HUBCAP

Study of Security of the HUBCAP Sandbox Architecture

Tomas Kulik, Hugo Daniel Macedo, Prasad Talasila and Peter Gorm Larsen

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

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Agenda

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

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Contribution

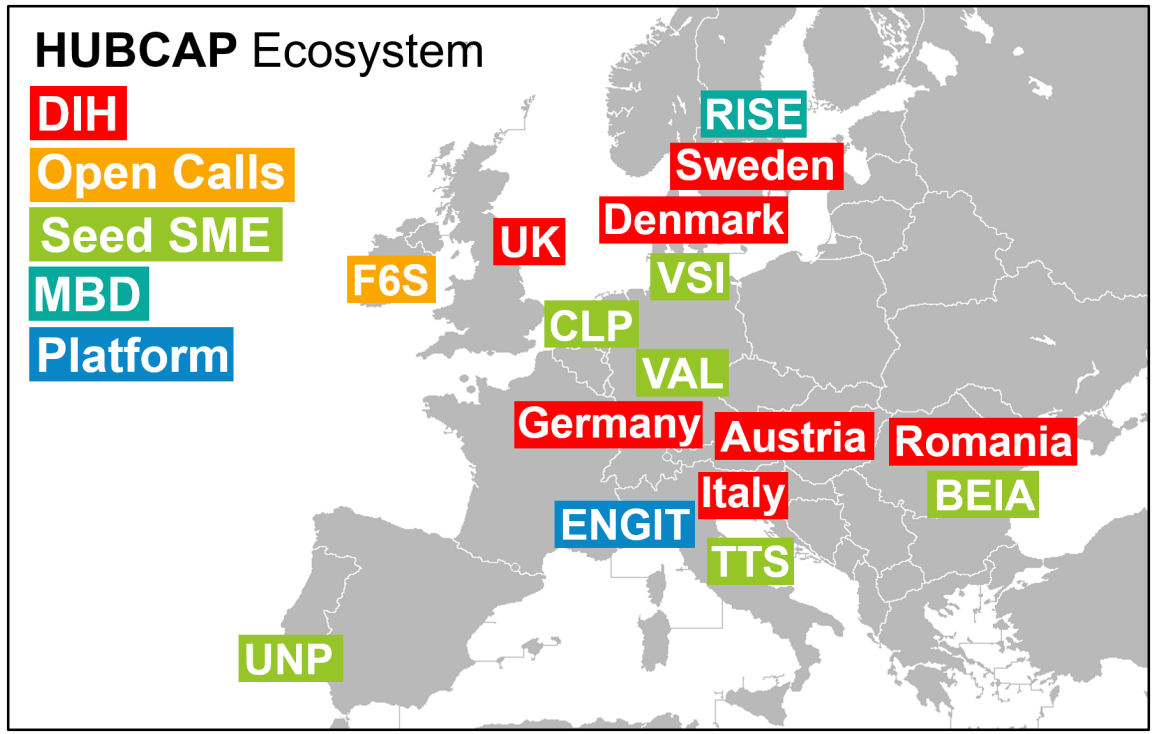
- Formally analyzed sandbox access model
- Improvements to the access model
- Move towards distributed Sandboxing
- Increased Security for Model based Engineering platform
- Approach to combinatorial testing of access control properties

