17-423/723: Designing Large-scale Software Systems

Design for Change

Jan 29, 2025

Leaning Goals

- Describe different types and causes of changes in software systems
- Describe changeability and its importance as a quality attribute
- Evaluate the changeability of a system using a dependency model
- Apply the information hiding principle to reduce the unnecessary impact of changes

Course Roadmap

- Foundational concepts & techniques for design
  - Domain & design modeling, quality attributes & trade-offs, design space, generating design alternatives, design review, design processes
- Designing for quality attributes
  - **Design for change**, interoperability, testability, reuse, scalability, robustness, security, usability, AI, ethics

Software Evolution

"The only constant in life is change"

software

Heraclitus

c. 500BC

Software Evolution

- Software systems continually evolve over time!

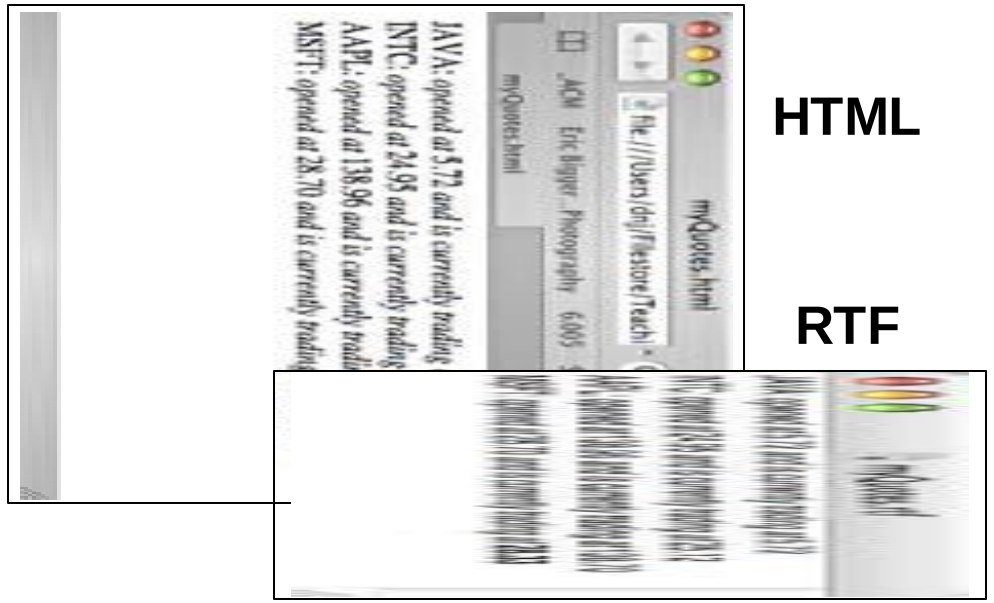
Types of Changes: Example

- Data structures
- Service/component logic
- Interface definitions
- Global and class variables; variable names and types
- Use of language features and library routines
- Data-size constraints (e.g., array declarations and loop limits)
- Business rules such as the laws, regulations, policies, and procedures that are embedded into a system
- Input and output formats
- **Q. other examples?**

What’s driving these changes?

- Causes of changes
  - New or updated client requirements
  - Maintenance: Refactoring, bug fixing, debloating
  - Resolving delayed or “default” design decisions
  - Quality improvement: Scalability, availability, reusability, security…
  - Adapting to external changes: New platform or hardware devices, libraries, languages, etc.,
- Nearly all changes are driven by a quality attribute(s)!

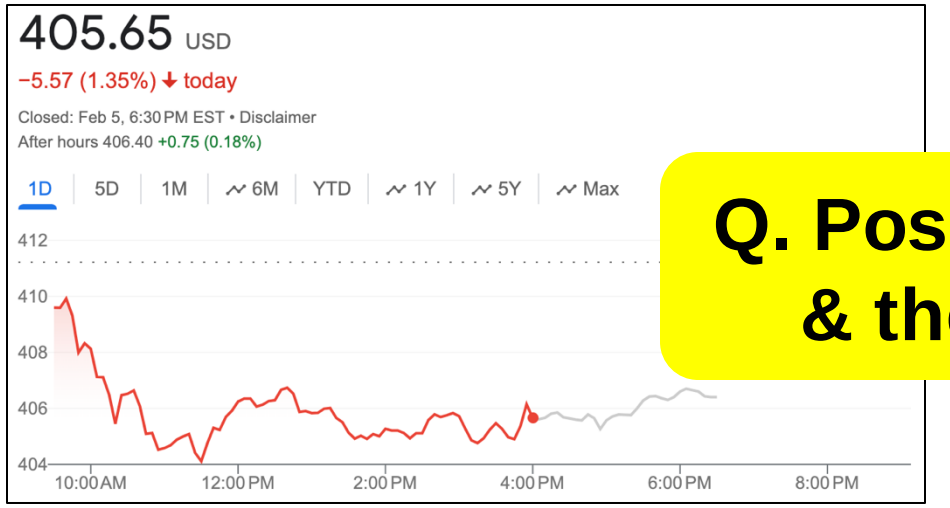
Today's Example: Stock Tracker App



- Get a list of stock quotes (prices) from an external source (e.g., Google)
- Produce the output report in HTML or RTF format
- Put the quote in **bold** if the change since the opening is > 1%

Based on an example by Daniel Jackson & Rob Miller

Today's Example: Stock Tracker App



Q. Possible changes & their causes?

- Get a list of stock quotes (prices) from an external source (e.g., Google)
- Produce the output report in HTML or RTF format
- Put the quote in **bold** if the change since the opening is > 1%

