

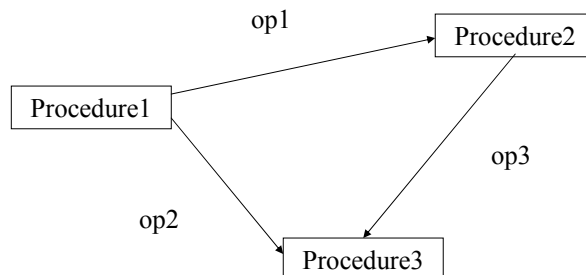
Solution 4: Implicit Invocation

- Component integration based on shared data, but
- The interface to the data is more abstract (like OO)
 - Storage formats are not exposed
 - Data is accessed abstractly (e.g. as a list or a set)
- Computations are invoked implicitly as data is modified based on active data model

190

What is Implicit Invocation?

- Explicit invocation
 - Procedure invocation, function call



191

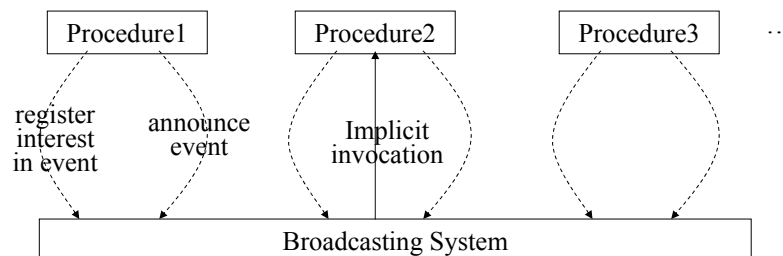
What is Implicit Invocation?

- Instead of invoking a procedure directly ...
 - A **component** can announce (or broadcast) one or more events.
 - Other **components** in the system can register an interest in an event by associating a procedure with the event.
 - When an event is announced, the broadcasting system (**connector**) itself invokes all of the procedures that have been registered for the event.

192

What is Implicit Invocation?

- An event announcement “implicitly” causes the invocation of procedures in other modules.



193

Event Systems: Model

- Components: objects or processes
 - Interface defines a set of incoming procedure calls
 - Interface also defines a set of outgoing events
- Connections: event-procedure bindings
 - Procedures are registered with events
 - Components communicate by announcing events at “appropriate” items
 - When an event is announced the associated procedures are implicitly invoked
 - Order of invocation is non-deterministic

194

Implicit Invocation Style

- Suitable for applications that involve loosely-coupled collection of components, each of which carries out some operations and may in the process enable other operations.
- Particularly useful for applications that must be reconfigured on the fly:
 - Changing a service provider.
 - Enabling or disabling capabilities.

195

Implicit Invocation Invariants

- Announcers of events do not know which components will be affected by those events.
- Components cannot make assumptions about the order of processing.
- Components cannot make assumptions about what processing will occur as a result of their events (perhaps no component will respond).

196

Implicit Invocation Specializations

- Often connectors in an implicit invocation system also include the **traditional procedure call** in addition to the bindings between event announcements and procedure calls.

197

Implicit Invocation Advantages

- Problem decomposition
 - Objects more independent than with explicit invocation
 - Interaction policy can be separated from interacting objects
- System maintenance and reuse
 - Static name dependencies not wired in, so dynamic reconfiguration is easy
 - Any component can be introduced into a system simply by registering it for the events of that system.
 - Eases system evolution since components may be replaced by other components without affecting the interfaces of other components in the system.
- Performance
 - Possibility of parallel handling events