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

HUBCAP

ABOUT HUBCAP

- **Project setup**

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

Open Calls
- Engage early-adopters
- 3 open call series

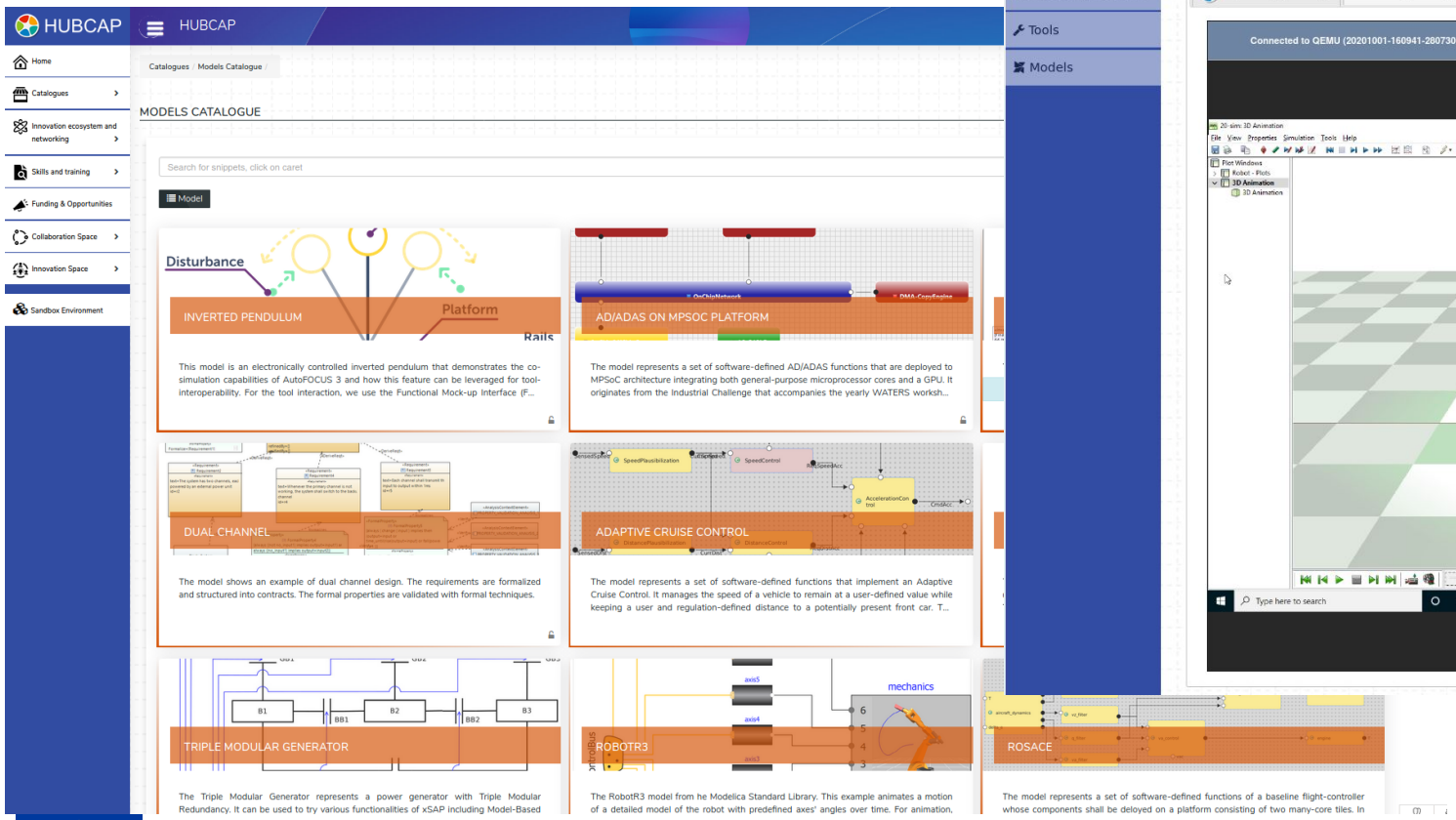
Seed SMEs
- Enabling quicker start for the platform
- Early-stage prototypical usage of HUBCAP
- Awareness-raising demonstrations

