

# Komponentenbasierter Taschenrechner mit CORBA

---

Von  
Balamuhunthan Balarajah  
Olaf Märker  
Jan Zimmermann

# Gliederung

---

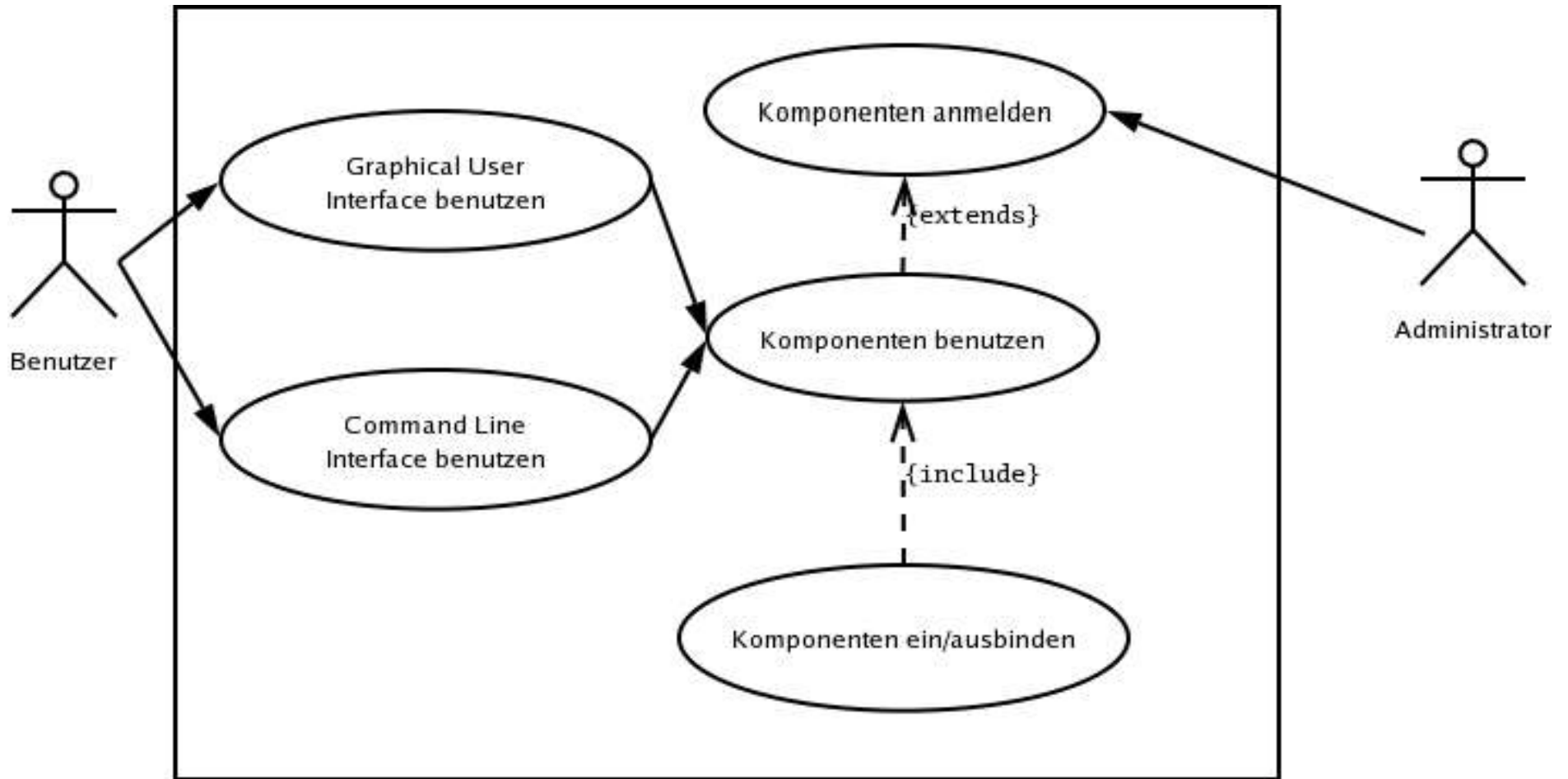
- Überlegungen
- Entwurf
- Verwendete Entwurfsmuster
- Testen
- Vorführung
- alternative Implementierung mit JNDI
- Ausblick

