Programming and Simulating Robots with Microsoft Robotics Studio

Ben Axelrod

http://www.benaxelrod.com

http://www.corobot.net

Agenda

- What is it?
  - In a nutshell
  - “Supported” hardware
  - Simulator
  - Architecture
- Mobile Manipulator example
- Look at some code
- Running MSRS
- Downsides
- Demo (if time)

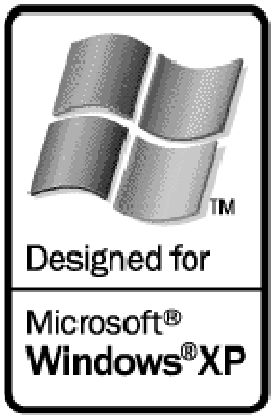
In a Nutshell…

- A distributed asynchronous service-oriented architecture (for robotics)
- CCR (Concurrency and Coordination Runtime)
  - Message oriented programming model
- DSS (Decentralized System Services)
  - service oriented application model
- Built on .NET
- A physics based simulator
- A visual programming language

Implications of .NET

- .NET is Microsoft's new development environment
- Choice of languages: C#, VB, C++, Python…
- Requires: Windows XP, CE, Vista
- Most robots will be "tethered"

eBox-2300



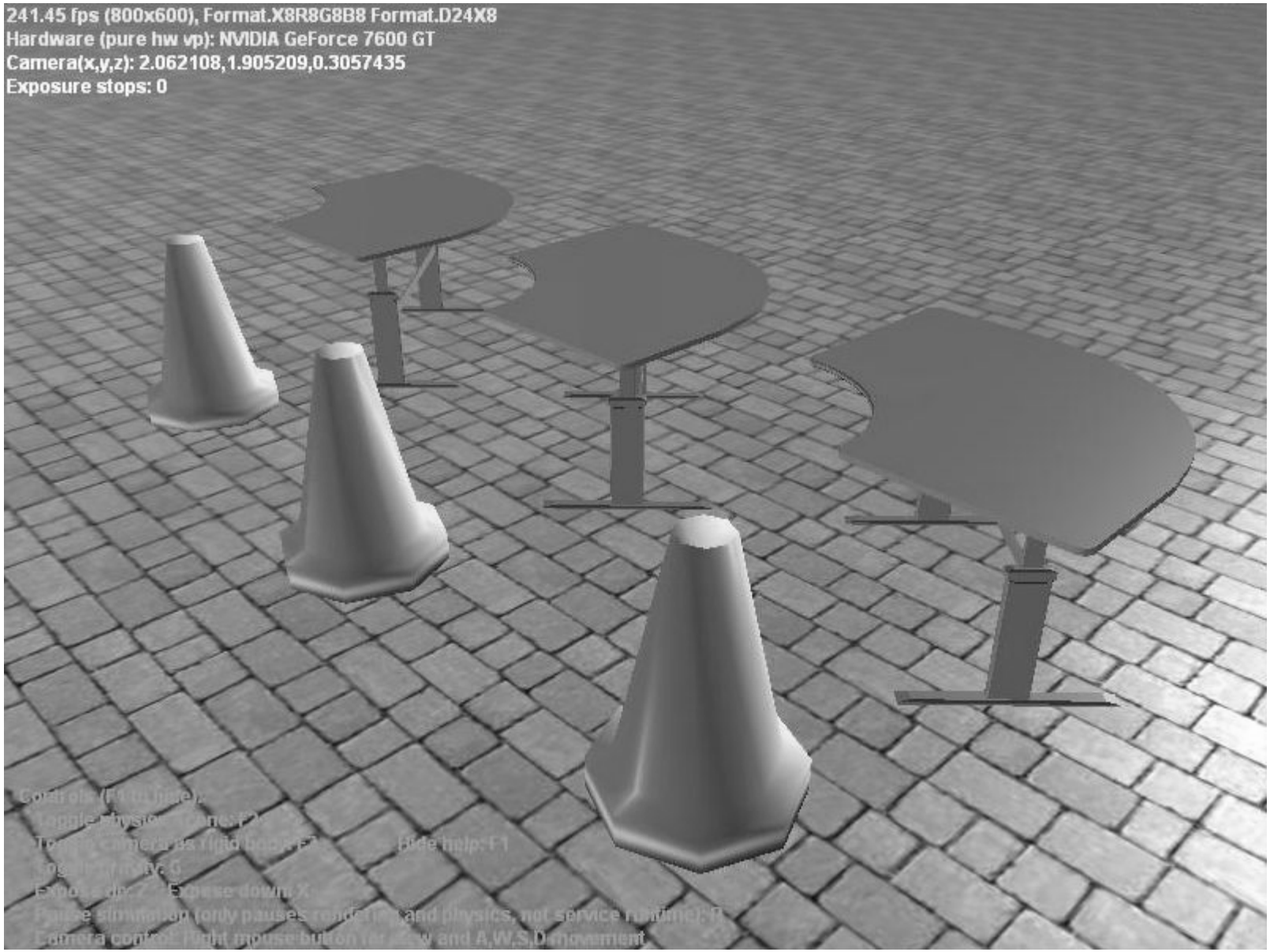
“Supported” Hardware

- Included with MSRS:
  - Lego RCX + NXT
  - Parallax BOE Bot
  - fischertechnik
  - iRobot Create
  - KHR-1
  - MobileRobots Pioneer P3DX
- Third party:
  - CoroWare CoroBot
  - Parallax Scribbler
  - Segway RMP
  - Robotics Connection Traxster + Stinger
  - Princeton PAVE UGC car
- Sensors:
  - SICK LRF
  - Webcams
  - IP Webcams
  - Phidgets
  - MS GPS

