

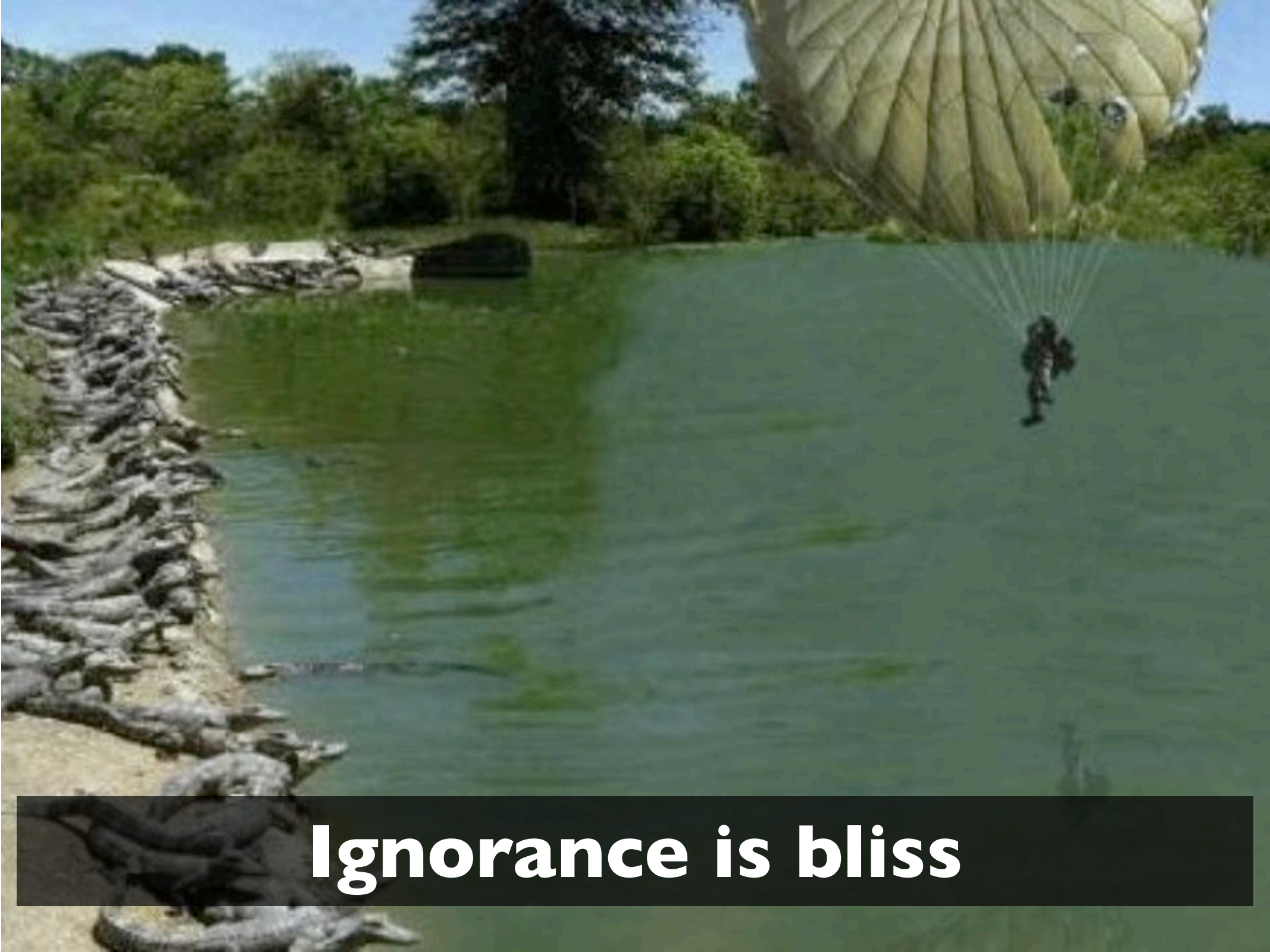
I will backup my laptop every day  
I will backup my laptop every day  
I will backup my laptop every day  
I will backup my laptop every day

# OSGi Lessons Learned: Best and Worst Practices

Martin Lippert (SpringSource, a division of VMware)