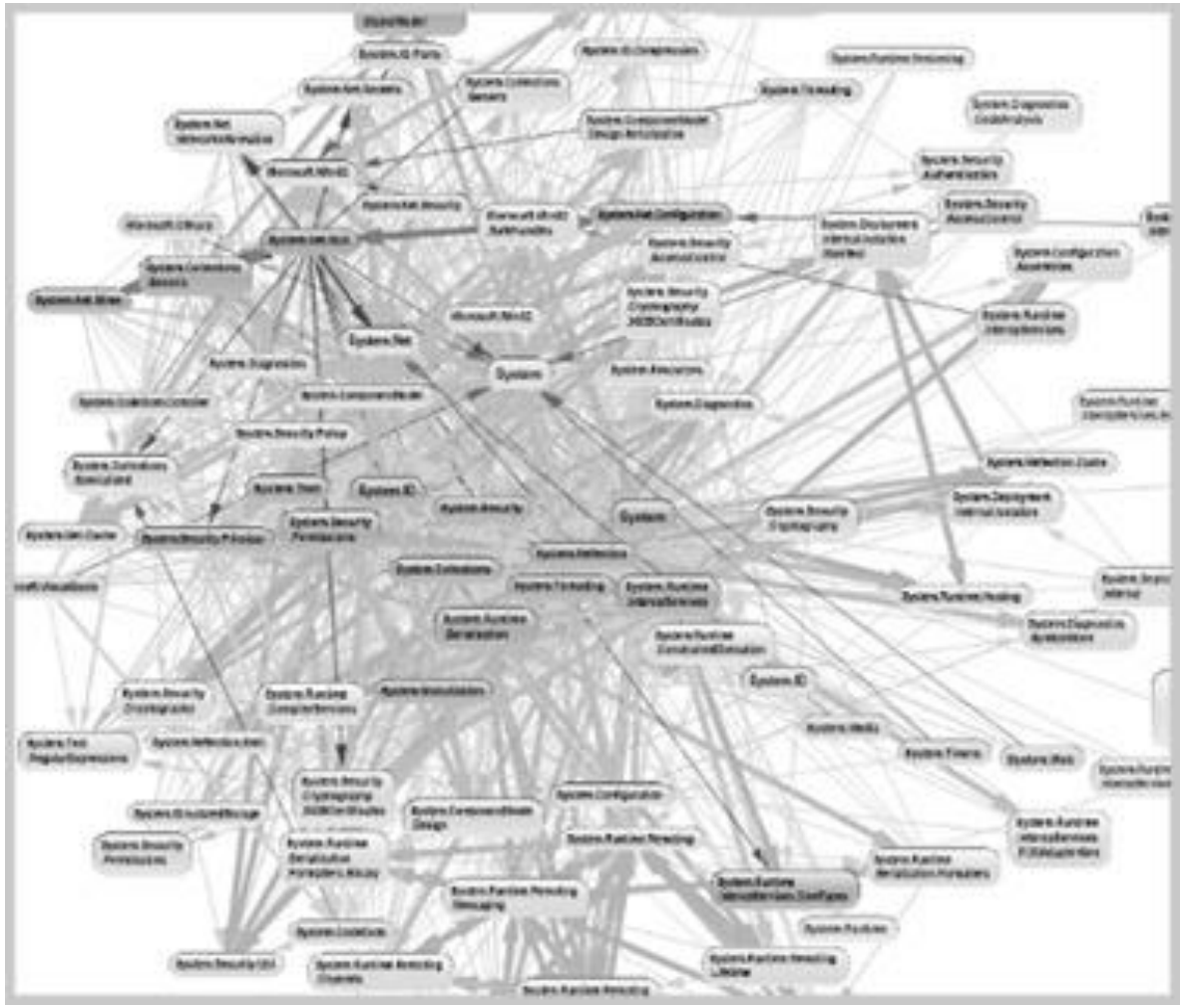
Consequences of a Change

- What happens when one part of a system changes? How does it impact the rest of the system?
- A (reckless) change can:
  - Cause unexpected/unnecessary changes in other parts of the system
  - Degrade one or more quality attributes (e.g., reliability, security)
  - Increase the overall system complexity
  - Contribute to increased development and maintenance costs
  - Break software written by other teams or organizations
  - Break software that users & stakeholders rely on

Photo: Michael Hession

Why MacOS Catalina Might Break Some of Your Apps, and What to Do About It

"...Catalina—unlike most macOS releases—breaks a lot of apps, especially older productivity software and games. And not just in the “some things don’t work right but they can still run” sense, but in the “it’s not possible to run the apps at all anymore” sense.'



Big Ball of Mud

Reckless changes increase complexity
Eventually, nobody understands how the system works

Q. Any examples that you know?

http://www.laputan.org/mud/mud.html#BigBallOfMud

(In)Flexibility to change is a product (dis)advantage!

This problem was exacerbated by the fact that, at the time, the Windows development environment was a mishmash of tools stitched together over many years.

The more code we added, the more complexity we created, the larger the team got, the bigger the ecosystem, the harder it became to leapfrog the competition.

An architecture that had been massively successful during the nineties became bogged down a decade later because the world around us was changing ever more rapidly while the organization struggled to keep up with it.

[What really happened with Visa: An insider's retrospective]

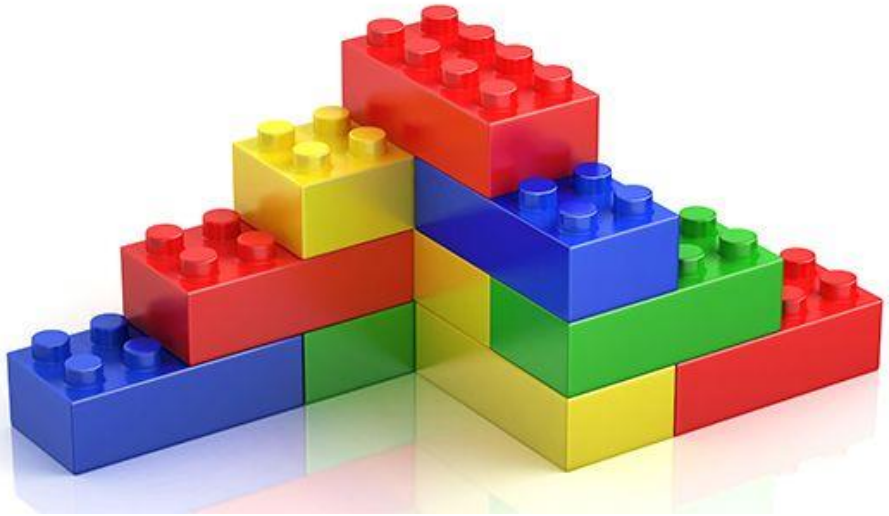
Changeability & Dependency

Changeability

- A measure of the amount of effort involved in making a change to a system
- Usually qualitative (i.e., yes/no), but sometimes quantified in terms of numerical metrics (e.g., lines of code changed)
- Quality attribute specifications – examples:
  - “A new publisher can be added without having to change any of the existing subscribers”
  - “New types of stocks can be added without changing the format of how each stock is displayed”
  - “Improving the performance of the C++ compiler does not affect the correctness of the parser output”
  - “Adding a new type of sensor in a self-driving vehicle requires changing only the image processing module”

Related Concepts

- Modularity
  - Degree to which different parts of the system can be substituted with alternative parts without affecting the rest of the system
  - Closely related to changeability: Modularity supports changeability!