# Überlegungen

---

- Eine Komponente kann jeweils mehrere Funktionen anbieten.
- Komponenten bringen kein UI mit
- Abfragen der angebotenen Funktionen
- Verteilungstransparenz
- Realisierung von Funktionen mit unterschiedlichen und beliebig vielen Parametern
- Interoperabilität
- Sprachunabhängigkeit

# Use Cases



# TR.idl

---

```
module TRPackage {
    exception TRFehler {string info;};

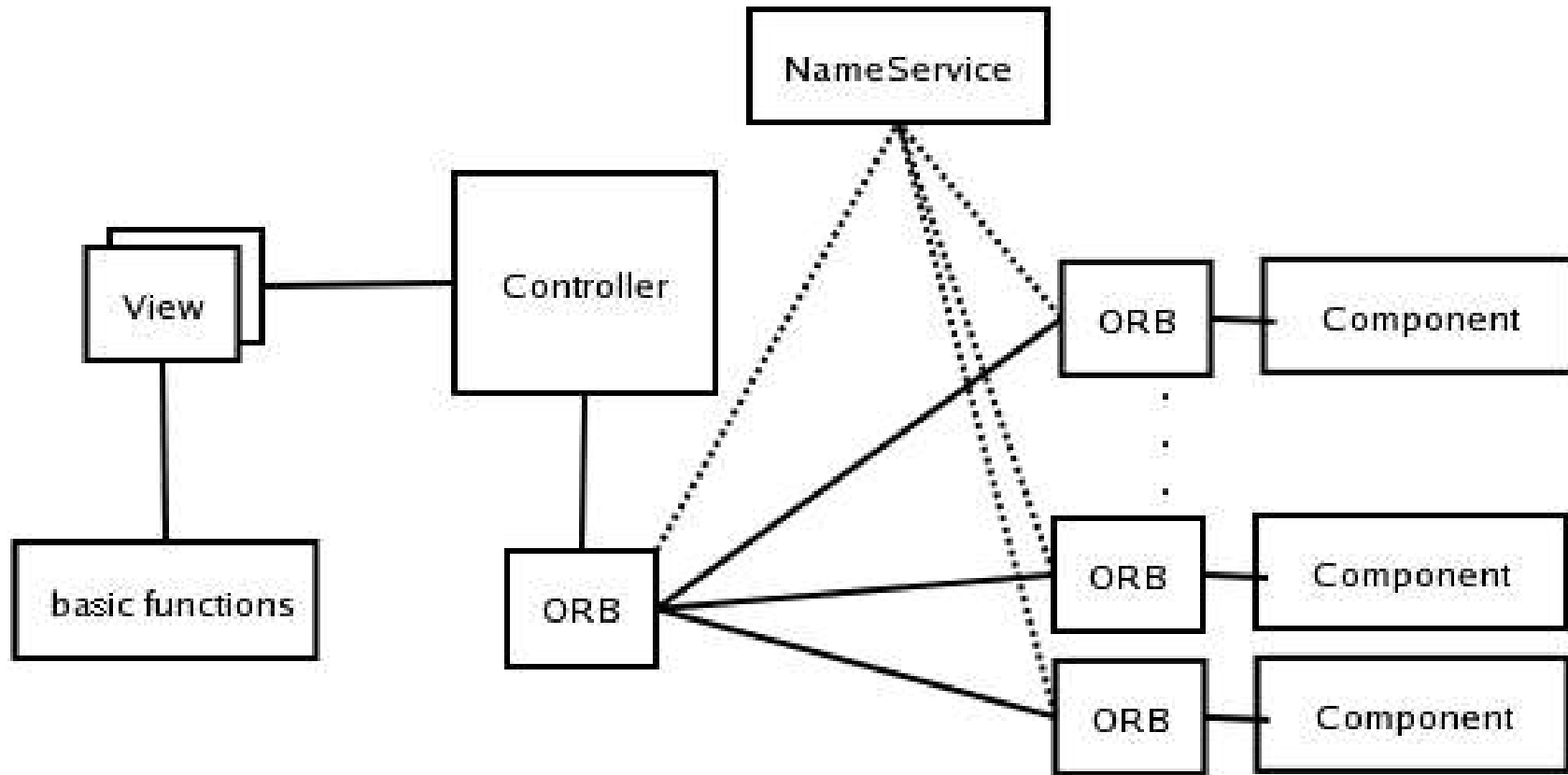
    typedef sequence <string> Parameter;

    struct Funktion {
        string funktionsname;
        long parameteranzahl;};

    typedef sequence <Funktion> Funktionen;

    interface TR {
        string berechne (in string function, in Parameter param)
            raises (TRFehler);
        Funktionen getFunktionen ();
    };
};
```

