Modules as Objects in Newspeak

Gilad Bracha

Peter von der Ahe

Vassili Bykov

Ministry of Truth

Yaron Kashai

Cadence Design Systems

William Maddox

Adobe Systems

Eliot Miranda

Teleplace

Hardware Modules

Hardware Modules

Plug'n Play

Hardware Modules

Plug'n Play

aka: Dynamic Configuration

Hardware Modules

Plug'n Play

In to each other!

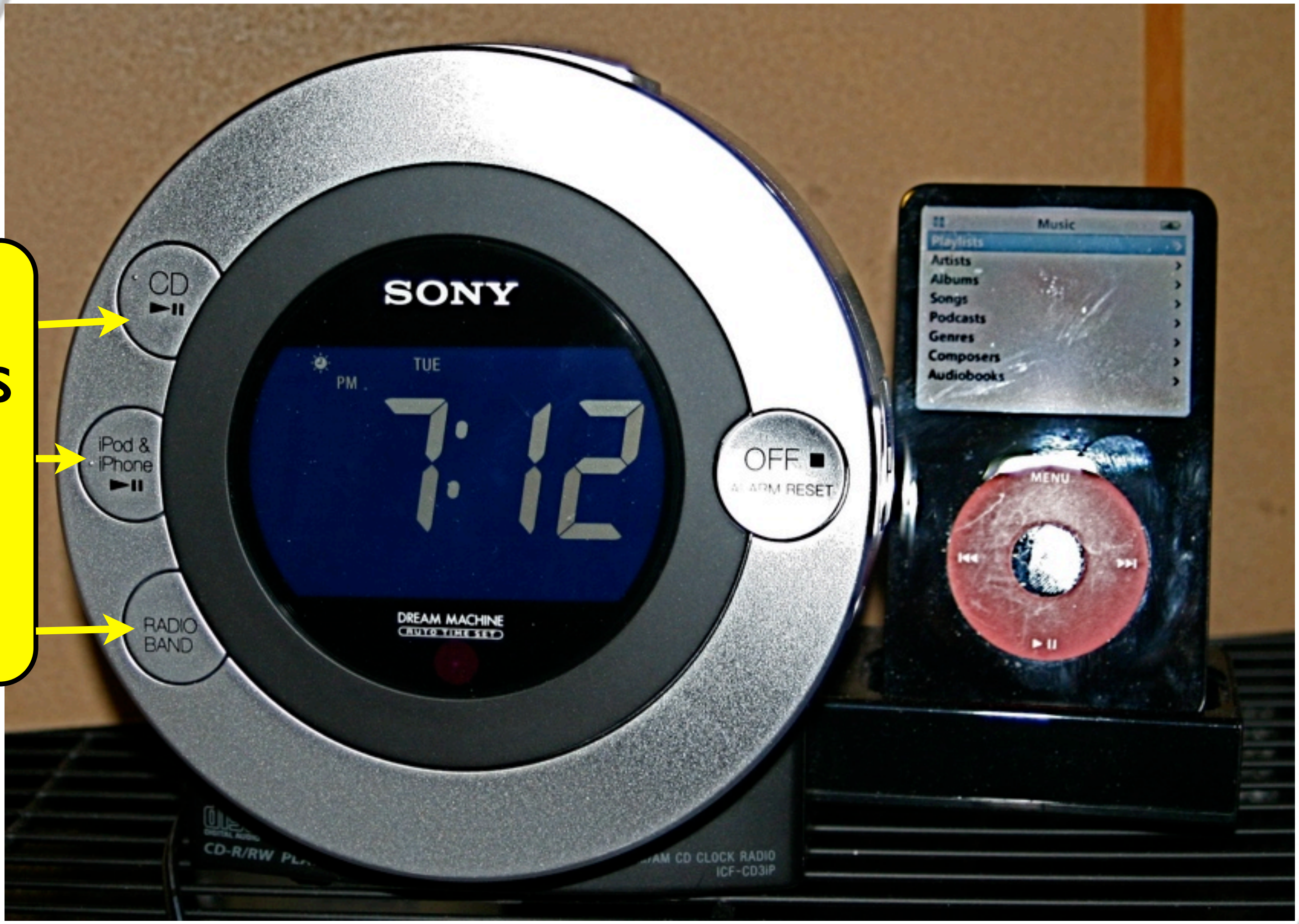
Hardware Modules

Plug'n Play

aka: Mutual Recursion

In to each other!