Related Concepts

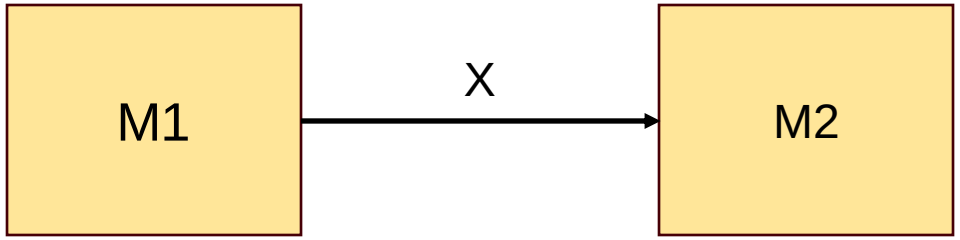
- Modularity
  - Degree to which different parts of the system can be substituted with alternative parts without affecting the rest of the system
  - Closely related to changeability: Modularity supports it!
- Extensibility
  - Usually refers to the ability of a system to support new features or requirements
- Modifiability
  - Often used interchangeably with changeability

Dependency

- Degree to which one component relies on another component to fulfill its responsibility
- Dependency is a key concept to understanding changeability
- Changeability is inversely correlated with the amount of dependency among components
  - Strong (weaker) the dependency, larger (smaller) the amount of effort required to implement a change

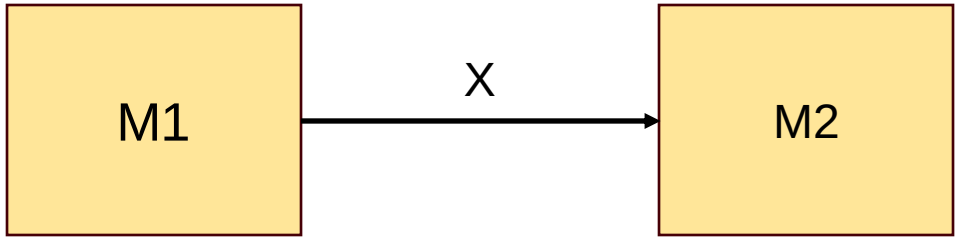
Dependency Model

- A variant of component model
- Can be used to communicate & evaluate the changeability of a system



Dependency Model

- To fulfill its responsibility, component M1 **depends** on M2 through connection X
- If M2 changes in a way that affects X, M1 may need to change to continue fulfilling its responsibility

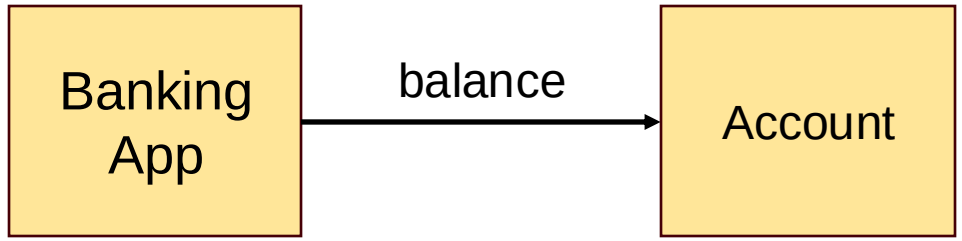


Dependency: Example

```
public class BankingApp {
   private Map<String,Account> accounts;

   public void depositDollar(String uid,
      int dollars) {
      Account acc = accounts.get(uid);
      acc.balance += dollars;
   }
}
```

```
public class Account {
   public String accountID;
   public int balance; // in dollars
}
```



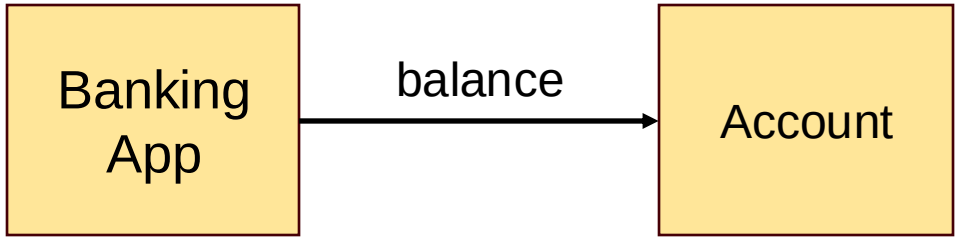
- Component responsibilities
  - Banking App: Fulfill user requests for depositing into account
  - Account: Stores the account balance in US dollars

Dependency: Example

```
public class BankingApp {
   private Map<String,Account> accounts;

   public void depositDollar(String uid,
      int dollars) {
      Account acc = accounts.get(uid);
      acc.balance += dollars;
   }
}
```

```
public class Account {
   public String accountID;
   public int balance; // in cents
}
```



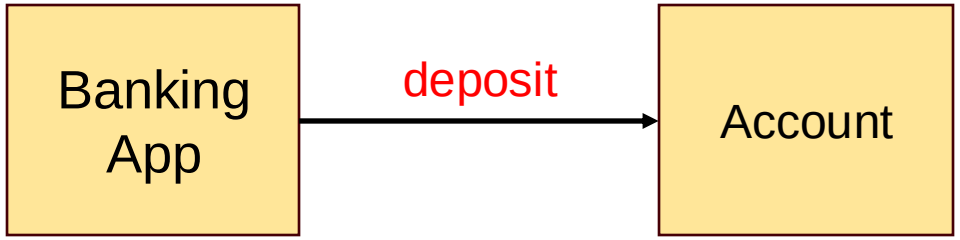
- **Q. What if the meaning of "balance" is changed to cents? How does it impact Banking App?**

Alternative Design

```
public class BankingApp {
   private Map<String,Account> accounts;

   public void depositDollar(String uid,
      int dollars) {
      Account acc = accounts.get(uid);
      acc.deposit(dollars);
   }
}
```

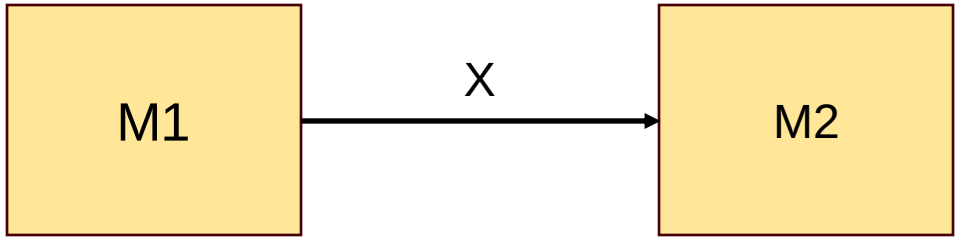
```
public class Account {
   public String accountID;
   public int balance; // in cents
   public void deposit(int dollars) {
      balance += dollars*100;
   }
}
```



- Banking App no longer directly depends on “balance”
- Changes to “balance” (e.g., from type int to float) does not impact Banking App as long as “deposit” behaves the same

Dependency Model

- To fulfill its responsibility, component M1 depends on M2 through connection X
- If M2 changes in a way that affects X, M1 may need to change to continue fulfilling its responsibility
- **Change propagates both ways!** If M1's responsibility changes in a way that affects X, M2 may need to change to accommodate M1

