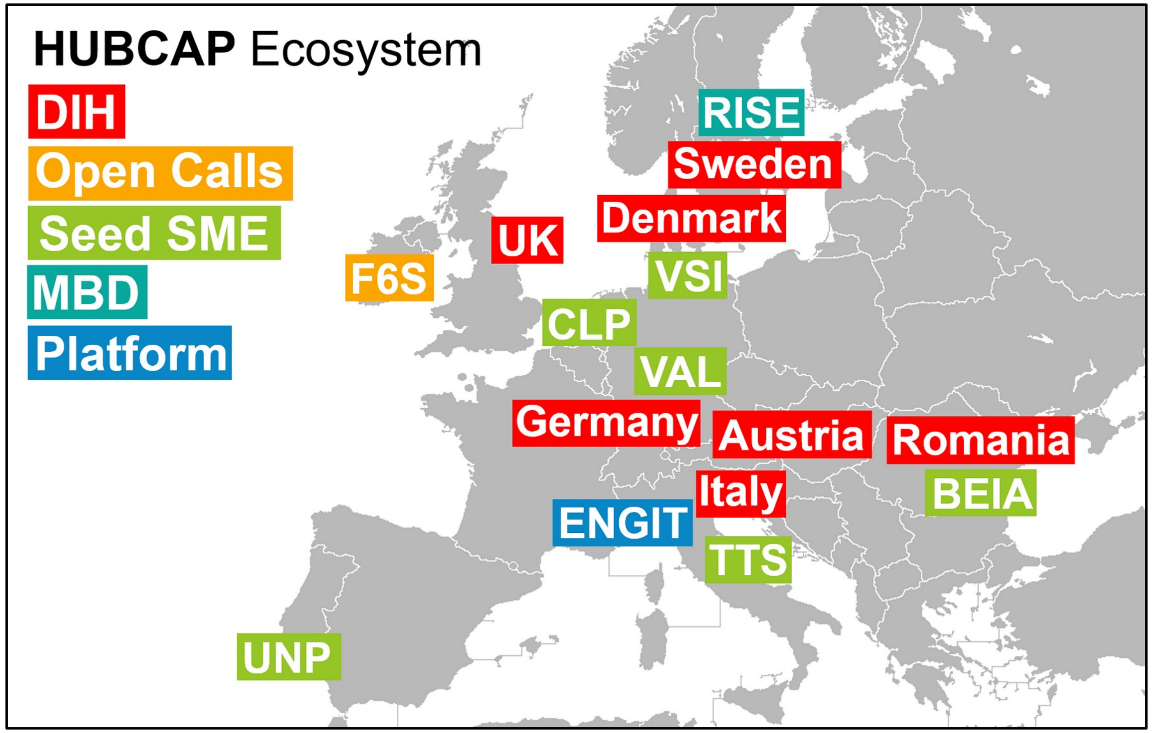
- Formally analyzed sandbox access model
- Improvements to the access model
- Continuous feature assessment
- Increased Security for Model based Engineering platform
- Approach to combinatorial testing of expanded access control properties

Horizon 2020 Programme
Grant Agreement #872698

ABOUT HUBCAP

• Who we are

- **Innovation Action** co-financed by the European Commission, DT-ICT-01-2019 Smart Anything Everywhere initiative.
- **Coordinator** Aarhus University, Denmark
- **Project duration** January 2020 - December 2022, 36 months
- **Total EC contribution** EUR ~7.95M
- HUBCAP will provide a one-stop-shop for European SMEs wanting to join the Cyber-Physical Systems (CPS) revolution using Model-Based Design (MBD) techniques.
- **Vision** Lower barriers for SMEs to realize the potential of growing autonomy in CPS by accessing advanced model-based design (MBD) technology, providing training and guidance.



Horizon 2020 Programme
Grant Agreement #872698

ABOUT HUBCAP

- Project setup

Network of DIHs:

- Inventory of service offerings
- Ecosystem building
- Cross-DIH collaboration
- Network sustainability

Seed SMEs

- Enabling quicker start for the platform
- Early-stage prototypical usage of HUBCAP
- Awareness-raising demonstrations

Collaboration Platform:

- Cloud-enabled, based on DIHIWARE
- “Access to” and “Collaborate with”:
  - Ecosystem
  - Community-building
  - Marketplace
  - **Sandbox**

