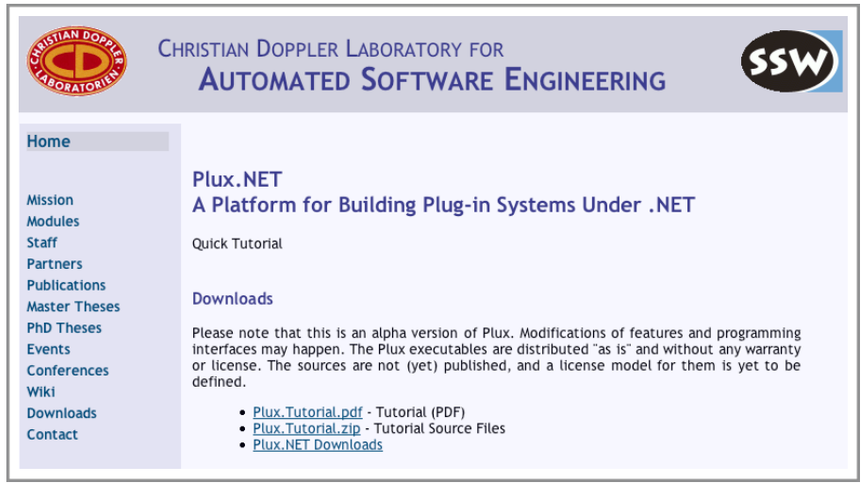
Plux.NET

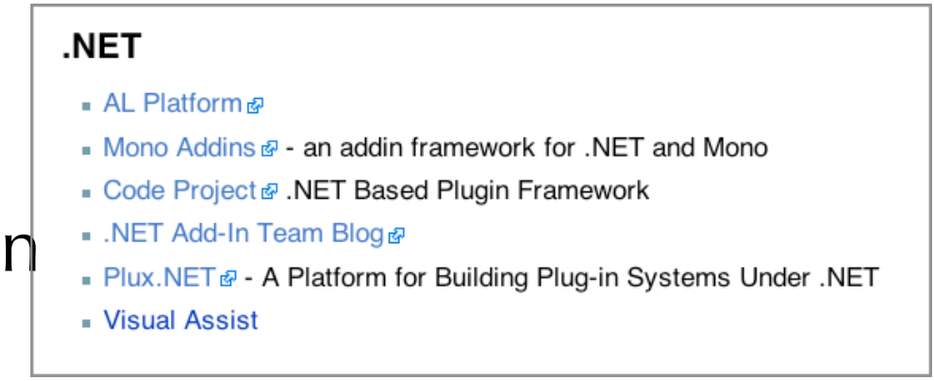
Reinhard Wolfinger
Christian Mittermair, Markus Jahn
Christian Doppler Laboratory
für Automated Software Engineering
Johannes Kepler University, Linz, Austria

Plux.NET v0.2 - Überblick

Neue Programmierschnittstelle (API)

- von Grund auf überarbeitet
- Aufbau und Abläufe dokumentiert
- Tutorial online (verlinkt auf Wikipedia)
- Funktionsumfang erweitert
  - einfache Szenarien weiterhin einfach
  - bei Bedarf feinere Eingriffsmöglichkeiten (für Konfigurationswerkzeuge)





Neue Laufzeitumgebung

- leichtgewichtiger
- flexibler
- (viel) schneller
- Fehlerursachen erkennbarer
- besser konfigurierbar

Slot & Extension

Slot & Extension


```
[Extension("MyApp")]


class MyApp {

}
```

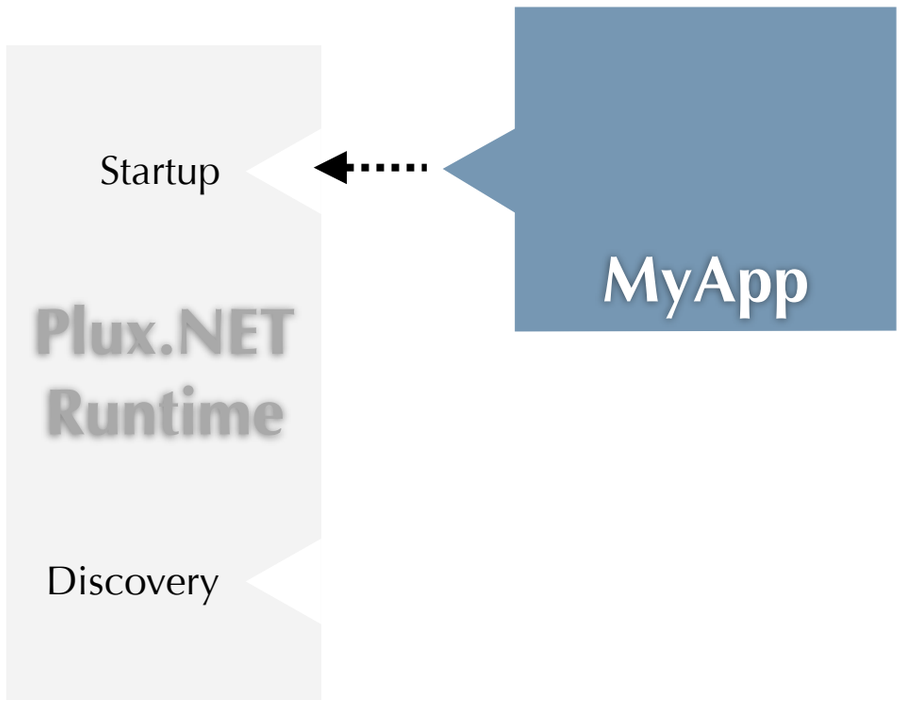
Slot & Extension


```
[Extension("MyApp")]


class MyApp {

}
```

Slot & Extension

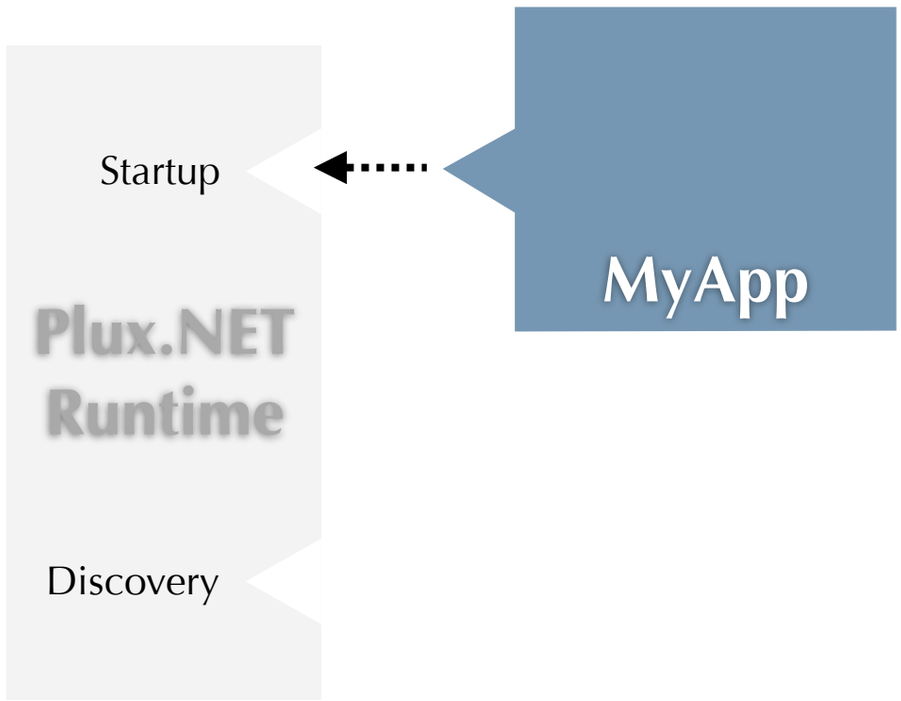

```
[Extension("MyApp")]
[Plug("Startup")]

class MyApp : IStartup {
  void Run() { ... }
}
```

Slot & Extension




```
[Extension("MyApp")]
[Plug("Startup")]

class MyApp : IStartup {
  void Run() { ... }
}
```

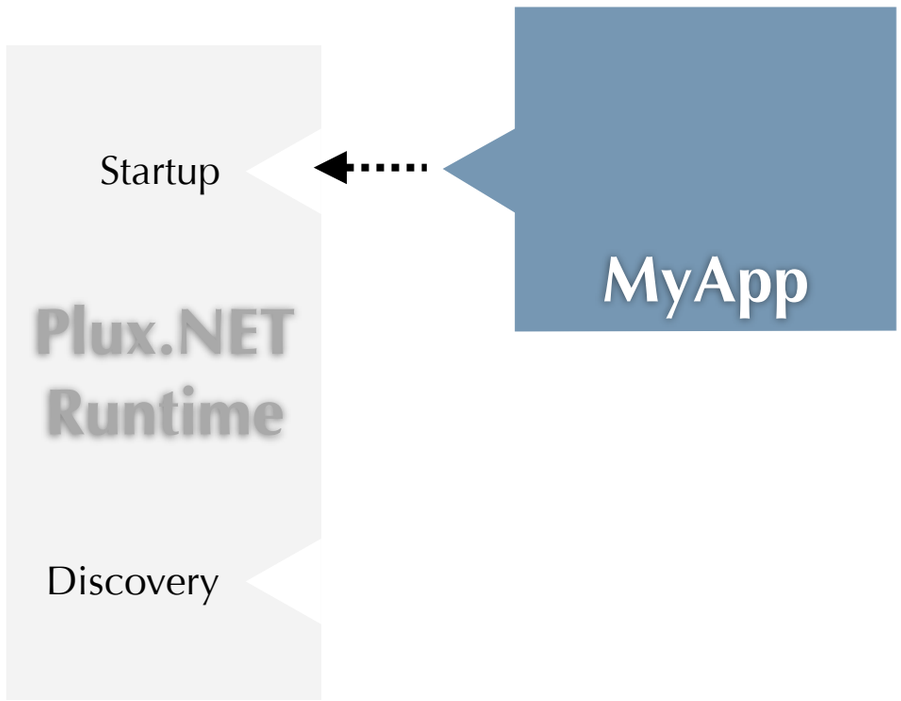
Slot & Extension




```
[Extension("MyApp")]
[Plug("Startup")]

class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]

interface ILogger {
  void Print(string msg);
}
```

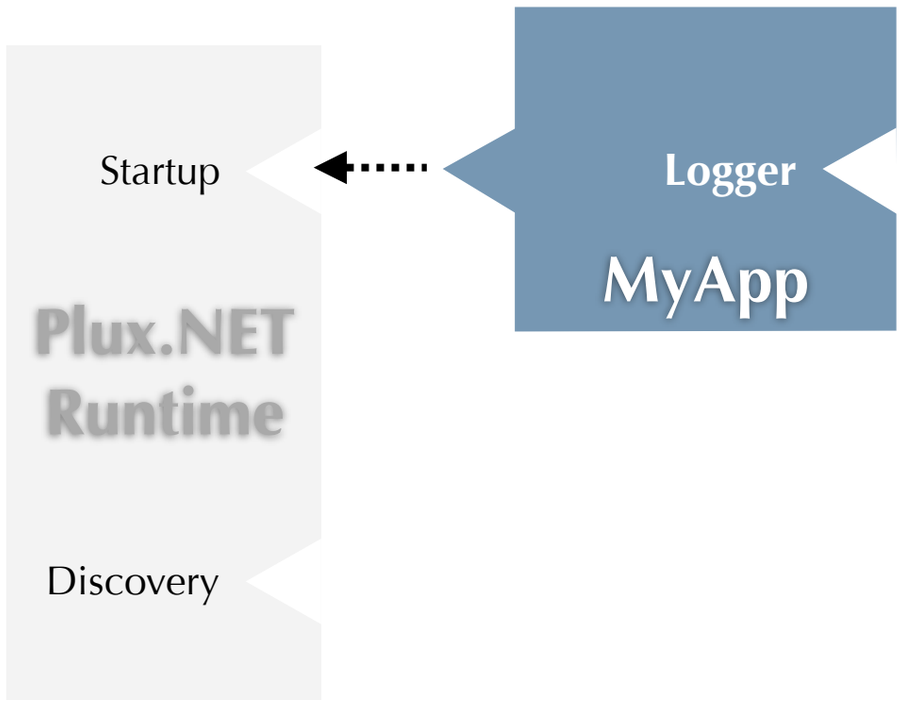
Slot & Extension




```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]

interface ILogger {
  void Print(string msg);
}
```

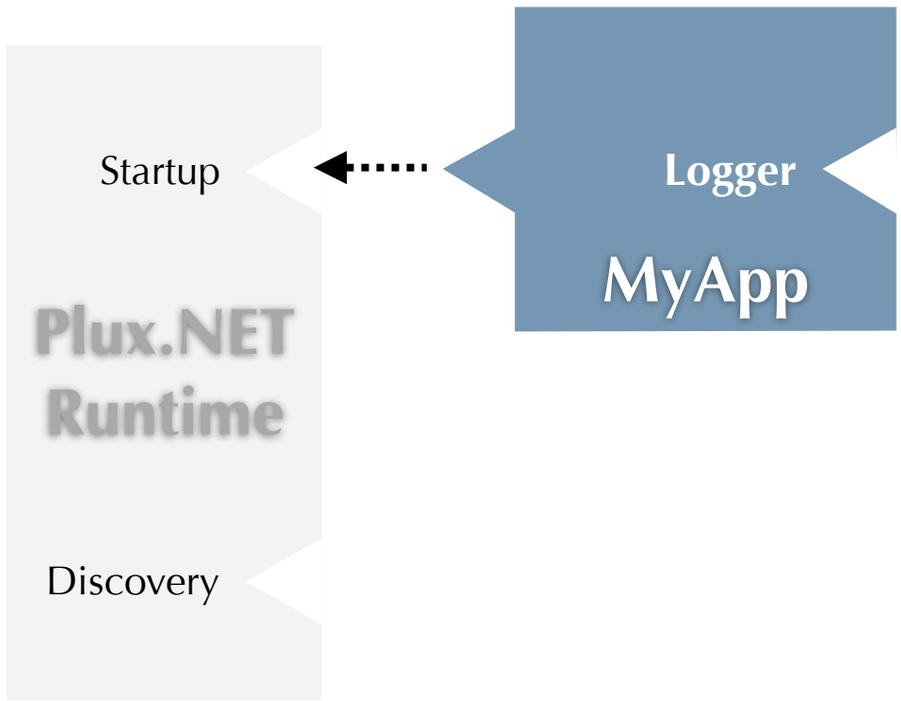
Slot & Extension




```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]

interface ILogger {
  void Print(string msg);
}
```

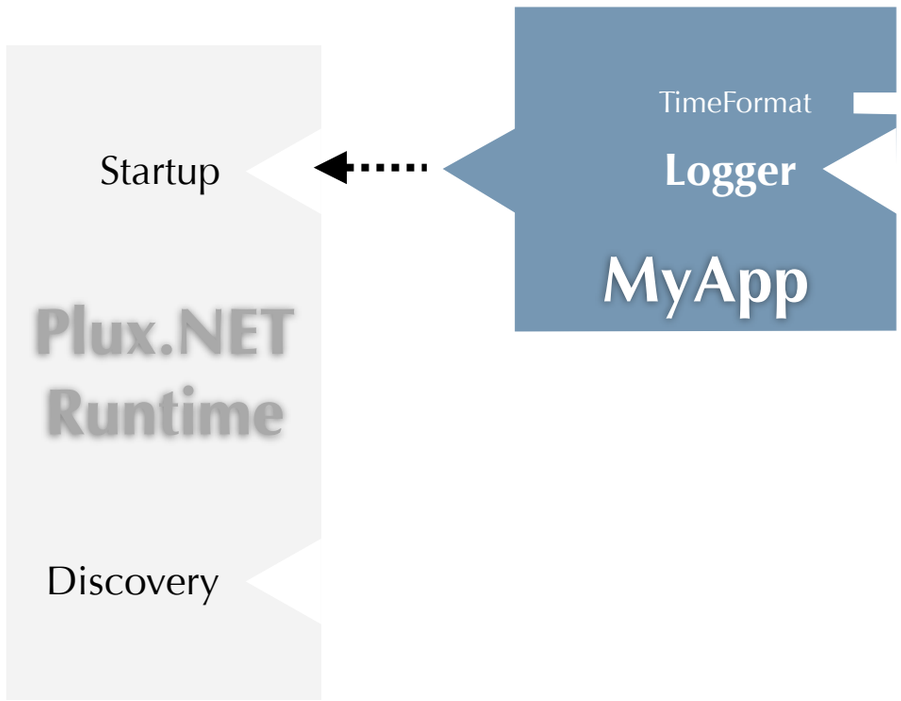
Slot & Extension




```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

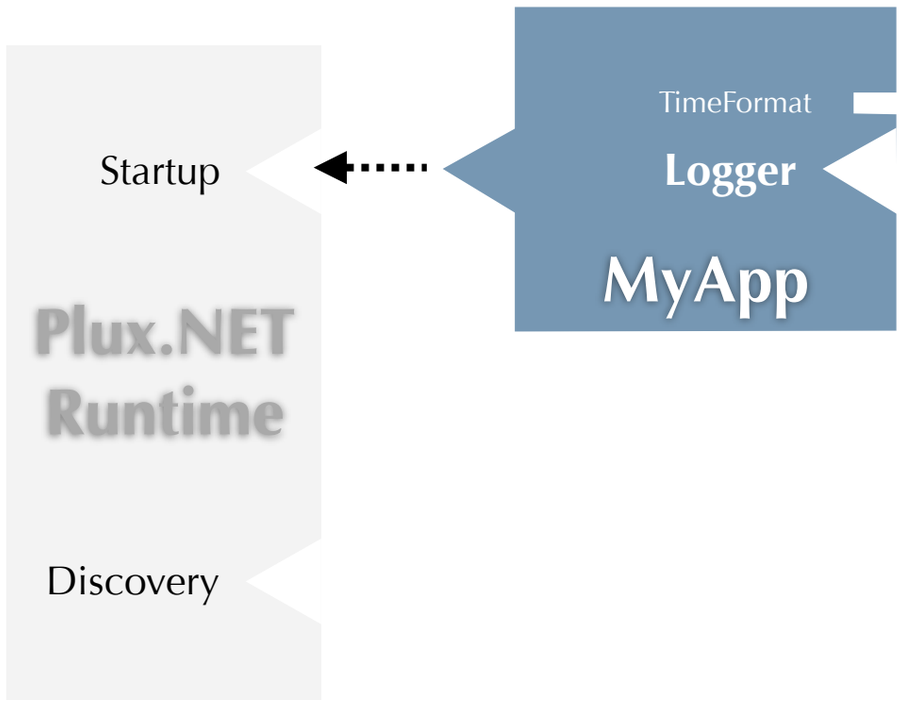
Slot & Extension




