Revit External Services

Make built-in features behaving your way

Arnošt Löbel

Sr. Principal Software Engineer, Revit R&D

Who’s Arnošt Löbel?

- Structural engineering background
- Revit developer 9+ years
- Core Revit frameworks
  - Transactions
  - Events, External Event
  - External Services
  - API safety
- Revit API Forums, API Help

Arnost.Lobel@Autodesk.com

Key learning objectives

At the end of this class, you will be able to:

1. Tell what Revit services are and how could they apply to you
2. Know **the** (hi)**story** behind external services
3. Familiarize with the **terminology** and major **classes**
4. Understand **classifications** of external services
5. Programmatically **Find** and **Execute** services
6. **Implement**, **Register**, and **Activate** external servers
7. Quite possibly know more than the average Revit developer

Class schedule

The main sections of the class cover:

1. The five-mile view - philosophy and design
2. Closer look at the implementation and workflows
3. Concrete code samples
4. Questions

The five-mile view

A Brief history of Revit customization

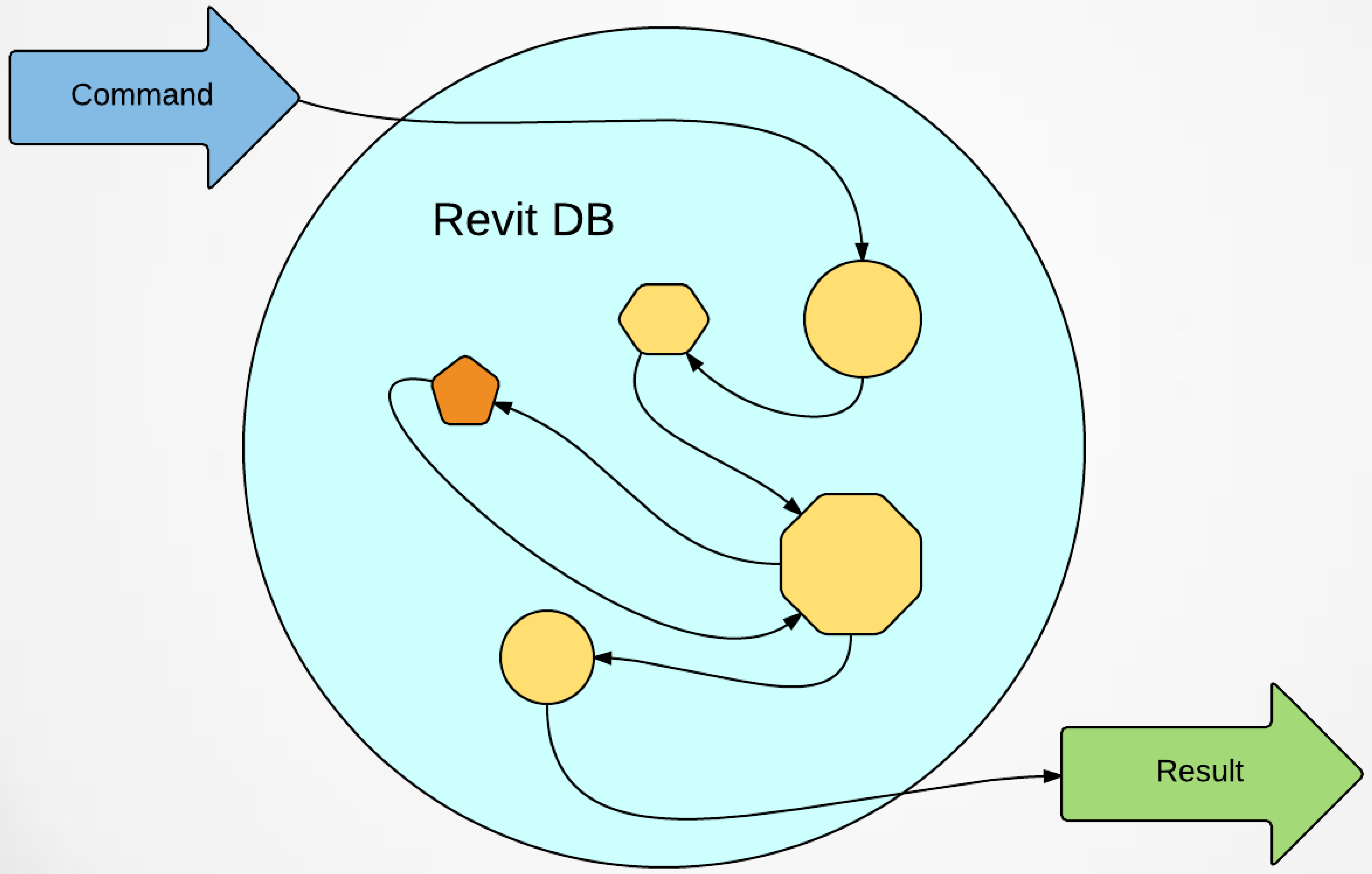
- **Phase 1** – External commands, Macros
  - Commands typically unrelated to existing Revit commands
- **Phase 2** – Events and Command overrides
  - Replacing or extending Revit's standard commands
- **Phase 2 ¾** – Dynamic Updaters
  - Reacting to model changes and regeneration
- **Phase 3** – External Services
  - Redefining what it means when Revit does something

What are External Services?