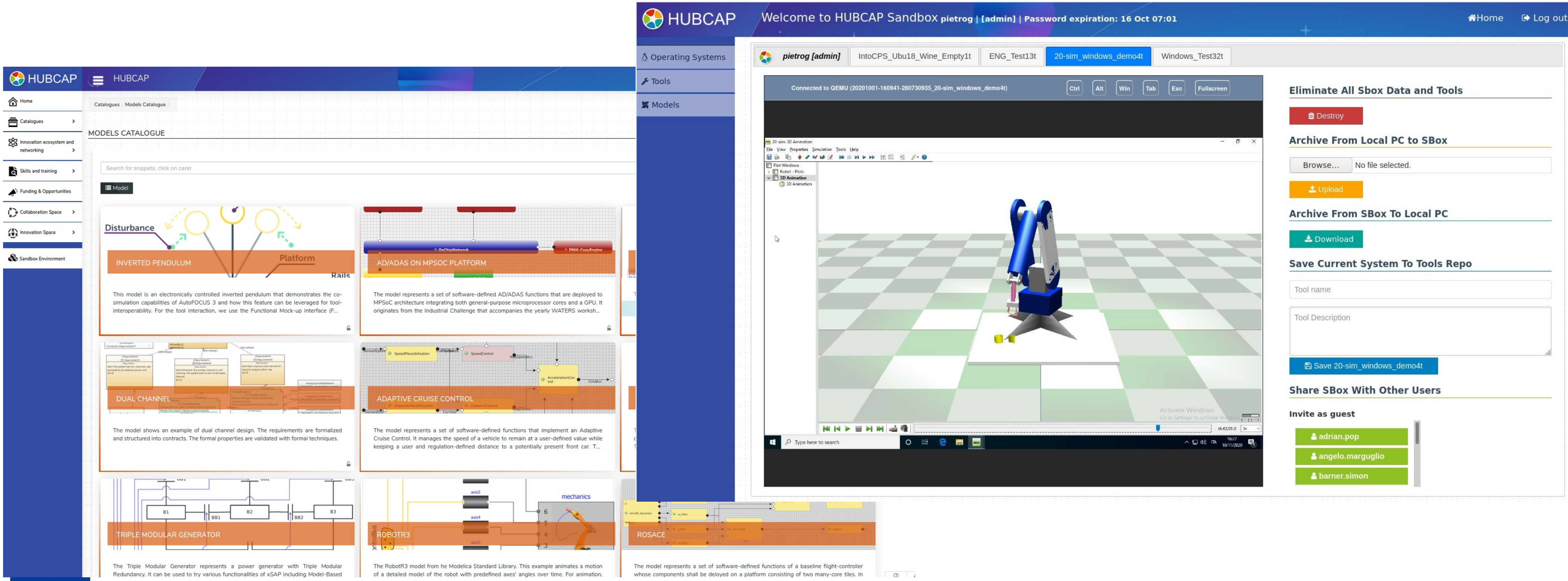
Open Calls

- Engage early-adopters
- 3 open call series

Model-Based Design:

- Populating the platform
- Enabling model-based services in sandbox
- Multi-user, validation and logging capabilities

HUBCAP Sandbox



INVERTED PENDULUM

This model is an electronically controlled inverted pendulum that demonstrates the co-simulation capabilities of AutoFOCUS 3 and how this feature can be leveraged for tool-interoperability. For the tool interaction, we use the Functional Mock-up Interface (F...

AD/ADAS ON MPSOC PLATFORM

The model represents a set of software-defined AD/ADAS functions that are deployed to MPSoC architecture integrating both general-purpose microprocessor cores and a GPU. It originates from the Industrial Challenge that accompanies the yearly WATERS worksh...

DUAL CHANNEL

The model shows an example of dual channel design. The requirements are formalized and structured into contracts. The formal properties are validated with formal techniques.

ADAPTIVE CRUISE CONTROL

The model represents a set of software-defined functions that implement an Adaptive Cruise Control. It manages the speed of a vehicle to remain at a user-defined value while keeping a user and regulation-defined distance to a potentially present front car. T...

TRIPLE MODULAR GENERATOR

The Triple Modular Generator represents a power generator with Triple Modular Redundancy. It can be used to try various functionalities of xSAP including Model-Based

ROBOTR3

The RobotR3 model from he Modelica Standard Library. This example animates a motion of a detailed model of the robot with predefined axes' angles over time. For animation,

ROSACE

The model represents a set of software-defined functions of a baseline flight-controller whose components shall be deloyed on a platform consisting of two many-core tiles. In

Welcome to HUBCAP Sandbox pietrog | [admin] | Password expiration: 16 Oct 07:01

Connected to QEMU (20201001-160941-280730935_20-sim_windows_demo4t)

Eliminate All Sbox Data and Tools

Archive From Local PC to SBox

No file selected.