```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

Slot & Extension

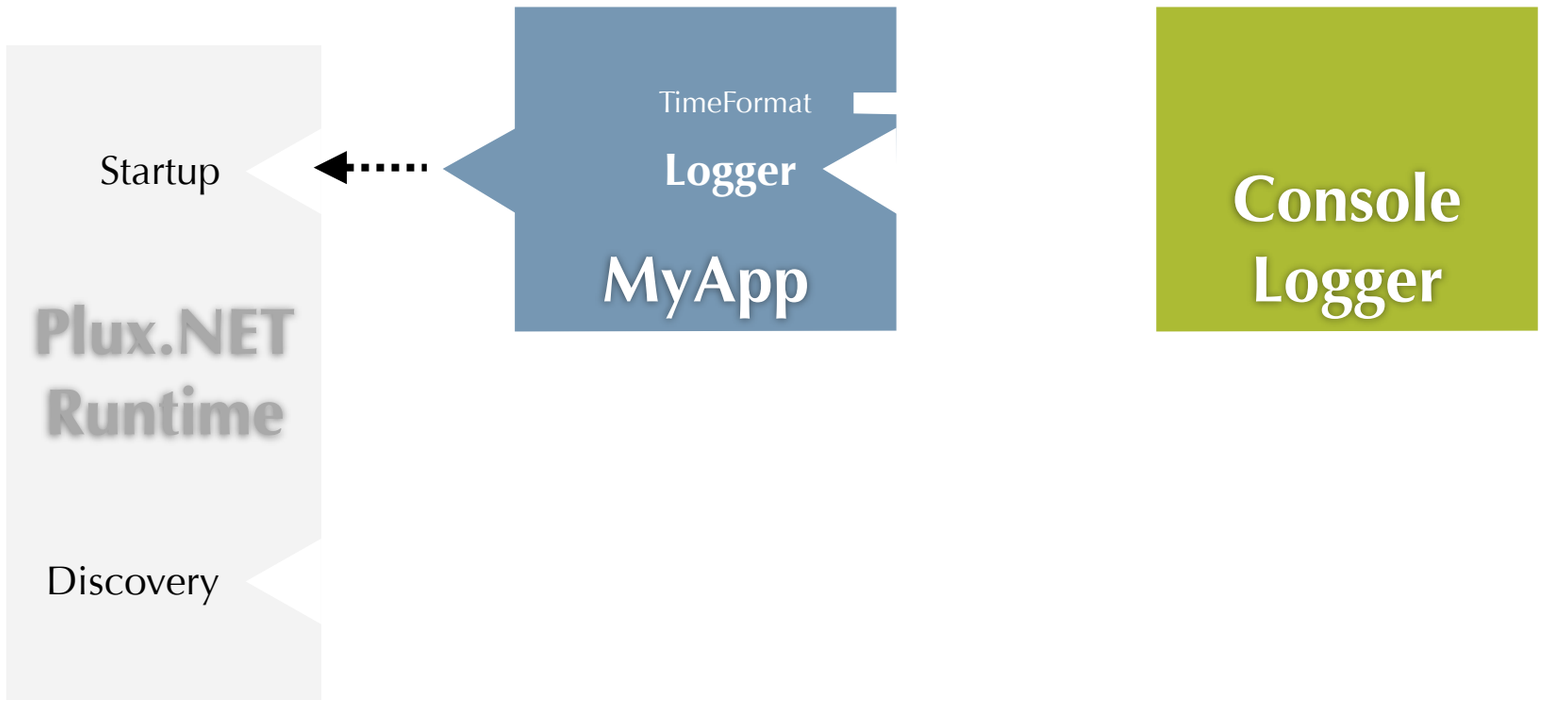



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

```
[Extension("ConsoleLogger")]


class ConsoleLogger {

}
```

Slot & Extension

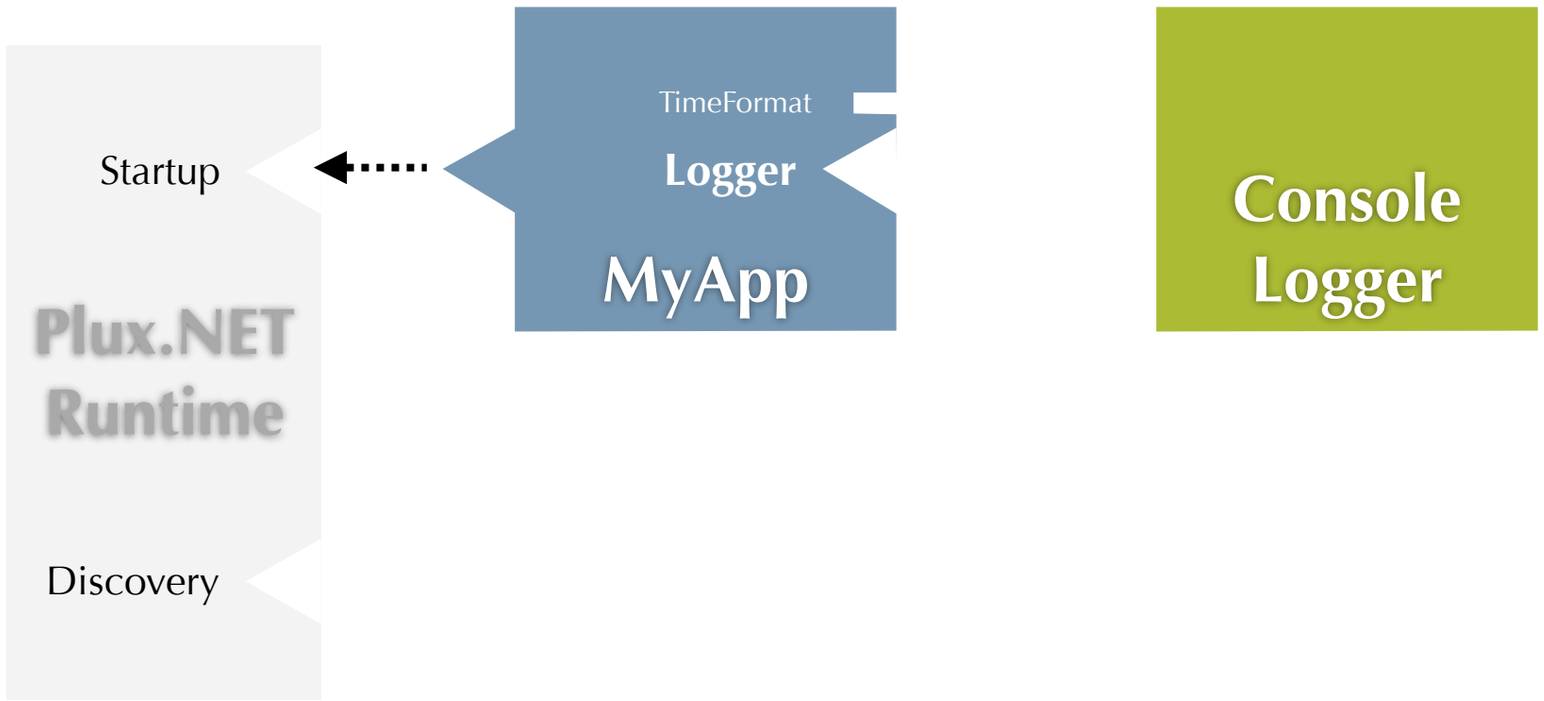



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

```
[Extension("ConsoleLogger")]


class ConsoleLogger {

}
```

Slot & Extension

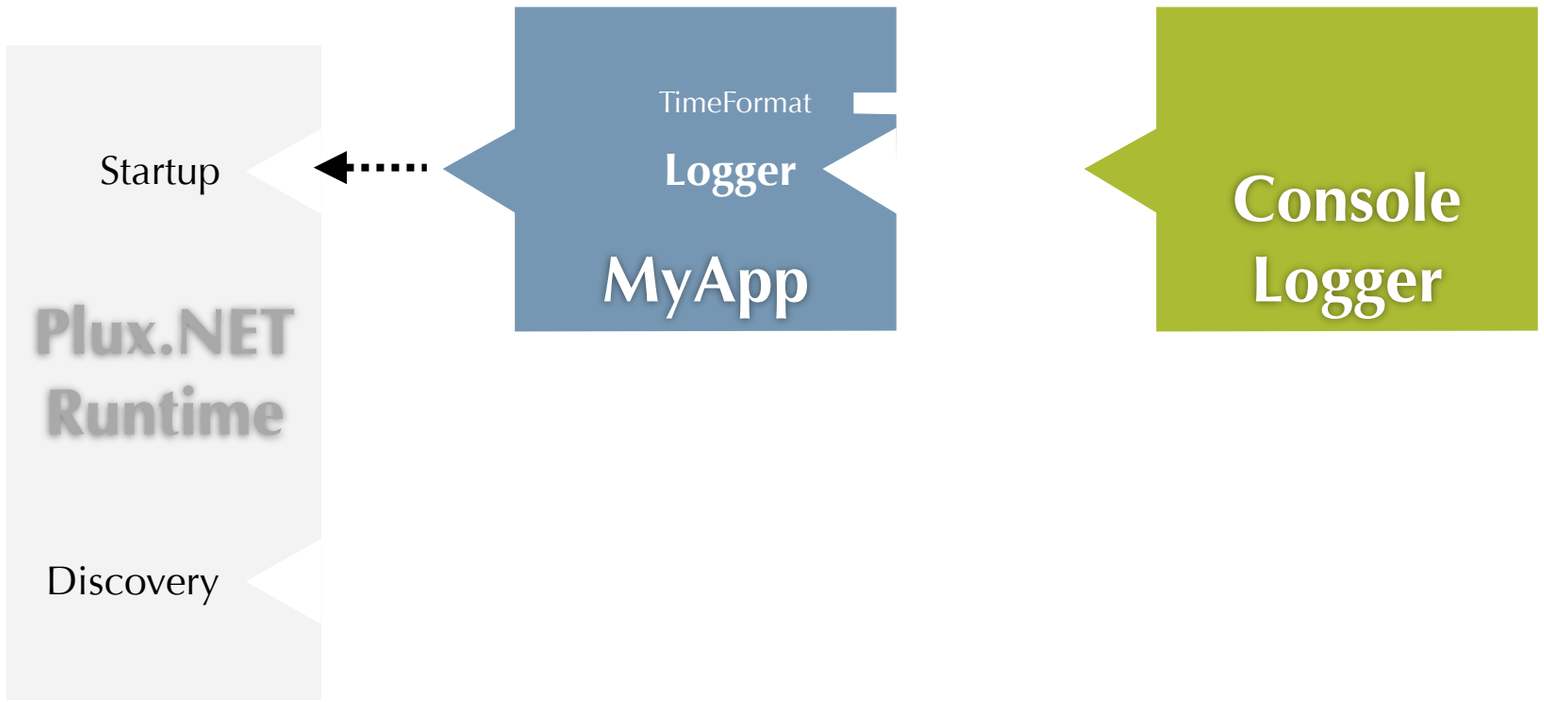



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

```
[Extension("ConsoleLogger")]
[Plug("Logger")]

class ConsoleLogger : ILogger {
   void Print(string msg) { ... }
}
```

Slot & Extension

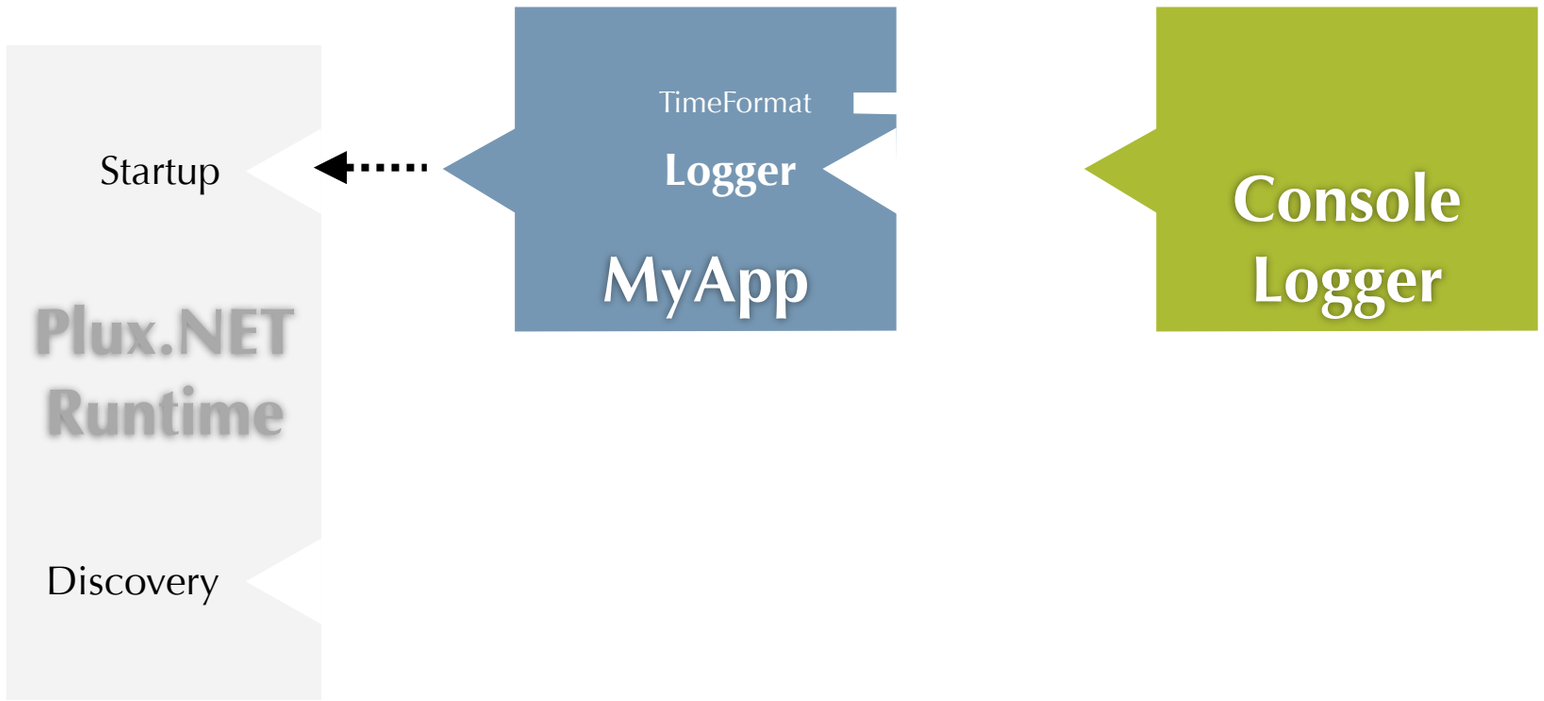



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

```
[Extension("ConsoleLogger")]
[Plug("Logger")]

class ConsoleLogger : ILogger {
   void Print(string msg) { ... }
}
```

Slot & Extension

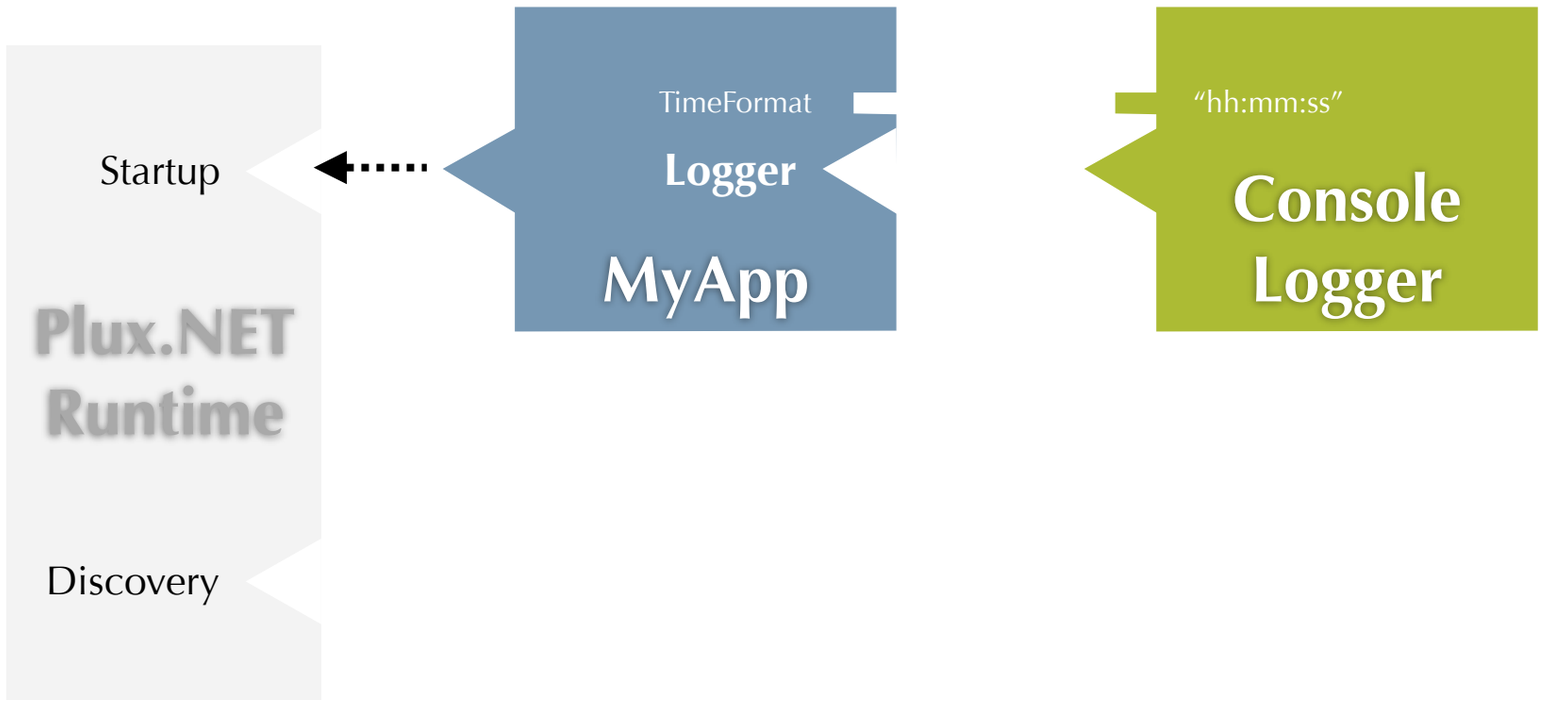



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

```
[Extension("ConsoleLogger")]
[Plug("Logger")]
[ParamValue("TimeFormat", "hh:mm:ss")]
class ConsoleLogger {
  void Print(string msg) { ... }
}
```

Slot & Extension

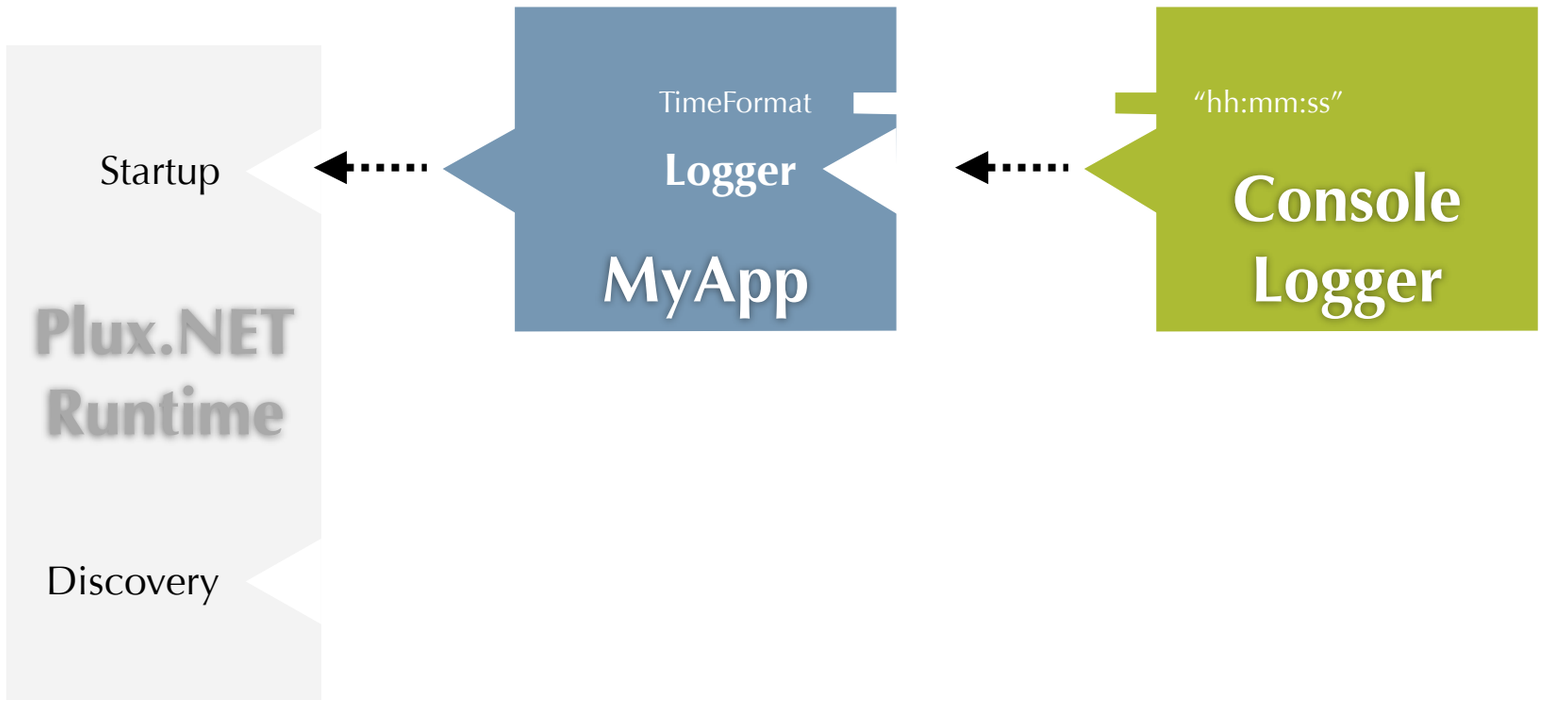



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

```
[Extension("ConsoleLogger")]
[Plug("Logger")]
[ParamValue("TimeFormat", "hh:mm:ss")]
class ConsoleLogger {
  void Print(string msg) { ... }
}
```