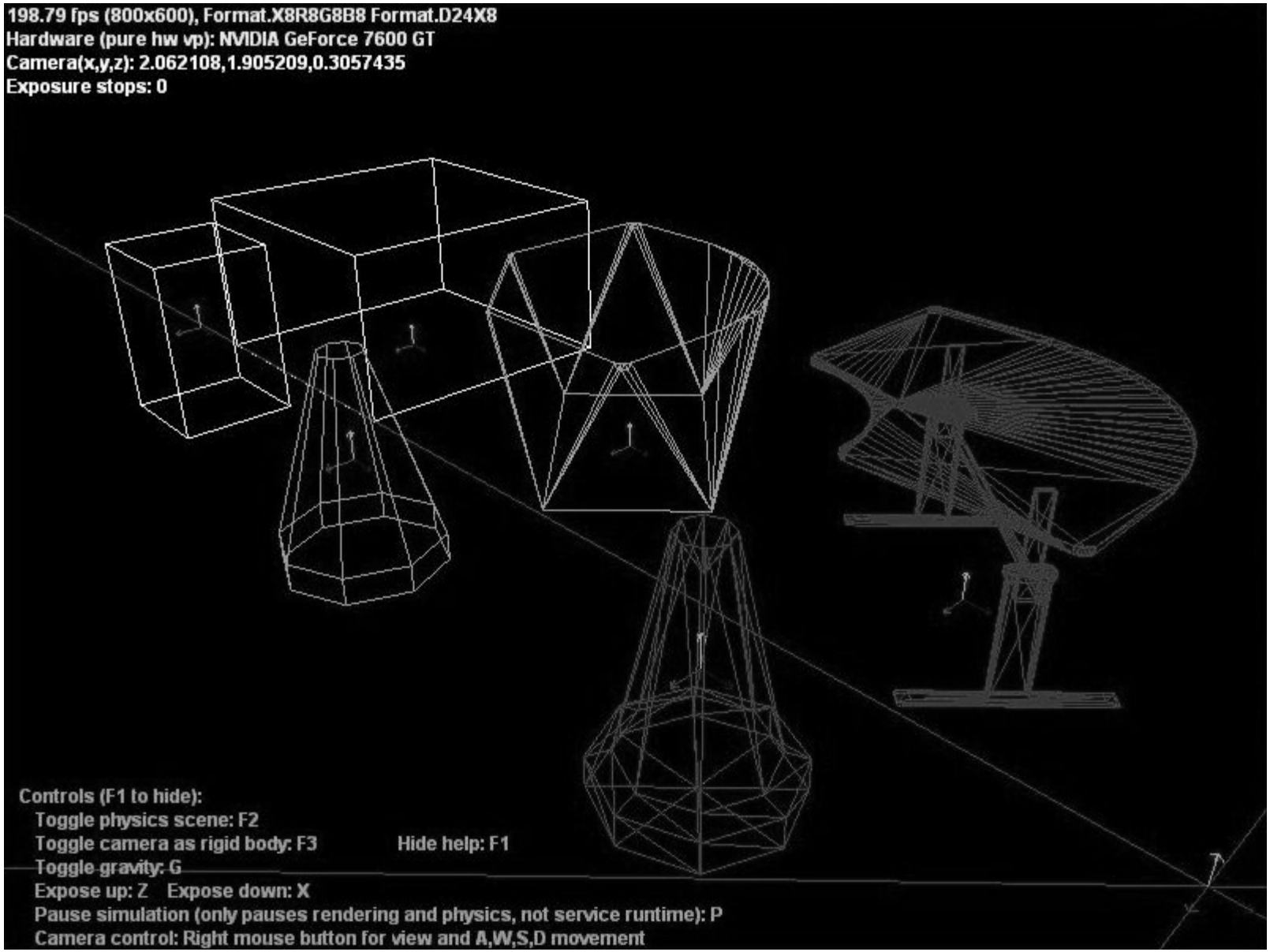
Simulator


Robotics Simulation Visualizer
57.50 fps (800x600), Format.X8R8G8B8 Format.D24X8
Hardware (pure hw vp): NVIDIA Quadro FX 540
Camera(x,y,z): 2.232749,0.7742481,-1.42905
Press F1 for help

Simulator

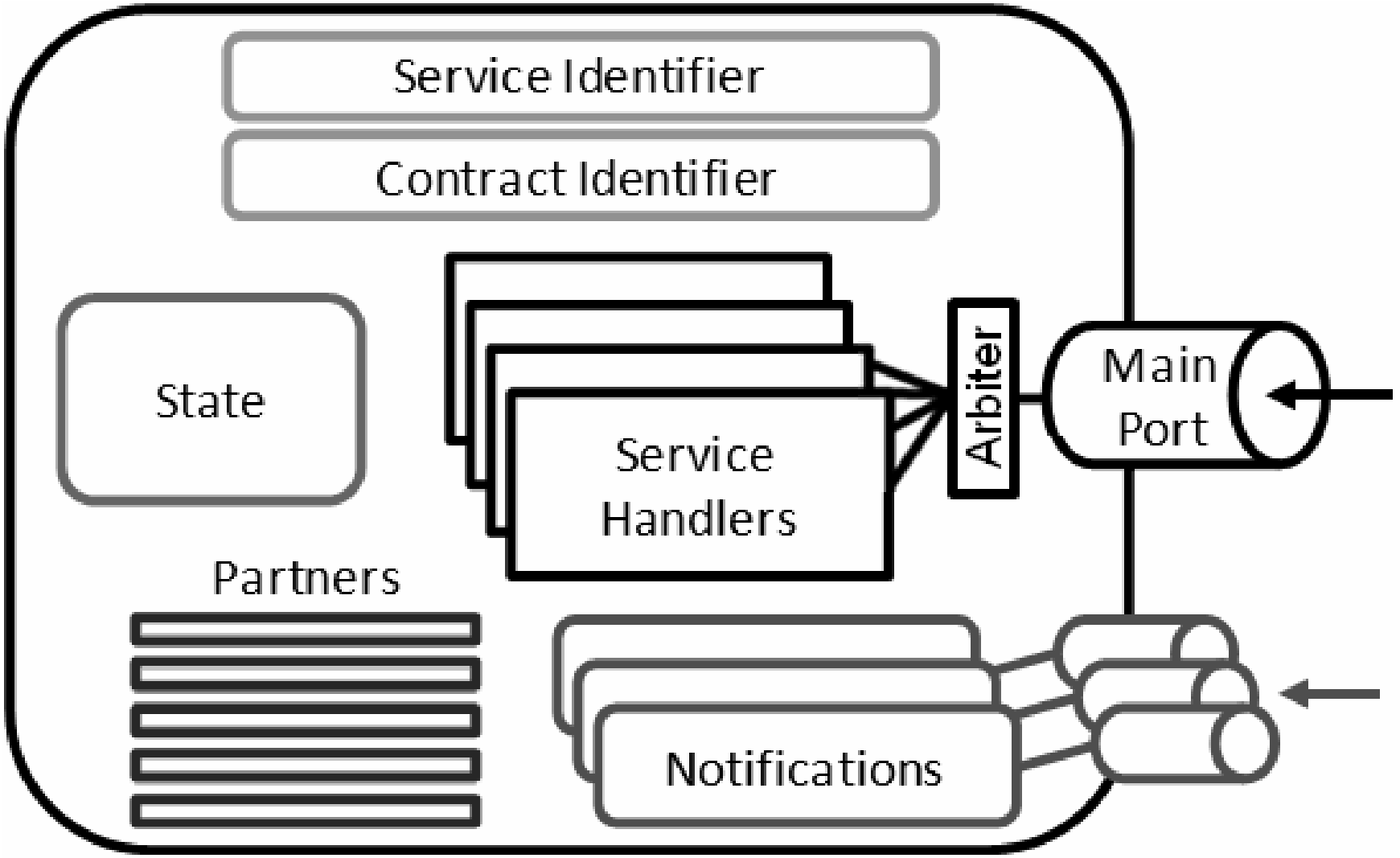

241.45 fps (800x600), Format.X8R8G8B8 Format.D24X8
Hardware (pure hw vp): NVIDIA GeForce 7600 GT
Camera(x,y,z): 2.062108,1.905209,0.3057435
Exposure stops: 0