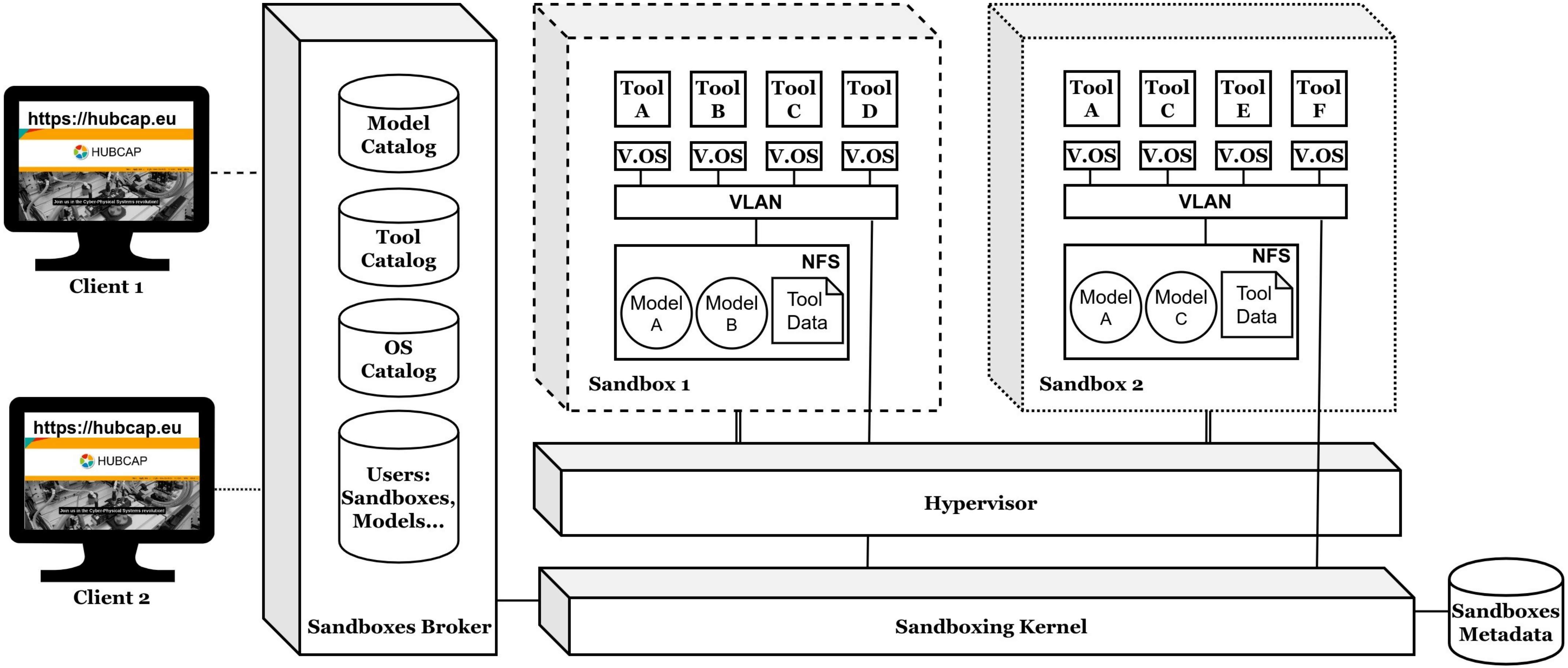
Archive From SBox To Local PC

Save Current System To Tools Repo

Share SBox With Other Users

Invite as guest

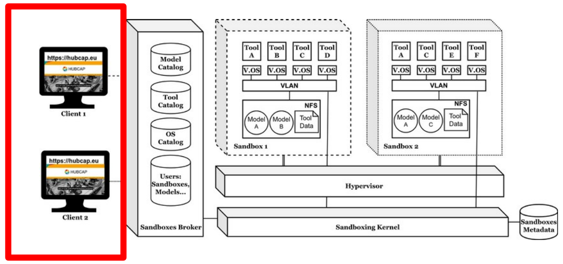
HUBCAP Platform architecture - Sandbox



Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks

- **Client**

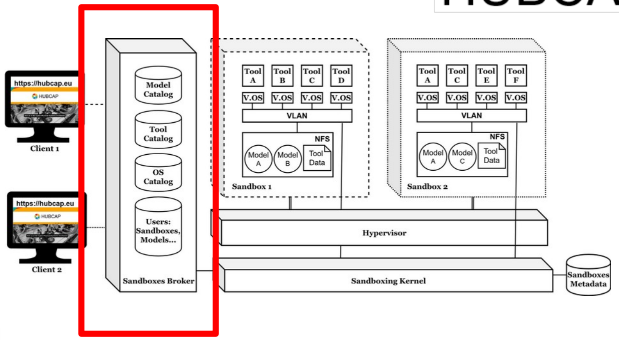


- **Remote access to sandbox** → Connect to Sandbox remotely based on the access rights (unique identity clients)
- **Interact with the HUBCAP Platform** → Manage existing Sandboxes or create new ones
- **Interact with the HUBCAP Sandboxes** → Download results of the tools within a Sandbox