Slot & Extension

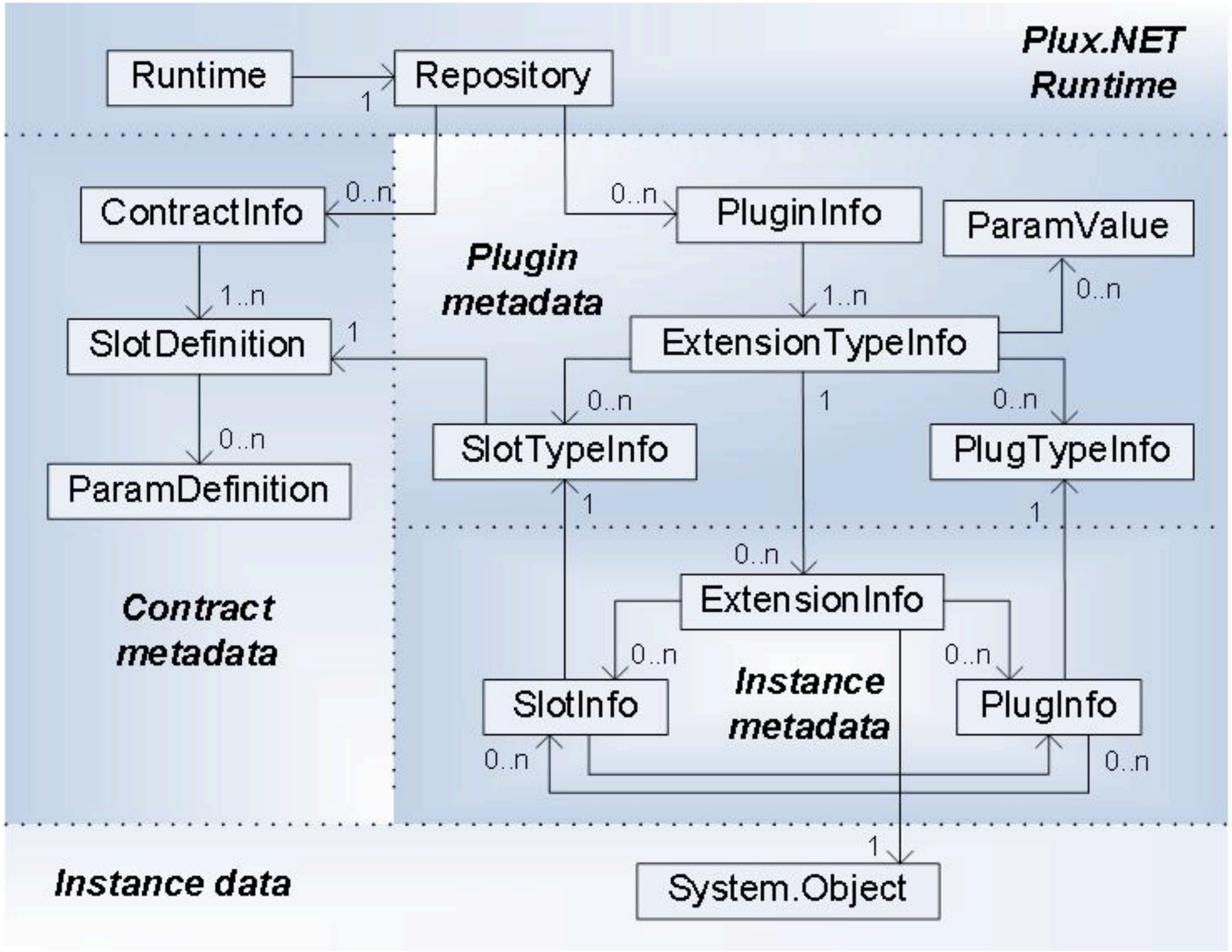



```
[Extension("MyApp")]
[Plug("Startup")]
[Slot("Logger")]
class MyApp : IStartup {
  void Run() { ... }
}
```

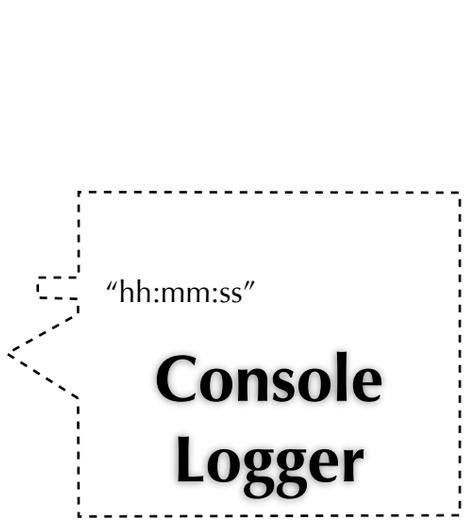
```
[SlotDefinition("Logger")]
[Param("TimeFormat", typeof(string))]
interface ILogger {
  void Print(string msg);
}
```

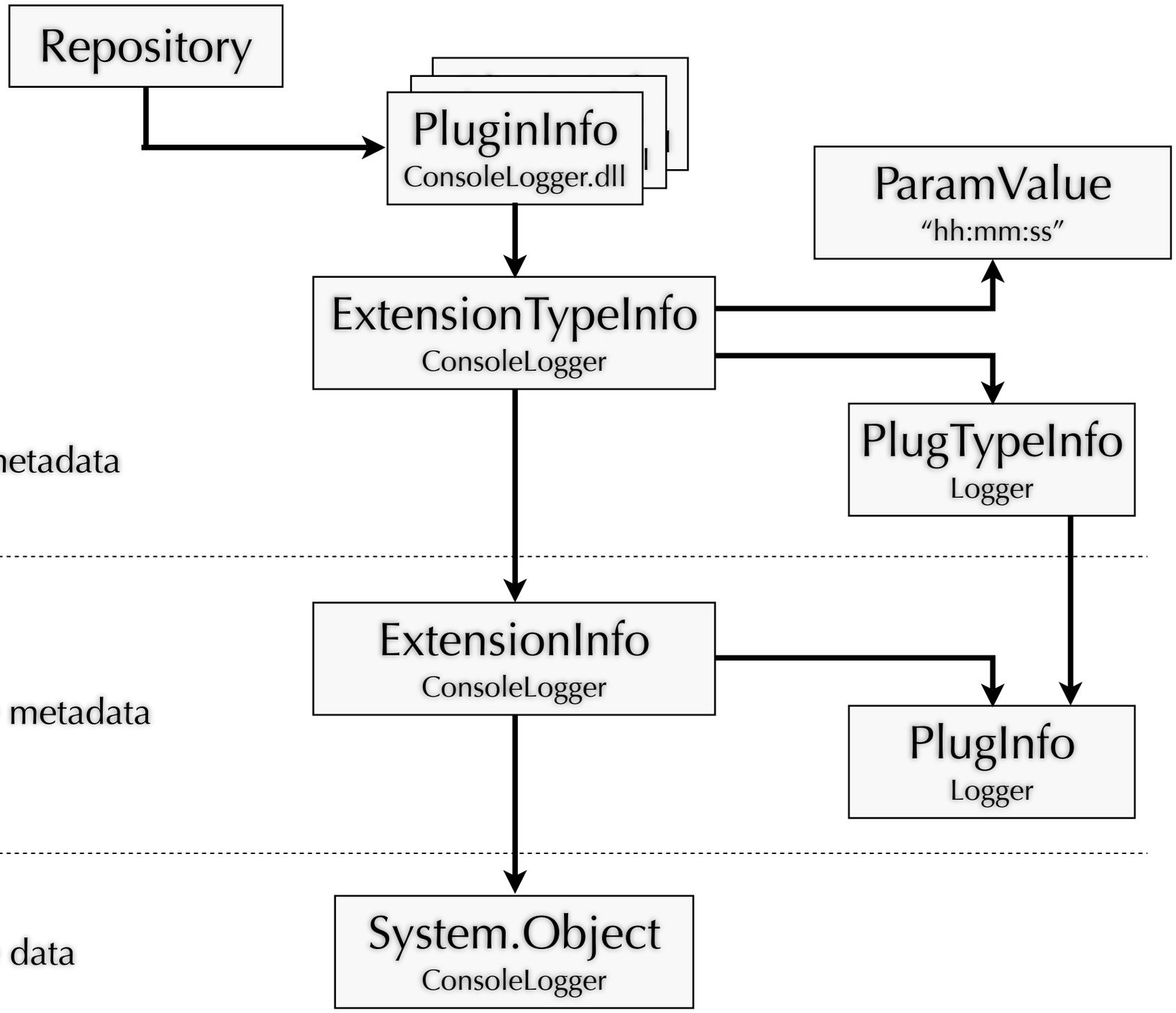
```
[Extension("ConsoleLogger")]
[Plug("Logger")]
[ParamValue("TimeFormat", "hh:mm:ss")]
class ConsoleLogger {
  void Print(string msg) { ... }
}
```

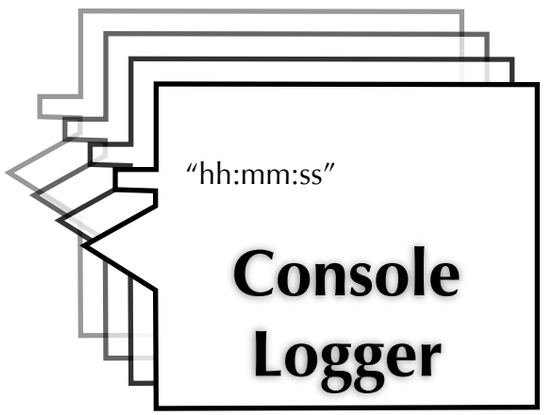
Plux.NET API


Plux.NET
Runtime
Runtime
Repository
1
ContractInfo
0..n
PluginInfo
0..n
ParamValue
Plugin
metadata
1..n
1..n
0..n
SlotDefinition
1
ExtensionTypeInfo
0..n
1
0..n
0..n
SlotTypeInfo
PlugTypeInfo
ParamDefinition
1
1
0..n
ExtensionInfo
Contract
metadata
0..n
0..n
Instance
metadata
SlotInfo
PlugInfo
0..n
0..n
1
Instance data
System.Object

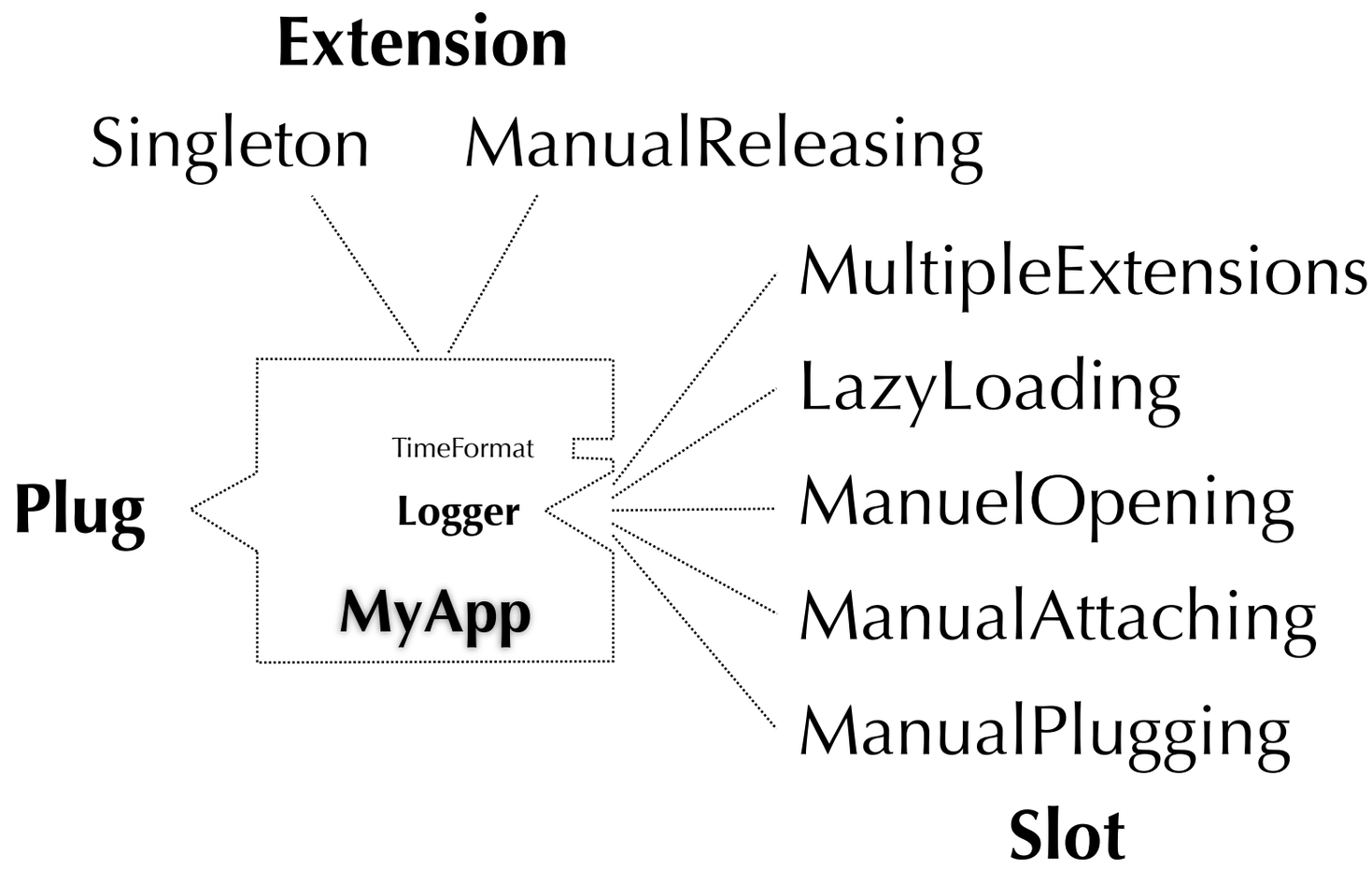
Beispiel


"hh:mm:ss"
Console
Logger


Repository
PluginInfo
ConsoleLogger.dll
ParamValue
"hh:mm:ss"
ExtensionTypeInfo
ConsoleLogger
PlugTypeInfo
Logger
Plugin metadata
ExtensionInfo
ConsoleLogger
PlugInfo
Logger
Instance metadata
System.Object
ConsoleLogger
Instance data

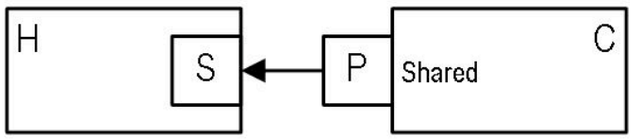

"hh:mm:ss"
Console
Logger

Slot & Extension

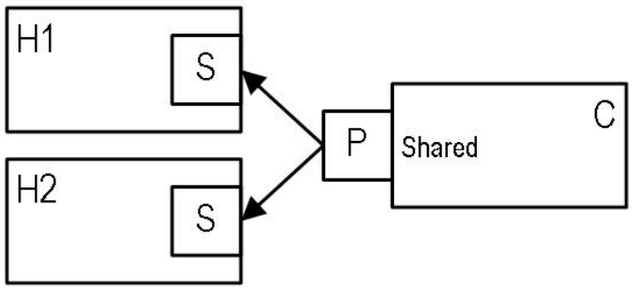


Muster

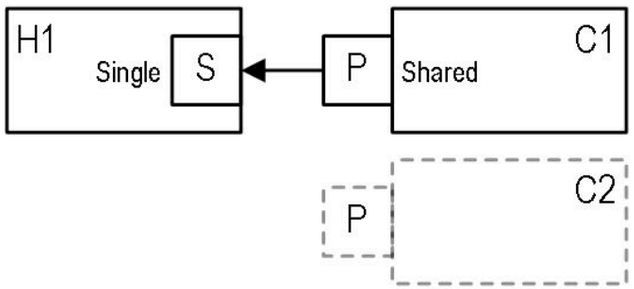
Automatically plug shared extension



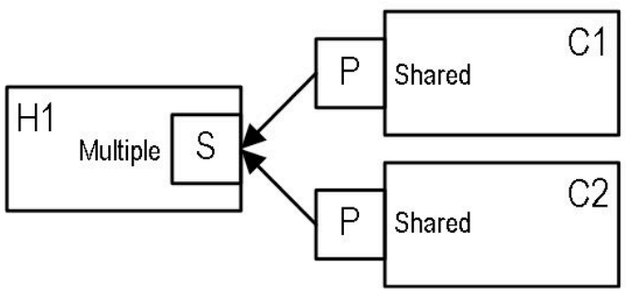
Automatically plug shared extension in multiple hosts



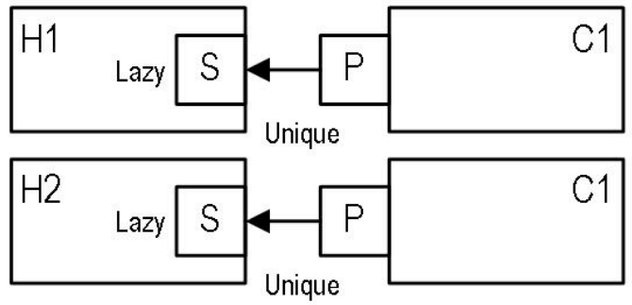
Constrain slot to single contributor



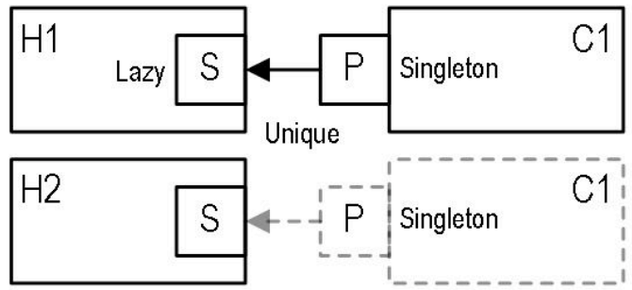
Allow multiple contributors



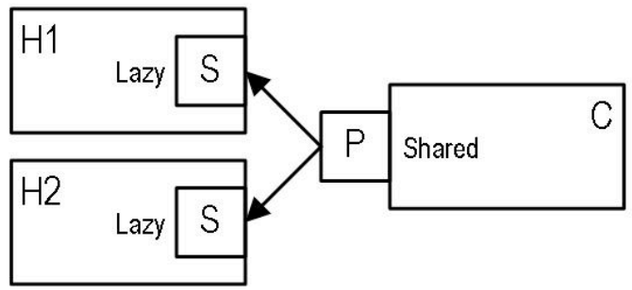
Semi-automatically plug unique contributors



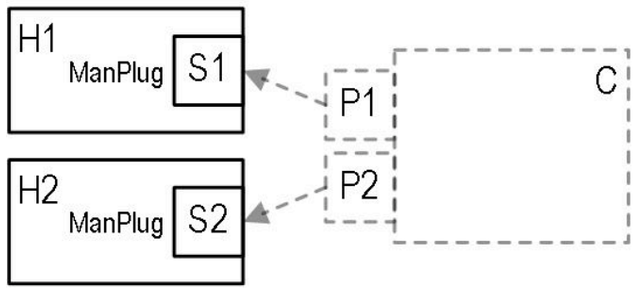
Constrain contributor to single instance