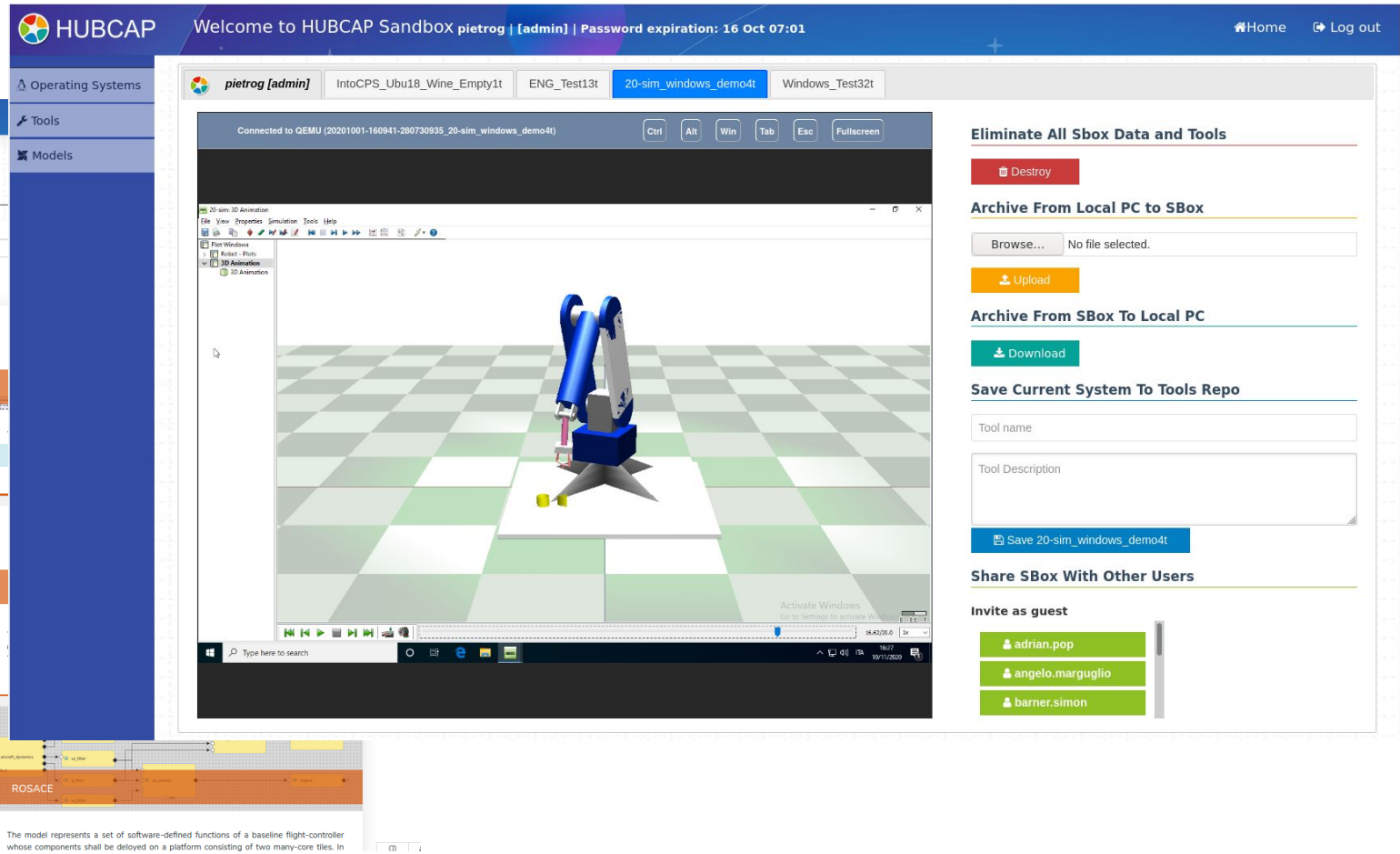
Model-Based Design:
- Populating the platform
- Enabling model-based services in sandbox
- Multi-user, validation and logging capabilities