Special thanks to Chris Aniszczyk (Red Hat), Jeff McAffer (EclipseSource),  
Paul VanderLei (Band XI)

**Don't Program  
OSGi**



**Ignorance is bliss**





**It's bad  
mojo to  
pollute the  
POJO**



**Your code sucks.  
Versioning can help.**





**Good fences  
make good  
neighbors**



A photograph of a brown donkey rearing up on its hind legs while pulling a wooden cart. The cart is heavily loaded with numerous cardboard boxes, which are stacked high and appear to be shifting. A man in a white shirt is standing at the front of the cart, holding the reins. In the background, another man in a blue shirt is walking, and there are other people and buildings in a dusty, outdoor setting. The word "FAIL!" is written in large, bold, black letters in the upper right corner of the image.

**FAIL!**

**Size Does Matter**



**Manage your dependencies**







**Use  
services**

```

Activator
public class Activator implements BundleActivator {
    private BundleContext context;
    private EmergencyMonitor monitor;
    private ServiceTracker gpsTracker;
    private IGps gps;
    private ServiceTracker airbagTracker;
    private IAirbag airbag;

    public void start(BundleContext context) throws Exception {
        this.context = context;
        monitor = new EmergencyMonitor();

        // Start tracking IGps services.
        ServiceTrackerCustomizer gpsCustomizer =
            createGpsCustomizer();
        gpsTracker = new ServiceTracker(context,
            IGps.class.getName(),
            gpsCustomizer);
        gpsTracker.open();

        // Start tracking IAirbag services.
        ServiceTrackerCustomizer airbagCustomizer =
            createAirbagCustomizer();
        airbagTracker = new ServiceTracker(context,
            IAirbag.class.getName(),
            airbagCustomizer);
        airbagTracker.open();
    }

    public void stop(BundleContext context) throws Exception {
        // Stop tracking IAirbag services.
        airbagTracker.close();

        // Stop tracking IGps services.
        gpsTracker.close();
    }

    private ServiceTrackerCustomizer createGpsCustomizer() {
        return new ServiceTrackerCustomizer() {
            public Object addingService(ServiceReference reference) {

```

```

                Object service = context.getService(reference);
                synchronized (this) {
                    if (Activator.this.gps == null) {
                        Activator.this.gps = (IGps) service;
                        Activator.this.bind();
                    }
                }
                return service;
            }
        }
    }

    public void removedService(
        ServiceReference reference, Object service) {
        synchronized (this) {
            if (service != Activator.this.gps)
                return;
            Activator.this.unbind();
            Activator.this.bind();
        }
    }

    public void modifiedService(ServiceReference reference,
        Object service) {
        // No service property modifications to handle.
    }
};

private ServiceTrackerCustomizer createAirbagCustomizer() {
    return new ServiceTrackerCustomizer() {
        public Object addingService(ServiceReference reference) {
            Object service = context.getService(reference);
            synchronized (this) {
                if (Activator.this.airbag == null) {
                    Activator.this.airbag = (IAirbag) service;
                    Activator.this.bind();
                }
            }
            return service;
        }
    }
}