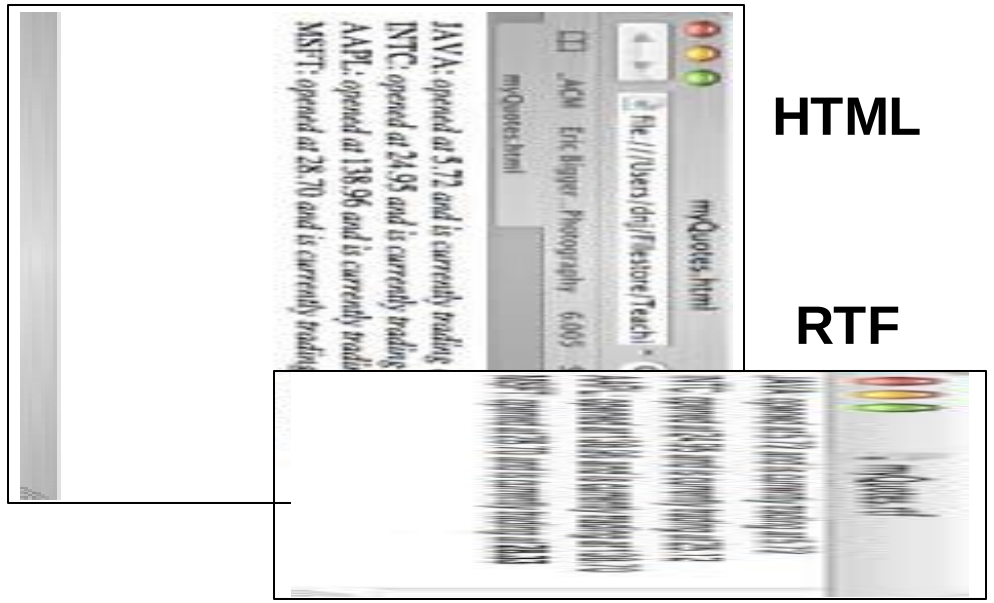


Change can propagate both directions!

- Component responsibilities
  - Stock Tracker App: Fulfills requests from the user to display the current quote (i.e., price) of a stock
  - Stock Quoter: Obtain the quote of a stock from an external source & return to Stock Tracker App
- **Q. What if the client wants to see the total price change since the opening (in addition to the current quote)?**

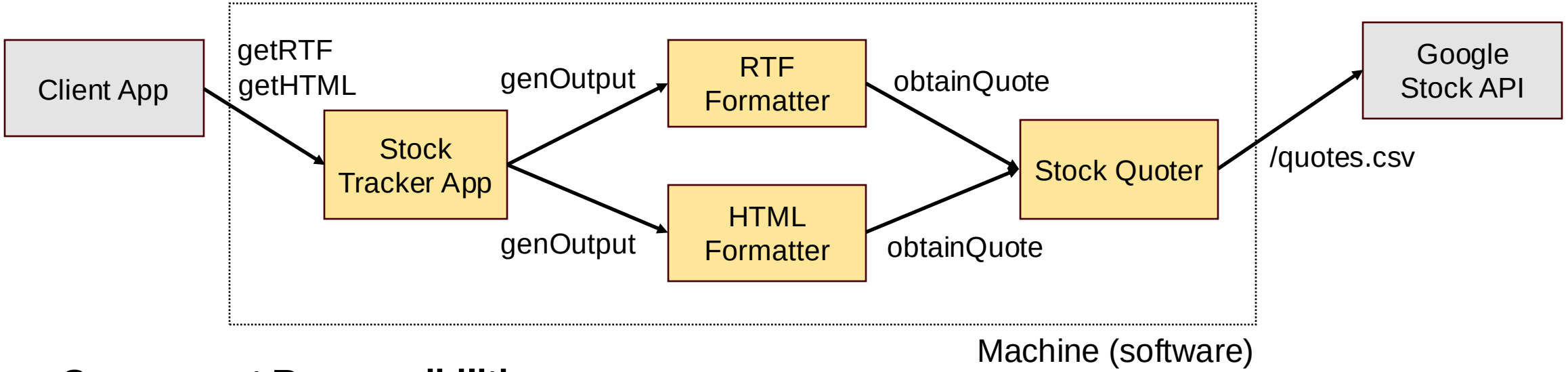
Change in Tracker App's responsibility forces Quoter to change!

Example: Stock Tracker App



- Get a list of stock quotes from an external source
- Produce the output report in HTML or RTF format
- Put the quote in **bold** if the change since the opening is > 1%

Stock Tracker App: Possible Design



Component Responsibilities

- Stock Trakcer App: Fulfills requests from a client for a quote in a certain format
- RTF/HTML formatter: Get quote from Stock Quoter & generate output in the right format
- Stock Quoter: Invoke Google API to get quote & return the result to Formatter

Domain Assumption

- Google Stock API: Returns a quote in CSV format

Stock Quoter

```
public class Quoter {
   private URL url;
   public Quoter (String symbol) throws MalformedURLException {
      url = new URL("http://quote.google.com/d/quotes.csv?s=“ +symbol+"&f=noa");
   }

   public String obtainQuote() throws IOException {
      // Read the stock quote from the URL and return the resulting CSV
      BufferedReader in = new BufferedReader(new InputStreamReader(url.open..));
      String csv = in.readLine();
      in.close();

      return csv;
   }
}
```

- Invoke Google API to get quote & return the result to Formatter

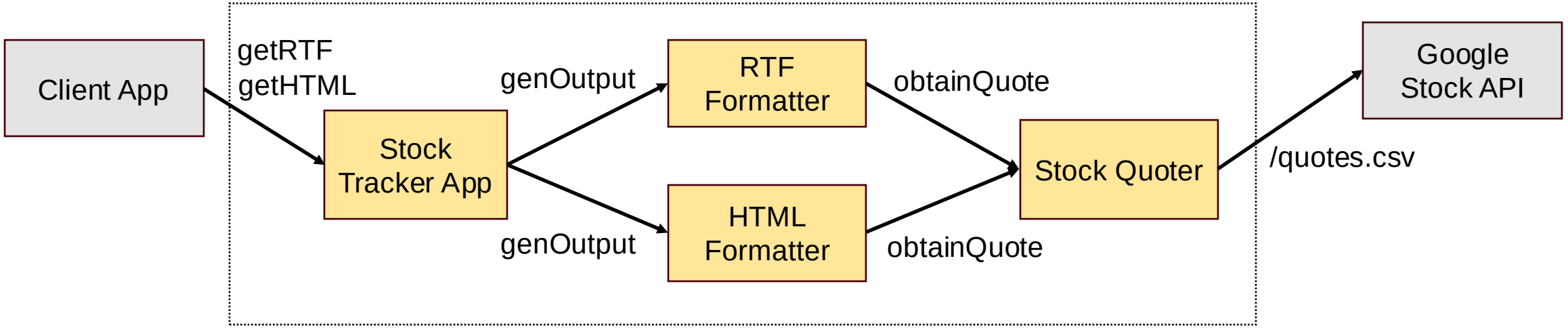
Formatter

```
public class HTMLFormatter {
   private final Set<String> symbols = new HashSet<String> ();

   public void generateOutput() throws IOException {
      PrintStream out = new PrintStream(new FileOutputStream (...));
      out.println("<html>");
      for (String symbol: symbols) {
         Quoter q = new Quoter (symbol);
         String quoteStr = q.obtainQuote();
         // Format & write HTML version of the quote into "out"
         ...
      }
      out.println("</html>");
      out.close();
    }
}
```