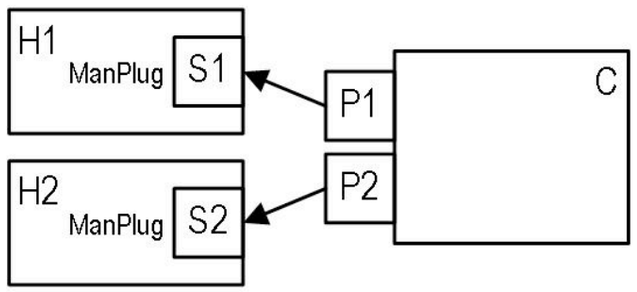


Lazy-load contributor

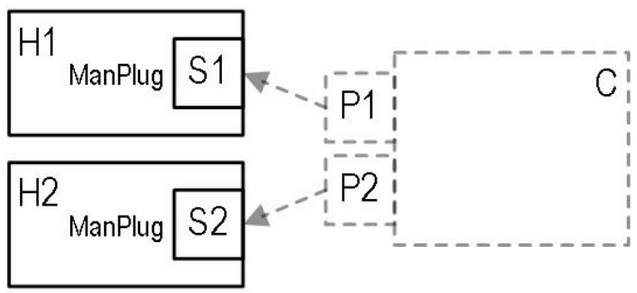


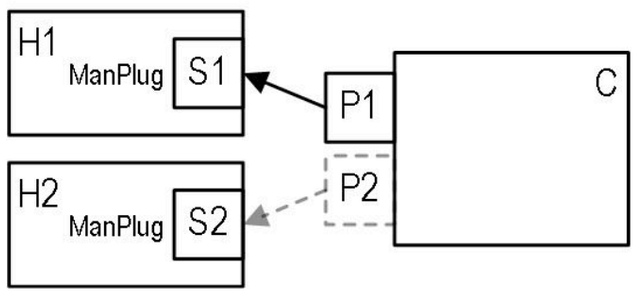
Manually plug all plugs



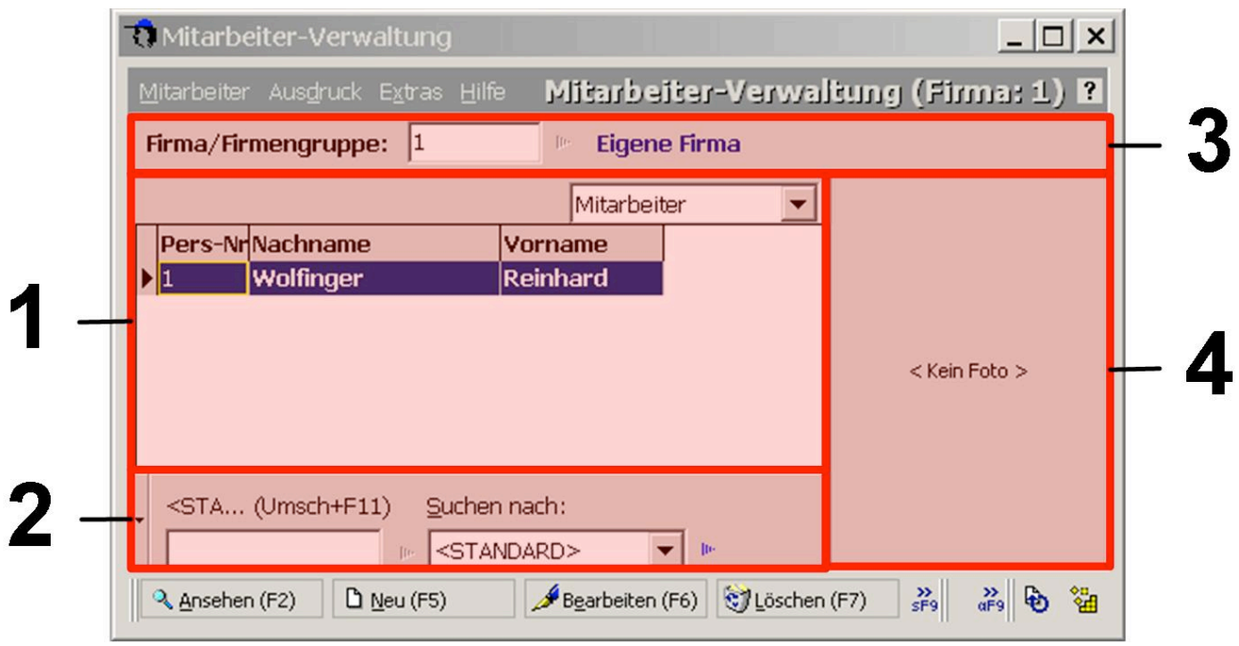


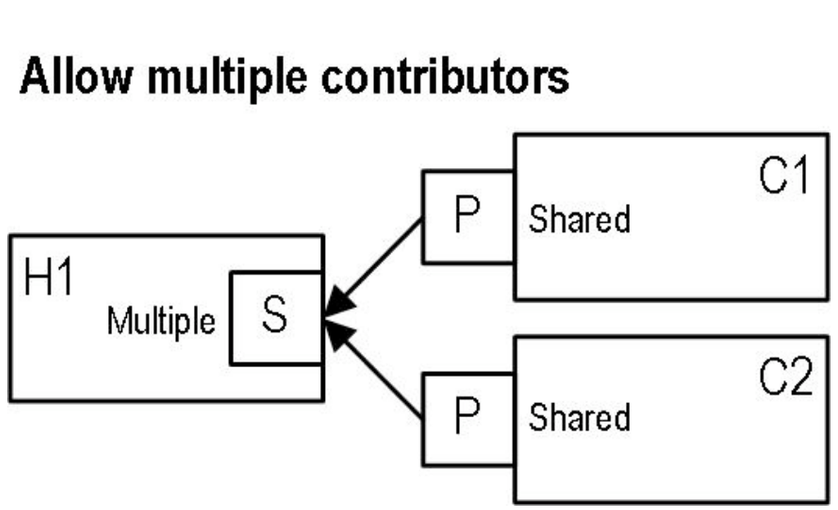
Manually plug individual plugs

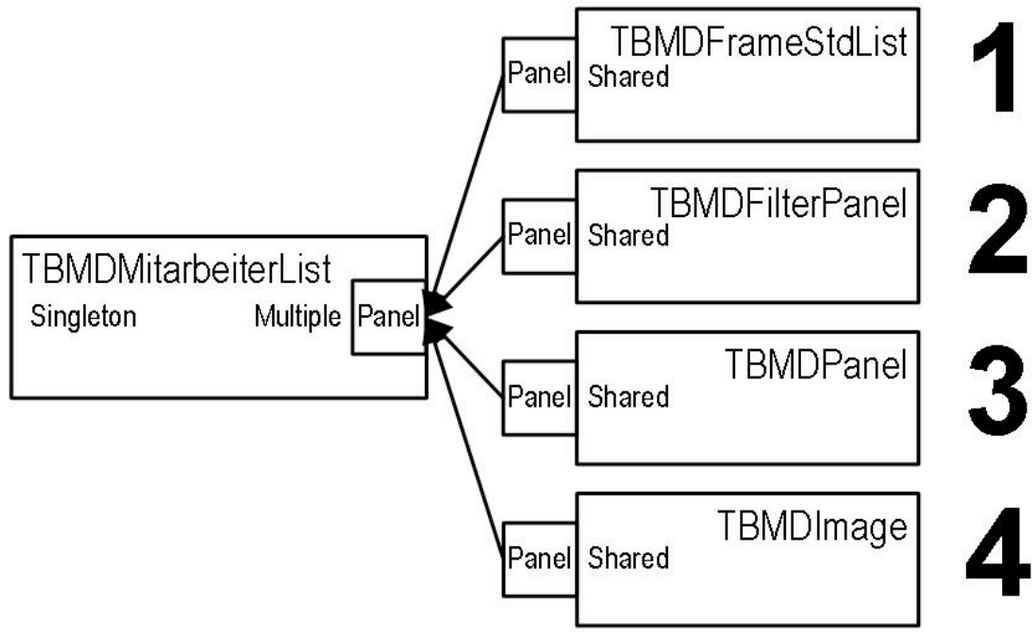




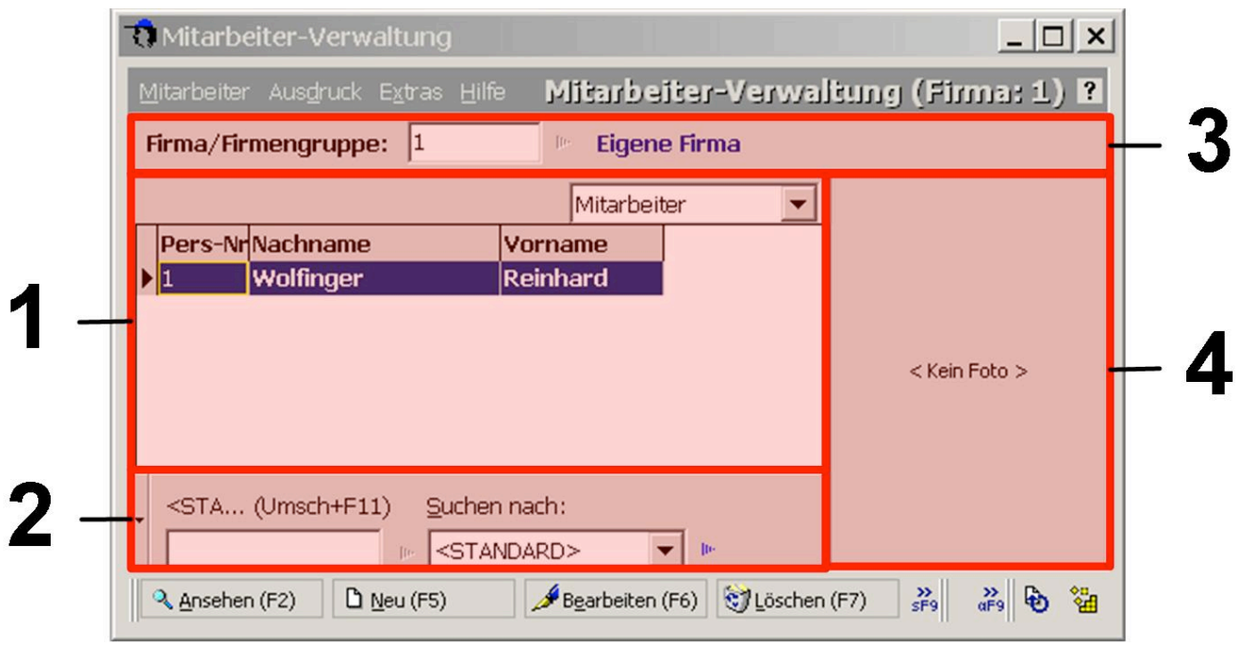
Beispiel 1 - Panels



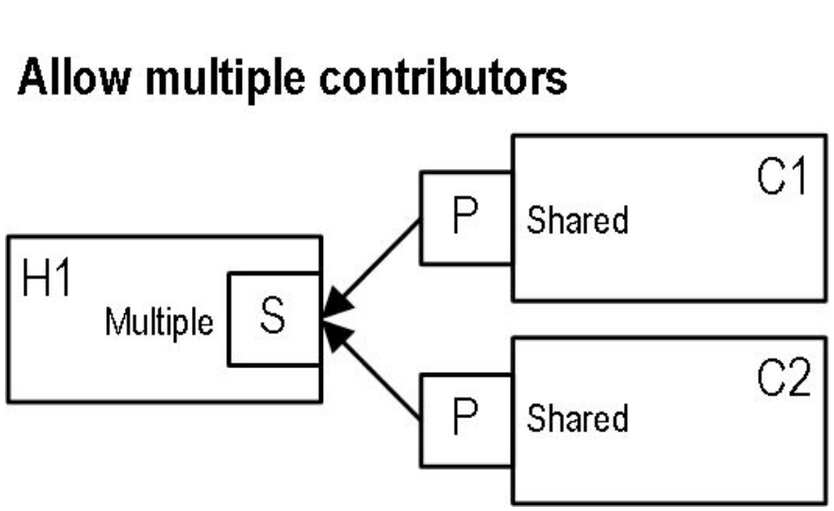


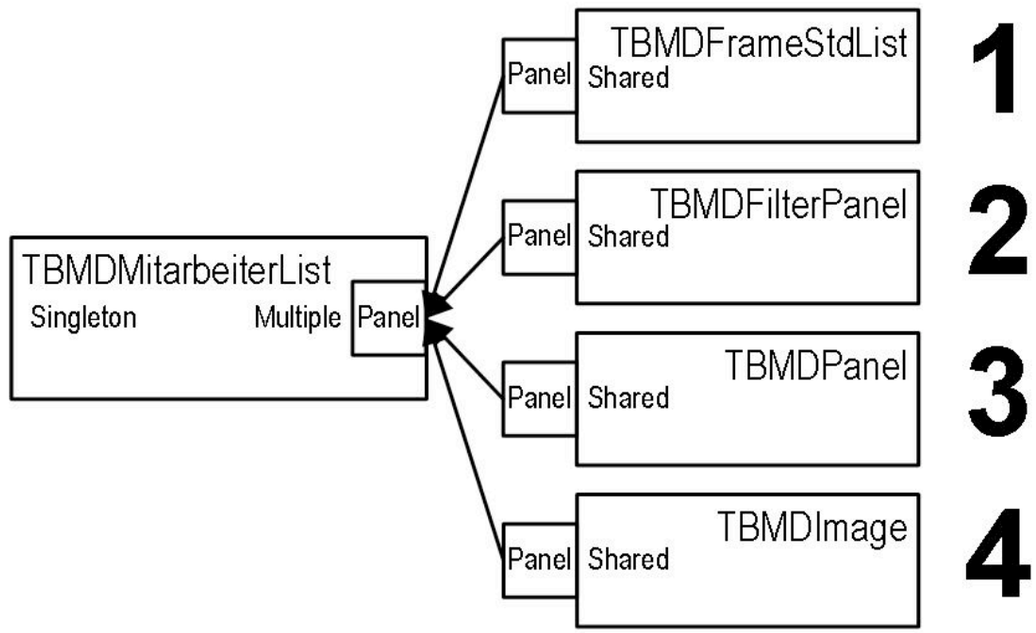


Beispiel 1 - Panels

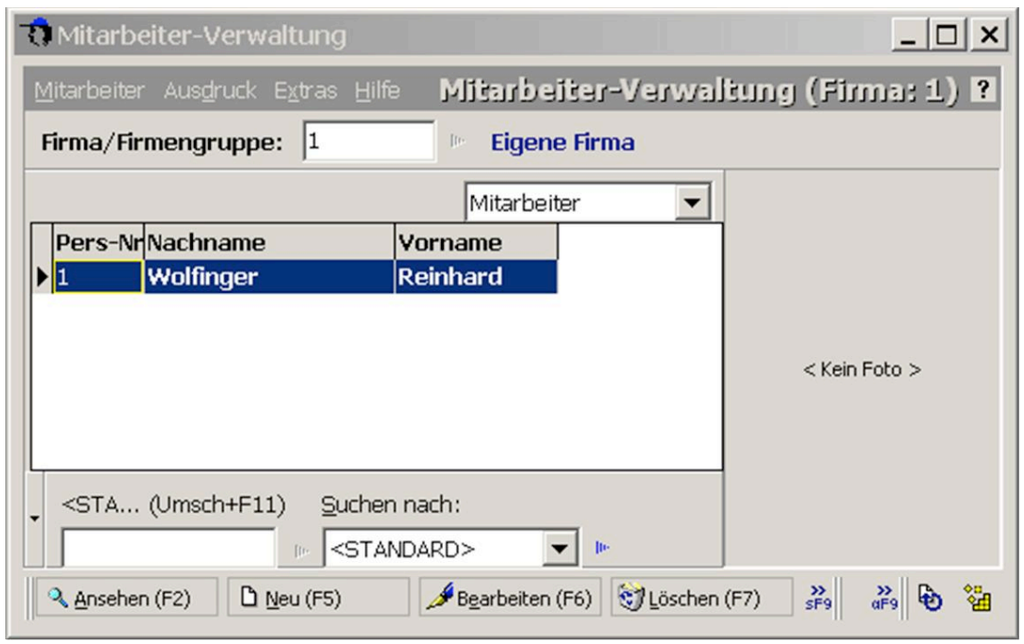


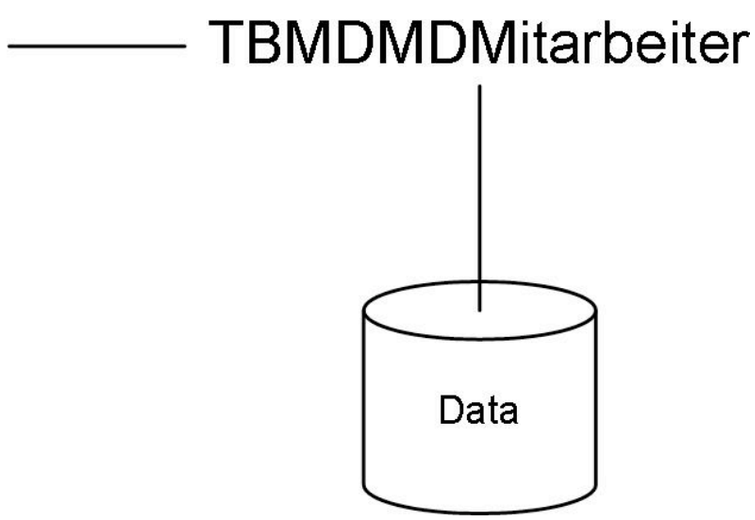

```
type TBMDFRMMitarbeiterList =
                    class(TBMDFRMPersonList)
  protected
     FListFrame: TBMDFrameStdList;
     FFilterPnl: TBMDFilterPanel;
     FPnlTop: TBMDPanel;
     FImage: TBMDImage;
end;
```