- Feature **Componentizing**
  A way to de-couple a particular behavior or feature in Revit, and package it nicely for another application to implement.
- Behavior **Customization**
  A level of customization (think “standardization”, “localization”) that goes beyond anything we have had in Revit so far.
- We call it: **The External Services Framework**

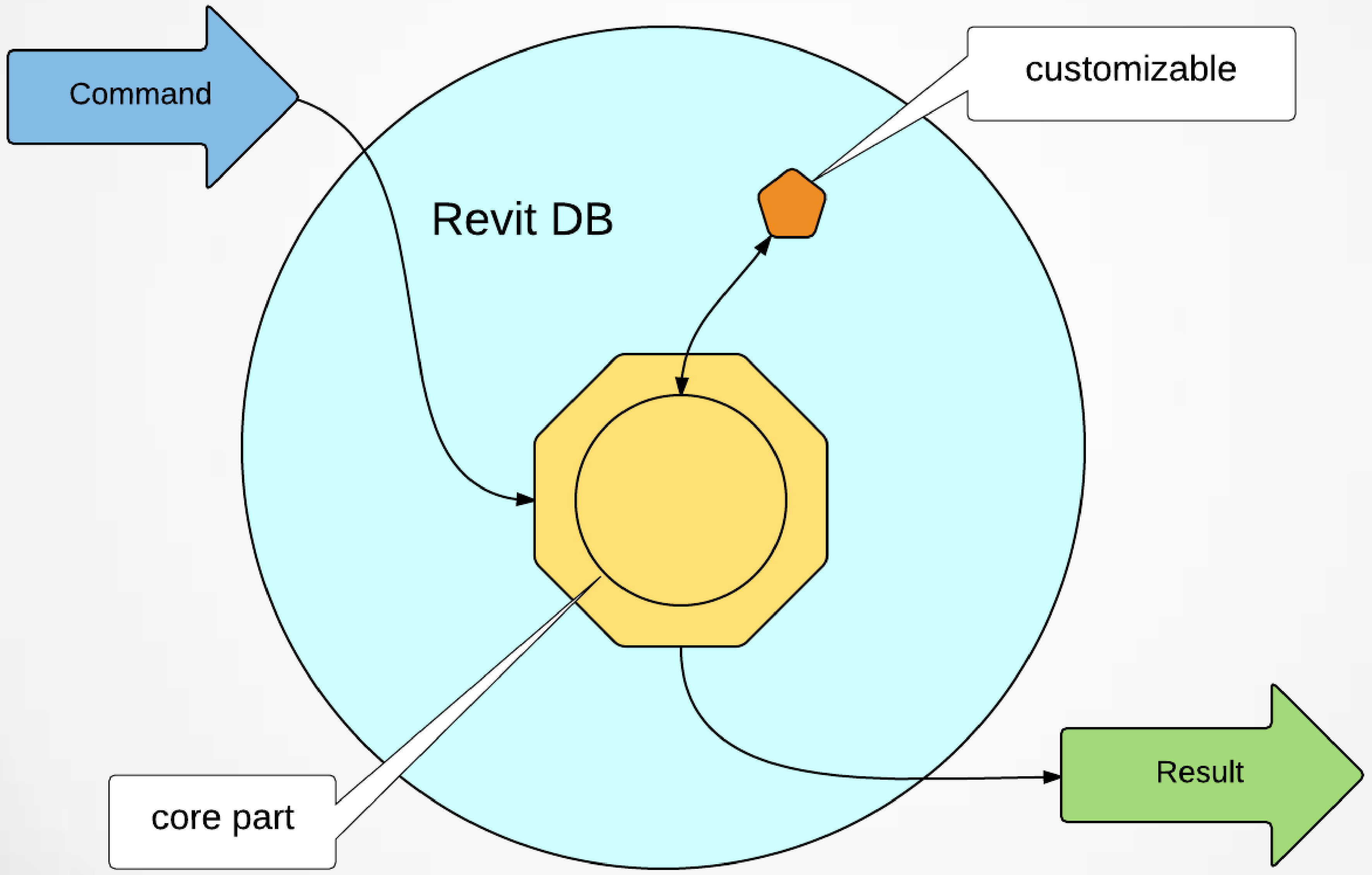