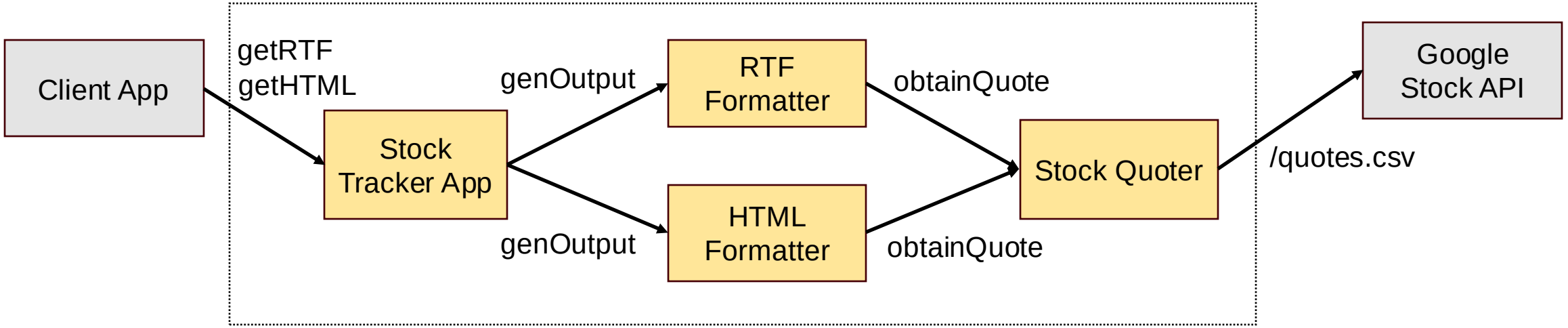
- Get quote from Quoter & generate output in the HTML format

Stock Tracker App



- Possible change: Google changes the quote format from CSV to JSON
- **Q. What is the impact of this change on the rest of the system?**

Stock Tracker App

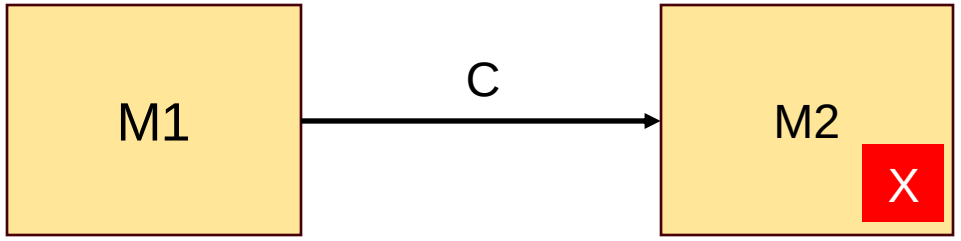


- Possible change: Google changes the quote format from CSV to JSON
- **Q. What is the impact of this change on the rest of the system?**
- **Q. How would you change the design to reduce the impact?**

Information Hiding

Principle: Information Hiding

- A principle for reducing dependency between components
- If M1 depends on M2, M2 should hide design decisions (i.e., "**secrets**") that are likely to change
- Benefit: Even if a secret in M2 changes, M1 can continue to fulfill its responsibility without changing!

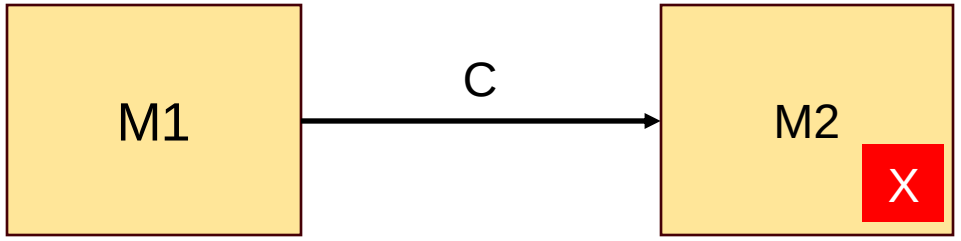


C: Connection to **M2**
X: Secret hidden in **M2**
Design task
C should be designed so that changing **X** does not affect it

Benefit
Changing **X** does not affect **M1**!

Principle: Information Hiding

- A principle for reducing dependency between components
- If M1 depends on M2, M2 should hide design decisions (i.e., "**secrets**") that are likely to change
- Benefit: Even if a secret in M2 changes, M1 can continue to fulfill its responsibility without changing!



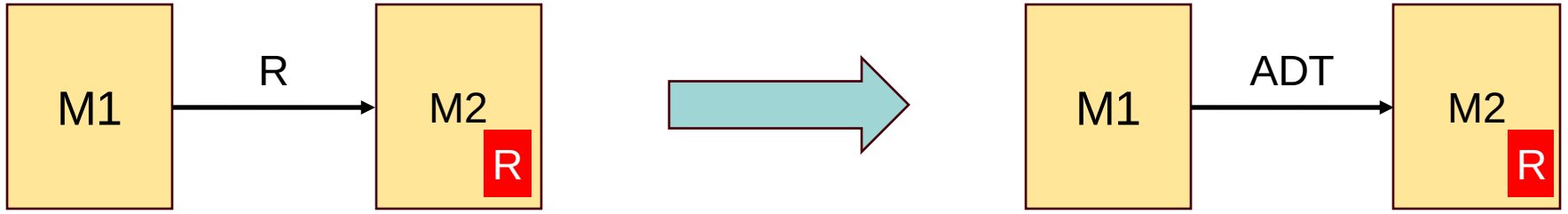
C: Connection to **M2**
X: Secret hidden in **M2**
Design task
C should be designed so that changing **X** does not affect it

Benefit
Changing **X** does not affect **M1**!

But how do we achieve this?