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

Horizon 2020 Programme
Grant Agreement #872698

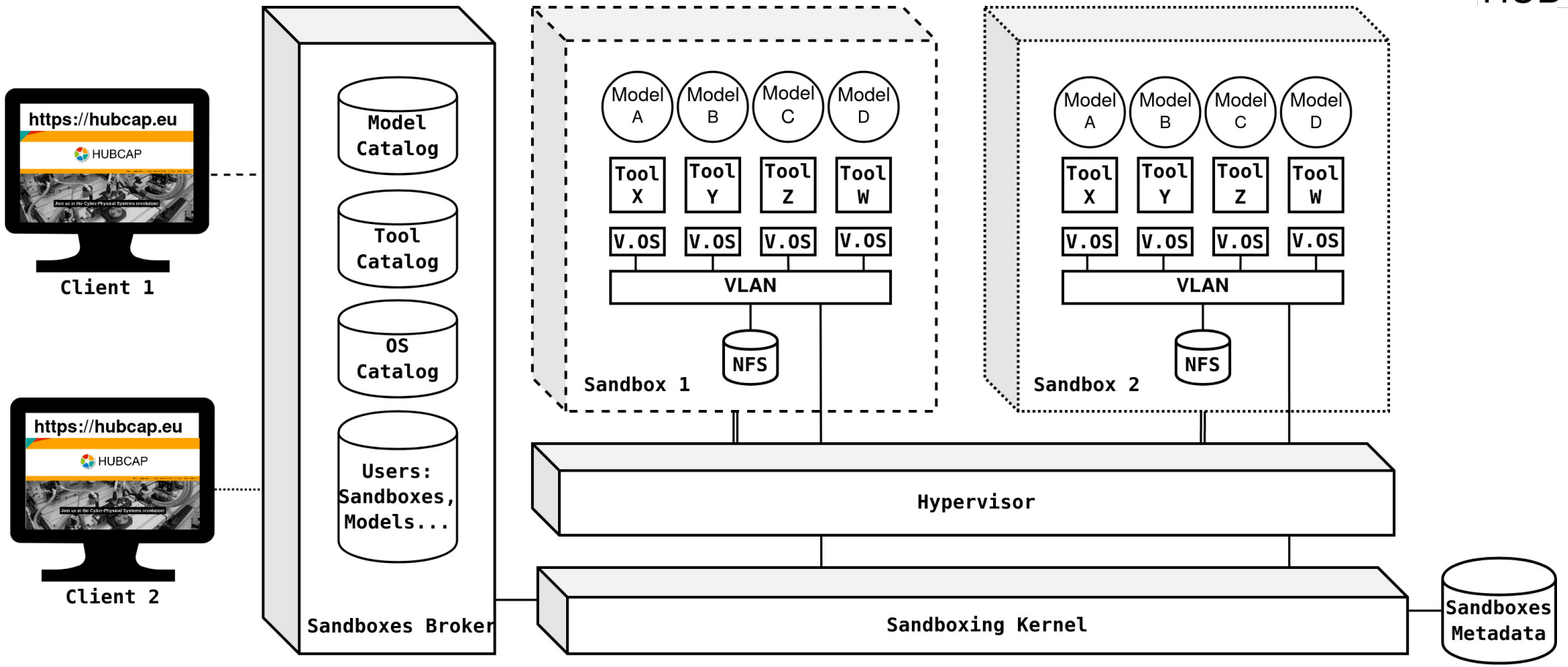
HUBCAP Sandbox





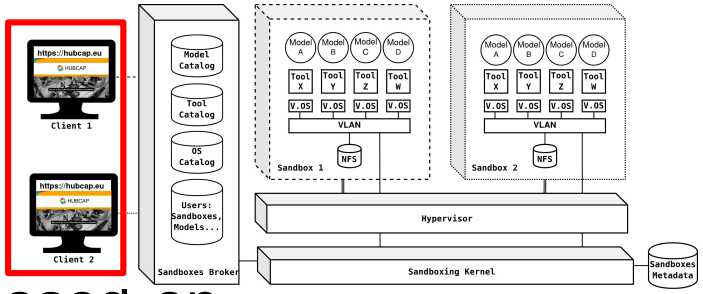
Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture - Sandbox



Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks

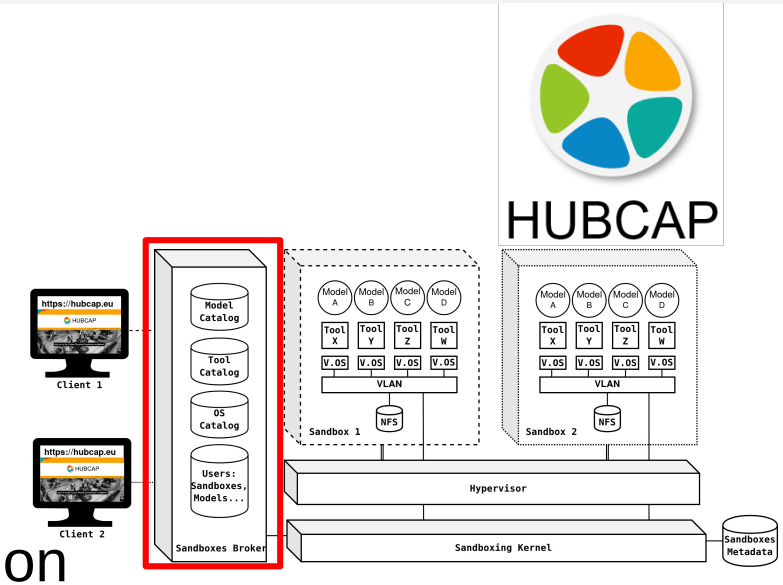


- **Client**

- **Remote access to sandbox** → Connect to Sandbox remotely based on the access rights
- **Interact with the HUBCAP Platform** → Manage existing Sandboxes or create new ones
- **Interact with the HUBCAP Platform** → Select existing repository items or provide new ones