Making an external service – Phase 1

collecting the pieces



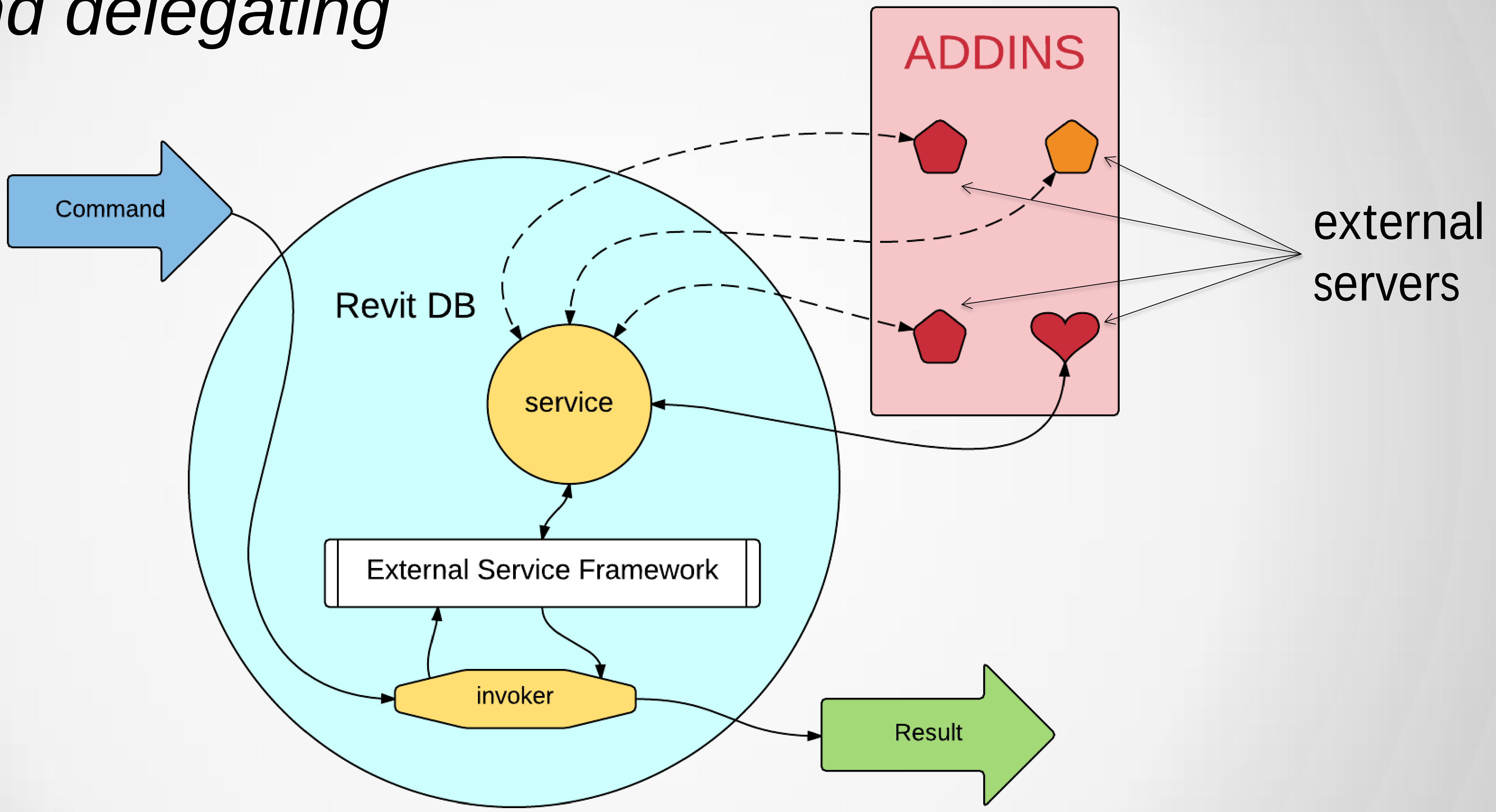
Making an external service – Phase 2

cleaning up and componentizing



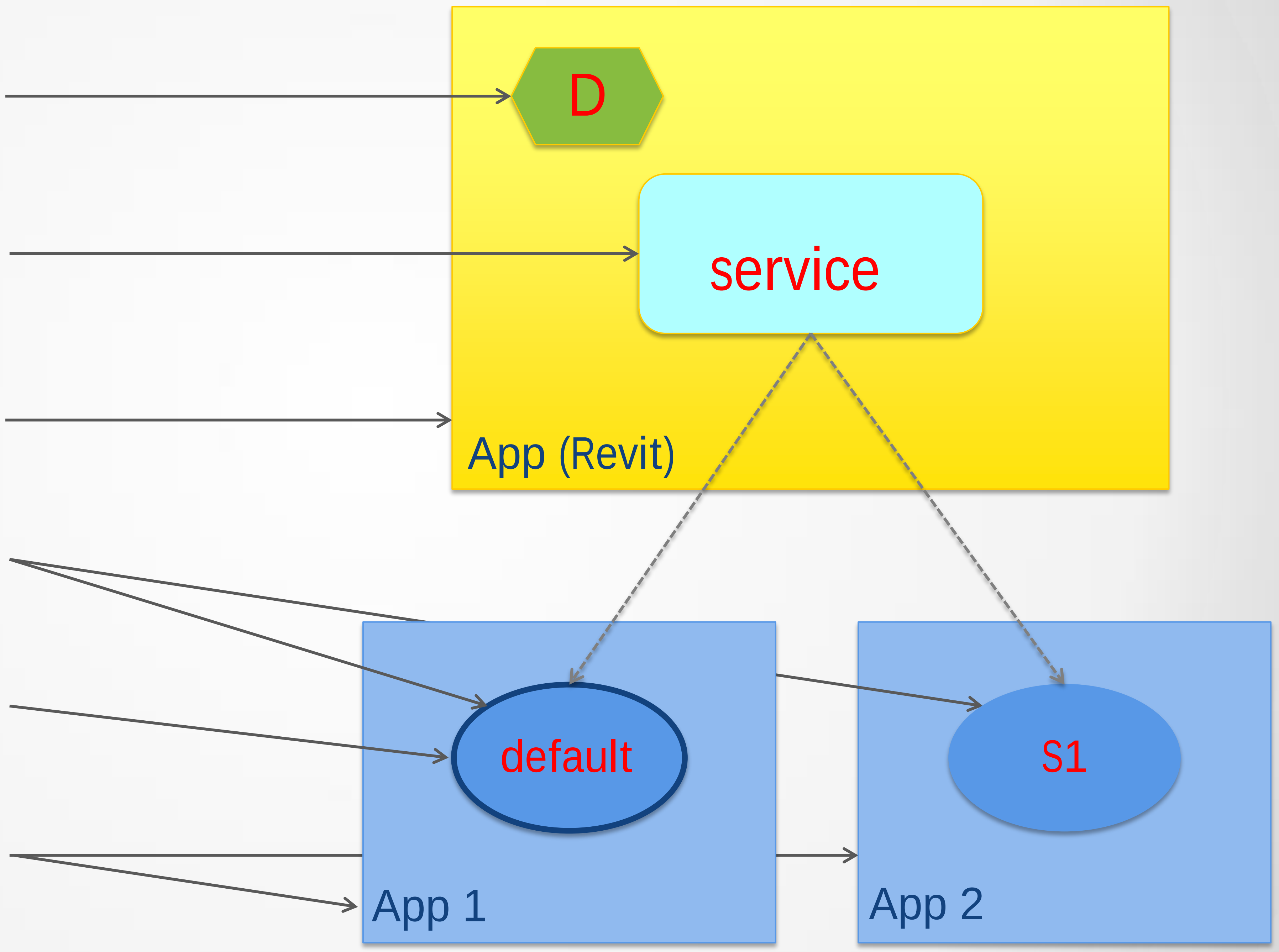
Making an external service – Phase 3

unifying and delegating



Terminology – External Services Framework

- Service Data
- Service
- Service Owner
- Server
- Default Server
- Server Provider