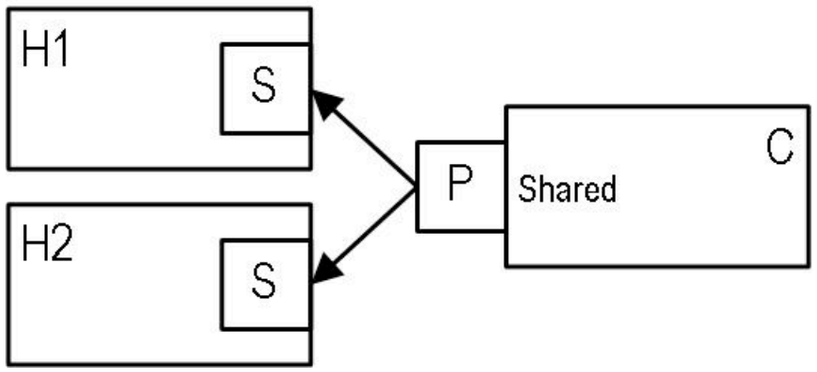


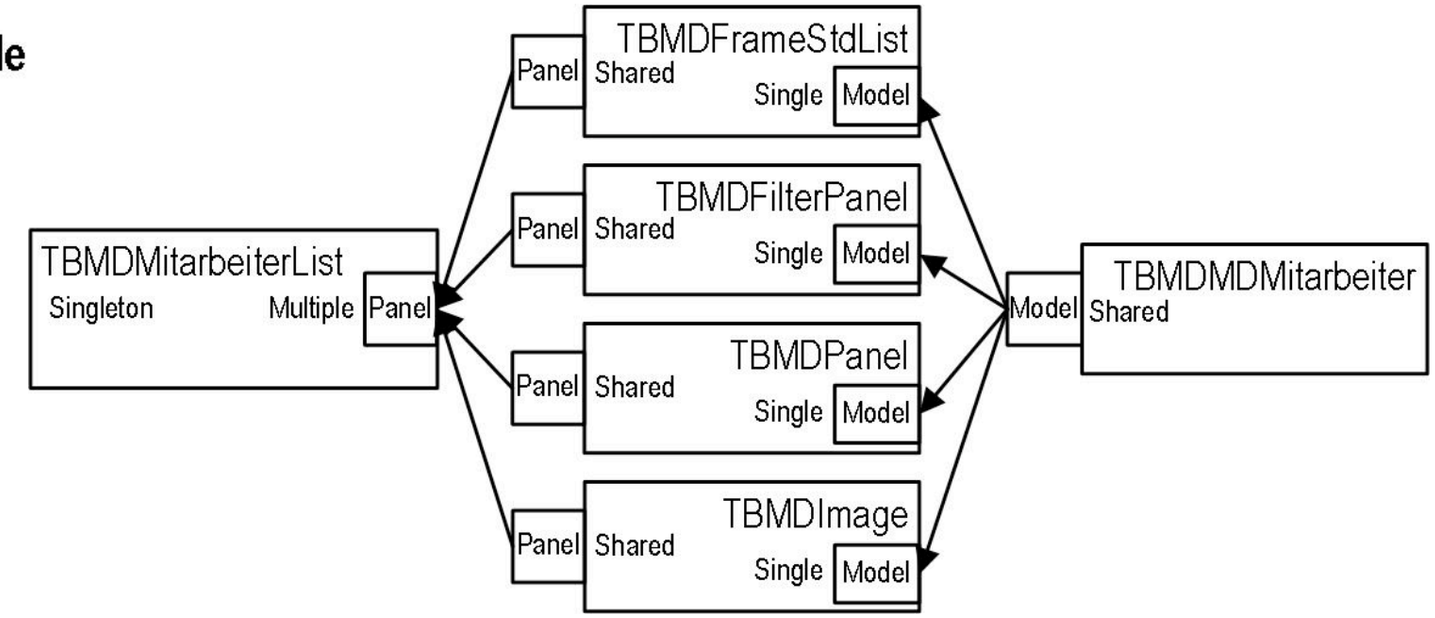
Beispiel 2 - Data Sources



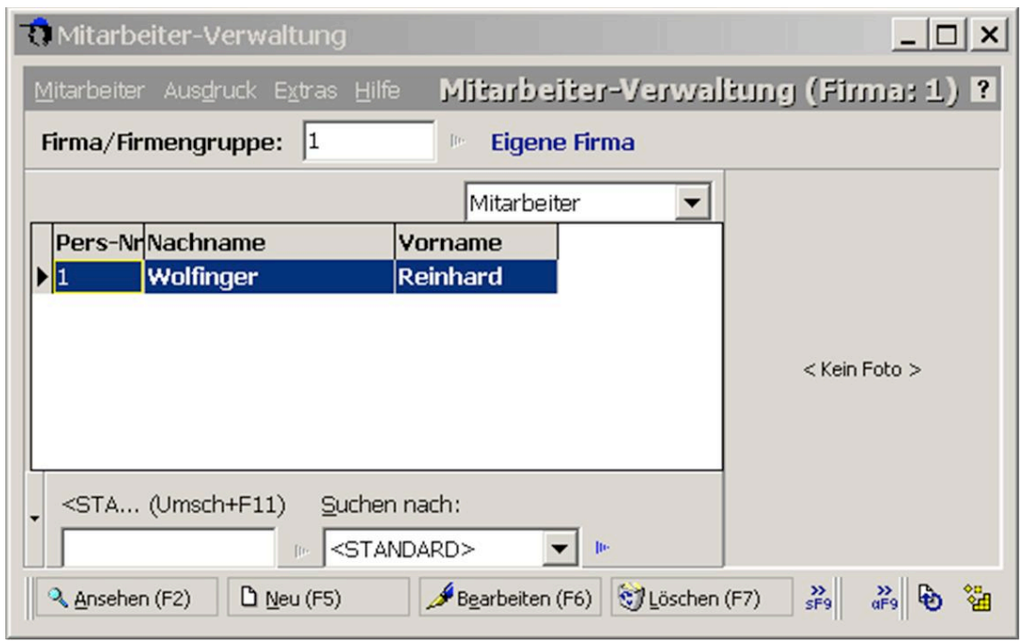


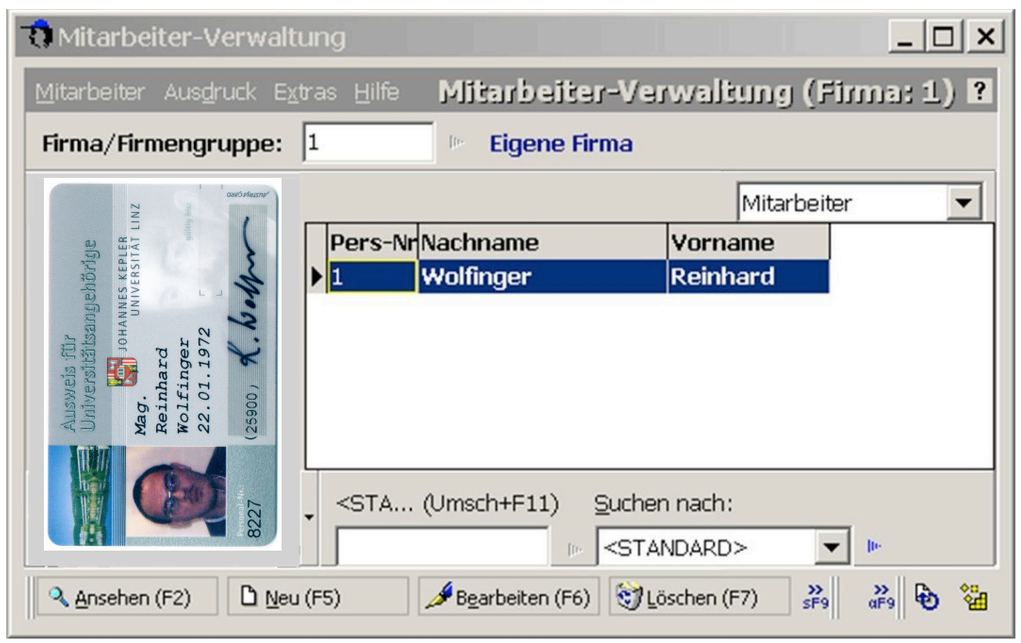
Automatically plug shared extension in multiple



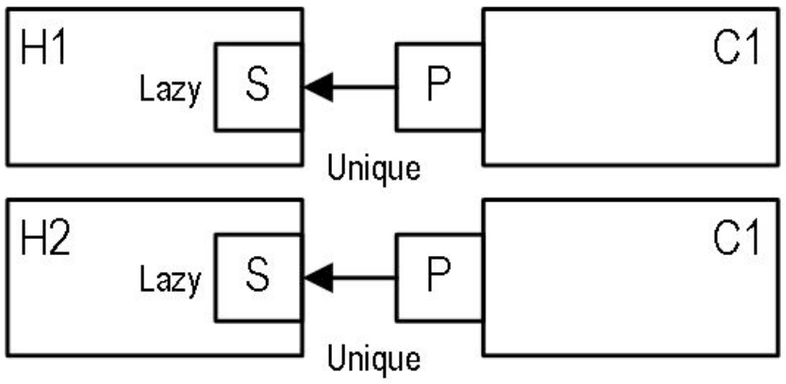


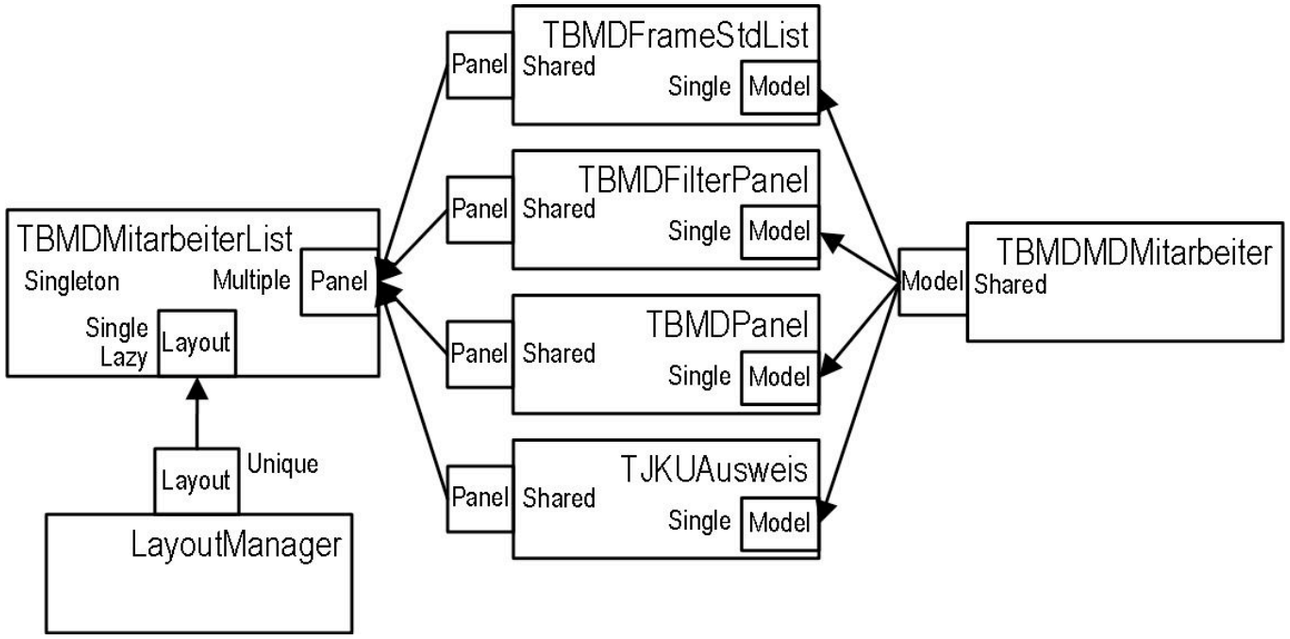
Beispiel 3 - LayoutManager



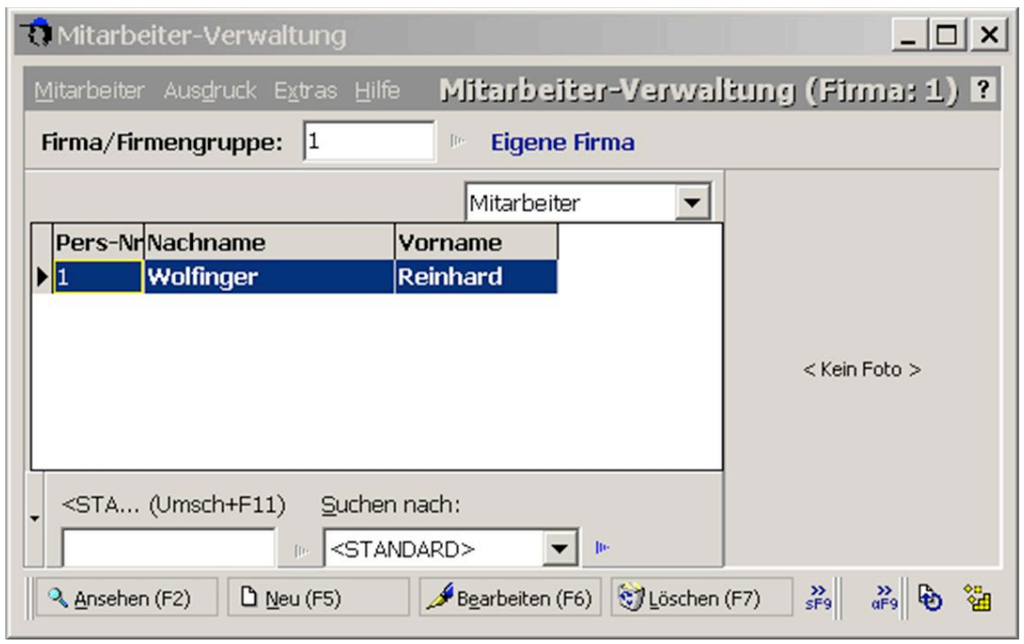


Semi-automatically plug unique contributors

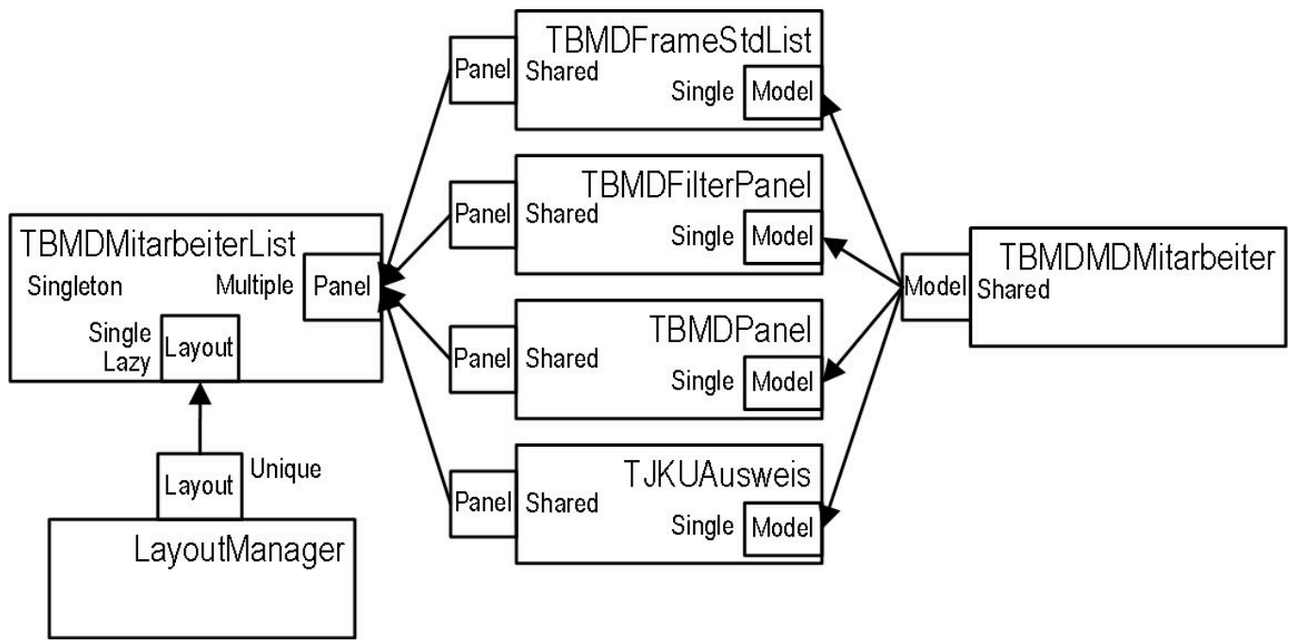




Beispiel 3 - LayoutManager




```
[Extension("TBMDMitarbeiterList")]
[Plug("Workbench.View")]
[Slot("Layout", LazyLoading=true,
  OnCreate="OnCreated",
  OnAttach= "OnAttached")]
class TBMDMitarbeiterList : IView {
  ExtensionInfo me = null;
  void OnCreated(ExtensionEventArgs a) {
    me = a.ExtensionInfo;
  }
  void OnAttached(ExtensionExventArgs a) {
    ExtensionInfo e = a.ExtensionTypeInfo.CreateUniqueInstance();
    e.PlugInfos["Layout"].Plug(me.SlotsInfos["Layout"]);
  }
}
```



Werkzeuge und Anwendungen

Neue Werkzeuge zur Konfiguration und Analyse

- Plux-Console
- Plux-Explorer
- Plux-Monitor

Neue Anwendungen mit Plux.NET

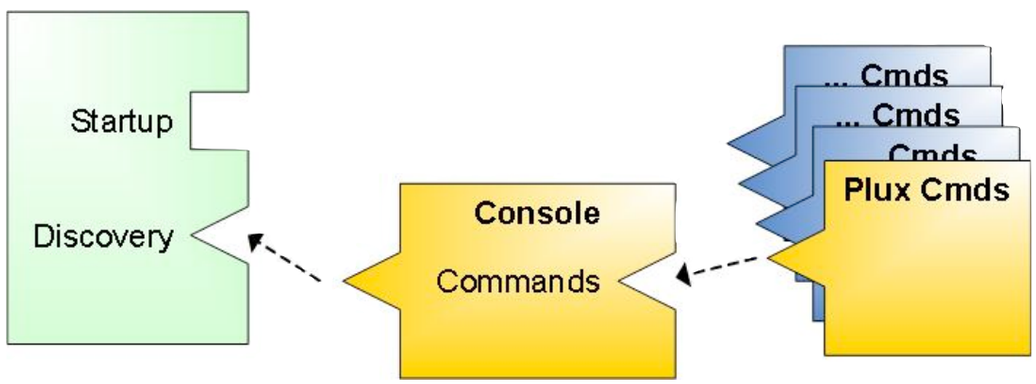
- CompilerMaster (.NET 2.0, Windows Forms)
- ContentWatcher (.NET 3.0, Windows Presentation Foundation)

Plux-Console

Plux-Console

- Konfigurations- und Analysewerkzeug für die Kommandozeile
  - steuern der Plux.NET Runtime
  - konfigurieren von Plug-in Anwendungen
  - analysieren von Konfigurationen
- Konfiguration laden und speichern
- Erweiterbar durch Befehls-Plugins
  - z.B. Kennzahlen zum Resourcenverbrauch

Demo

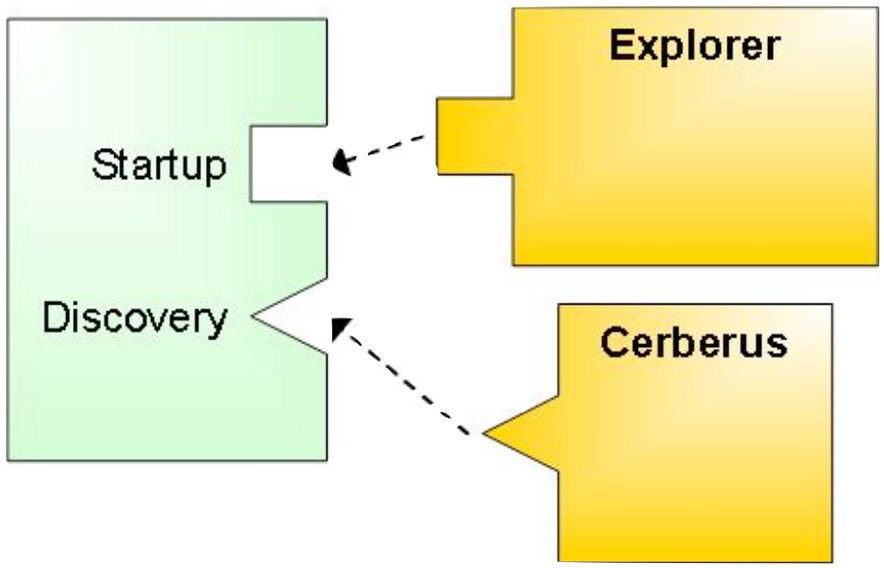


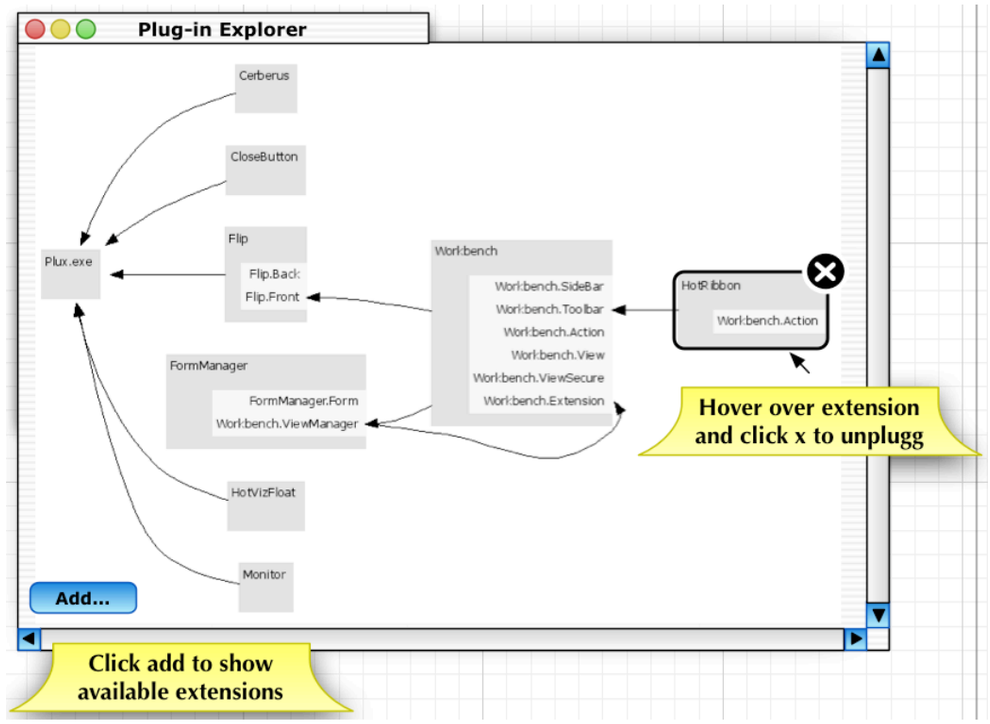

```
PluxConsole
Plux.NET Version 0.2.800.0
[Plux.NET Version 0.2.791.0]
Copyright (C) Christian Doppler Laboratory for Automated Software Eng
2006-2008. All rights reserved.

plux> help

 list plugininfos
 list extensioninfos
 list plugtypeinfos
 install
 attach
 autoattach off|on
 help

plux> list plugininfos

id   State          PluginInfo (Version)
1    ACTIVE (1)     Plux.dll (0.2.791.0)
2    ACTIVE (1)     console.dll (0.1.0.39340)

plux>
```