Information Hiding Techniques

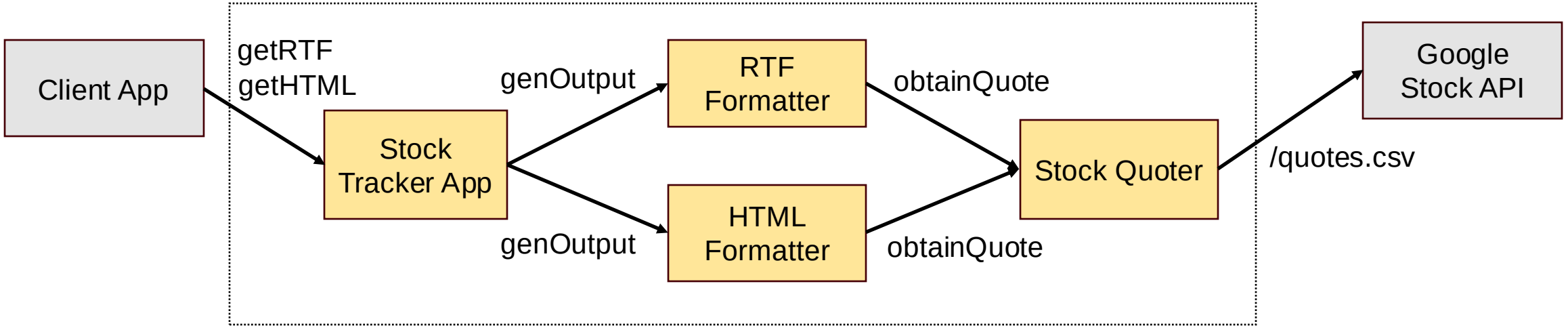
- **Data abstraction**
  - Hide the underlying **representation (secret)** of a data type behind an **abstract data type**
  - When the representation changes, other components can continue to use that data type without having to change



R: Data representation (secret)
ADT: Abstract data type

Communicate

Data Abstraction: Example



- Possible change: Google changes the quote format from CSV to JSON
- **Q. How would you hide the secret in Stock Quoter to reduce the impact of this change?**

Stock Quoter: Original version

```
public class Quoter {
   private URL url;
   public Quoter (String symbol) throws MalformedURLException {
      url = new URL("http://quote.google.com/d/quotes.csv?s=“ +symbol+"&f=noa");
   }

   public String obtainQuote() throws IOException {
      // Read the stock quote from the URL and return the resulting CSV
      BufferedReader in = new BufferedReader(new InputStreamReader(url.open..));
      String csv = in.readLine();
      in.close();

      return csv;
   }
}
```

- Invoke Google API to get quote & return the result to Formatter

Stock Quoter ver. 2

Data abstraction
Hide the representation of the quote as CSV string!

```
class Quoter {
    RL url;
    tring open,
    nt change;

    oter (String
    ew URL("http:
        +symb

    4alformedl
    om/d/quot

    ring getOpen () {return open;}
    ring getAsk () {return ask;}
    t getChange () {return change;}
    id obtainQuote () throws IOException {
    dReader in = new BufferedReader(new InputStreamReader(url.open..));
        csv = in.readLine();
    in.close();

    String[] fields = csv.split(",");
    open = fields[1];
    ask = fields[2];
    change = (int)(100 *(Float.valueOf(ask)-Float.valueOf(open))
        / Float.valueOf(open));
}
```

Provide functions for accessing different parts of the quote

Parses the CSV string & stores different parts

machine.
design
use?

of

BufferedReader is also a state machine. Draw it. What design pattern does it use?

HTML Formatter ver. 2

Data abstraction
Formatter no longer depends on the quote being stored as CSV!

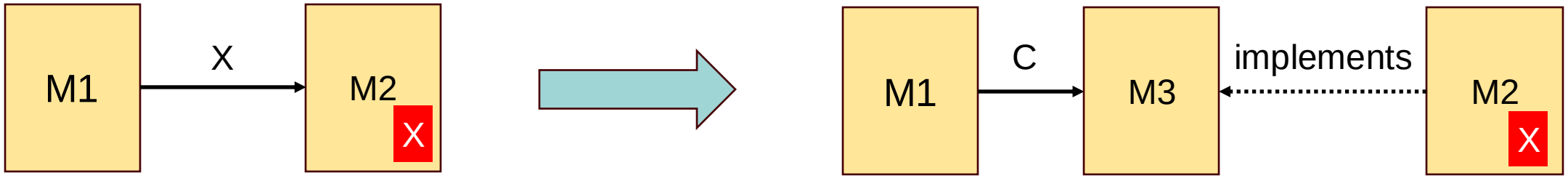
Access different parts of the quote & write them into HTML

```
HTMLFormatter {
private final Set<String> symbols = new Hash
...
public void generateOutput () throws IOExce
    PrintStream out = new PrintStream(new Fi                    ;
    out.println("<html>");
    for (String symbol: symbols) {
        Quoter q = new Quoter (symbol);
        q.obtainQuote();
        out.println(symbol + ": "
                    + "<i>opened at</i> " + q.getOpen ()
                    + "<i> and is currently trading at </i>");
        boolean bigChange = Math.abs (q.getChange()) >= 1;
        if (bigChange) out.println("<b>");
        out.println(q.getAsk ());
        if (bigChange) out.println("</b>");
        out.println("<br>");
    }
    out.close();
}
}
```

How would the RTF version differ? What's undesirable about this choice?

Information Hiding Techniques

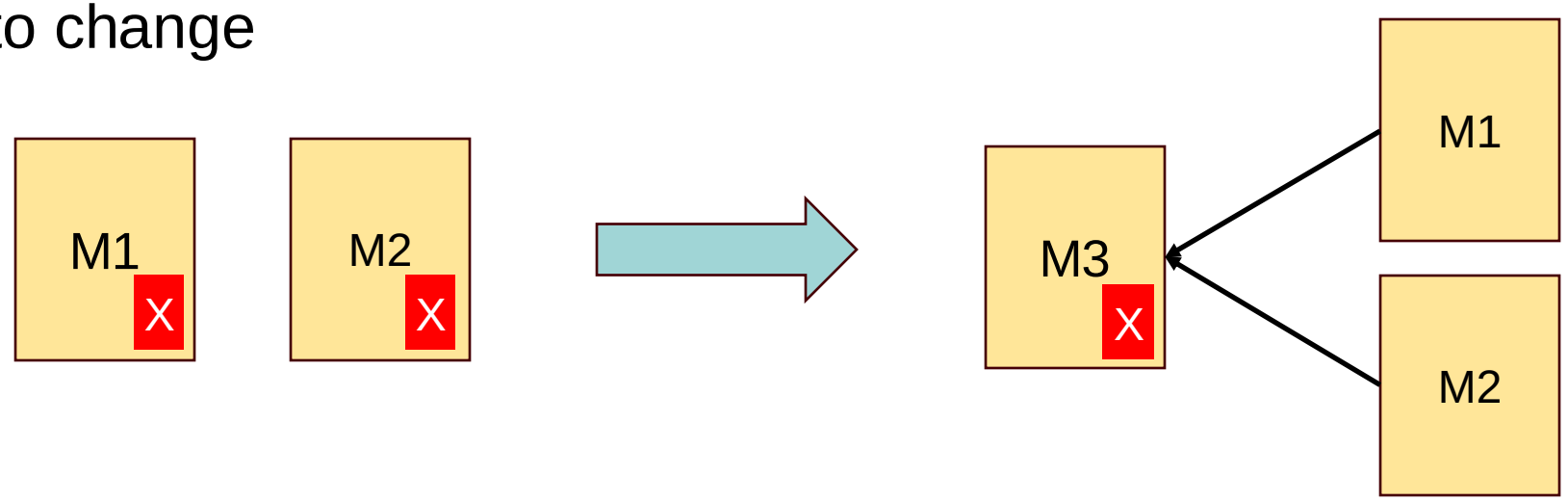
- **Data abstraction**
- **Interface abstraction**
  - A more general form of abstraction
  - Hide the **logic** of how a function/service is carried out behind an **interface**
  - When the logic changes, other components can continue to use that service without having to change