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

HUBCAP Platform architecture – Primary building blocks



• Broker

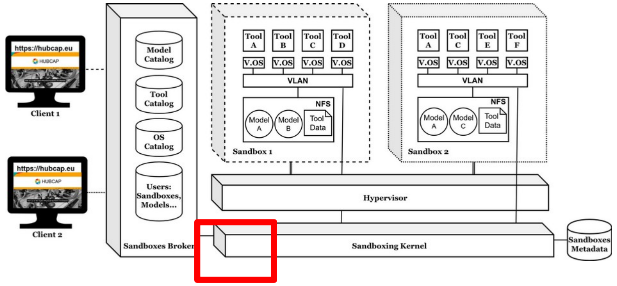
- **Connection handling to a Sandbox** → Facilitates client connection to a Sandbox
- **Component management within the platform** → Manages components of the HUBCAP platform such as the repository and Sandboxes, including starting of new Sandboxes
- **Persistence of Sandbox settings** → The broker records Sandbox metadata such as identities of the servers under the Sandbox

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks

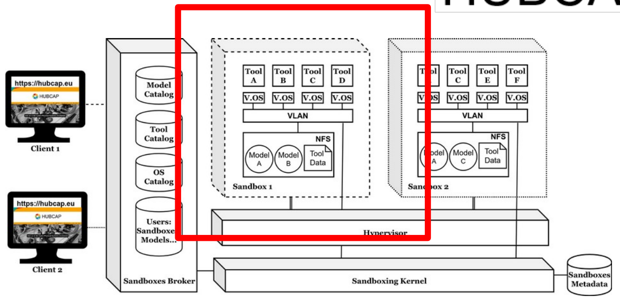
- **Gateway**

- **Direct connection from Client** → Keeps client connection to a Sandbox open
- **Server connection handling** → Manages connections from clients towards specific servers constituting a Sandbox, including the disconnections



Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks



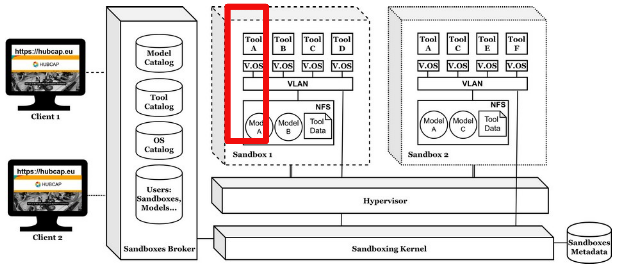
- **Sandbox**

- **Container for Servers** → A Sandbox is a collection of servers combining different tools and potentially models
- **Isolated internal network** → Sandbox provides isolated environment from the rest of the platform
- **Collaboration space** → Used for collaboration on specific modeling tasks for multiple clients, each Sandbox carries a unique identity

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks

- **Server**



- **Virtualization** → Servers within the Sandbox are considered virtual machines, facilitating fast deployment
- **Different Operating Systems** → Servers can be installed with different operating systems
- **Tool Hosting** → Servers host different tools utilized for model based engineering, results could be downloaded