Plux-Explorer

Plux-Explorer

- graphisches Konfigurations- und Analysewerkzeug
  - installiert im Modus Endbenutzer Features
  - installiert im Modus Architekt einzelne Extensions

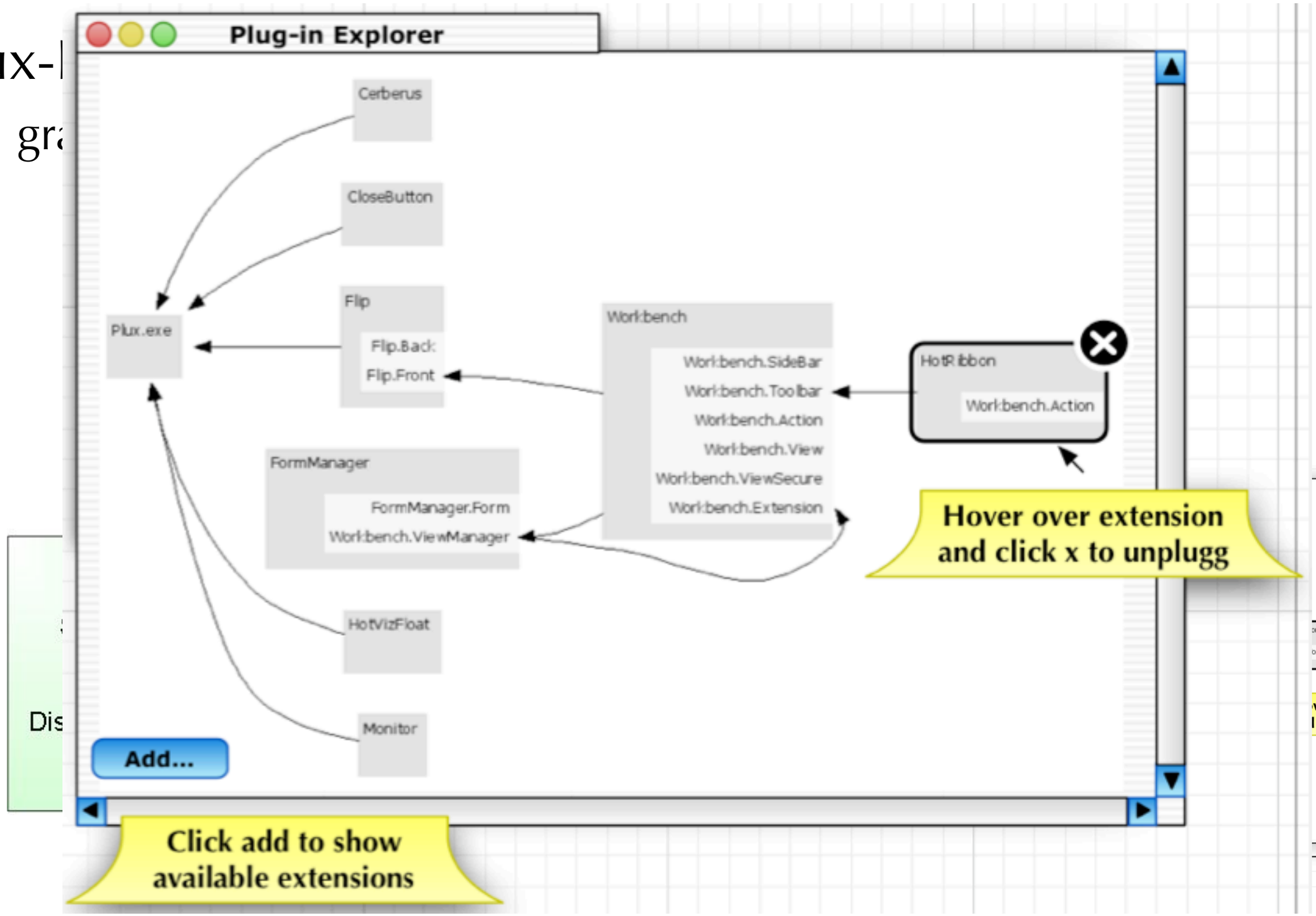




Plux-Explorer

Plux-

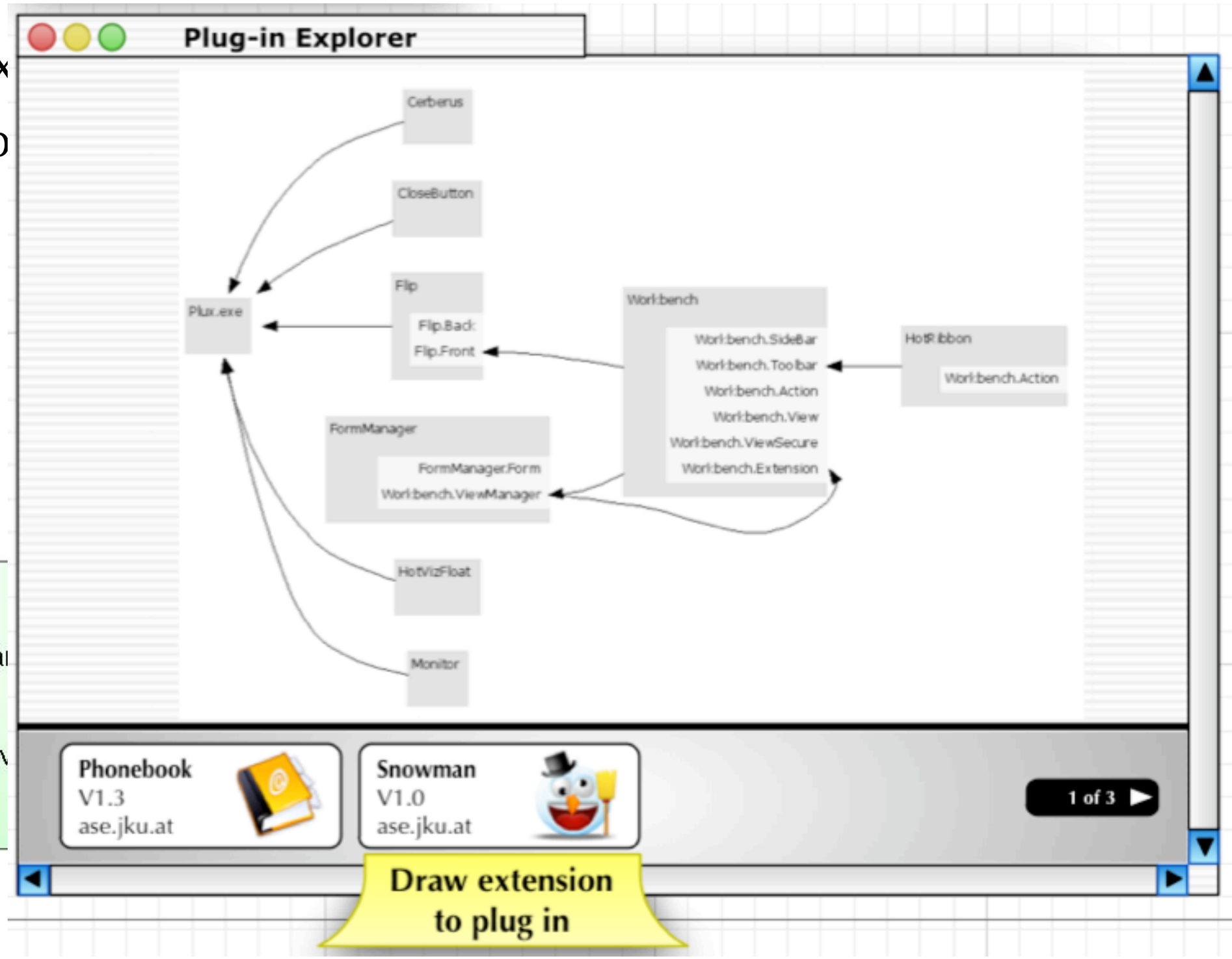
- gr



Plux-Explorer

Plux-Ex

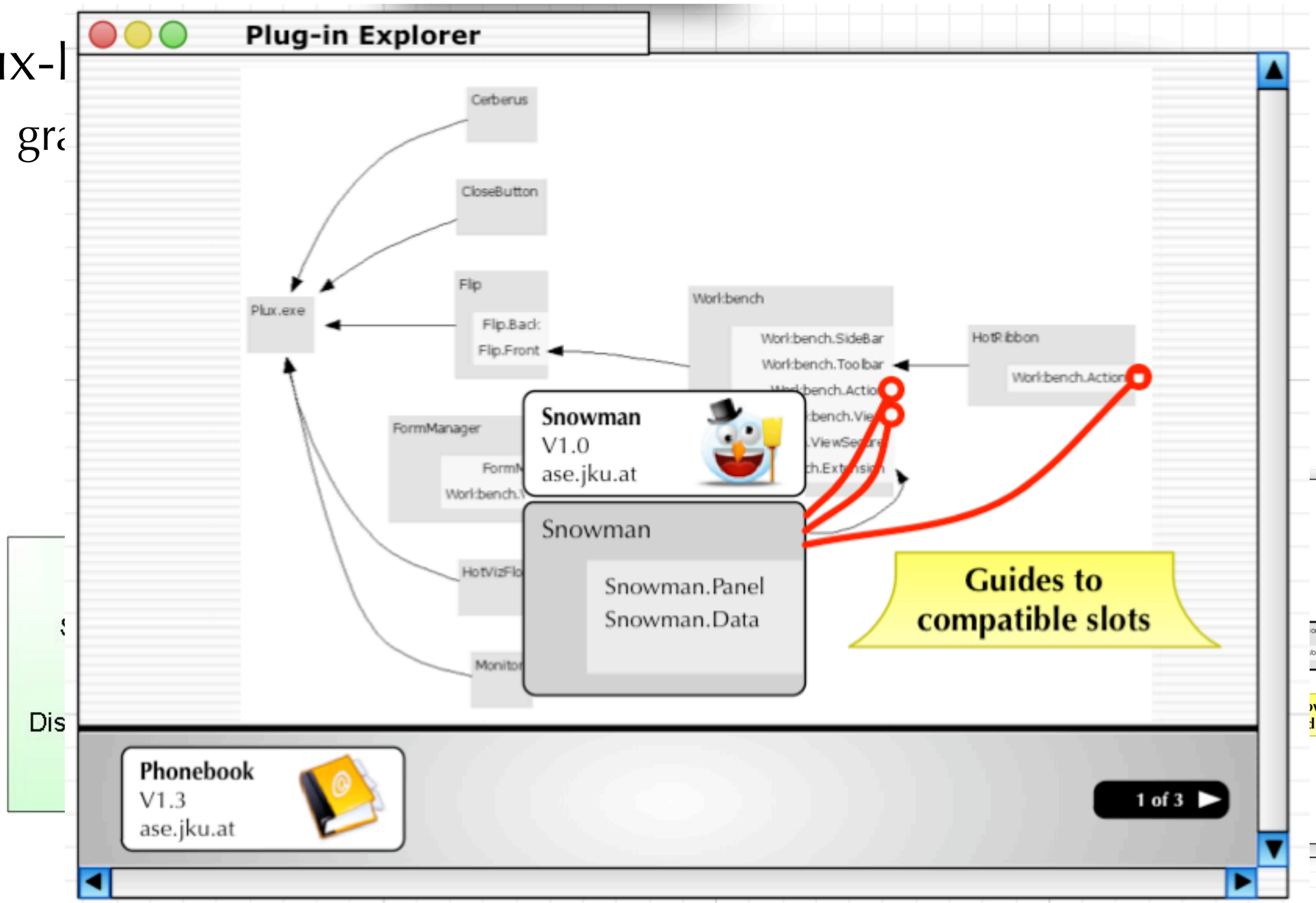
- grap
  -
  -



Plux-Explorer

Plux-I

- gra



Plux-Monitor (1)

Kennzahlen zur Diagnose, Überwachung und Steuerung

- Basiskennzahlen
  - Resourcenverbrauch, z.B. Anzahl aktuell geladener Extensions, oder Anzahl geladener Extensions seit Start
  - Nutzungsverhalten, z.B. Zeitdauer die eine Extension geladen ist
- abgeleitete Kennzahlen
  - durchschnittliche Zeitdauer die eine Extension geladen ist

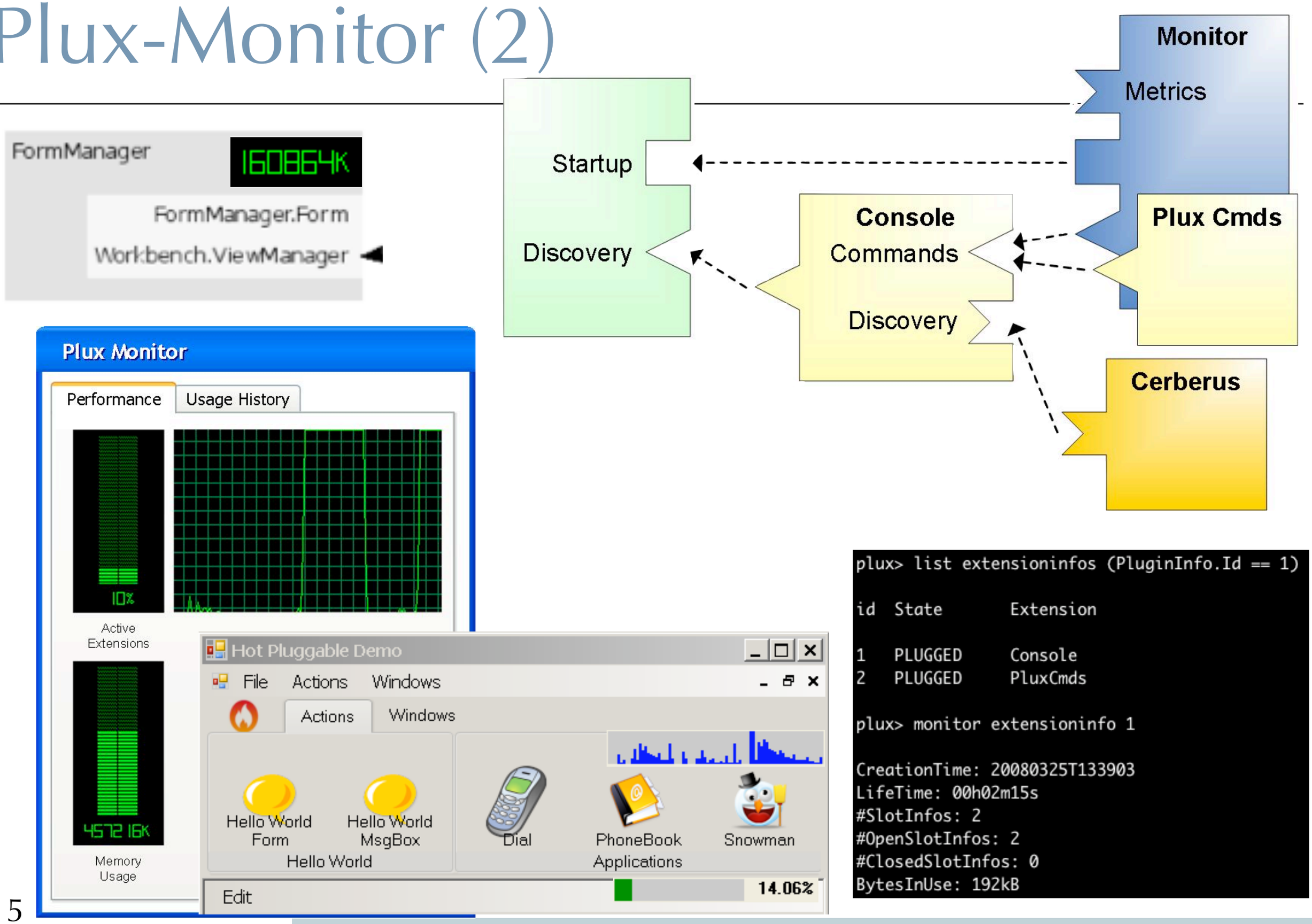
Kennzahlen zur Evaluierung

- Frage: sind Plug-in Anwendungen wirklich schlanker?
- Frage: wie wird Individualisierung genutzt?

Kennzahlen stehen zur Verfügung...