Simulator

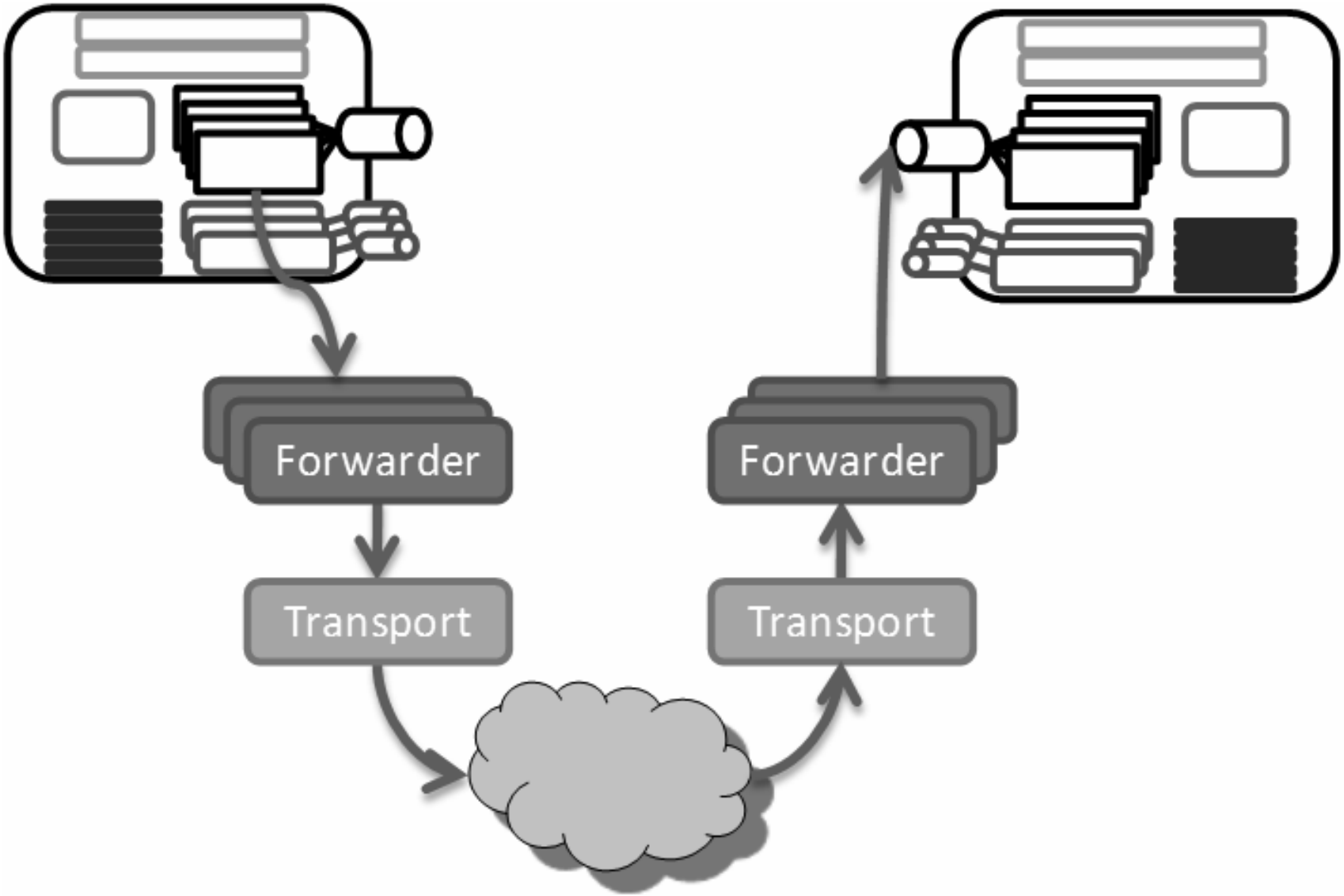

198.79 fps (800x600), Format.X8R8G8B8 Format.D24X8
Hardware (pure hw vp): NVIDIA GeForce 7600 GT
Camera(x,y,z): 2.062108,1.905209,0.3057435
Exposure stops: 0
Controls (F1 to hide):
Toggle physics scene: F2
Toggle camera as rigid body: F3
Hide help: F1
Toggle gravity: G
Expose up: Z Expose down: X
Pause simulation (only pauses rendering and physics, not service runtime): P
Camera control: Right mouse button for view and A,W,S,D movement