Hardware Modules



Multiple Instances of the same design

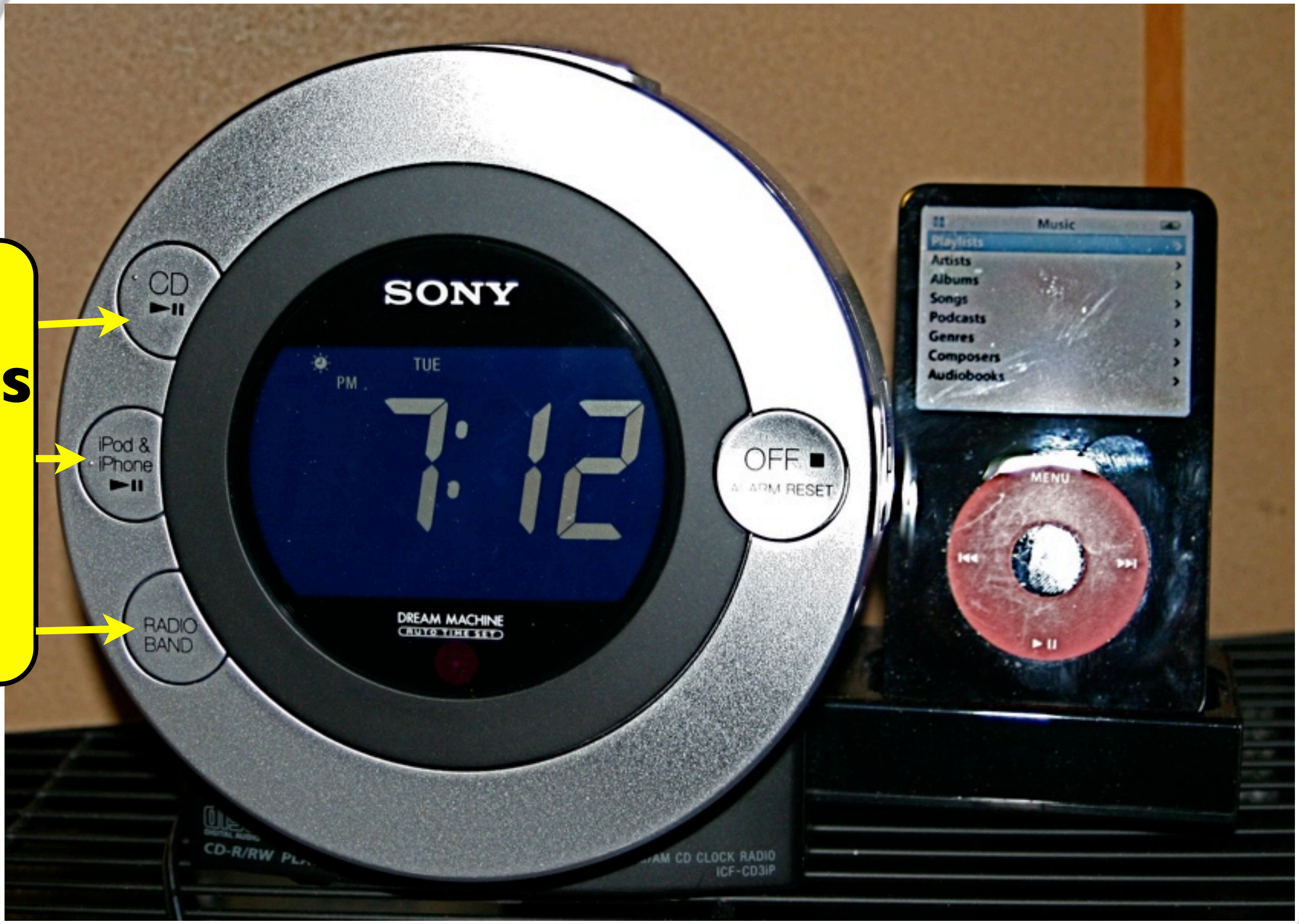
Mainstream Module Problems

- No mutual recursion
- Single instance of a design per run
- No distinction between module definition and module instances
- Awkward to define multiple configurations