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks



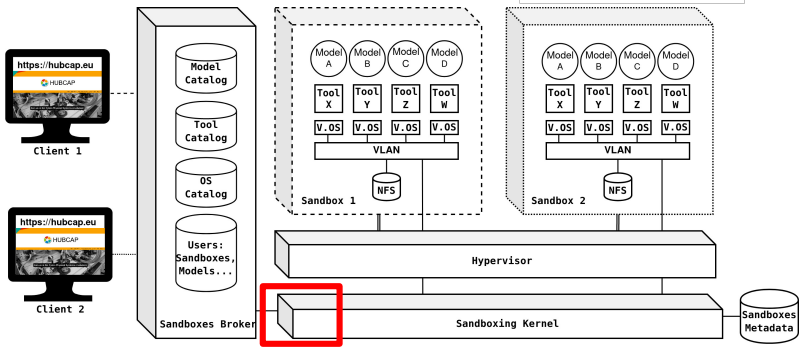
- **Broker**

- **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
- **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**

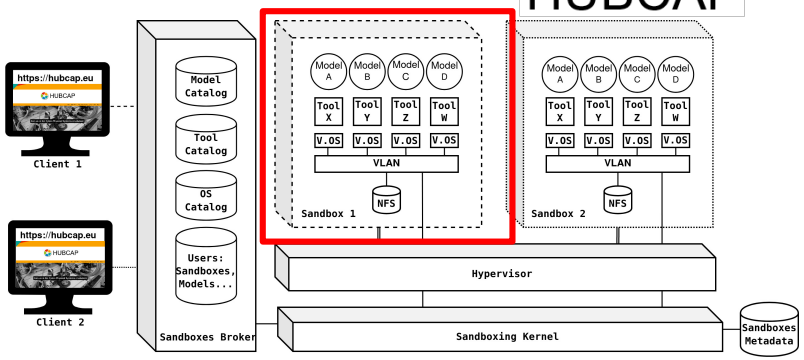


- **Direct connection from Client to** → Keeps client connection to a Sandbox open
- **Server connection handling** → Manages connections from clients towards specific servers constituting a Sandbox

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP Platform architecture – Primary building blocks

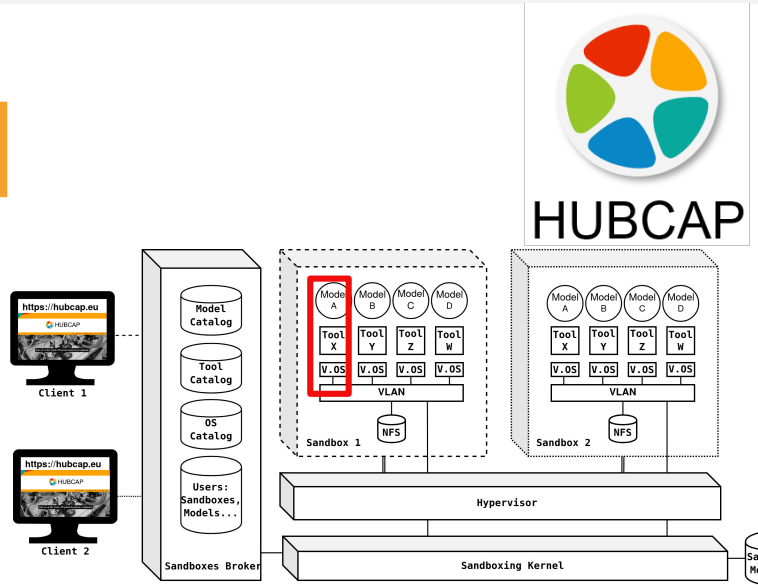
- **Sandbox**



- **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

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

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

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

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Sandbox Access Control

Feature	Provider Owner	Provider Guest	Consumer Owner	Consumer Guest
Access to remote viewer	X	X	X	X
Upload archive	X	X	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 tool	X			