What is a service

- Separate state and behavior

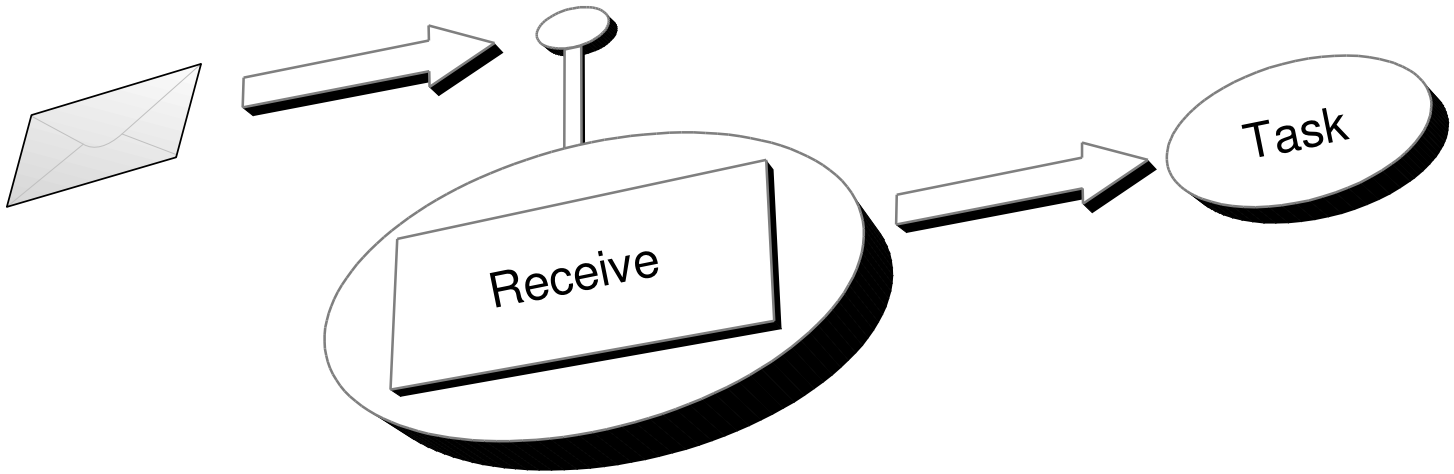


Message transport



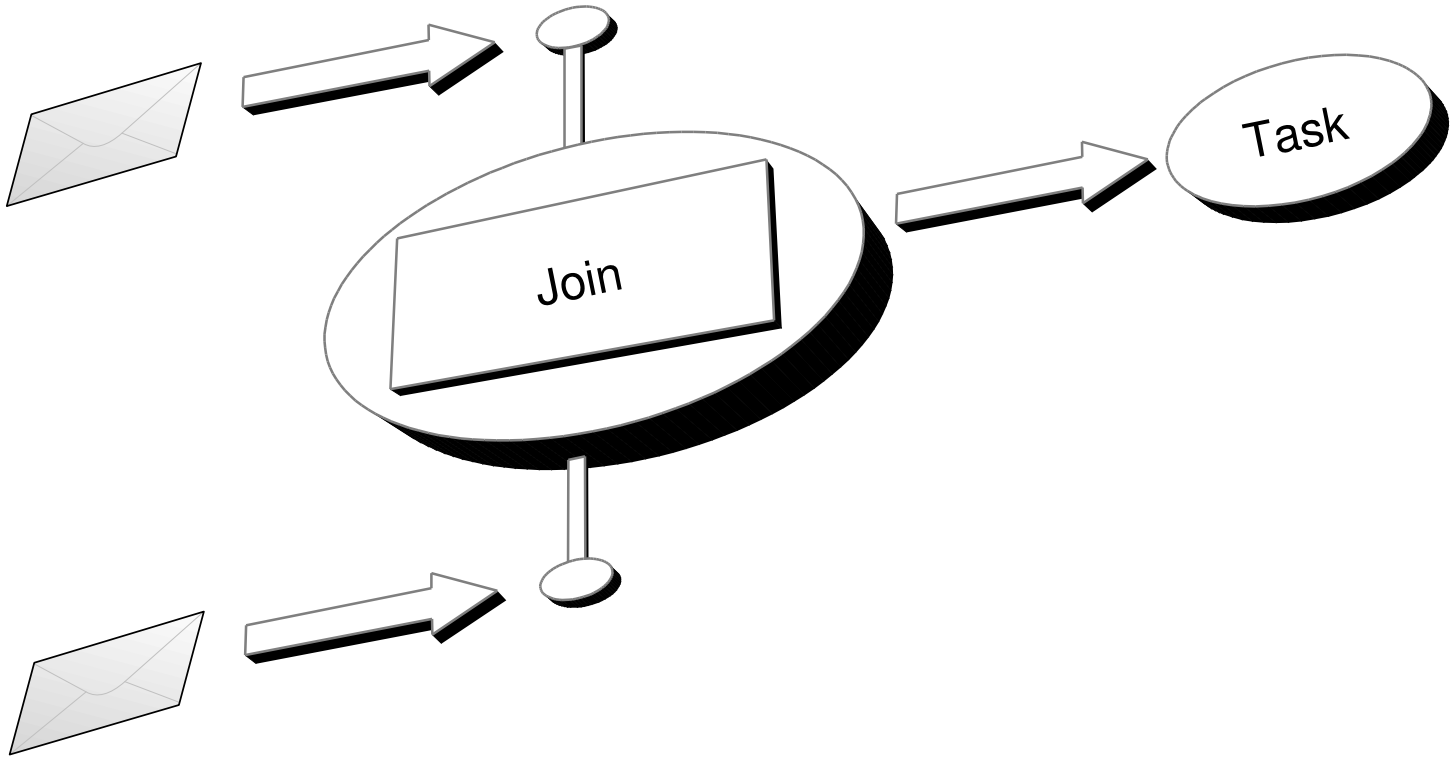
CCR

- Coordinating asynchronous tasks



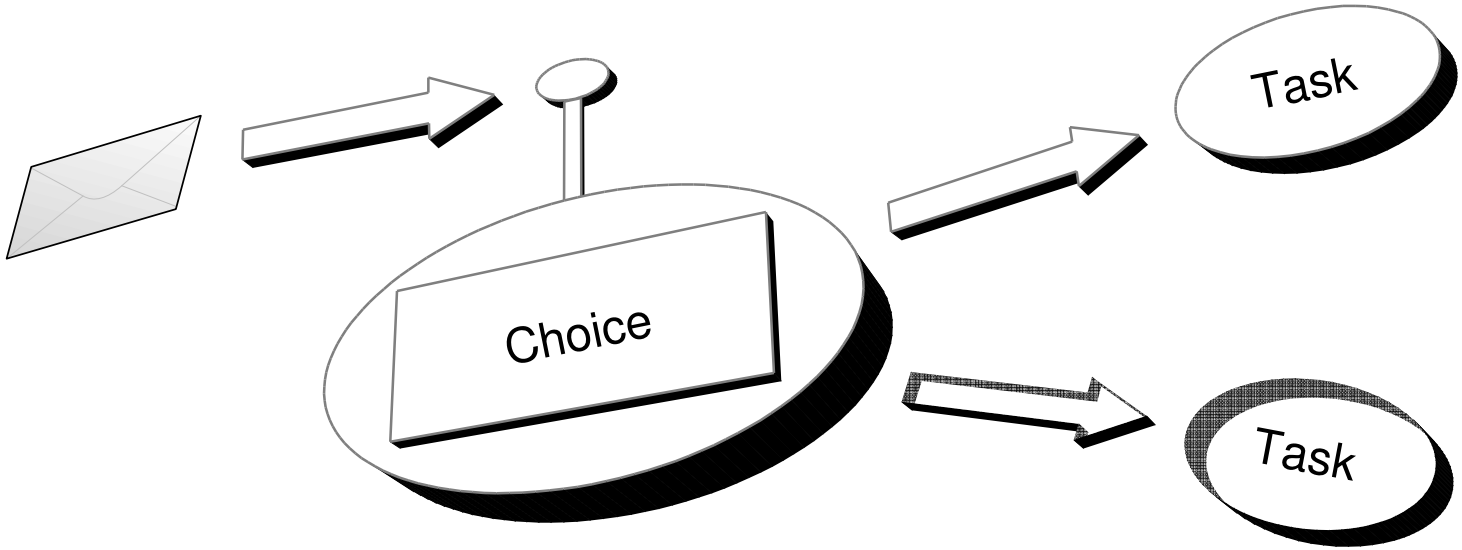
CCR

- Coordinating asynchronous tasks