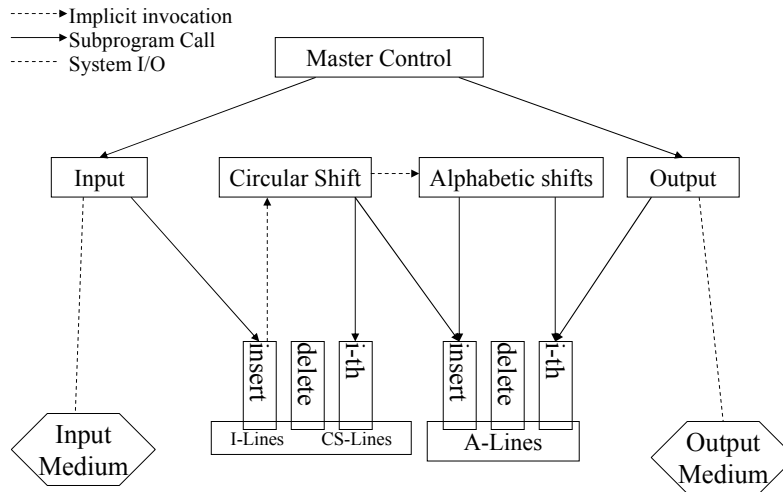
198

Implicit Invocation Disadvantages

- Problem decomposition
 - No control over sequencing of invocations
 - Function call semantics problematic
 - Cycles may be problematic
- System maintenance and reuse
 - Needs central management to keep track of events, registrations, and dispatch policies
 - Event handling may interact badly with other run time mechanisms, e.g., event control loops of RPC, etc.
- Performance
 - Indirection may incur overhead
- When a component announces an event:
 - it has no idea what other components will respond to it,
 - it cannot rely on the order in which the responses are invoked,
 - it cannot know when responses are finished.

199

Architecture of Solution 4



200

Solution 4: Modularization

- Module 1: Input
 - Operation read: data lines from the input medium through operation insert. A new line is put into the line buffer "I-Lines".
- Module 2: Circular Shift
 - Using operation i-th to read I-Lines and produce shifted lines and store them in a separate abstract line buffer, CS-Lines
- Module 3: Alphabetizer
 - Using operation (2nd) i-th to read I-Lines and produce shifted lines and store them in a separate abstract line buffer "A-Lines"
- Module 4: Output
 - Access A-Lines and print out
- Module 5: Master Control
 - Invokes Input and Output explicitly

201

Non-Functional Requirements

- Modifiability
 - Changes in processing algorithms, e.g.
 - Line shifting
 - One at a time as it is read
 - All after they are read
 - On demand when the alphabetization requires a new set of shifted lines
 - batch alphabetizer vs incremental alphabetizer
 - Changes in data representation, e.g.
 - Storing characters, words and lines
 - in 1-D/2-D array/linked list, compressed/uncompressed
 - Explicitly/implicitly (as pairs of index and offset)
 - Core storage/secondary storage
- Enhanceability
 - Enhancement to system function, e.g.,
 - eliminate noise words (a, an, the, and, etc.),
 - The user deletes lines from the original or shifted lines

Not affects others

Not affects others

Additions are invisible to other modules (registration only)

202

Non-Functional Requirements

- Performance
 - Space and time
 - May use more space than OO and Shared Data for natural representations
 - Can be inefficient due to triggering
- Reusability
 - To what extent can the components serve as reusable entities – high since implicit invocation relies only on the existence of certain externally triggered events

203

Implicit Invocation

- Components: processes & event
- Connectors:
 - Broadcasting system
- Styles: Implicit invocation
- Rationale: NFRs
- Constraint: the order of invocations is non-deterministic

204

Summary of KWIC

	Pipe & Filter	Shared Data	ADT/OO	Implicit Innovation
Change algorithm	++	-	+	+
Change data representation	++	-	+	+
Add function	+	-	+	++
Space	-	++	+	+
Time	-	++	+	-
Reusability	++	-	+	+
Intuitiveness	++	-	+	-

205

Implicit Invocation Examples

- Retail stores
- Graphical User Interface (GUI)
- OS
- Java

206

GUI

- Graphical user interfaces are composed of a set of primitive components (or widgets)
 - Buttons, scroll bars, menus, dialogue boxes, text entry fields
- These widgets should be customizable and reusable in any context, not hand-coded for every application
- Therefore, should not have knowledge of the architecture in which they are placed

207

GUI

- GUI components issue (raise) events when they are invoked
 - Buttons issue buttonClicked events
 - Menus issue menuSelection events
 - Scroll bars issue scroll events
 - Text fields issue text changed events
 - etc...