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

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

HUBCAP

Client VDM-SL Model

```
-- InviteGuest
InviteGuest: ClientId * ClientId * SandboxId ==> ()
InviteGuest(cId, cguestId, sId)==
  BrokerAddGuest(cId, cguestId, sId)
pre cId in set validClients;

-- Select OS
SelectOS : ClientId * OSId ==> ()
SelectOS(cId, osId)==
  clientst.selectedOS := SelectOperatingSystem(osId, cId)
pre cId in set validClients;

-- Select Tool
SelectToolFromRepository: ClientId * ToolId ==> ()
SelectToolFromRepository(cId, tId) ==
  clientst.selectedTool := SelectTool(tId, cId)
pre cId in set validClients;
```

State

```
types

ClientSt::
selectedTool :
SelectedTool
selectedOS : SelectedOS
selectedModel :
SelectedModel;

ClientId = nat;
KnownSandboxes = set of
SandboxId;
Errors = set of token;
SelectedTool = [nat];
SelectedOS = [nat];
SelectedModel = [nat]
```

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

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
BrokerInitiateSandboxAccess: ClientId * SandboxId ==> bool
BrokerInitiateSandboxAccess(cId, sId) ==
    let sandboxes = GetSystemSandboxes(),
        servers = dunion {s.sandboxServers | s in set rng sandboxes &
s.sandboxId = sId}
    in
      (for all x in set servers do
         UpdateConnections(cId, x, sId, true);
       return true) -- one needs to enable the UpdateConnections
operation to report if it was ok
pre not ClientIsNull(cId, brokerst.providers, brokerst.consumers,
brokerst.owners, brokerst.guests)
and
((cId in set dom brokerst.owners and sId in set brokerst.owners(cId))
    or(cId in set dom brokerst.guests and sId in set
brokerst.guests(cId)));
```

State

```
types

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;
...
```

Gateway + Server + Sandbox VDM-SL Model

Gateway

```
types
GateWaySt ::
connectedClients : ConnectedClients
connectedServers : ConnectedServers;

ConnectedClients = set of ClientId;
ConnectedServers = set of ServerId;
```

Server

```
types
ServerId = nat;
```

Sandbox

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

Sandbox::
 sandboxId : SandboxId
 sandboxServers : SandboxServers
```