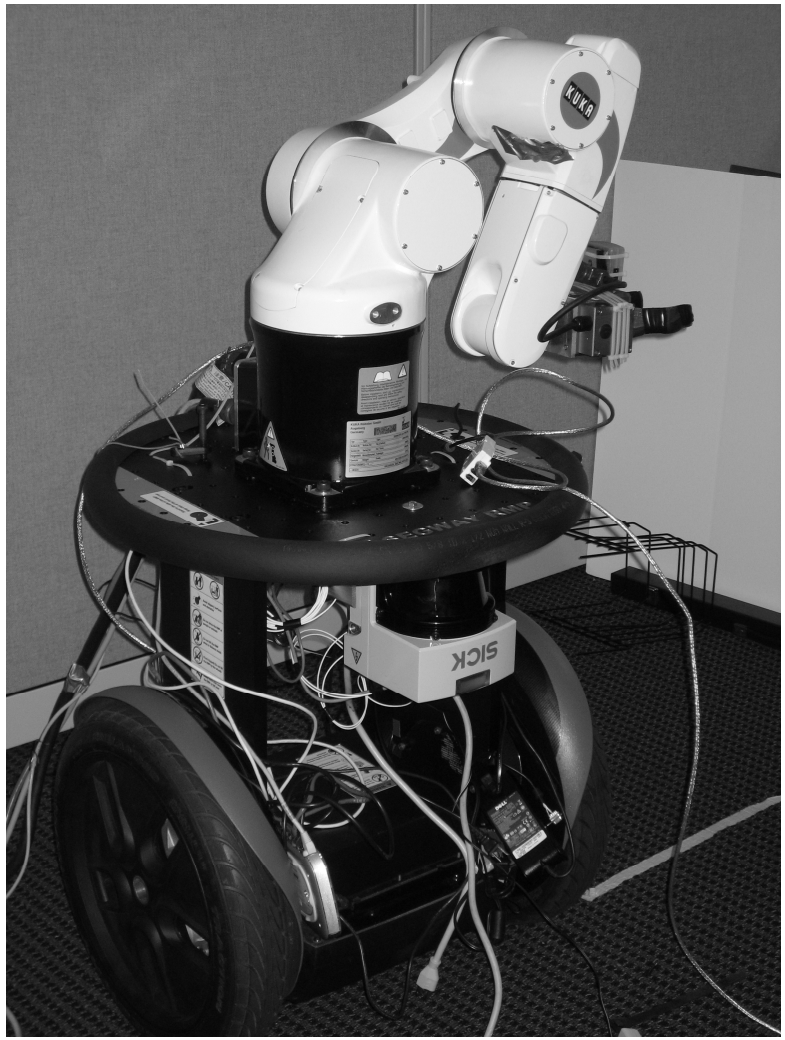
CCR

- Coordinating asynchronous tasks



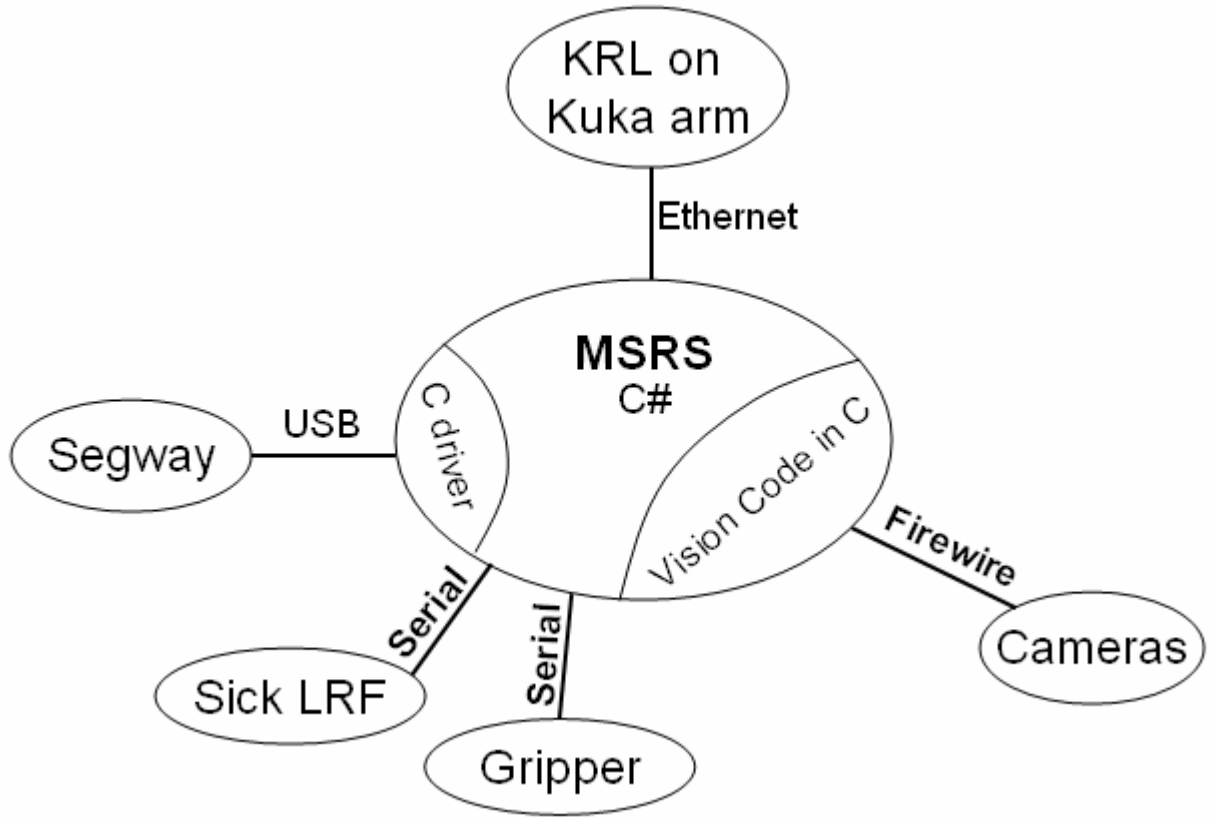
Example – mobile manipulation

- GT Class Project
- KUKA KR-5 sixx R650
- Schunk PG-70 parallel gripper
- Segway RMP 200
- SICK LMS 291
- Objective: Serve coffee

