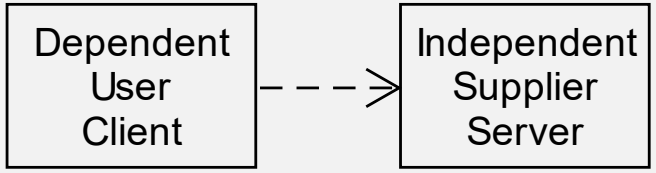
DEPENDENCY

Uses / Using

Delegation

Delroy A. Brinkerhoff

DEPENDENCY: ROLES AND MEANING



- Various role names
  - Formal names are "dependent" & "independent"
  - I prefer "client" and "supplier"
- Best understood in terms of responsibilities
  - The client class *depends* on the supplier class to fulfill its responsibilities
  - The client class *uses* the supplier class to fulfill its responsibilities
  - The client class *delegates* some of its responsibilities to the supplier class

DEPENDENCY PROPERTIES AND IMPLEMENTATION

PARAMETER

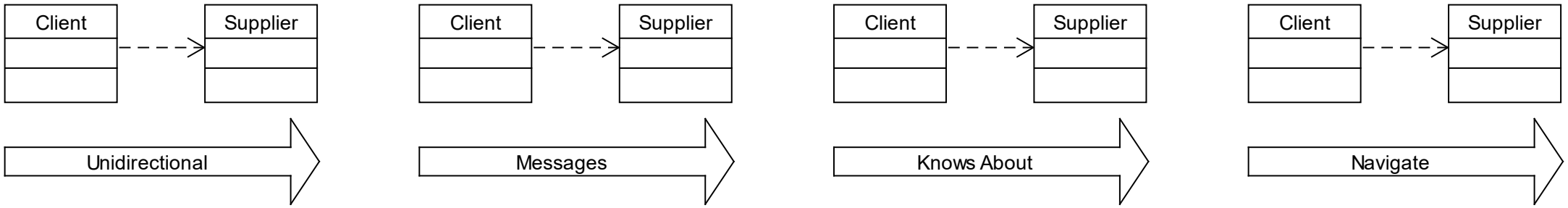
```
class bar { ... }

class foo
{
    private:
        T f1(bar b) { ... }
        T f2();
};
```

LOCAL VARIABLE

```
T foo::f2()
{
    bar b;
      ...
    b.g();
}
```

DETAILS

- Binding is intentionally temporary
- Implemented with local variables & parameters
- Lifetimes are independent
  - Relationship is created with the function call
  - Relationship ends when the function returns
  - Return types are insignificant
  - Suppliers may be pointers or references
- Sharing:
  - Sharable when implemented as a local variable
  - Shared when implemented as a parameter

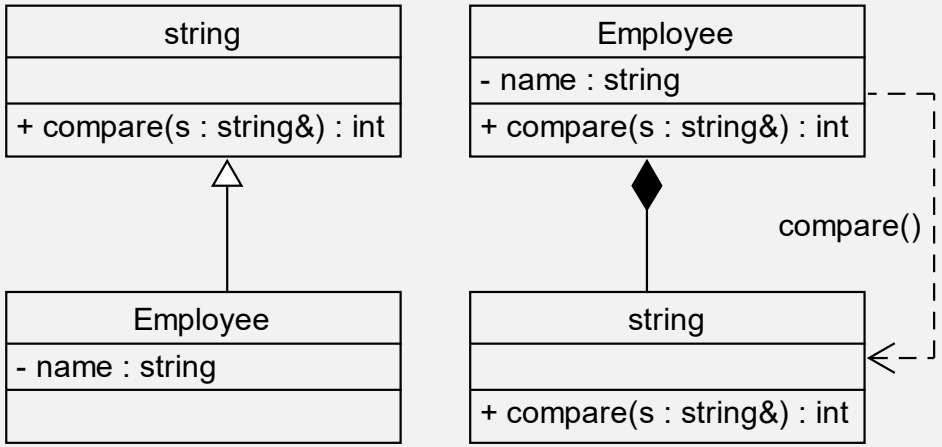


DEPENDENCY DIRECTIONALITY

SOME DEPENDENCY PROBLEMS

- Dependency is "a semantic relationship between two model elements in which a change to one element (the independent one) may affect the semantics of the other element (the dependent one)." (Booch, Rumbaugh, & Jacobson)
- "A **dependency** exists between two elements if changes to the definition of one element (the **supplier** or target) may cause changes to the other (the **client** or source)." (Fowler)
- Using these definitions, *all* relationships are dependencies!
- "Trying to show all the dependencies in a class diagram is an exercise in futility; there are too many, and they change too much." (Fowler)
- Understanding objects' behavior in a dependency relationship is more important than diagramming it.

IMPLEMENTATION INHERITANCE




```
int Employee::compare(string& other)
{
        return name.compare(other);
}
```