Horizon 2020 Programme
Grant Agreement #872698

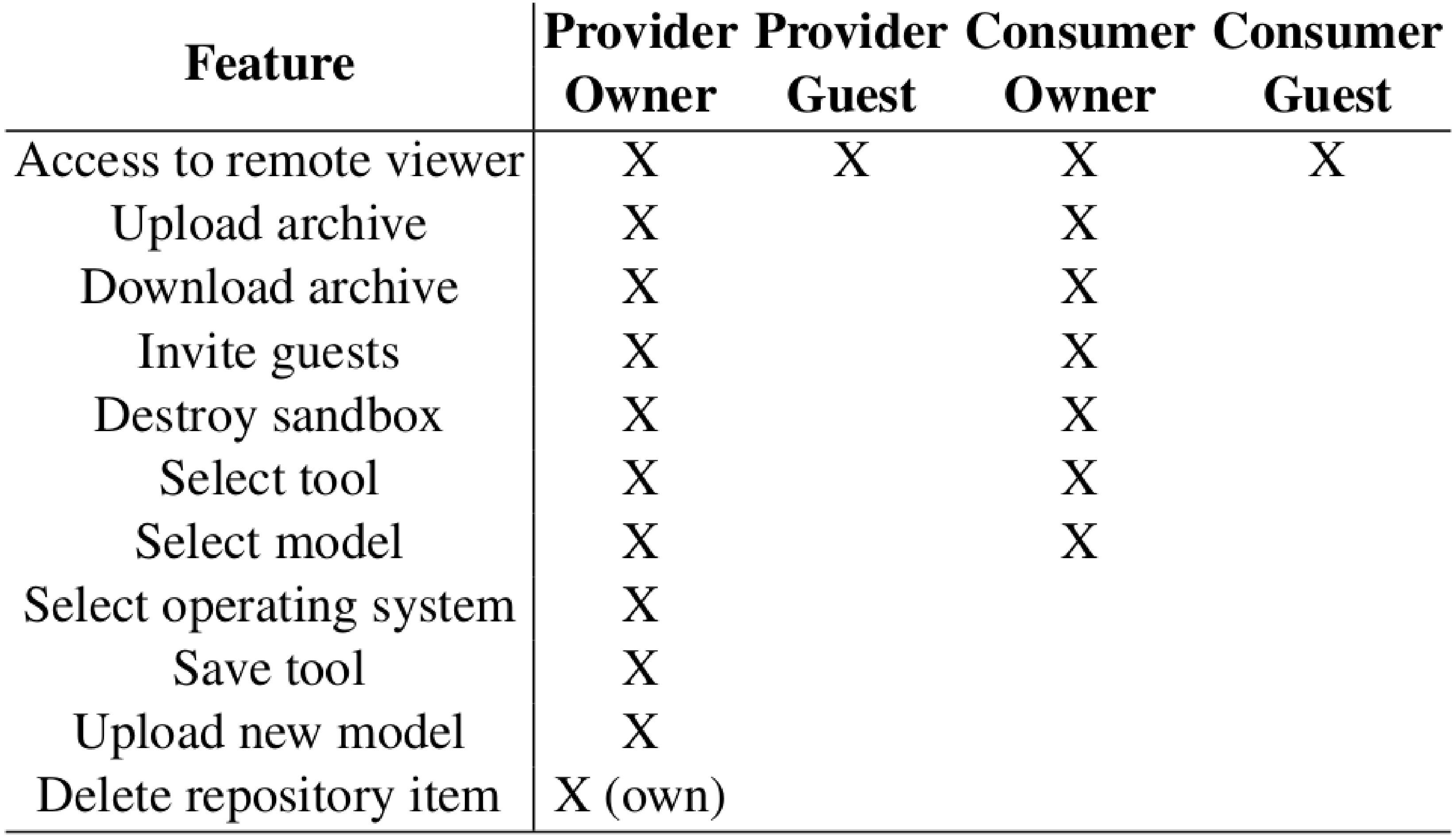
Sandbox Access Control

- Limit the user access to Sandbox
- Distinguish between providers and consumers
- Distinguish between owners and guests

- Limit the functionality based on the role or profile
- Ensure intellectual property protection
- Iterate to accommodate new functionality (from previous)

Horizon 2020 Programme
Grant Agreement #872698

Sandbox Access Control



Feature	Provider Owner	Provider Guest	Consumer Owner	Consumer Guest
Access to remote viewer	X	X	X	X
Upload archive	X		X	
Download archive	X		X	
Invite guests	X		X	
Destroy sandbox	X		X	
Select tool	X		X	
Select model	X		X	
Select operating system	X			
Save tool	X			
Upload new model	X			
Delete repository item	X (own)			

Horizon 2020 Programme
Grant Agreement #872698

VDM-SL Model

- Model the different components
- Model the System behavior on top
- Capture the access table

- Single module model
- Multiple traces for analysis
- Use of Combinatorial Testing

Horizon 2020 Programme
Grant Agreement #872698

Client VDM-SL Model

```
-- Select Tools
SelectToolsFromRepository: ClientId * SelectedTools ==> ()
SelectToolsFromRepository(cId, tIds) ==
     clientst.selectedTools := SelectTools(tIds, cId)
pre cId in set validClients and
          GetPrivateToolsByToolId(tIds, brokerst.validTools) <> {} =>
               forall t in set GetPrivateToolsByToolId(tIds,
brokerst.validTools) & t.owner = cId;

-- Select Models
SelectModelsFromRepository: ClientId * SelectedModels ==> ()
SelectModelsFromRepository(cId, mIds) ==
     clientst.selectedModels := SelectModels(mIds, cId)
pre cId in set validClients;

-- Launch Sandbox
LaunchNewSandbox: ClientId ==> ()
LaunchNewSandbox(cId) ==
     def r = StartNewSandbox(cId, clientst.selectedTools,
clientst.selectedModels)
     in(clientst.latestSandbox := r)
pre cId in set validClients;
```

State

```
types

ClientSt::
     selectedTool : SelectedTool
     selectedOS : SelectedOS
     selectedModel : SelectedModel
     selectedTools : SelectedTools
     selectedModels : SelectedModels
     downloadedData : DownloadedData
     latestSandbox : SandboxId;


ClientId = nat;
SelectedTool = [nat];
SelectedOS = [nat];
SelectedModel = [nat];
SelectedTools = set of nat;
SelectedModels = set of nat;
DownloadedData = set of Data;
```