Classification of External Services

1. **By Server Policy**
2. By Availability
3. By Accessibility
4. By Serialization

Services by server policies

- Multiple servers can be registered for <u>any</u> service
- **Single-server** service:
  - Only <u>one server</u> may be active
  - Only one server gets executed
- **Multi-server** service:
  - <u>More than one</u> servers may be active at the same time
  - Any or all of them can anticipate in the services' execution

Classification of External Services

1. By Server Policy
2. **By Availability**
3. By Accessibility
4. By Serialization

Services by availability

- **Mandatory** service:
  - Must be present in order for Revit to run
  - Must have at least one server active at all time
  - Must have a "*default*" server assigned
- **Optional** service:
  - Not required to be available
  - May have no active server or even no registered server

Classification of External Services

1. By Server Policy
2. By Availability
3. **By Accessibility**
4. By Serialization

Services by accessibility

- **Private** service:
  - May be executed by its owner only (in most cases - Revit)
- **Public** service:
  - May be executed by anyone
- In both cases
  - All services are query-able via the API
  - Executors must know how and when to execute it

Classification of External Services

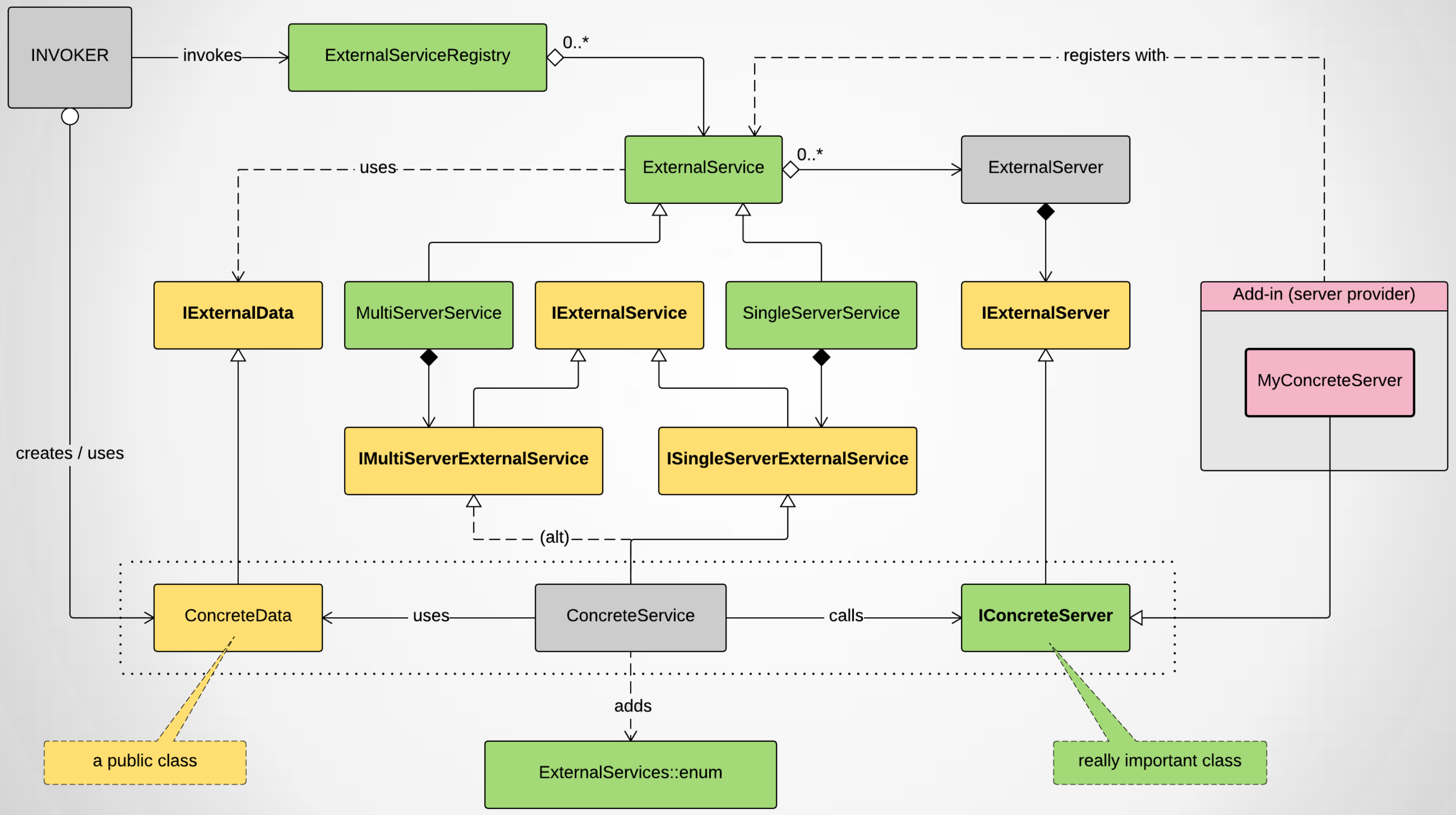
1. By Server Policy
2. By Availability
3. By Accessibility
4. **By Serialization**

Services by serialization

- **Recordable** service:
  - Every execution gets recorded in the document
  - Executions require a modifiable document
  - Presence of executed servers checked upon file opening
  - Specific policies for work-shared models
- **Non-recordable** service:
  - Nothing stored in the model
  - No requirement on future availability of servers
  - Currently – that's <u>all</u> services in Revit

Looking closer

Simplified class diagram (using: Autodesk.Revit.DB.ExternalService)