Newspeak

- Newspeak is a dynamic, class based language with two defining properties:
  - All names are late bound
  - No global namespace

Hardware Modules



Multiple **Instances** of the same design

Designs are instantiated

Classes are instantiated

Use classes as unit of modular design

Classes Define Modules

Newspeak modularity is based *exclusively* on classes

- No packages, modules, bundles, templates ...

Newspeak

- Newspeak is a dynamic, class based language with two defining properties:
  - All names are late bound
  - No global namespace

No References to Variables

Representation Independence

- Always use slots via accessors

No References to Classes

- Always use accessors
- Classes are first class objects
  - Concepts are phenomena
- Classes are always virtual
- Classes are always mixins
- Class hierarchy inheritance

Newspeak

- Newspeak is a dynamic, class based language with two defining properties:
  - All names are late bound
  - No global namespace

The Insidious Import

```
module BraveNewWorldExplorer;
    import Collections.List;
```

The Insidious Import

```
module BraveNewWorldExplorer;
    import Collections.List;
```

nested within “module”

Global name!

The Insidious Import

module definition

```
module BraveNewWorldExplorer;

    import Collections.List;
```

module configuration

Module Definition

```
class BraveNewWorldExplorer usingLib: platform = (
    |
    List = platform collections List.
    ...
     |
    )( ...)
```

Module Configuration

```
main: platform args: as = (
    platform HopscotchFramework
      HopscotchWindow
          openSubject:
          ((BraveNewWorldExplorer
          usingPlatform: platform) FileSubject
          onModel: (as at: 1)
          ))
```

Module Configuration

```
main: platform args: as = (
  platform HopscotchFramework
    HopscotchWindow
        openSubject:
        ((BraveNewWorldExplorer
        usingPlatform: platform) FileSubject
        onModel: (as at: 1)
        ))
```

Module Configuration

```
class BraveNewWorldExplorerApp
    fileBrowserClass: fb <BraveNewWorldExplorer> = (
        | BraveNewWorldExplorer = fb. |
        )(
        main: platform args: as = (...)
        )
```

Module Configuration

Instantiate ***BraveNewWorldExplorerApp*** using tools (e.g., IDE).

Module Deployment

BraveNewWorldExplorerApp instance can be deployed via object serialization.

Module Loading

Serialized instance of ***BraveNewWorldExplorerApp*** can be loaded via object deserialization, followed by invocation of ***main:args:***.

Modules are Sandboxes

Factory method parameters are objects/capabilities that determine per-module sandbox