# Aufbau



..... bind or resolve

— use

# Verwendete Entwurfsmuster

---

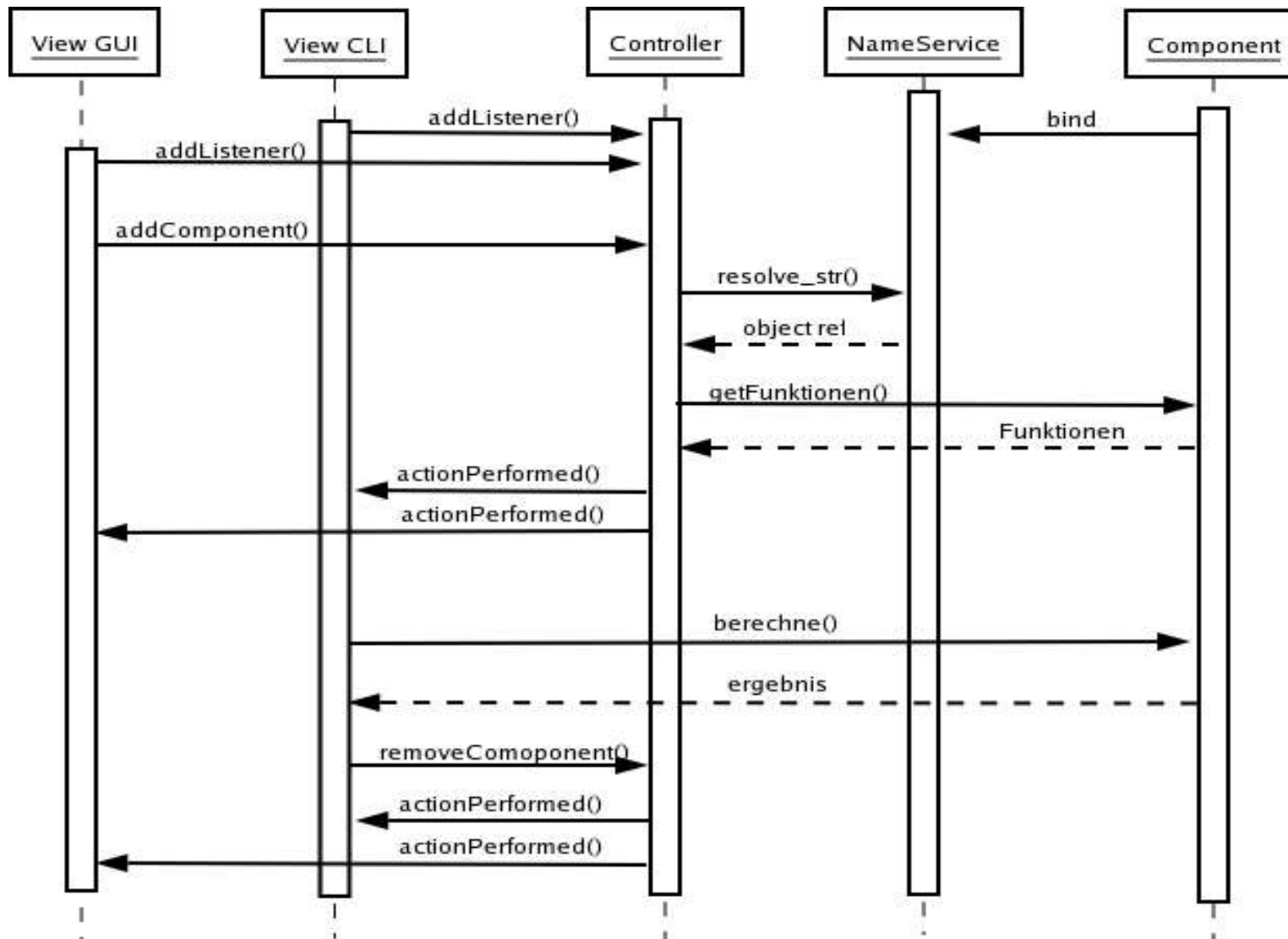
- Model-View-Controller
- ActionListener (Observer)
- Bridge (CORBA Interface)