Registering Services

- **Time of registrations**
  - `OnStartup` (before the ApplicationInitialized event)
- **Registering Options**
  - `IsPublic` – access to Execute
  - `IsRecordable` – serialization
  - `IsSelfsynchronizing` – worksharing policy
- **Execution Policy** (multi-server services only)
  - `FirstApplicableServer`
  - `AllApplicableServers`
- **Execution Key** – required to execute

Implementing a server – part 1

```
public partial class MyServer : IConcreteServerInterface
{
    public override System.String GetName()
    {
        return "Short name";
    }
    public override System.String GetVendorId()
    {
        return "MYID";
    }
    public override System.String GetDescription()
    {
        return "Brief Description";
    }
}
```

Implementing a server – part 2

```
public partial class MyServer : IConcreteServerInterface
{
    public override Guid GetServerId()
    {
        return new Guid("3B8D358F-27B1-43F5-922C-21F59168AEA8");
    }
    public override ExternalServiceId GetServiceId()
    {
        return ExternalServices.BuiltInExternalServices.SomeService;
    }
    public override int Compute(int x, int y)
    {
        return x + y;     // Service-specific!
    }
}
```

Registering servers

1. Find a service
   a) `BuiltInExternalServices` enumeration
   b) Iterating through all services in `ExternalServiceRegistry`
2. Acquire access to the service
   - `GetService` on the `ExternalServiceRegistry` class
3. Register a server with the service
   - `AddServer` on `ExternalService` class

Activating and Deactivating servers

- Service must have an active server in order to be executed
- Only a registered server can be activated
- Activation/Deactivation depends on service
  - Single-Server: `SetActiveServer, UnsetActiveServer`
  - Multi-Server: `SetActiveServers`
- Activation applicability
  - Application-wide (all and any document)
  - Document-wide (one document only)
- Deactivated servers remain registered!

Executing Services

- Methods `ExternalServiceRegistry.ExecuteService`
  - Only “Public” services can be executed by anyone
- Single-Server service
  - The one currently active (if any) server gets executed
- Multi-Server service
  - Executing of servers depends on execution policy
  - Service gets asked per each active server - `CanExecute`
  - `FirstApplicableServer` : the first one that can be executed
  - `AllApplicableServers` : all that can be executed

ExternalService class members

About servers

1. AddServer
2. GetServer
3. RemoveServer
4. GetDefaultServerId
5. GetRegisteredServerIds
6. IsRegisteredServerId
7. NumberOfServers

About provider

8. Name
9. Description
10. ServiceId
11. ProviderId

Options

12. GetOptions
13. IsSerializable
14. GetPublicAccessKey

(3) If **GetDefaultServerId** returns a valid Guid it means the service is mandatory (i.e. it must have an active server at all times.

(13) The method **IsSerializable** is a shortcut for GetOptions.IsRecordable

(14) The method **GetPublicAccessKey** can only be called for a public service (i.e. GetOptions.IsRecordable returns True)

ExternalServiceRegistry class members

Registering

1. RegisterService (ISingleServerService service, ExternalSeriviceOptions options)
2. RegisterService (ISingleServerService service, Guid defaultServer, ExternalSeriviceOptions options)
3. RegisterService (IMultiServerService service, ExternalSeriviceOptions options, ExecutionPolicy policy)