- für die Plux-Console über ein Befehls-Plugin
- für den Plux-Explorer
- für andere Plug-ins über API

Plux-Monitor (2)

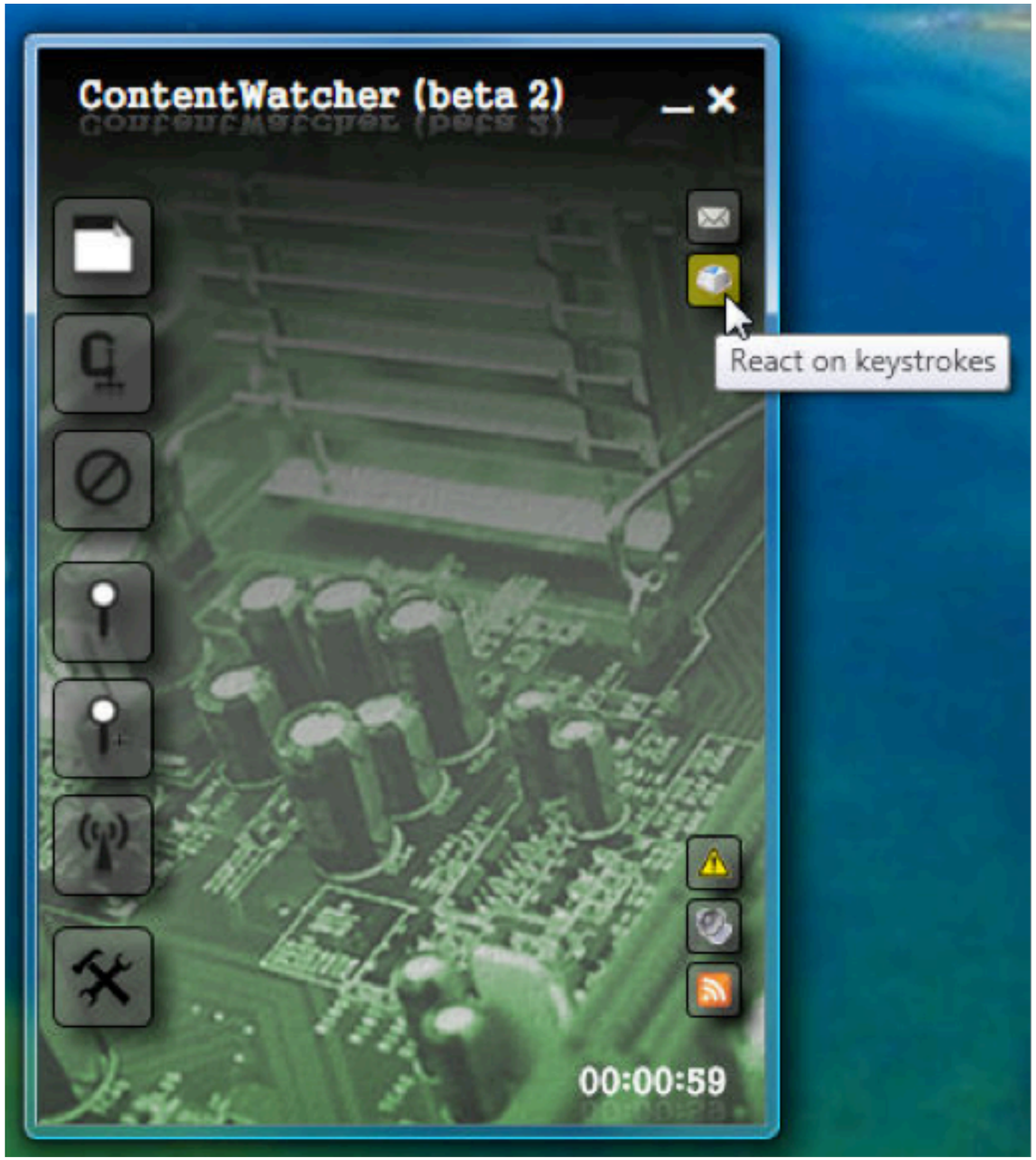



```
plux> list extensioninfos (PluginInfo.Id == 1)

id  State       Extension

1   PLUGGED     Console
2   PLUGGED     PluxCmds

plux> monitor extensioninfo 1

CreationTime: 20080325T133903
LifeTime: 00h02m15s
#SlotInfos: 2
#OpenSlotInfos: 2
#ClosedSlotInfos: 0
BytesInUse: 192kB
```