```

```

        public void removedService(
            ServiceReference reference, Object service) {
            synchronized (this) {
                if (service != Activator.this.airbag)
                    return;
                Activator.this.unbind();
                Activator.this.bind();
            }
        }

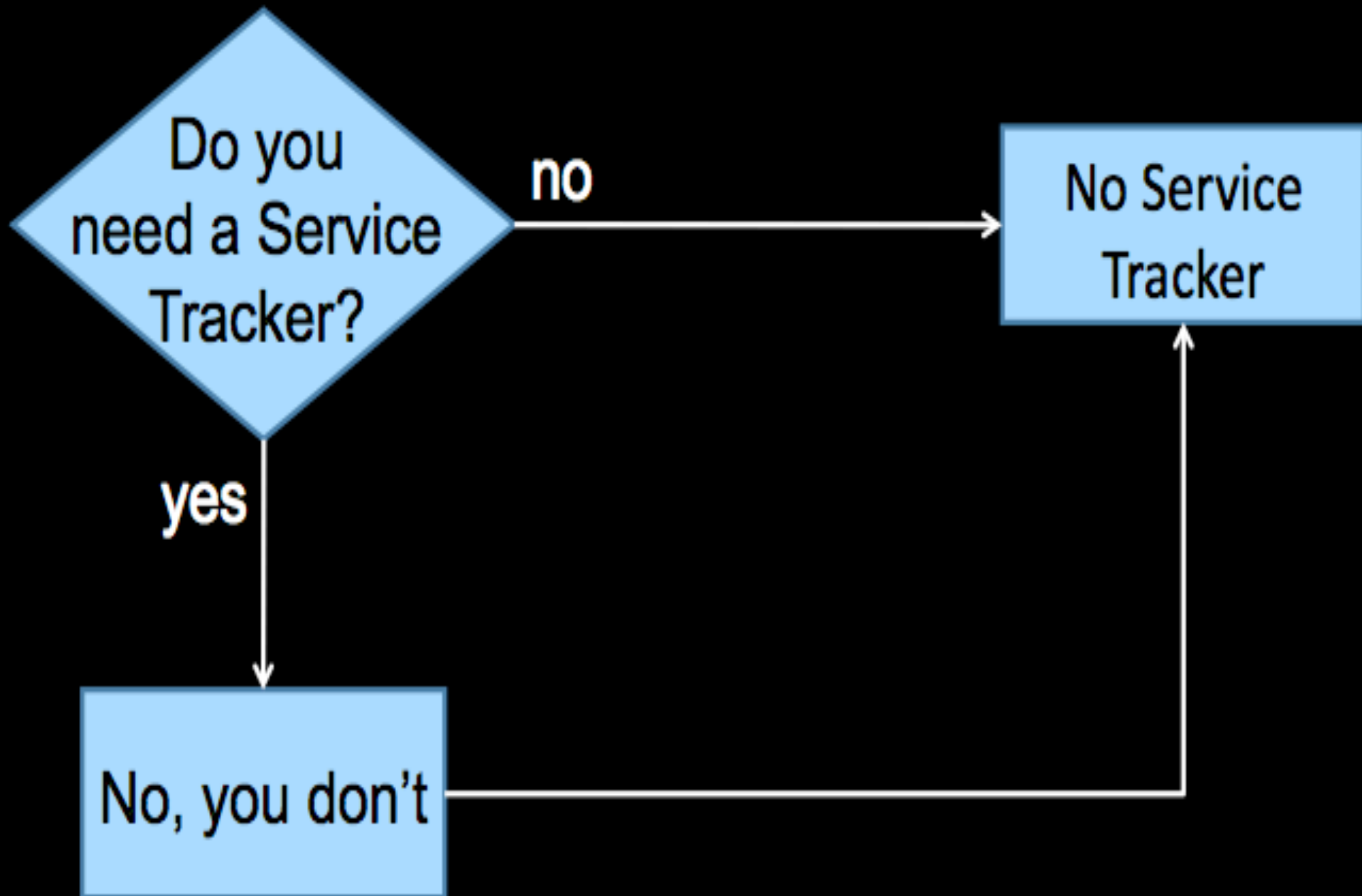
        public void modifiedService(ServiceReference reference,
            Object service) {
            // No service property modifications to handle.
        }
};

private void bind() {
    if (gps == null) {
        gps = (IGps) gpsTracker.getService();
        if (gps == null)
            return; // No IGps service.
    }
    if (airbag == null) {
        airbag = (IAirbag) airbagTracker.getService();
        if (airbag == null)
            return; // No IAirbag service.
    }
    // Bind IGps and IAirbag to the EmergencyMonitor
    monitor.bind(gps, airbag);
}

private void unbind() {
    if (gps == null || airbag == null)
        return;
    monitor.unbind();
    gps = null;
    airbag = null;
}
}

```







**Not everything  
should be a service**





*... and OSGi made all your apps dynamic!*



**Free!**





**If you don't test it,  
it doesn't work!**



**OSGi is not a religion**



**Now some controversy**



# Require-Bundle



# Import-Package





# Services or Extensions?



A close-up photograph of spaghetti with a red tomato sauce. The spaghetti is tangled and messy, with the sauce coating the strands. The text is overlaid in the center of the image.

**Your System still  
looks like this?**



**Red or Blue?**





# Q&A

**Martin Lippert**  
**[mlippert@vmware.com](mailto:mlippert@vmware.com)**