Horizon 2020 Programme
Grant Agreement #872698

Broker VDM-SL Model

```
functions
ClientIsNull: ClientId * Providers * Consumers * Owners * Guests -> bool
ClientIsNull(cId, ps, cs, os, gs)==
  cId not in set ps and
  cId not in set cs and
  cId not in set dom os and
  cId not in set dom gs;

operations
-- upload archive to a specific Sandbox
UploadArchiveToSandbox: ClientId * SandboxId * token ==> ()
UploadArchiveToSandbox(cId, sbId, arch) ==
     systemSandboxes(sbId).uploadedData := systemSandboxes(sbId).uploadedData union
{arch}
pre (cId in set brokerst.providers or cId in set brokerst.consumers) and
          (cId in set dom brokerst.owners and sbId in set brokerst.owners(cId))
post card systemSandboxes(sbId).uploadedData = card
systemSandboxes~(sbId).uploadedData + 1;

-- download archive from a specific Server
DownloadArchiveFromServer: ClientId * ServerId * SandboxId ==> token
DownloadArchiveFromServer(cId, sId, sbId) ==
if systemSandboxes(sbId).sandboxServers(sId).data <> mk_token(nil) then
     return systemSandboxes(sbId).sandboxServers(sId).data
else
     return mk_token(nil)
pre sbId in set brokerst.owners(cId);
```

```
types
```

State

```
BrokerSt ::
 providers : Providers
 consumers : Consumers
 validModels : ValidModels
 activeSandboxes : ActiveSandboxes
 validTools : ValidTools
 validOSs : ValidOSs
 owners : Owners
 guests : Guests
 errorLog : ErrorLog
 sandboxModels : SandboxModels
 sandboxTools : SandboxTools
 sandboxOSs : SandboxOSs;

Owners = map ClientId to set of SandboxId;
Guests = map ClientId to set of SandboxId;
ActiveSandboxes = set of SandboxId;
...
```

Tool + Server + Sandbox VDM-SL Model

Server

```
types
ServerId = nat;
Data = token;

Server::
    serverId : ServerId
    toolId : ToolId
    data : Data
```

Tool

```
types

Version = nat;
Private = bool;
OsOnly = bool;
Owner = ClientId;

Tool::
    osId : OSId
    version : Version
    private : Private
    osOnly : OsOnly
    owner : Owner
```

Sandbox

```
types
SandboxId = nat;
SandboxServers = set of ServerId;
UploadedData = set of token;

Sandbox::
 sandboxId : SandboxId
 sandboxServers : SandboxServers
 uploadedData : UploadedData
```

Destroying a Sandbox

```
-- Destroying the sandbox removes it from known system sandboxes
DestroySandbox : ClientId * SandboxId ==> ()
DestroySandbox(cId, sId) ==
(systemSandboxes := {sId} <-: systemSandboxes;
brokerst.owners(cId) := brokerst.owners(cId) \ {sId})
pre cId in set dom brokerst.owners
and
sId in set brokerst.owners(cId)
and
not sId in set brokerst.activeSandboxes
post
sId not in set brokerst.owners(cId);
```