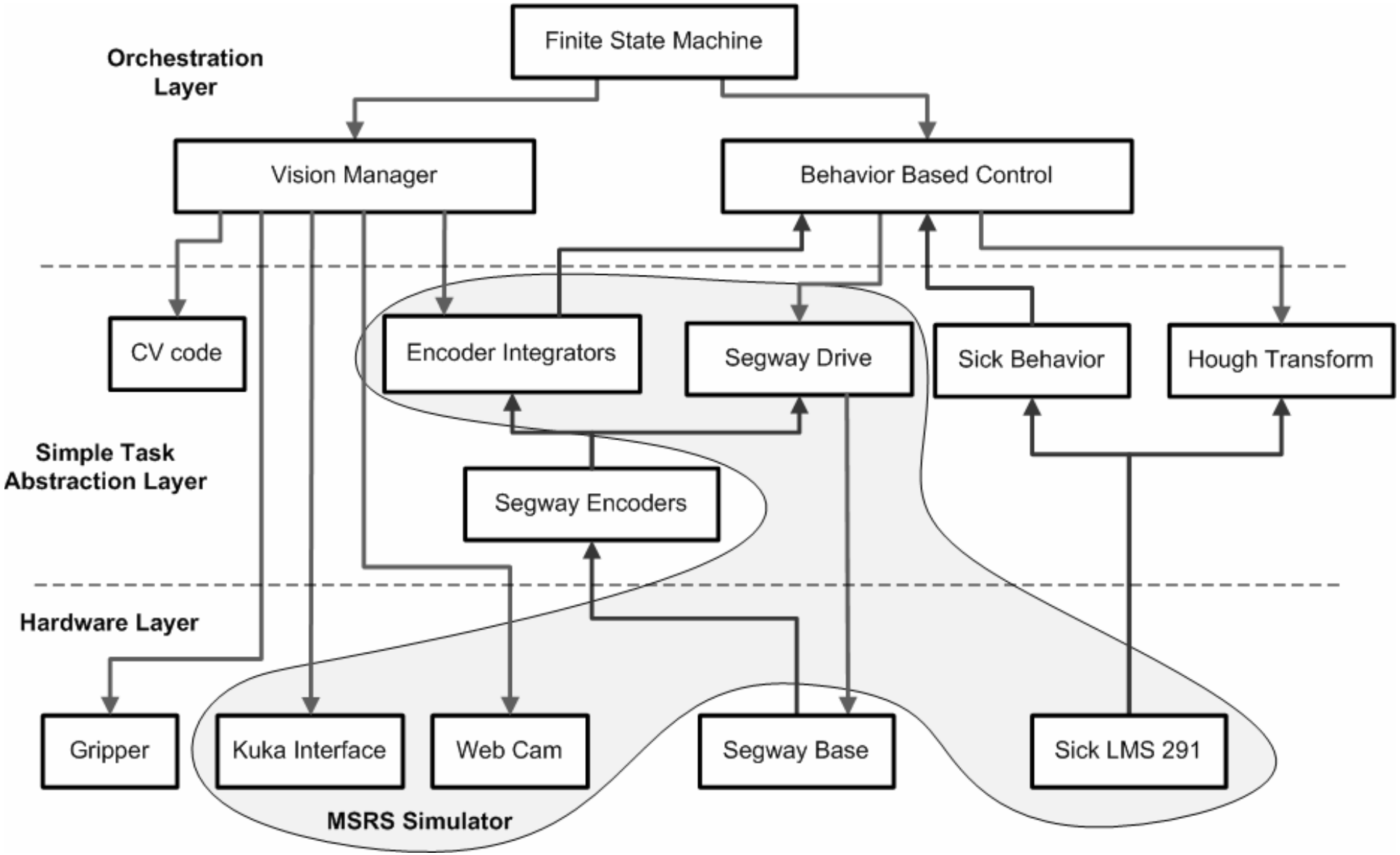


Example – mobile manipulation

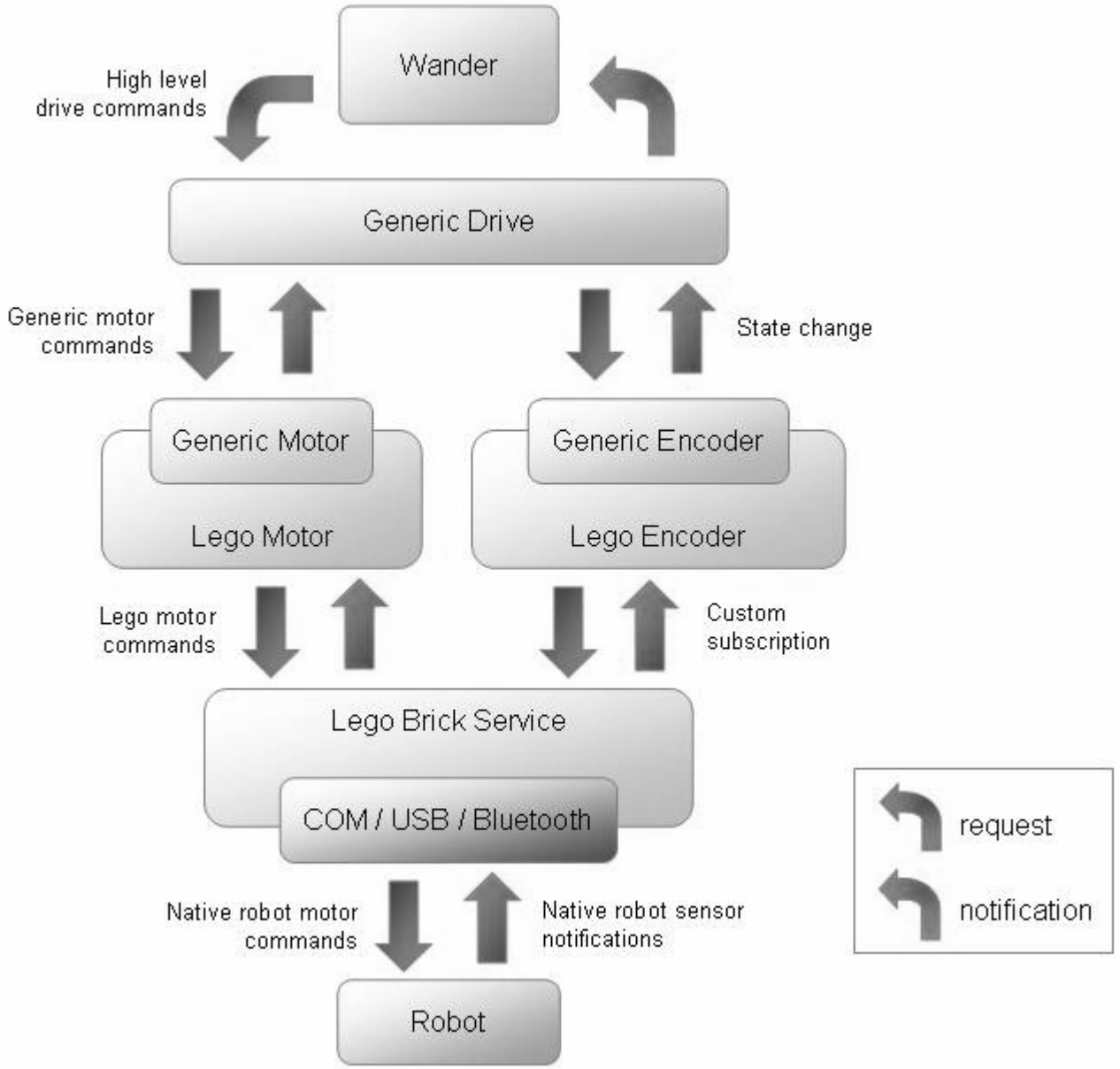
- MSRS used as ‘glue’ for robot system



Example – mobile manipulation



Standard Hierarchy



Code Example

- Custom line sensor
- We want to build a MSRS service that:
  - Has some simple state
  - Supports subscriptions
  - Can be calibrated with a special message

Code

- Contract and class

```
public sealed class Contract
{
    public const String Identifier = "http://schemas.tempuri.org/2008/03/linesensor.html";
}

[DataContract()]
public class LineSensorState
{
    [DataMember]
    public bool LineInView;

    [DataMember]
    public Position LineLocation;

    [DataMember]
    public DateTime TimeStamp;
}

[DataContract()]
public enum Position
{
    Left,
    Center,
    Right
}
```

Code

- Main port and message definitions

```
[ServicePort()]
public class LineSensorOperations : PortSet<
    DsspDefaultLookup,
    DsspDefaultDrop,
    Get,
    Replace,
    Subscribe,
    Calibrate>
{ }

public class Get : Get<GetRequestType, PortSet<LineSensorState, Fault>>
{ }

public class Replace : Replace<LineSensorState, PortSet<DefaultReplaceResponseType, Fault>>
{ }

public class Subscribe : Subscribe<SubscribeRequestType, PortSet<SubscribeResponseType, Fault>>
{ }

public class Calibrate : Update<CalibrateRequestType, PortSet<DefaultUpdateResponseType, Fault>>
{ }

[DataContract()]
public class CalibrateRequestType
{ }
```

Code

- Class

```
[DisplayName("LineSensor")]
[Description("The LineSensor Service")]
[Contract(Contract.Identifier)]
public class LineSensorService : DsspServiceBase
{
    private LineSensorState _state = new LineSensorState();

    [ServicePort("/LineSensor", AllowMultipleInstances=false)]
    private LineSensorOperations _mainPort = new LineSensorOperations();

    [Partner("SubMgr",
             Contract = submgr.Contract.Identifier,
             CreationPolicy = PartnerCreationPolicy.CreateAlways)]
    private submgr.SubscriptionManagerPort _submgrPort = new submgr.SubscriptionManagerPort();

    public LineSensorService(DsspServiceCreationPort creationPort) : base(creationPort) { }

    protected override void Start()
    {
        base.Start();

        if (_state == null)
        {
            _state = new LineSensorState();
            _state.LineInView = false;
            _state.TimeStamp = DateTime.Now;
        }
        //do custom setup
    }
```