Accesssing

4. GetService (ExternalServiceId id)
5. GetServices ()

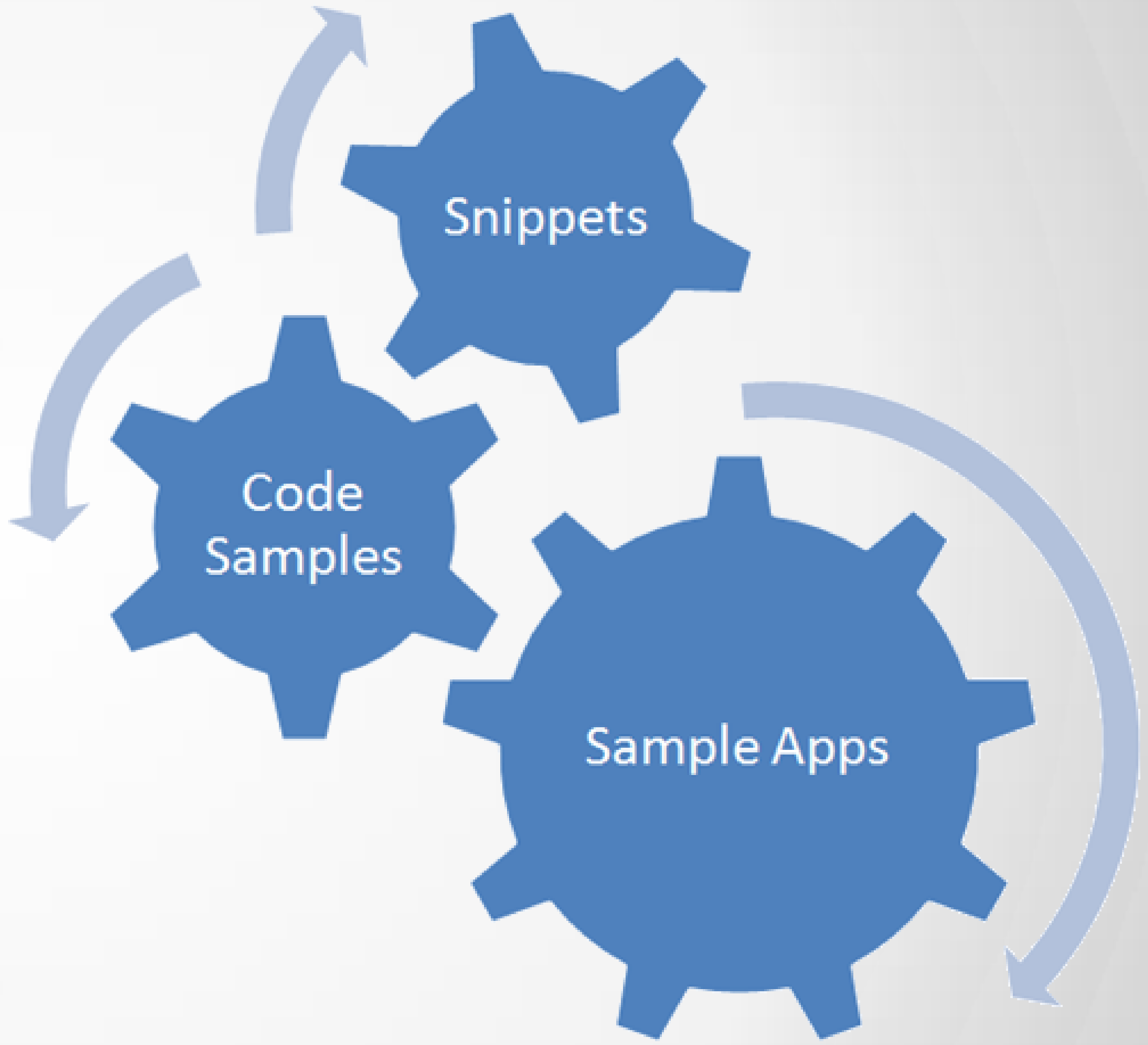
Executing

6. ExecuteService (Guid key, IExternalData data)
7. ExecuteService (Guid key, Document doc, IExternalData data)
8. ExecuteService (Guid key, Guid serverId, IExternalData data)

(2) The registering method that takes a default server declares a service as mandatory, which means there must be an active server set at all times. If none is set explicitly then the default server automatically becomes the active one.

(8) The execution method that operates with an explicit server can only be used with a non-recordable service. Also, this execution does does not alter the currently active that may currently be assigned for the service.

Examples



Existing applications

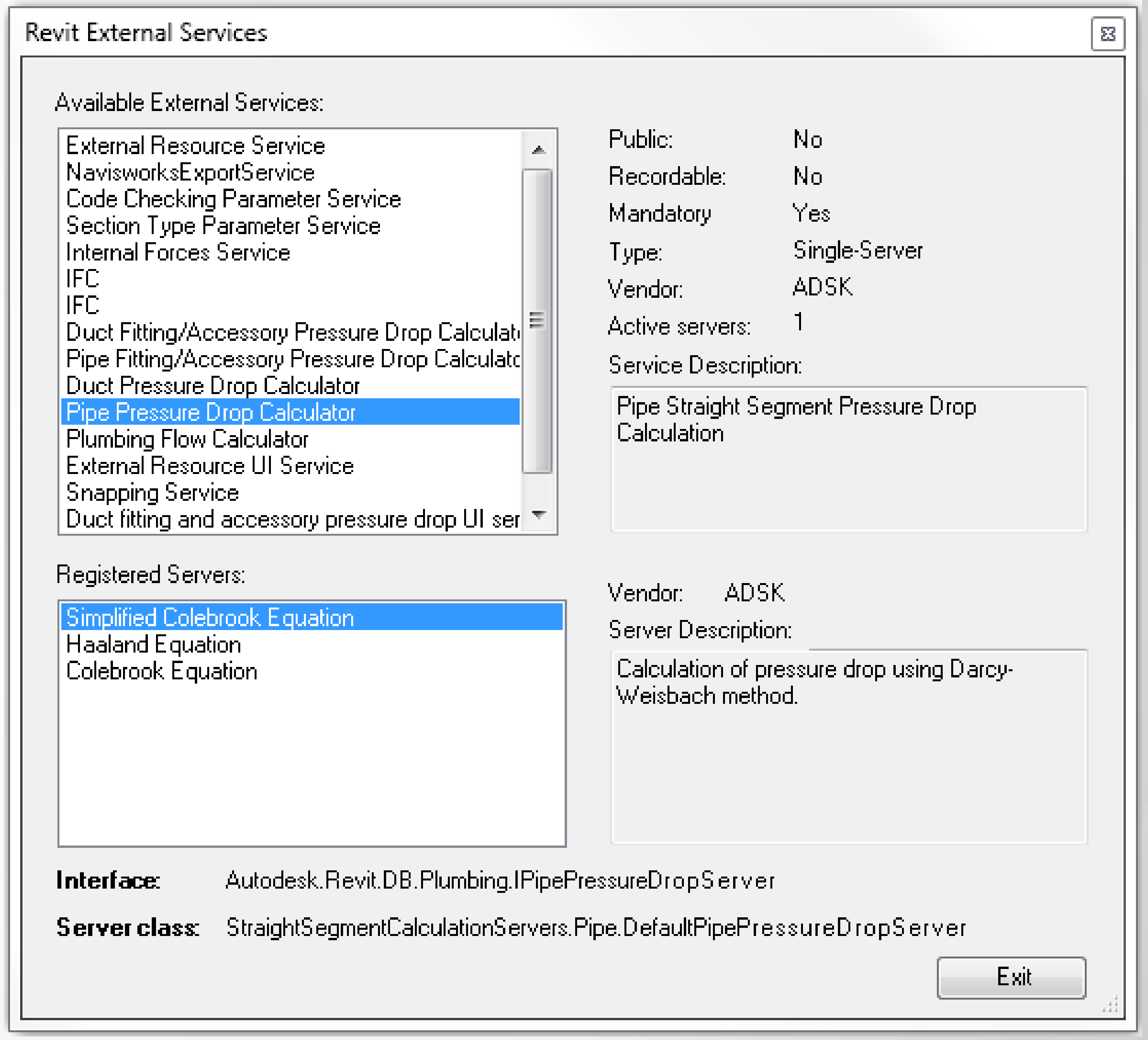
There already are many services available in Revit:

- Several MEP calculations
- External Resources
- IFC Export / Import
- Navisworks Export
- Code checking
- *many more in making*...