Horizon 2020 Programme
Grant Agreement #872698

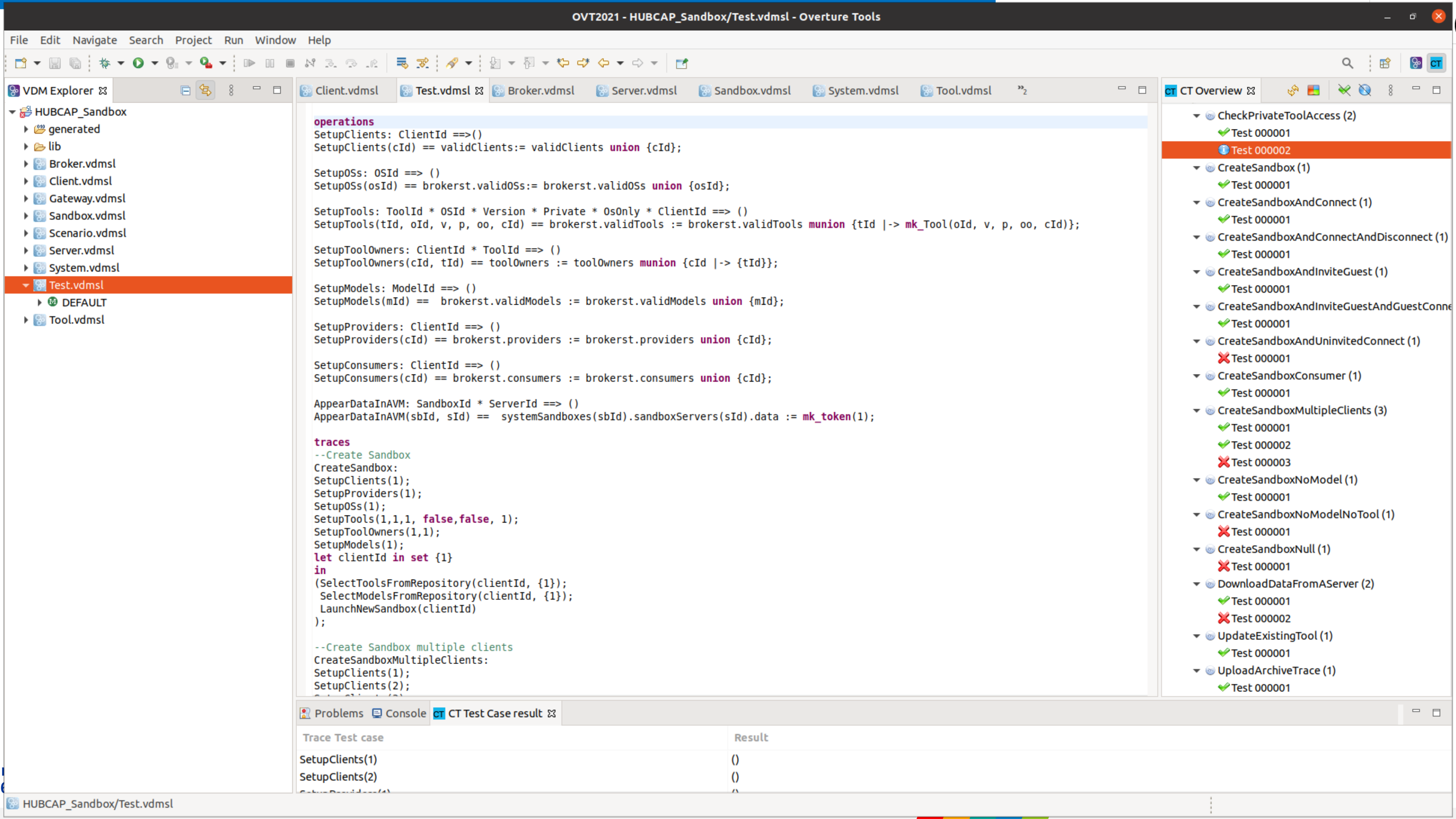
System VDM-SL Model

Initial State

```
state SystemSt of
    gatewayConnections : GatewayConnections
    gatewayConnectionsSandbox :
GatewayConnectionsSandbox
    systemSandboxes : SystemSandboxes
  toolOwners : ToolOwners
  modelOwners : ModelOwners
    brokerst : BrokerSt
    clientst: ClientSt
    validClients : ValidClients
    systemServers : SystemServers

inv ss == dom ss.gatewayConnections = dom
ss.gatewayConnectionsSandbox
init s == s = mk_SystemSt({|->},{|->},{|->},{|->},{|->},
                        mk_BrokerSt({},{},{},{},{|->},{},
{|->},{|->},[],{|->},{|->},{|->}),
                        mk_ClientSt(nil,nil,nil,{},{},
{},0),{},{})
end
```

```
-- Get private tools Ids by tool Id
pure GetPrivateToolsByToolId: set of ToolId * map ToolId to
Tool ==> set of Tool
GetPrivateToolsByToolId(tIds, valTools) ==(
    dcl tools: map ToolId to Tool := {|->};
    tools := tIds <: valTools;
    return {t | t in set rng tools & t.private = true})
pre tIds subset dom valTools;

-- Get the latest server Id
MaxServer : SystemServers -> nat
MaxServer(ss) ==
  if ss = {} then 0
  else let max in set ss be st forall d in set ss & d <=max
       in
         max;

GenerateNewServerId:()==> nat
GenerateNewServerId()==
    return MaxServer(systemServers) + 1;
```