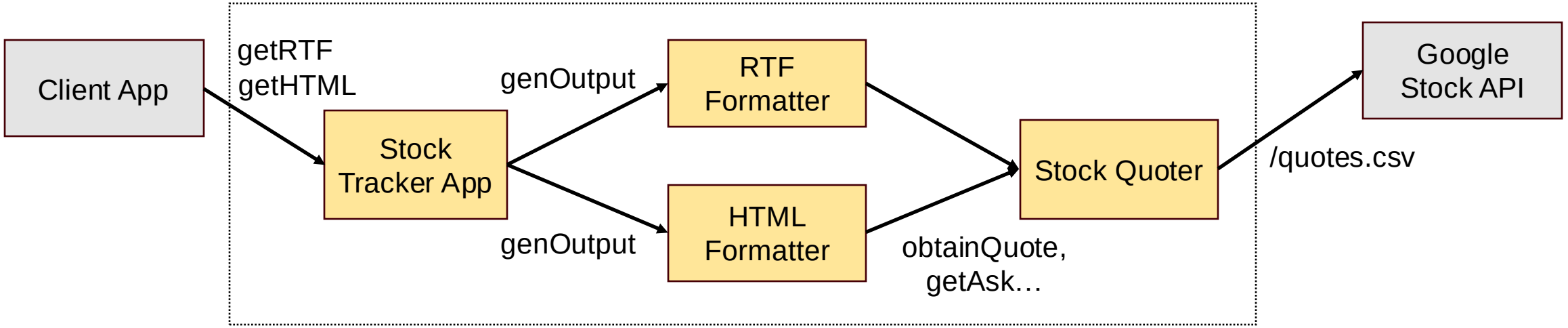
C and **M3** should be designed so that changing **X** does not affect **C**

Information Hiding Techniques

- **Data abstraction**
- **Interface abstraction**
- **Encapsulation**
  - When a design decision appears in multiple components, extract & isolate the decision (**secret**) into a single component
  - When the decision changes, only the encapsulating component (**M3**) need to change

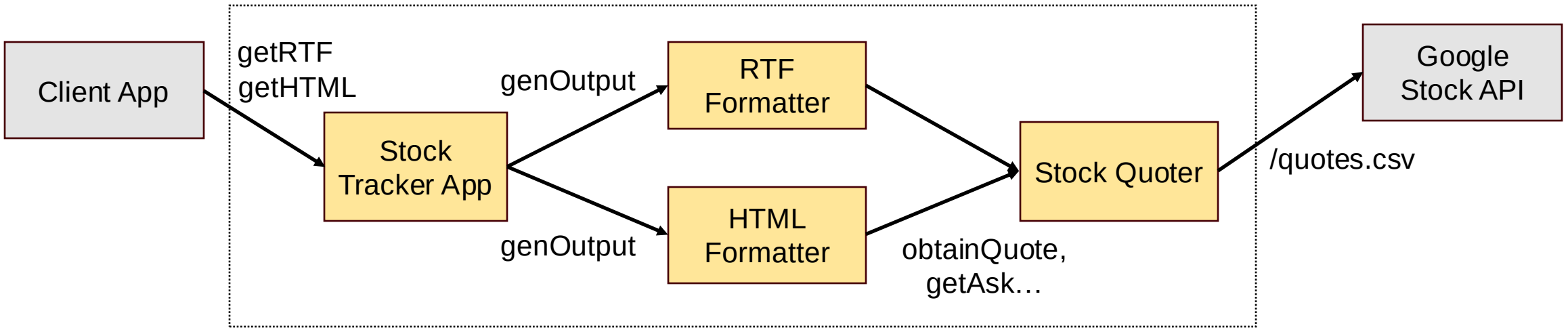


Stock Tracker App: Another Change



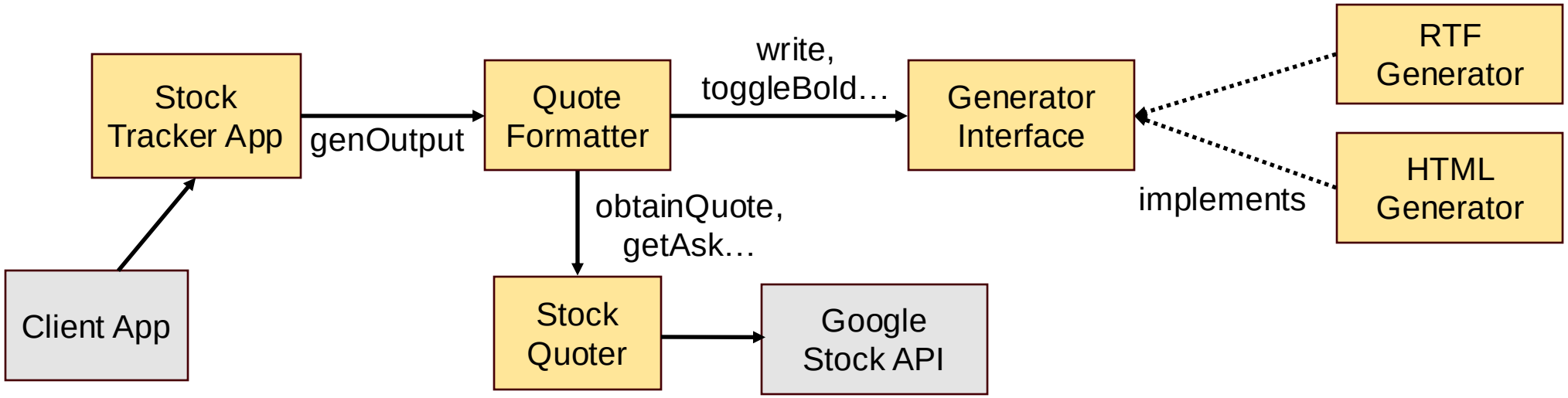
- Change in formatting requirement: If a stock changes >1% since the opening, *italicize* the quote instead of **bolding** it
- **Q. What is the impact of this change?**
- **Q. How would you change the design to reduce the impact?**

Shared Secret!