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

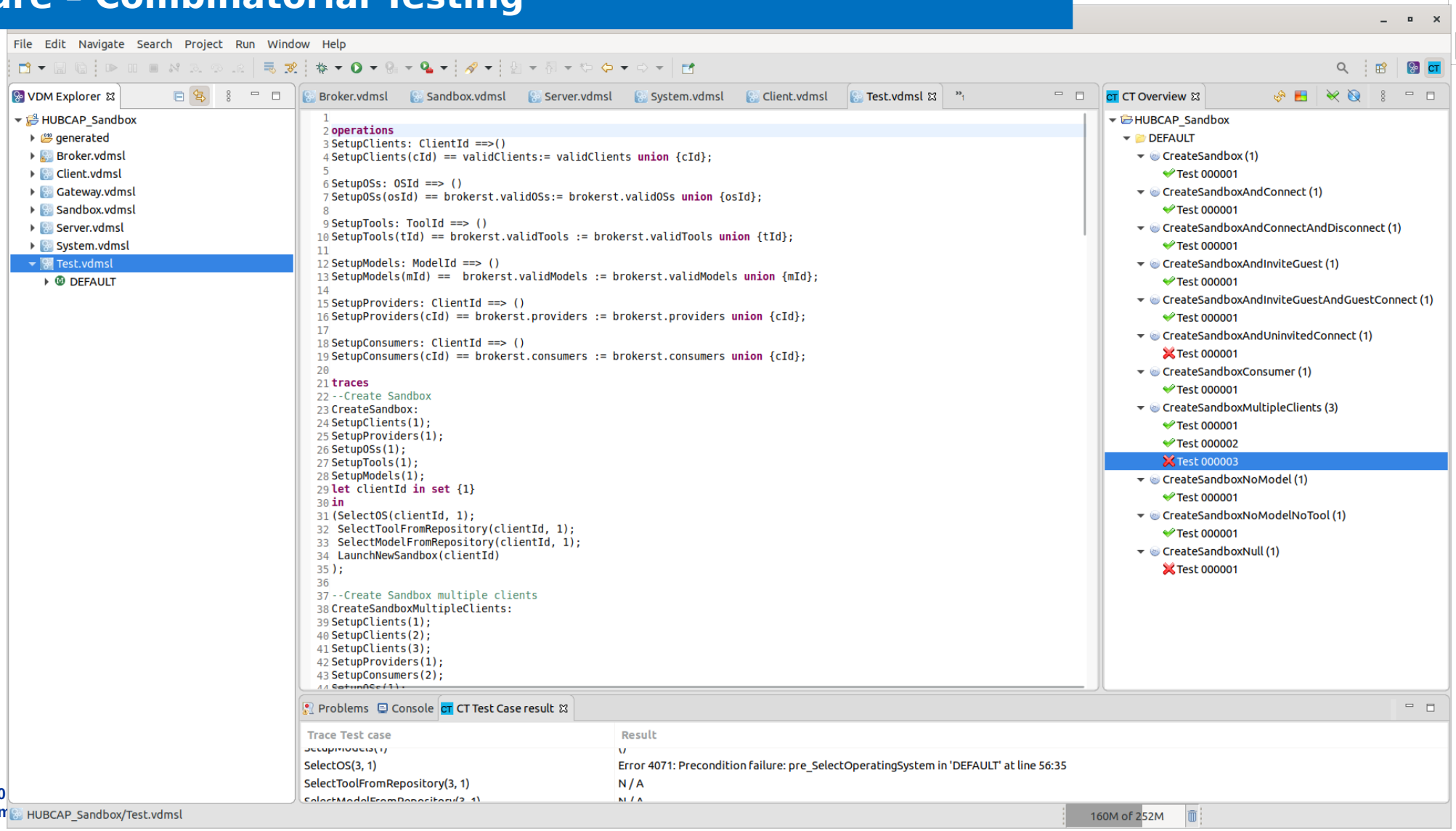
Initial State

```
state SystemSt of
gatewayConnections : GatewayConnections
gatewayConnectionsSandbox : GatewayConnectionsSandbox
systemSandboxes : SystemSandboxes
  toolOwners : ToolOwners
  modelOwners : ModelOwners
brokerst : BrokerSt
clientst: ClientSt
validClients : ValidClients
inv ss == dom ss.gatewayConnections = dom
ss.gatewayConnectionsSandbox
init s == s = mk_SystemSt({|->},{|->},{|->},{|->},{|-
>},
                          mk_BrokerSt(
{},{},{},{},{},{},{|->},{|->},[],{|->},{|->},{|->}),
                          mk_ClientSt(nil,nil,nil),{})
end
```

```
pure GetSystemSandboxes: () ==> map SandboxId to Sandbox
GetSystemSandboxes() ==
  return systemSandboxes;

pure GetGatewaySandboxConnections: () ==>
GatewayConnections
GetGatewaySandboxConnections() ==
  return gatewayConnectionsSandbox;

pure Max: SystemSandboxes ==> nat
Max(ss) ==
if ss = {|->} then
return 0
else
  let max in set dom ss be st forall d in set dom ss & d
<= max
  in
    return max;

GenerateNewSandboxId: () ==> nat
GenerateNewSandboxId() ==
  return Max(systemSandboxes) + 1;
```

Horizon 2020 Programme
Grant Agreement #872698

Overture – Combinatorial Testing



Horizon 2020
Grant Agreem

Combinatorial Testing traces

- **Validating the Sandbox Creation**

```
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};

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

CreateSandboxMultipleClients (3)
- Test 000001
- Test 000002
- Test 000003

```
CreateSandboxMultipleClients:
SetupClients(1);
SetupClients(2);
SetupClients(3);
SetupProviders(1);
SetupConsumers(2);
SetupOSs(1);
SetupTools(1);
SetupModels(1);
let clientId in set {1, 2, 3}
in
(SelectOS(clientId, 1);
 SelectToolFromRepository(clientId, 1);
 SelectModelFromRepository(clientId, 1);
 LaunchNewSandbox(clientId)
);
```

Null user

Horizon 2020 Programme
Grant Agreement #872698

HUBCAP

Results

- Suggestions to the implementation team
- Explicit roles for Sandbox creation
- Covered the current access functionality

- 11 traces expanding to 13 tests
- 2 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 model in order to iterate on the table of permissions
- Utilize combinatorial testing to cover more scenarios

Horizon 2020 Programme
Grant Agreement #872698

Thank you

Thank you for your attention!

Horizon 2020 Programme
Grant Agreement #872698