Code

- Message handlers part 1

```
[ServiceHandler(ServiceHandlerBehavior.Concurrent)]
public virtual IEnumerator<ITask> GetHandler(Get get)
{
    get.ResponsePort.Post(_state);
    yield break;
}

[ServiceHandler(ServiceHandlerBehavior.Exclusive)]
public virtual IEnumerator<ITask> ReplaceHandler(Replace msg)
{
    _state = msg.Body;

    base.SendNotification<Replace>(_submgrPort, _state);

    msg.ResponsePort.Post(DefaultReplaceResponseType.Instance);
    yield break;
}
```

Code

- Message handlers part 2

```
[ServiceHandler(ServiceHandlerBehavior.Concurrent)]
public virtual IEnumerator<ITask> SubscribeHandler(Subscribe subscribe)
{
    yield return Arbiter.Choice(
        SubscribeHelper(_submgrPort, subscribe.Body, subscribe.ResponsePort),
        delegate(SuccessResult success)
        {
            base.SendNotification<Replace>(_submgrPort, subscribe.Body.Subscriber, _state);
        },
        delegate(Exception e)
        {
            LogError(null, "Subscribe failed", e);
        }
    );

    yield break;
}
```

Code

- Callback

```
void HandleSensorCallback(bool leftSensor, bool rightSensor)
{
    LineSensorState newState = new LineSensorState();
    newState.TimeStamp = DateTime.Now;

    //do sensor logic
    newState.LineInView = leftSensor | rightSensor;
    if (leftSensor && rightSensor) newState.LineLocation = Position.Center;
    else if (leftSensor) newState.LineLocation = Position.Left;
    else if (rightSensor) newState.LineLocation = Position.Right;

    Replace rmsg = new Replace();
    rmsg.Body = newState;
    _mainPort.Post(rmsg);
}
```

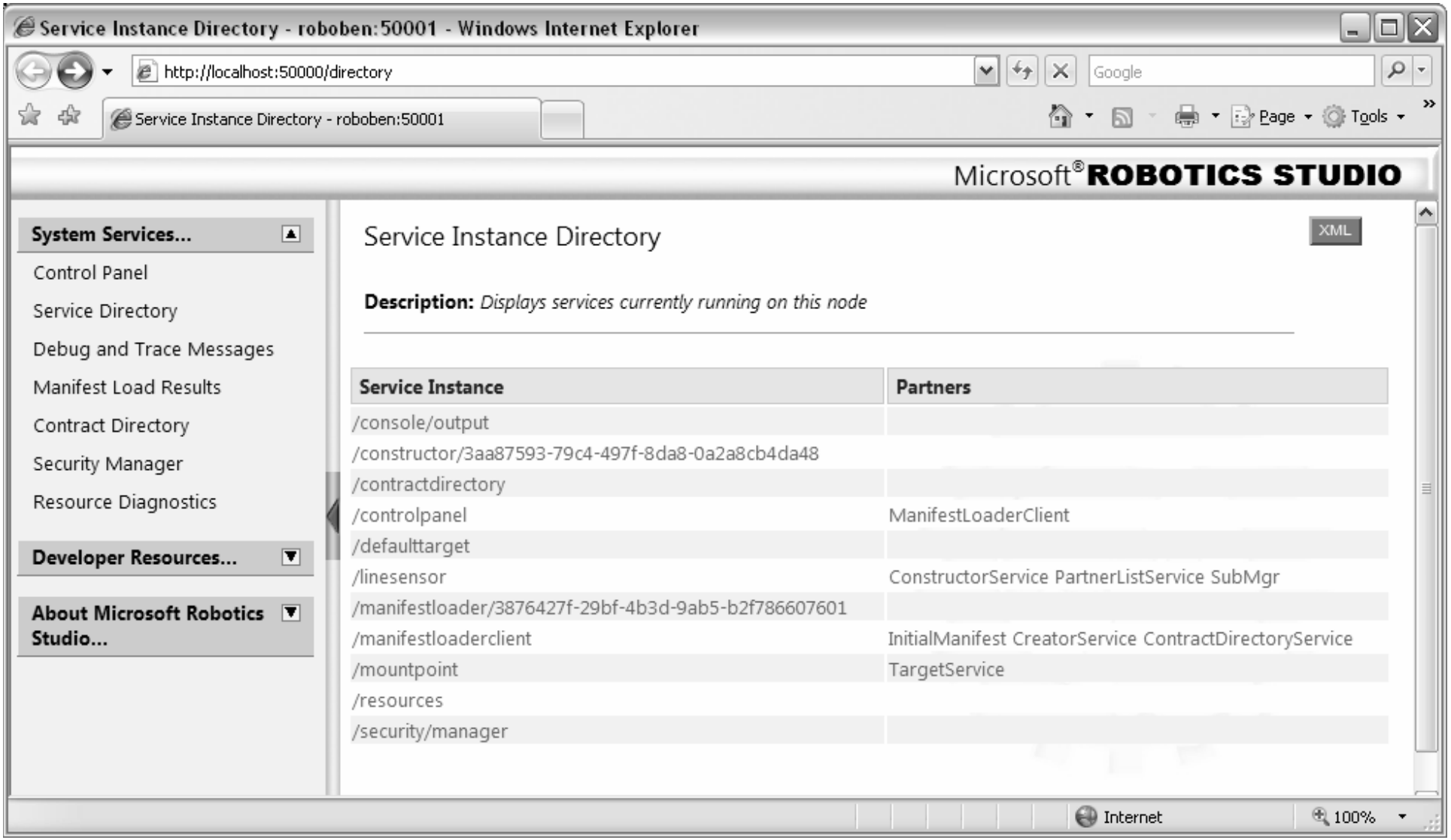
Usage

- Manifest

```
<?xml version="1.0" ?>
<Manifest
    xmlns="http://schemas.microsoft.com/xw/2004/10/manifest.html"
    xmlns:dssp="http://schemas.microsoft.com/xw/2004/10/dssp.html">
    <CreateServiceList>

        <ServiceRecordType>
            <dssp:Contract>http://schemas.tempuri.org/2008/03/linesensor.html</dssp:Contract>
        </ServiceRecordType>

    </CreateServiceList>
</Manifest>
```