CompilerMaster

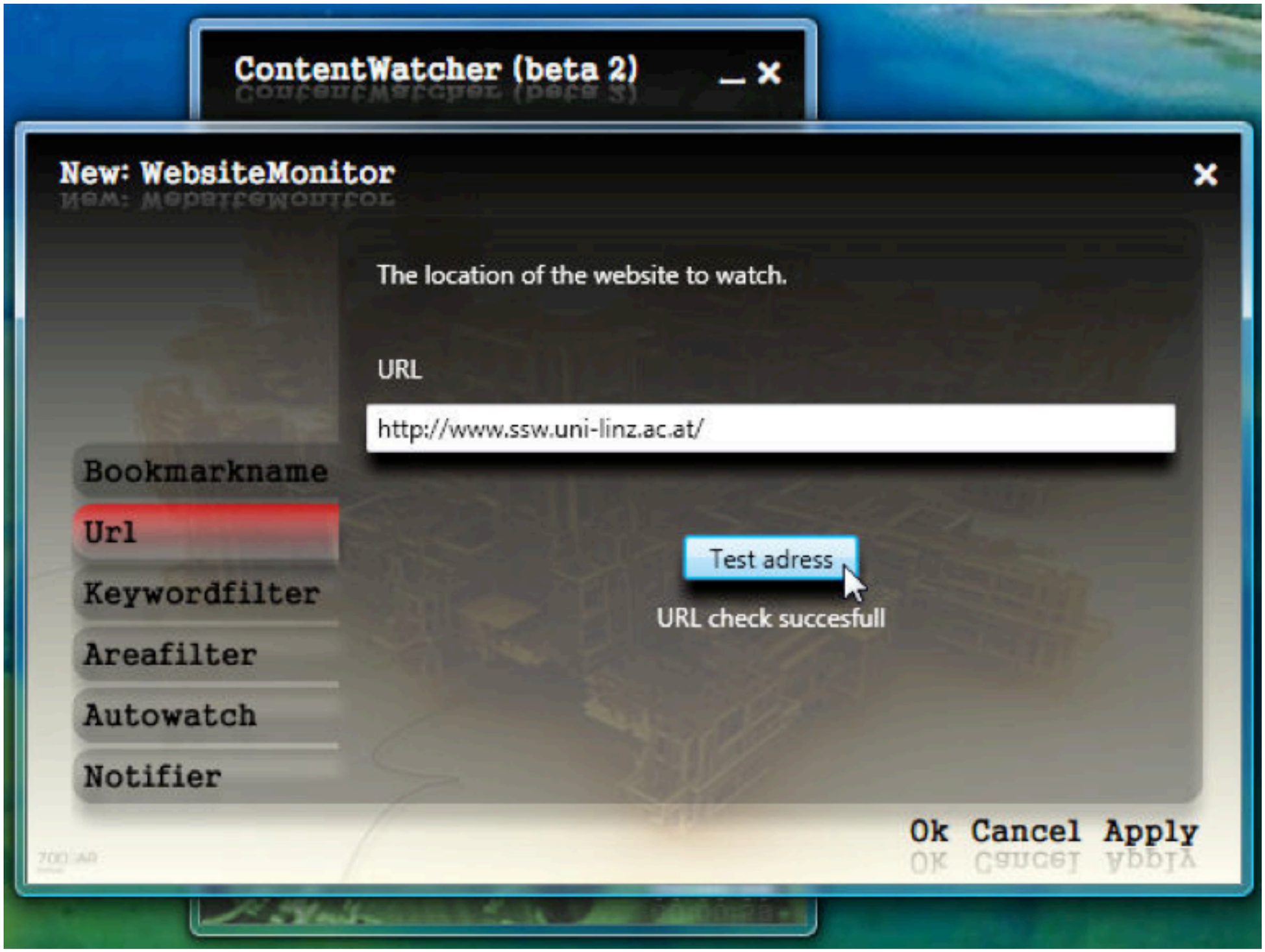
Demo

ContentWatcher

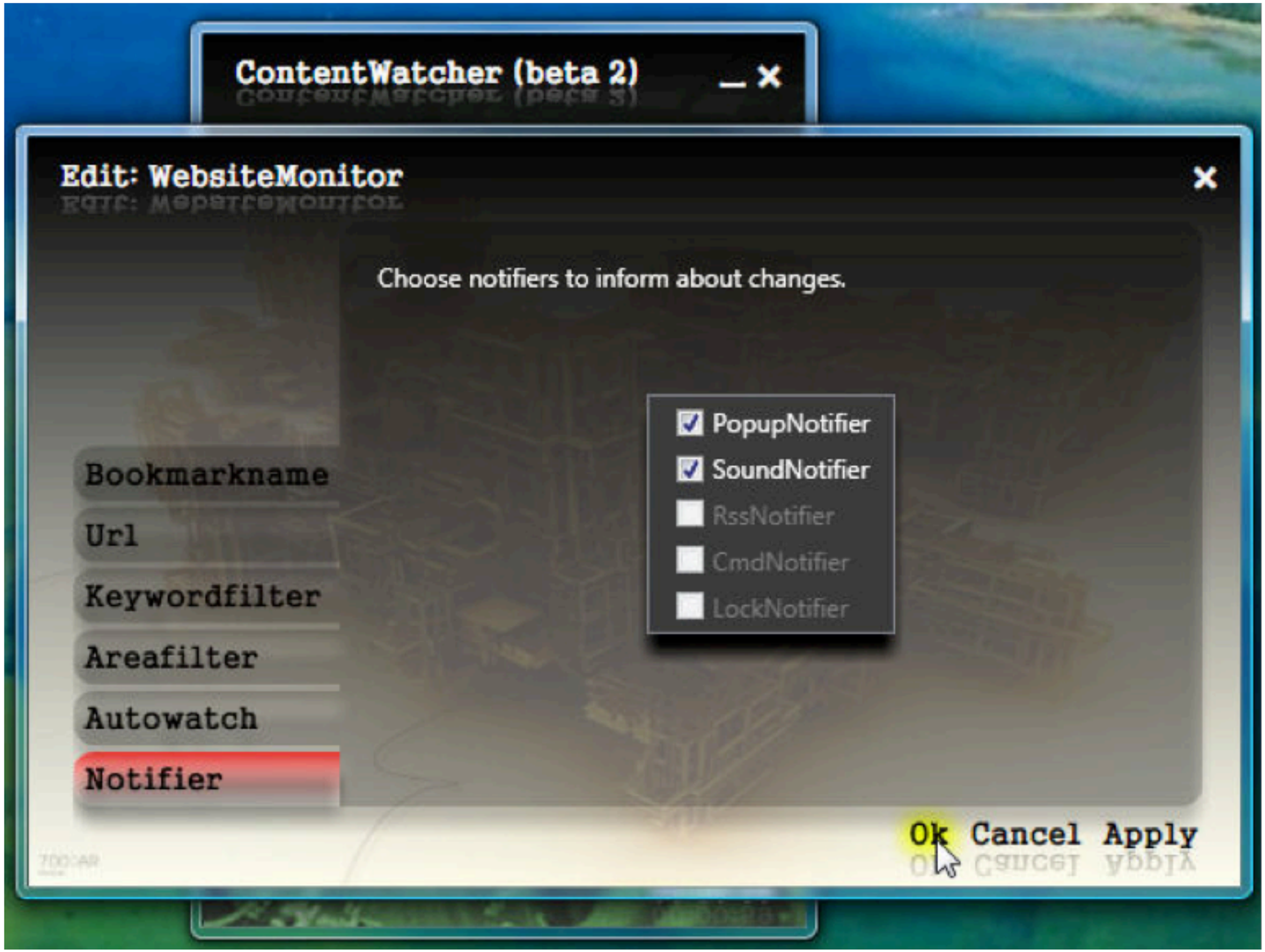


ContentWatcher

ContentWatcher

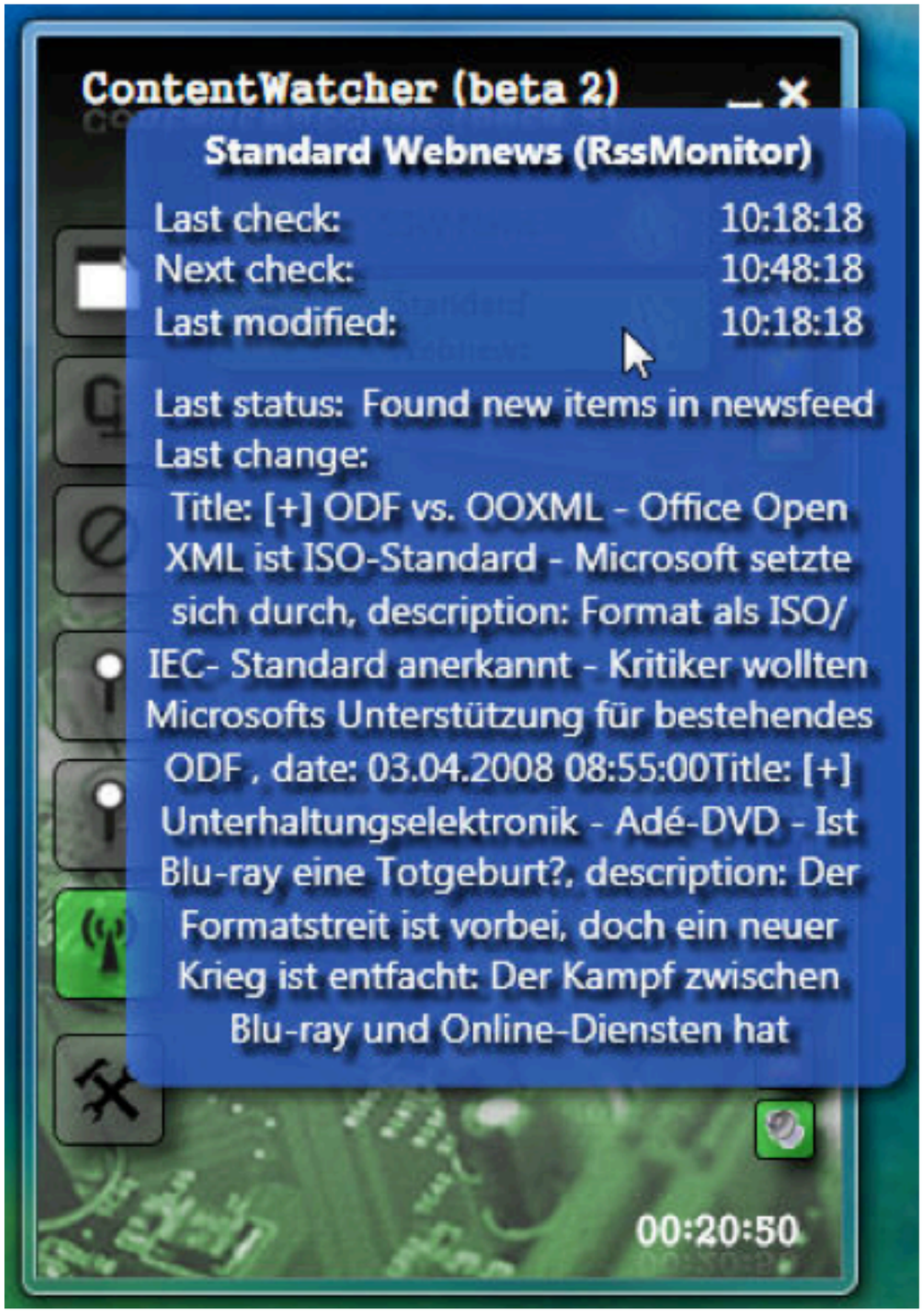


ContentWatcher



ContentWatcher

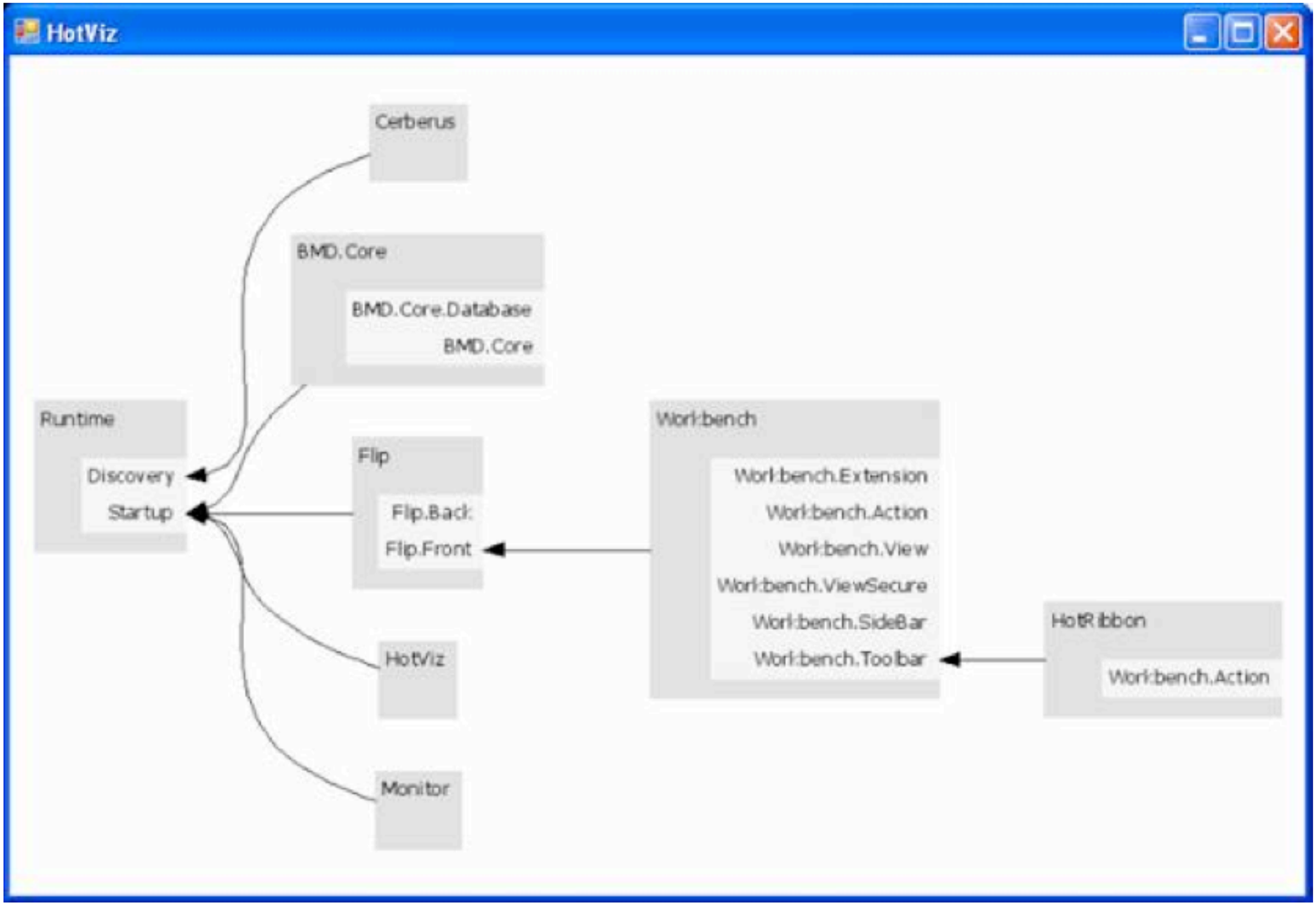
ContentWatcher



ContentWatcher

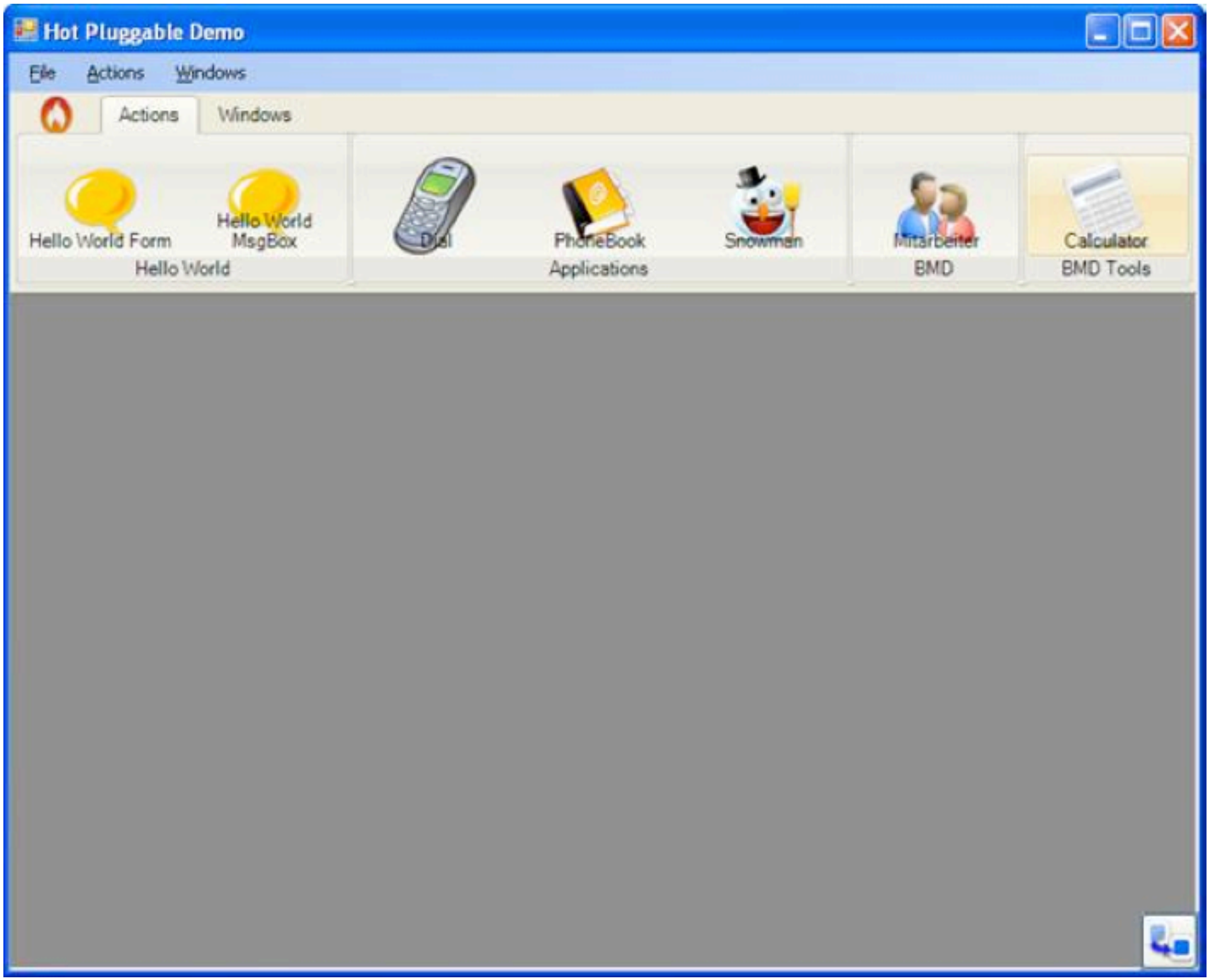
Organizer

Demo



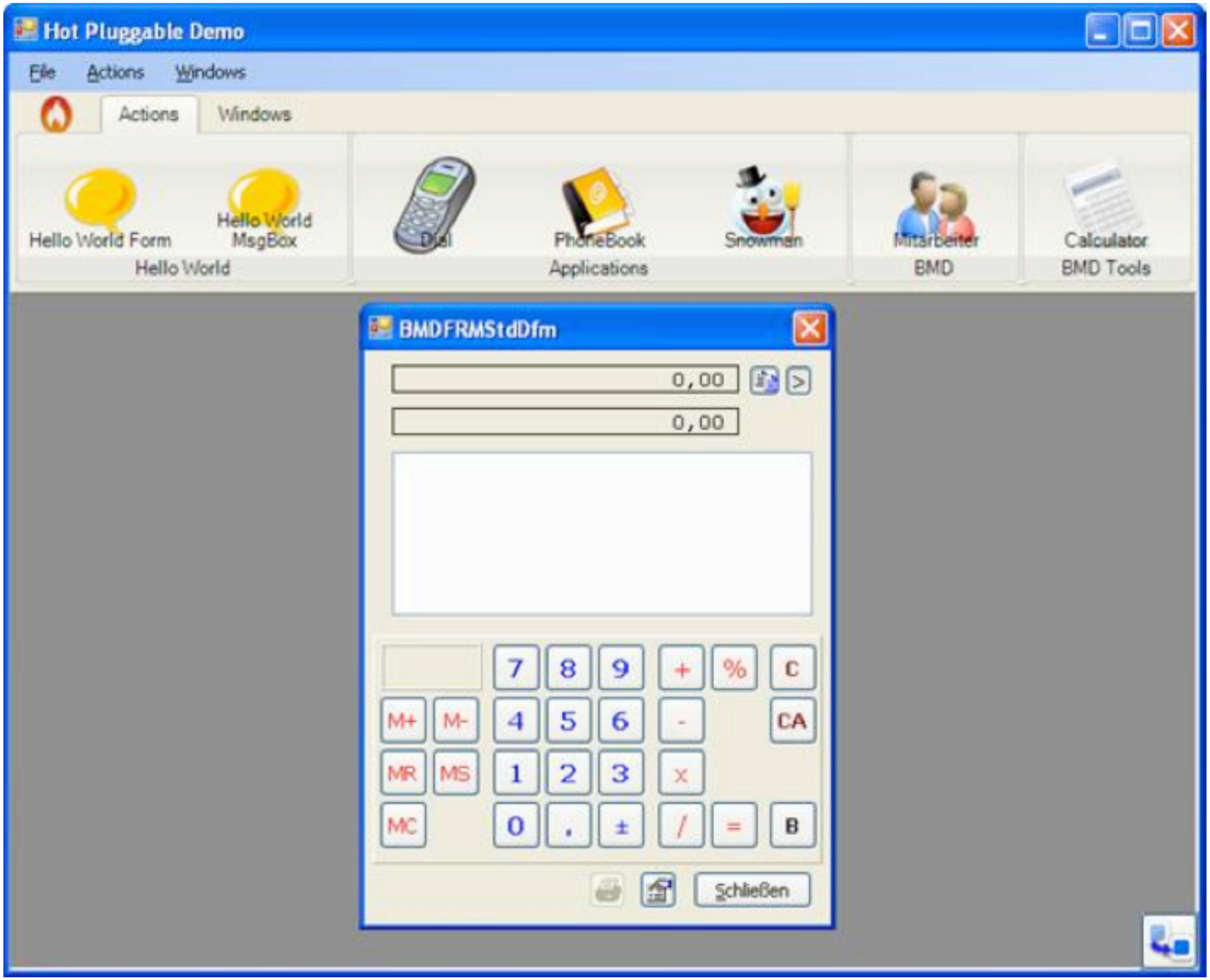
Organizer

Demo



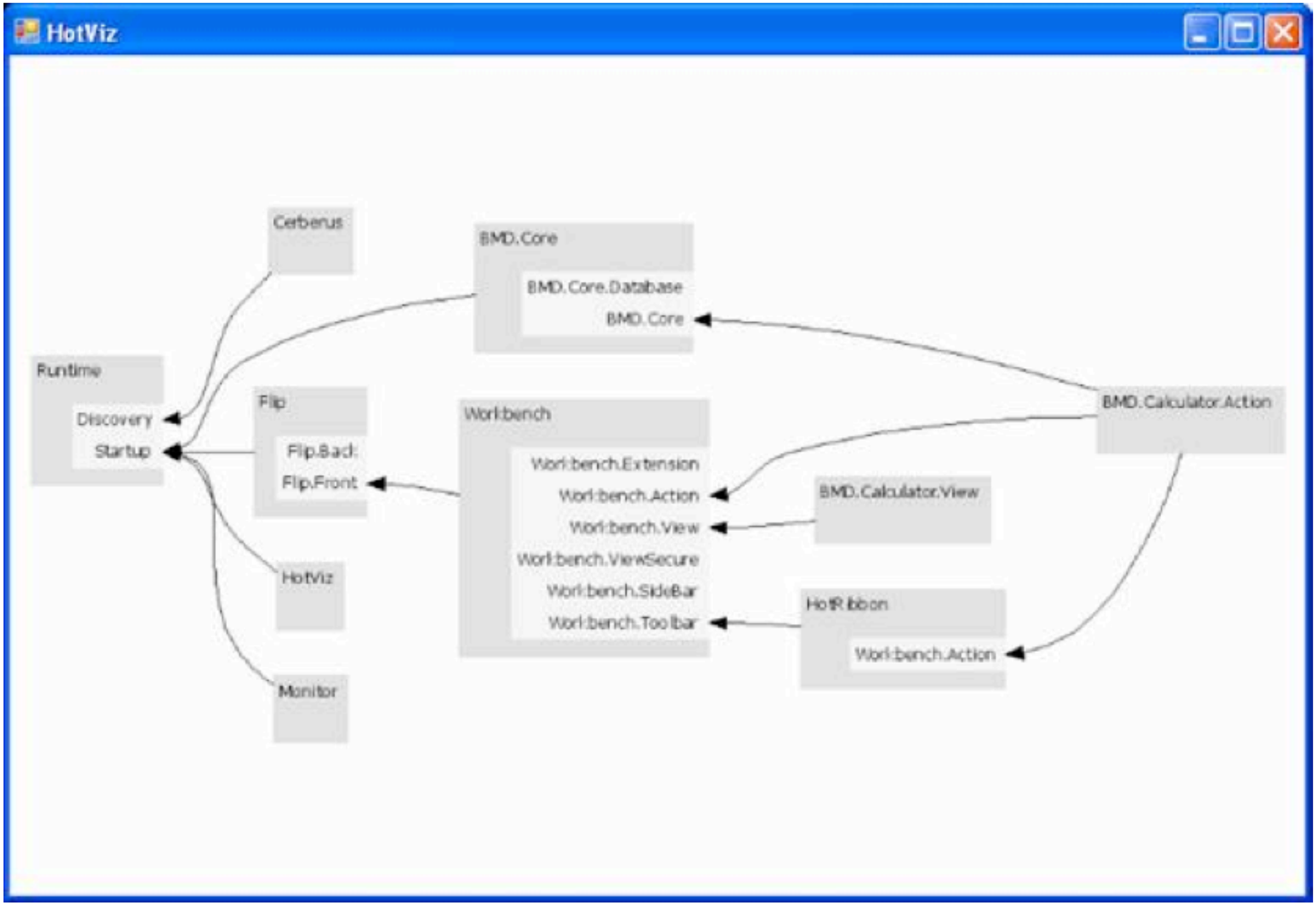
Organizer

Demo



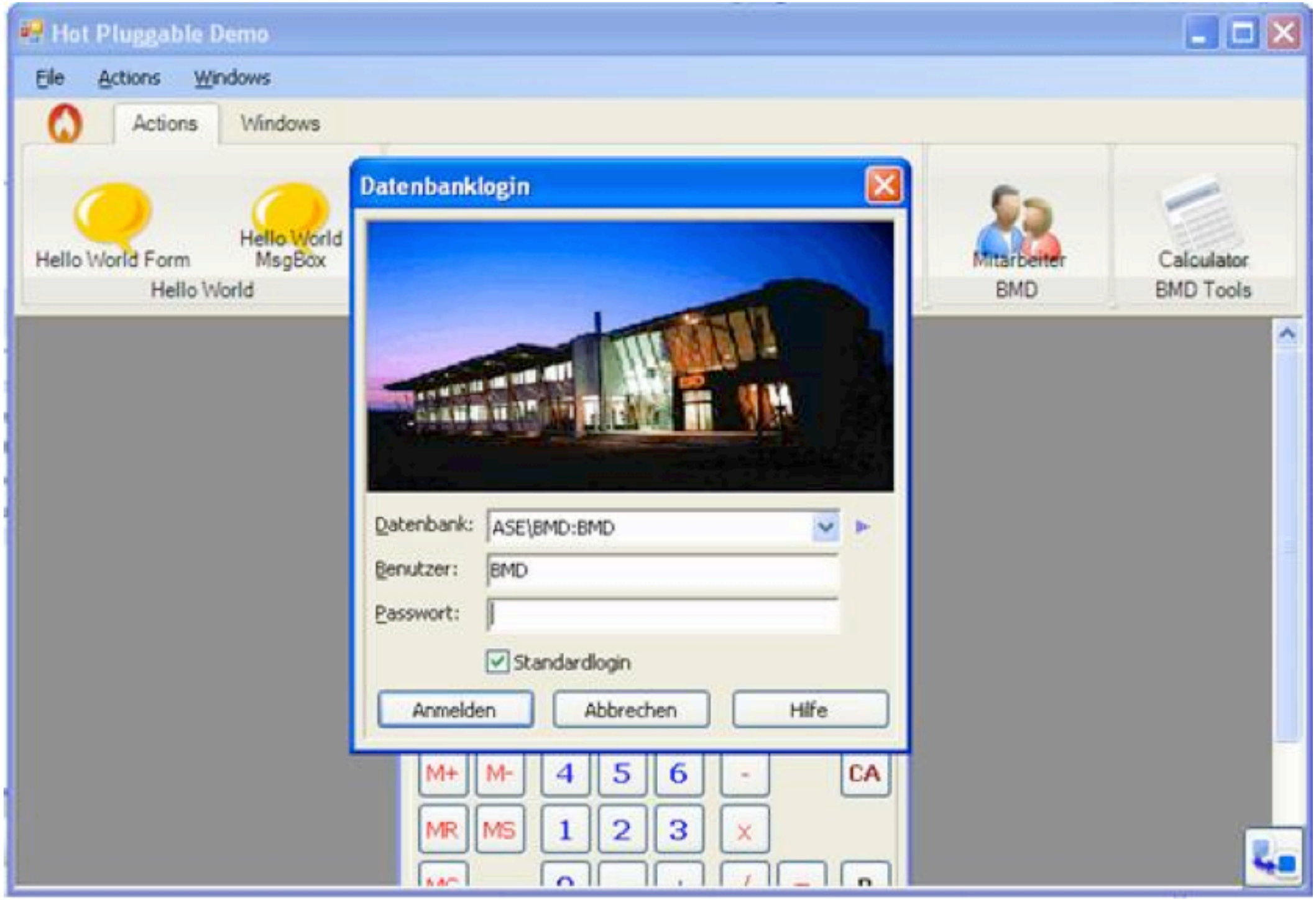
Organizer

Demo



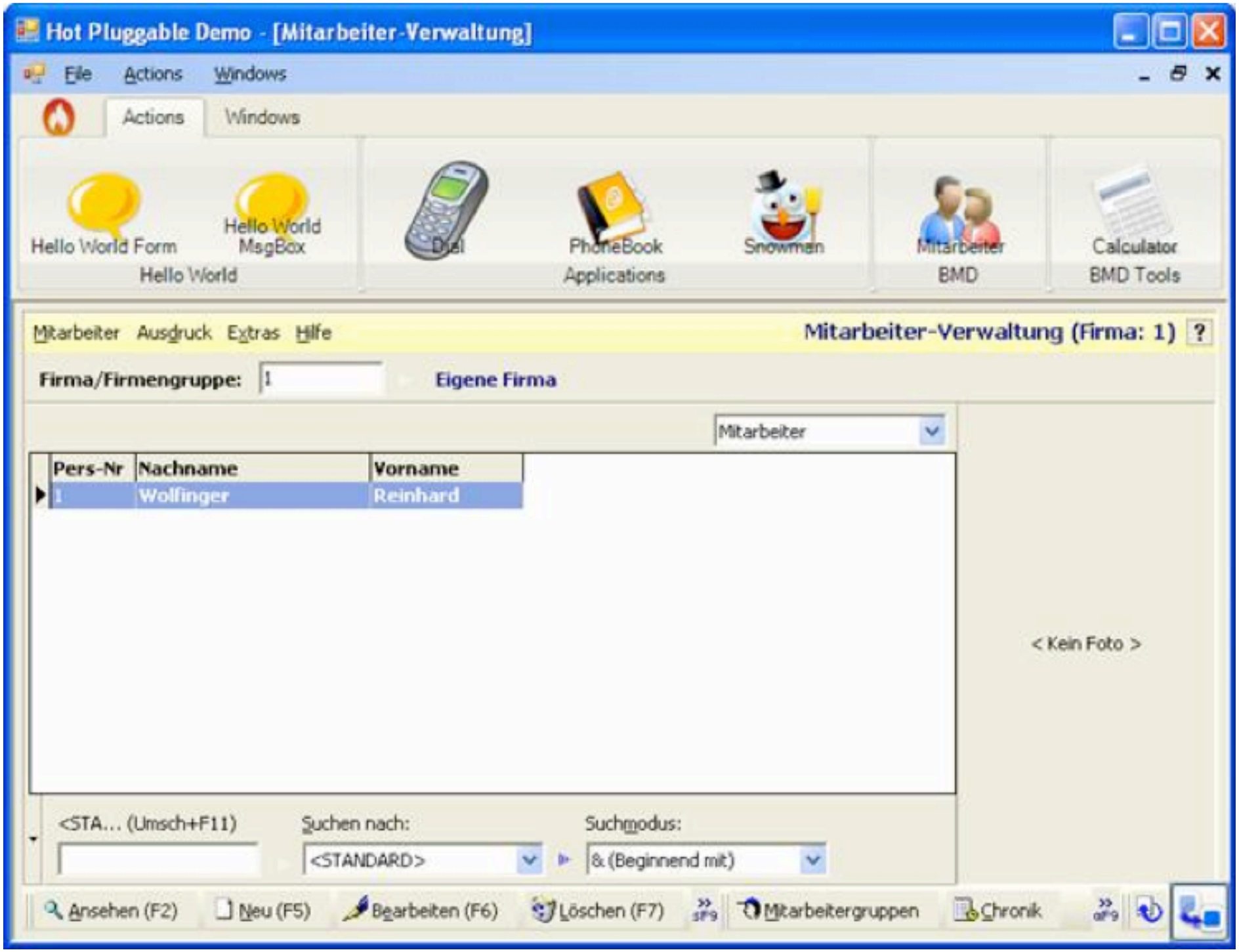
Organizer

Demo



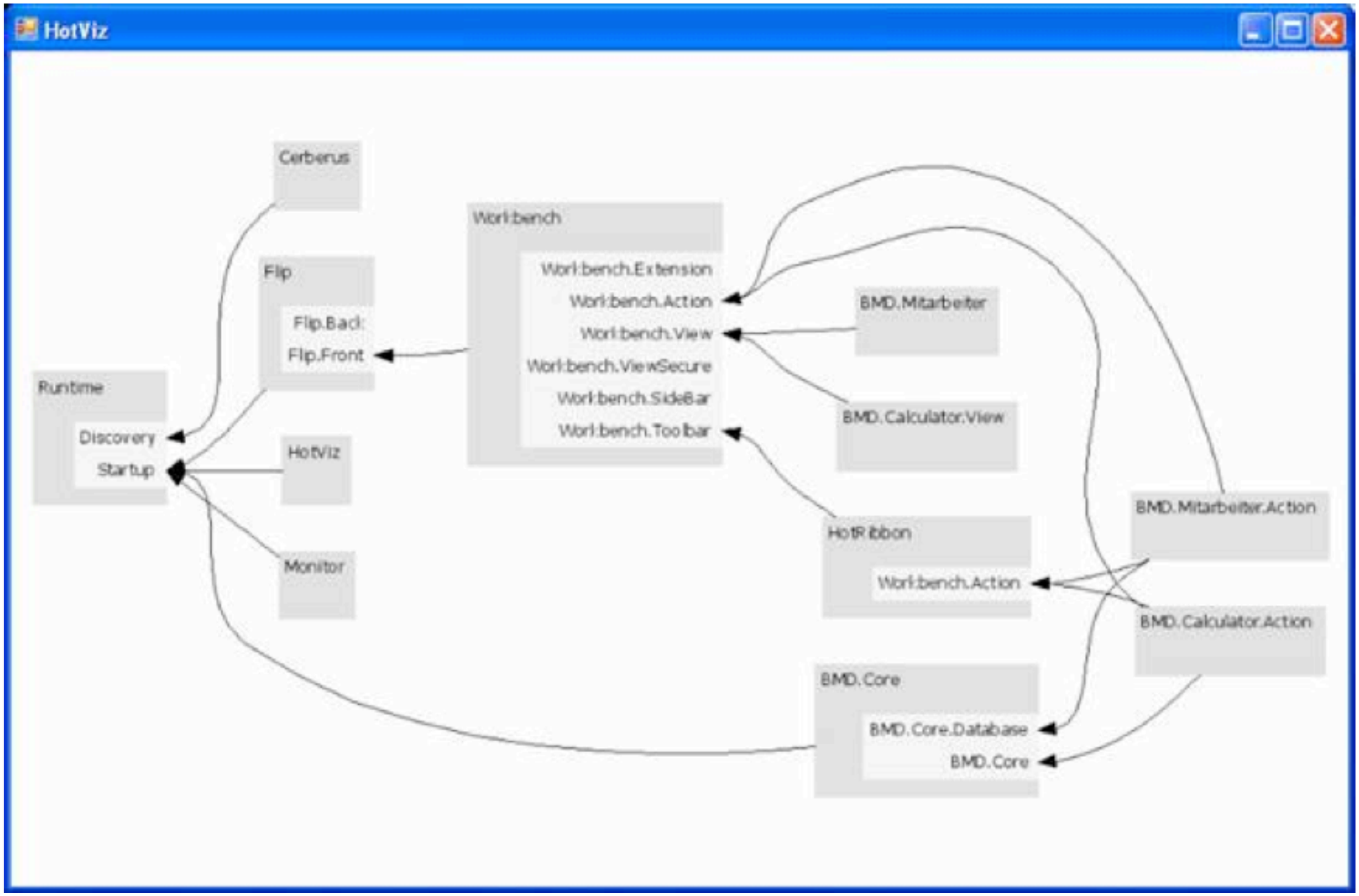
Organizer

Demo



Organizer

Demo



Projektplan 2009

Security Constraints

- Control who is allowed to contribute

Guidelines and Patterns

- Support architects build plug-in based applications

Scripting Language

- Specify and capture plug-in configurations

Component Contracts

- Verify quality of plug-ins

More BMD Pilot Projects