# Testen

---

- Zwei Möglichkeiten zum Testen
- Test, der als View auf den Controller aufsetzt
  - Testet alle Funktionen aller am NamingService gebundenen Komponenten mit allen möglichen Kombinationen aus Parametern
  - Vergleicht Ergebnisse mit vorher gespeicherten Werten
- CLI kann Scriptdateien mit Befehlen interpretieren (Eingabeumleitung)

# Vorführung Taschenrechner



# NameService: Binden mit CORBA

---

```
Wurzellmpl wurzellmpl = new Wurzellmpl();
wurzellmpl.setORB(orb);
// get object reference from the servant
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(wurzellmpl);
TR href = TRHelper.narrow(ref);

// get the root naming context
// NameService invokes the name service
org.omg.CORBA.Object objRef = orb.resolve_initial_references
    ("NameService");
// Use NamingContextExt which is part of the Interoperable
// Naming Service (INS) specification.
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

// bind the Object Reference in Naming
String name = "Wurzel";
NameComponent path[] = ncRef.to_name( name );
ncRef.rebind(path, href);
```

# NameService: Binden mit JNDI

---

```
Wurzellmpl wurzellmpl = new Wurzellmpl();
```

```
Context ctx = new InitialContext()
```

```
// bind the Object Reference in Naming
```

```
String name = "Wurzel";
```

```
ctx.bind(name,wurzellmpl);
```

# Auflösen in CORBA

---

```
public void addComponent(String name) {  
try {  
    // resolve the Object Reference in Naming  
    TR tr = TRHelper.narrow(ncRef.resolve_str(name));  
  
    Funktion[] func = tr.getFunktionen();  
    registrierteKomponenten.put(name,new Komponente(name,tr,func));  
    fireComponentChanged(new ActionEvent(this,ADD_COMPONENT,name));  
} catch (org.omg.CORBA.SystemException ex) {  
}
```

# Auflösen mit JNDI

---

```
Context ctx = new InitialContext();
```

```
public void addComponent(String name) {  
try {  
    TR tr = (TR) ctx.lookup(name);  
    Funktion[] func = tr.getFunktionen();  
    registrierteKomponenten.put(name, new Komponente(name, tr, func));  
    fireComponentChanged(new ActionEvent(this, ADD_COMPONENT, name));  
} catch (NamingException e) {  
    System.err.println("Problem looking up " + name + ": " + e);  
}
```

# Auflisten in CORBA

---

```
public ArrayList listAllBindedComponents(){
    ArrayList al=new ArrayList();
    BindingListHolder bl = new BindingListHolder();
    BindingIteratorHolder bi = new BindingIteratorHolder();
    ncRef.list(20,bl,bi);
    if(bl != null && bl.value != null){
        for(int i=0;i<bl.value.length;i++){
            //Komponenten können mit mehreren Namen am NameService gebunden sein.
            //Hier wird nur der erste berücksichtigt
            al.add(bl.value[i].binding_name[0].id);
        }
    }
    return al;
}
```

# Auflisten mit JNDI

---

```
public ArrayList listAllBindedComponents(){
    ArrayList al=new ArrayList();
    NamingEnumeration ne=ctx.list(„“)
    while(ne.hasMore()){
        al.add((String) next());
    }
    return al;
}
```

# Ausblick

---

- Benutzerfreundlichkeit:
  - Beschreibung zu Komponenten anbieten
- weitere Komponenten
- Überlegungen zu Multiuserfähigkeit
  - Bsp: Speicherkomponente
- Ausfallsicherheit
  - Was passiert, wenn eine Komponente ausfällt
- Beschreibung der Typen der Parameter einer Funktion

# Quellen

---

- <http://clorb.sourceforge.net>
- [www.omg.org](http://www.omg.org)
- <http://home.t-online.de/home/Martin.Both/VBOrb.html>
- <http://omniorb.sourceforge.net>
- <http://java.sun.com>
- [www.apl.juh.edu/~hall/lisp/Symbolic-Differentiation/](http://www.apl.juh.edu/~hall/lisp/Symbolic-Differentiation/)

[www.google.de](http://www.google.de)