Service Explorer sample

- Lists all available services
- Shows services' properties
- Lists servers of each service
- Shows servers' description
- Servers' interfaces
- Servers' actual classes

The sample's code and an [*.addin] file is available in the class' material.


Revit External Services

Available External Services:

External Resource Service
NavisworksExportService
Code Checking Parameter Service
Section Type Parameter Service
Internal Forces Service
IFC
IFC
Duct Fitting/Accessory Pressure Drop Calculat
Pipe Fitting/Accessory Pressure Drop Calculat
Duct Pressure Drop Calculator
Pipe Pressure Drop Calculator
Plumbing Flow Calculator
External Resource UI Service
Snapping Service
Duct fitting and accessory pressure drop UI ser

Public:	No
Recordable:	No
Mandatory	Yes
Type:	Single-Server
Vendor:	ADSK
Active servers:	1

Service Description:
Pipe Straight Segment Pressure Drop Calculation

Registered Servers:

Simplified Colebrook Equation
Haaland Equation
Colebrook Equation

Vendor: ADSK

Server Description:
Calculation of pressure drop using Darcy-Weisbach method.

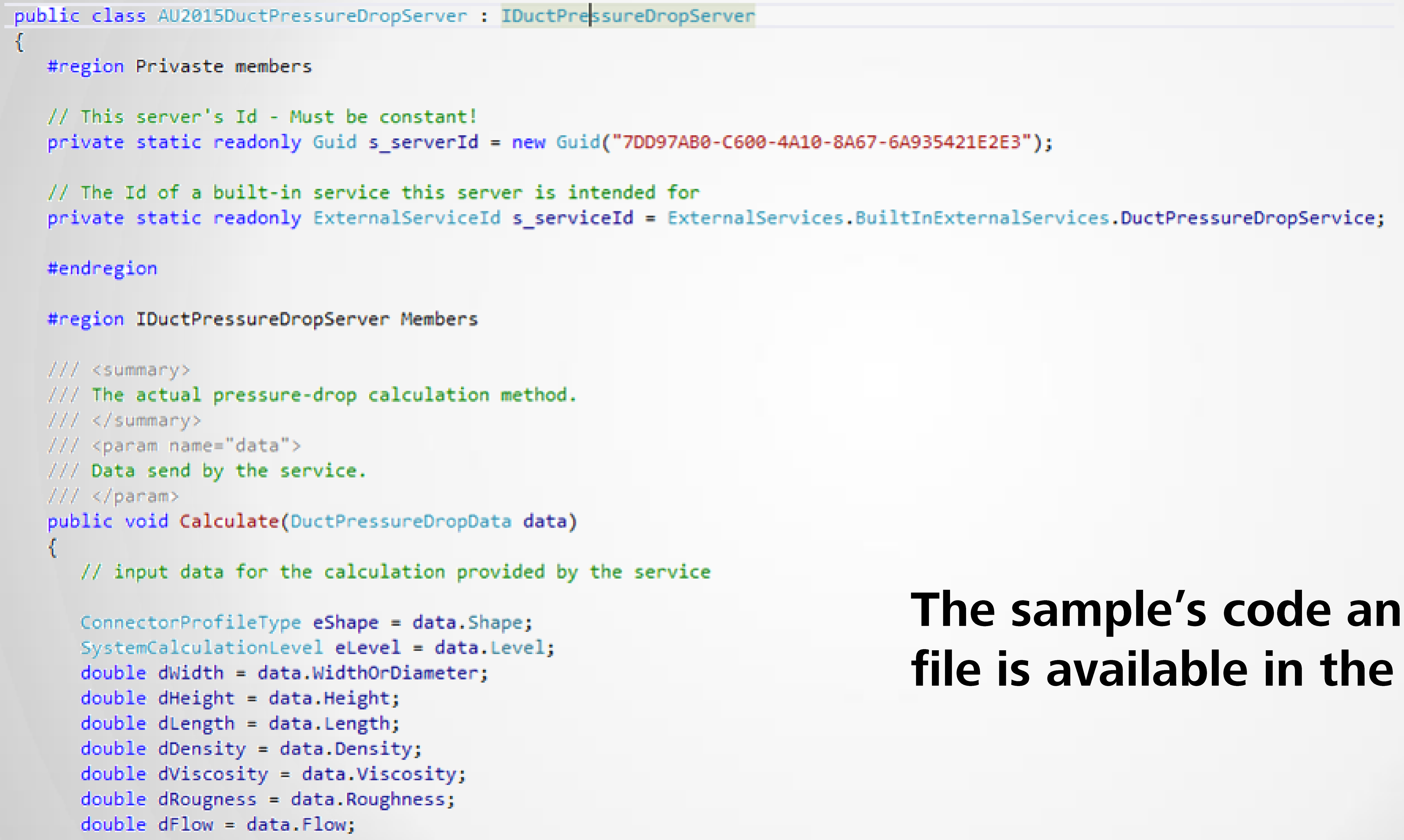
Interface: Autodesk.Revit.DB.Plumbing.IPipePressureDropServer

Server class: StraightSegmentCalculationServers.Pipe.DefaultPipePressureDropServer

Ids and Interfaces of common built-in service

Built-in service Id	Server Interface
PipePlumbingFixtureFlowService	**IPipePlumbingFixtureFlowServer**
PipePressureDropService	**IPipePressureDropServer**
DuctPressureDropService	**IDuctPressureDropServer**
DuctFittingAndAccessoryPressureDropService	**IDuctFittingAndAccessoryPressureDropServer**
DuctFittingAndAccessoryPressureDropUIService	**IDuctFittingAndAccessoryPressureDropUIServer**
PipeFittingAndAccessoryPressureDropUIService	**IPipeFittingAndAccessoryPressureDropUIServer**
ExternalResourceService	**IExternalResourceServer**
ExternalResourceUIService	**IExternalResourceUIServer**