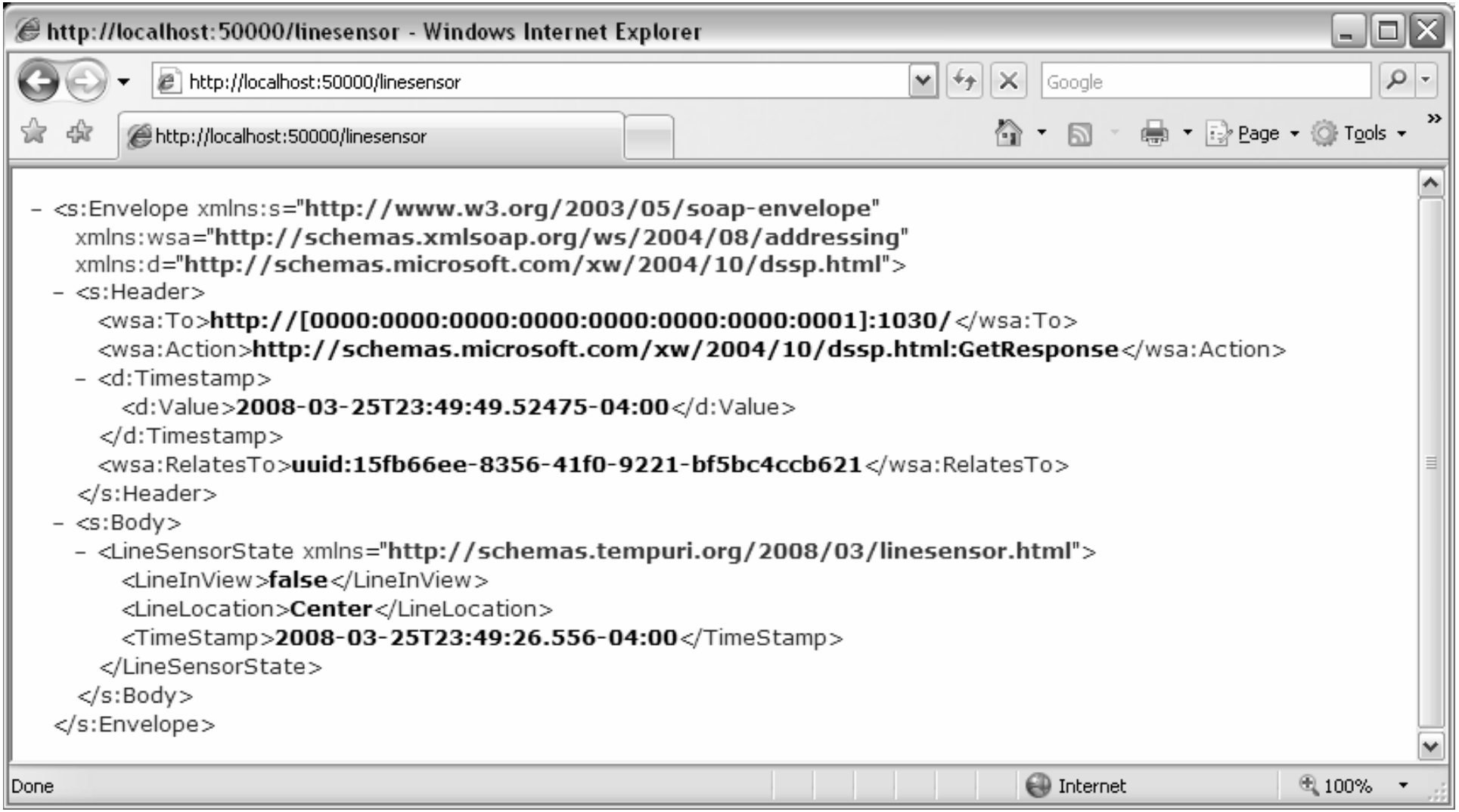
Usage

- Contract directory



Usage

- XML state



Synchronous Tasks – Bad

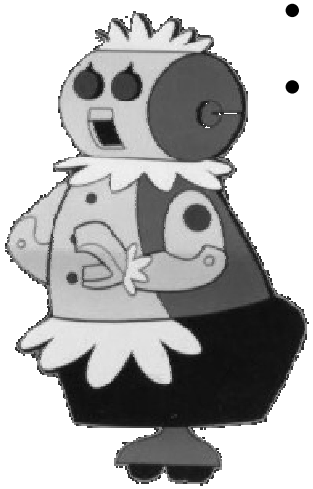
```
protected override void Start()
{
    base.Start();

    DrawSquare();
}

void DrawSquare()
{
    double ONE_METER = 1.0;
    double FULL_POWER = 1.0;
    double QUARTER_TURN = 90.0;

    System.Threading.Thread.Sleep(1000);

    for (int i = 0; i < 4; i++)
    {
        _drivePort.DriveDistance(new DriveDistanceRequest(ONE_METER, FULL_POWER));
        _drivePort.RotateDegrees(new RotateDegreesRequest(QUARTER_TURN, FULL_POWER));
    }
}
```

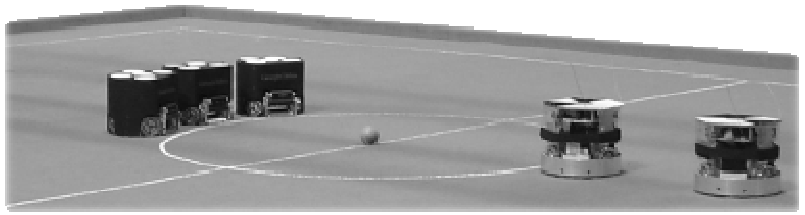
Synchronous Tasks - Good

```
protected override void Start()
{
    base.Start();

    SpawnIterator(DrawSquare);
}

IEnumerator<ITask> DrawSquare()
{
    double ONE_METER = 1.0;
    double FULL_POWER = 1.0;
    double QUARTER_TURN = 90.0;

    yield return Arbiter.Receive(false, TimeoutPort(2000), delegate(DateTime t) { });

    for (int i = 0; i < 4; i++)
    {
        yield return Arbiter.Receive<DefaultUpdateResponseType>(false,
             _drivePort.DriveDistance(new DriveDistanceRequest(ONE_METER, FULL_POWER)),
             delegate(DefaultUpdateResponseType rsp)
             { }
        );

        yield return Arbiter.Receive<DefaultUpdateResponseType>(false,
            _drivePort.RotateDegrees(new RotateDegreesRequest(QUARTER_TURN, FULL_POWER)),
            delegate(DefaultUpdateResponseType rsp)
            { }
        );
    }

    yield break;
}
```

Why I Like It

- .NET is great
- CCR nice for asynchronous code
  - MySpace uses it
- It's Microsoft
  - MS on a robot is laughable
    - Kuka uses it
  - This is the direction robotics is going
    - One home computer controlling all robots
    - Everything now is off-board computation





Downsides

- Steep learning curve
- Lots of boiler-plate code
- Can start services in many ways
  - Partnerships break down with large numbers
- Requires a fairly fast computer
  - I want to put code on a Gumstix
- Can be difficult to debug sometimes
- No contract inheritance
- Users must faithfully implement contracts
- Lots of pre-existing robotics code for Linux
- No distinction between “in” and “out” messages

Competitors

Platform	**Type**	
CLARAty (NASA JPL)	Platform	Open source
ERSP (Evolution Robotics)	Platform	Commercial
Microsoft Robotics Studio	Platform	Commercial or Free
iRobot AWARE	Platform	Commercial
OROCOS	Machine and robot control libraries	Open source & Free
Skilligent	Robot learning add-on	Commercial
URBI	Platform	Commercial
Webots	Simulation environment	Commercial
Player, Stage, Gazebo	Platform	Open Source & Free
OpenJAUS	Platform	Open source
Saphira (Mobile Robotics)		Commercial or Free
ORCA (Toshiba)		
DROS		Open Source & Free

http://www.linuxdevices.com/articles/AT5739475111.html

Resources

- Microsoft Robotics
  - http://www.microsoft.com/robotics
  - The team Blog, product downloads and community-support newsgroup are linked from this main page
  - Wiki: http://channel9.msdn.microsoft.com
- CoroWare, Inc.
  - Corporate: http://www.coroware.com
  - ClassPack demo: http://support.coroware.com/forums

Programming Microsoft Robotics Studio
by Sara Morgan

Professional Microsoft Robotics Studio
By Kyle Johns, Trevor Taylor

- RoboChamps: http://robochamps.com