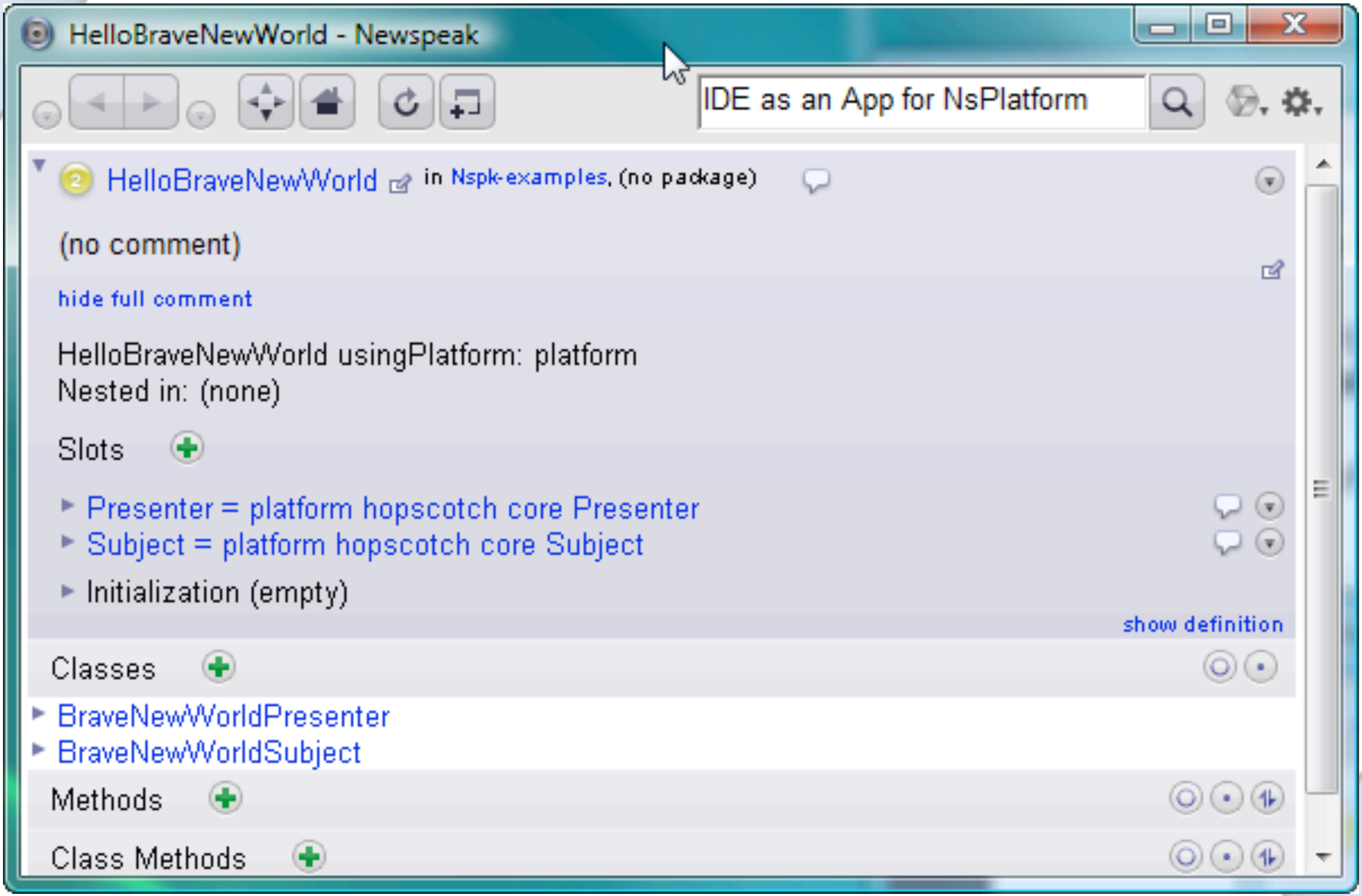
Side by Side Modules

```
platform:: Platform new.

m1:: NewspeakParsing
         using: platform
        parseLib: (CombinatorialParsing
                                usingLib: platform)

m2:: NewspeakParsing
         using: platform
        parseLib: (PackratParsing usingLib: platform)
```

Multiple Implementations

- Modules are objects, accessed via an interface
- Different implementations can co-exist

Status



HelloBraveNewWorld in Nspk-examples, (no package)

(no comment)

hide full comment

HelloBraveNewWorld usingPlatform: platform
Nested in: (none)

Slots

- Presenter = platform hopscotch core Presenter
- Subject = platform hopscotch core Subject
- Initialization (empty)

show definition

Classes

- BraveNewWorldPresenter
- BraveNewWorldSubject

Methods

Class Methods

Status

- Available at *http://newpeaklanguage.org*
  - open source under Apache 2.0 license
- Work in Progress
  - Expect some tweaks to syntax and semantics
  - Implementation still not complete - especially libraries

Related Work

- Self
- Smalltalk
- Beta, gBeta, Virtual Classes
- E
- PLT Scheme/Units
- ML
- CLU, Modula, Ada, Oberon ...
- Much more, see the paper

Conclusions

- Natural and powerful synergy between:
  - Message-based programming
  - Component style modularity
  - Virtual classes, mixins, class hierarchy inheritance
  - Object capability model and security
  - Mirror based reflection
  - Actor style concurrency
  - Pluggable types

The Newspeak eye ◉ used in the bullets, slide background etc. was designed by Victoria Bracha and is used by permission.