Overture – Combinatorial Testing



- HUBCAP_Sandbox
  - generated
  - lib
  - Broker.vdmsl
  - Client.vdmsl
  - Gateway.vdmsl
  - Sandbox.vdmsl
  - Scenario.vdmsl
  - Server.vdmsl
  - System.vdmsl
  - Test.vdmsl
    - DEFAULT
  - Tool.vdmsl

```
operations
SetupClients: ClientId ==>()
SetupClients(cId) == validClients:= validClients union {cId};

SetupOSs: OSId ==> ()
SetupOSs(osId) == brokerst.validOSs:= brokerst.validOSs union {osId};

SetupTools: ToolId * OSId * Version * Private * OsOnly * ClientId ==> ()
SetupTools(tId, oId, v, p, oo, cId) == brokerst.validTools := brokerst.validTools munion {tId |-> mk_Tool(oId, v, p, oo, cId)};

SetupToolOwners: ClientId * ToolId ==> ()
SetupToolOwners(cId, tId) == toolOwners := toolOwners munion {cId |-> {tId}};

SetupModels: ModelId ==> ()
SetupModels(mId) ==  brokerst.validModels := brokerst.validModels union {mId};

SetupProviders: ClientId ==> ()
SetupProviders(cId) == brokerst.providers := brokerst.providers union {cId};

SetupConsumers: ClientId ==> ()
SetupConsumers(cId) == brokerst.consumers := brokerst.consumers union {cId};

AppearDataInAVM: SandboxId * ServerId ==> ()
AppearDataInAVM(sbId, sId) ==  systemSandboxes(sbId).sandboxServers(sId).data := mk_token(1);

traces
--Create Sandbox
CreateSandbox:
SetupClients(1);
SetupProviders(1);
SetupOSs(1);
SetupTools(1,1,1, false,false, 1);
SetupToolOwners(1,1);
SetupModels(1);
let clientId in set {1}
in
(SelectToolsFromRepository(clientId, {1});
 SelectModelsFromRepository(clientId, {1});
 LaunchNewSandbox(clientId)
);

--Create Sandbox multiple clients
CreateSandboxMultipleClients:
SetupClients(1);
SetupClients(2);
```

- CheckPrivateToolAccess (2)
  - ✓ Test 000001
  - Test 000002
- CreateSandbox (1)
  - ✓ Test 000001
- CreateSandboxAndConnect (1)
  - ✓ Test 000001
- CreateSandboxAndConnectAndDisconnect (1)
  - ✓ Test 000001
- CreateSandboxAndInviteGuest (1)
  - ✓ Test 000001
- CreateSandboxAndInviteGuestAndGuestConne
  - ✓ Test 000001
- CreateSandboxAndUninvitedConnect (1)
  - ✗ Test 000001
- CreateSandboxConsumer (1)
  - ✓ Test 000001
- CreateSandboxMultipleClients (3)
  - ✓ Test 000001
  - ✓ Test 000002
  - ✗ Test 000003
- CreateSandboxNoModel (1)
  - ✓ Test 000001
- CreateSandboxNoModelNoTool (1)
  - ✗ Test 000001
- CreateSandboxNull (1)
  - ✗ Test 000001
- DownloadDataFromAServer (2)
  - ✓ Test 000001
  - ✗ Test 000002
- UpdateExistingTool (1)
  - ✓ Test 000001
- UploadArchiveTrace (1)
  - ✓ Test 000001

Trace Test case	Result
SetupClients(1)	()
SetupClients(2)	()

HUBCAP

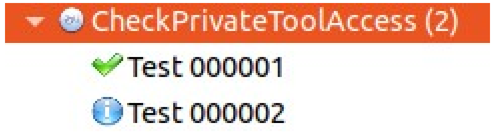
Combinatorial Testing traces

- **Validating the private tool handling**

```
operations
SetupClients: ClientId ==>()
SetupClients(cId) == validClients:= validClients union
{cId};

SetupOSs: OSId ==> ()
SetupOSs(osId) == brokerst.validOSs:= brokerst.validOSs
union {osId};

SetupProviders: ClientId ==> ()
SetupProviders(cId) == brokerst.providers :=
brokerst.providers union {cId};

SetupTools: ToolId * OSId * Version * Private * OsOnly *
ClientId ==> ()
SetupTools(tId, oId, v, p, oo, cId) ==
brokerst.validTools := brokerst.validTools munion {tId |->
mk_Tool(oId, v, p, oo, cId)};
```

```
CheckPrivateToolAccess:
SetupClients(1);
SetupClients(2);
SetupProviders(1);
SetupProviders(2);
SetupOSs(1);
UploadTool(1, 1, 1, true, false);
let clientId in set {1,2}
in(
 SelectToolsFromRepository(clientId, {1});
);
```

User not owning the private tool



CheckPrivateToolAccess (2)
✓ Test 000001
ⓘ Test 000002

Results

- Suggestions to the implementation team
- Explicit roles for private tools and Sandbox data access
- Covered the current access functionality

- 15 traces expanding to 19 tests
- 4 seconds analysis time
- Small effort in validation – security properties captured as pre and post conditions

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Conclusion and future work

- VDM-SL is a good fit for the access analysis
- Combinatorial Testing provides a powerful analysis tool
- An explicit permission for Sandbox creation proposed to the implementation team

- Expand the model to cover aspects of federated cloud
- Use the VDM model in a runtime verification approach
- Utilize combinatorial testing to cover more scenarios

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Thank you

Thank you for your attention!

Horizon 2020 Programme
Grant Agreement #872698