Sample of a duct-pressure drop Server



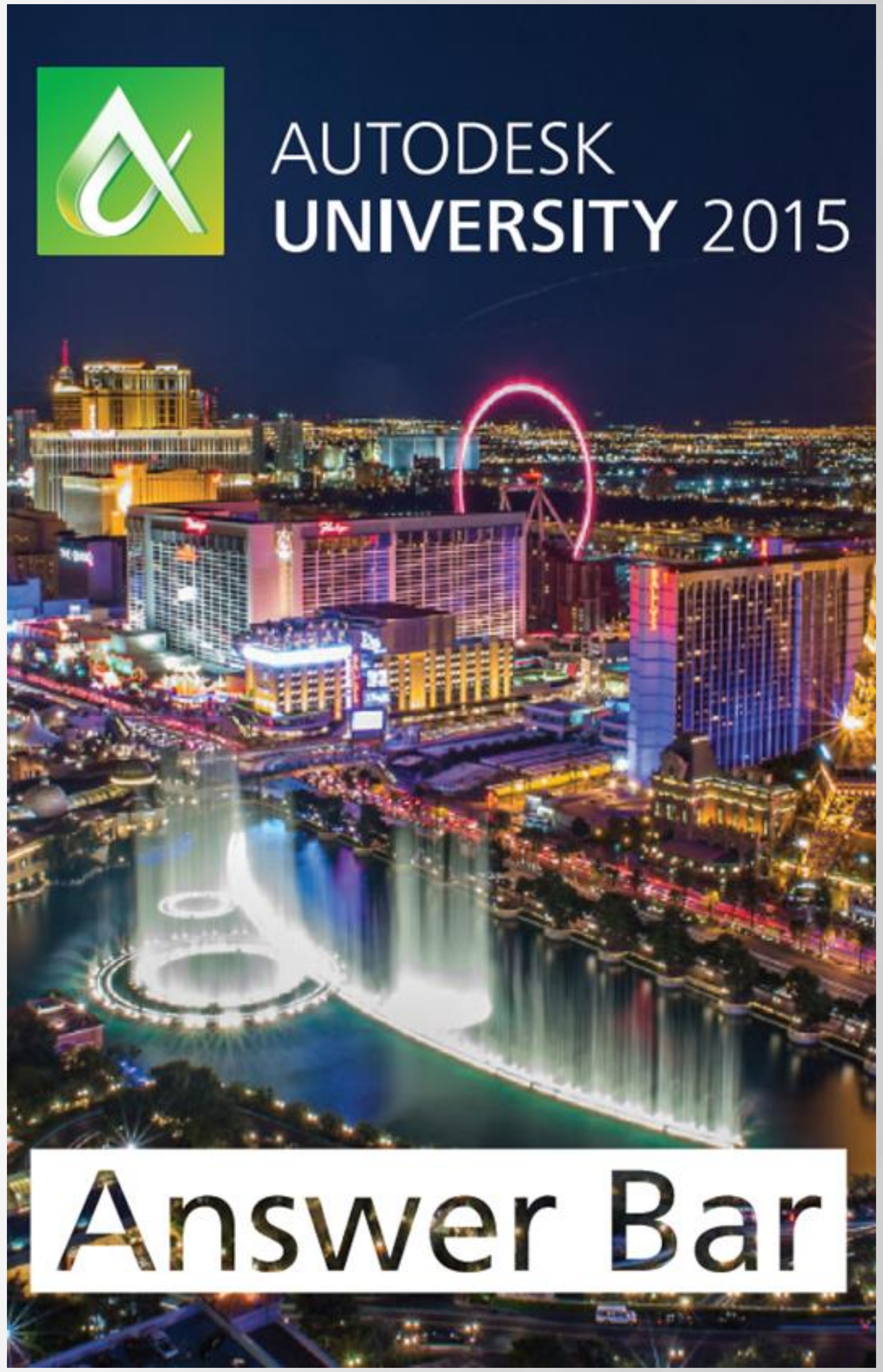
```
public class AU2015DuctPressureDropServer : IDuctPressureDropServer
{
    #region Privaste members

    // This server's Id - Must be constant!
    private static readonly Guid s_serverId = new Guid("7DD97AB0-C600-4A10-8A67-6A935421E2E3");

    // The Id of a built-in service this server is intended for
    private static readonly ExternalServiceId s_serviceId = ExternalServices.BuiltInExternalServices.DuctPressureDropService;

    #endregion

    #region IDuctPressureDropServer Members

    /// <summary>
    /// The actual pressure-drop calculation method.
    /// </summary>
    /// <param name="data">
    /// Data send by the service.
    /// </param>
    public void Calculate(DuctPressureDropData data)
    {
        // input data for the calculation provided by the service

        ConnectorProfileType eShape = data.Shape;
        SystemCalculationLevel eLevel = data.Level;
        double dWidth = data.WidthOrDiameter;
        double dHeight = data.Height;
        double dLength = data.Length;
        double dDensity = data.Density;
        double dViscosity = data.Viscosity;
        double dRougness = data.Roughness;
        double dFlow = data.Flow;
```

The sample's code and an [*.addin] file is available in the class' material.

Questions?

AU Answer Bar

- **Seek answers** to all of your technical product questions by visiting the Answer Bar.
- **Open daily** 8am-6pm Tues-Wed and 8am-4pm Thurs
- Located just outside of **Hall C on level 2**.
- **Staffed by Autodesk** developers, QA, & support engineers ready to help you through your most challenging technical questions.



AUTODESK®