- **Problem**: HTML/RTF Formatters know (1) **how to** generate HTML/RTF elements in different formats and (2) **what** should be bolded, underlined, etc.,
- (2) is a design decision that can be separated & hidden from components that generate HTML/RTF!

Stock Tracker App: New Design



- HTML/RTF Generator: Writes & formats a given string using HTML/RTF tags
- Generator interface: Hides the type of output file (HTML/RTF) from the client
- Formatter: Encodes which part of the quote should be bolded, italicized; does not know anything about HTML/RTF!

RTF Generator

```
ic class RTFGenerator implements Generator {
ivate boolean italic;
ivate boolean bold;
private final String filename;
private PrintStream stream;

public RTFGenerator (String filename) {
    this.filename = filename; }
public void open() throws FileNotFoundException {
    FileOutputStream fos = new FileOutputStream (filen
    stream = new PrintStream(fos);
    stream.println ("{\\rtf1\\mac"); }
public void close() {
    stream.println ("}"); stream.close(); }
public void newLine () {
    stream.println ("\\"); }
public void toggleBold() {
    stream.println (bold ? "\\f\\b0" : "\\f\\b");
    bold = !bold; }
...
}
```


Q. How is RTF Generator different from RTF Formatter?

Encapsulation
RTF Generator doesn't know about how the stock quote should be formatted; that secret now belongs to Quote Formatter!

Generator Interface

```
face Generator {
oid open () throws Exception;
oid close ();
oid newLine ();
public void toggleBold ();
public void toggleItalic ();
public void write (String s);
}

public class RTFGenerator implements Generator {
  public void open() throws FileNotFoundException { ... }
  ...}

public class HTMLGenerator implements Generator {
  public void open() throws FileNotFoundException { ... }
  ...}
```

Interface abstraction
Hide details about the type of output (HTML or RTF) and how it's generated!

Quote Formatter

```
eFormatter {
    ng> symbols = r
    r generator;

    Generator generator)
    erator ;
}

public
    (String symbol) {
    ;
}

public void generateOutput () throws Except
    generator.open ();
    for (String symbol: symbols) {
        Quoter q = new Quoter (symbol);
        q.obtainQuote();
        generator.write (symbol + ": ");
        generator.toggleItalic ();
        generator.write ("opened at ");
        generator.toggleItalic ();
        ...
        generator.close();
    }
}
```


An implementor of the Generator interface

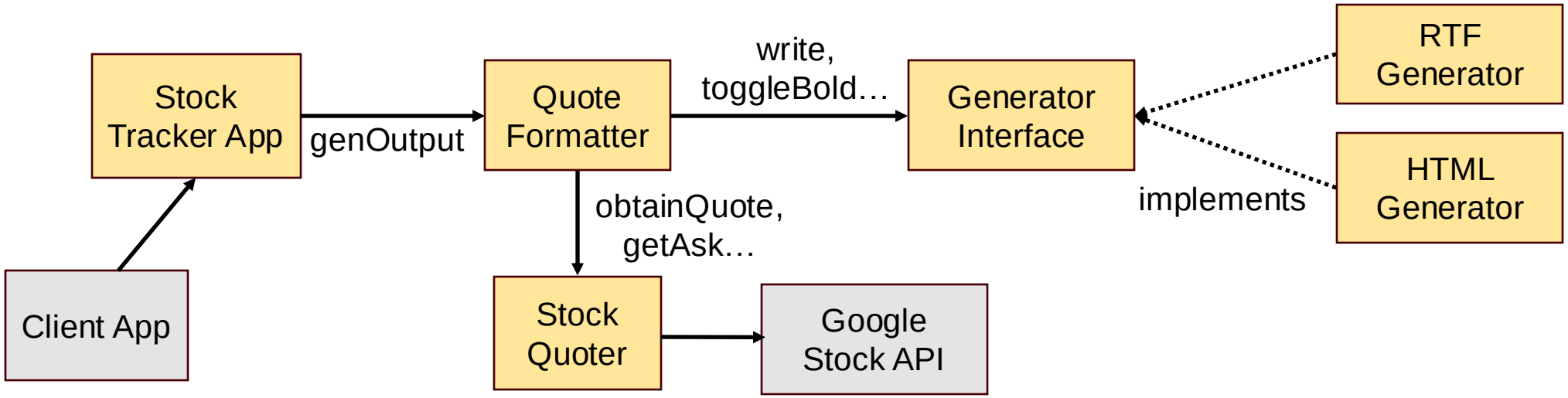
Decide how different parts of the quote should be formatted

But does not mention anything about HTML or RTF!

or is plugged into
atter

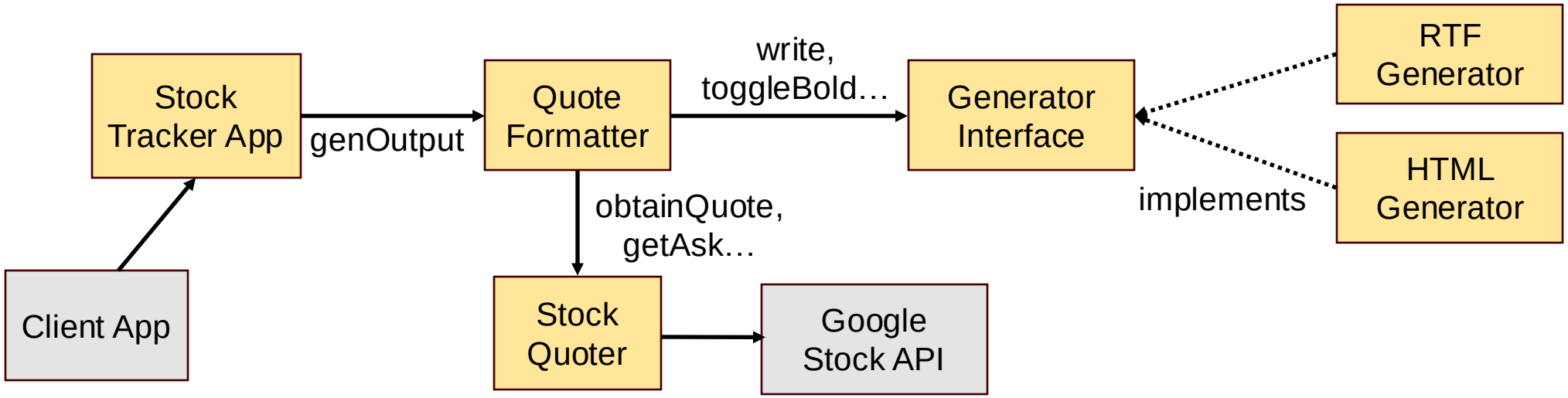
no mention of HTMLGener
or RTFGenerator anywhere!

Benefits of New Design



- Decisions about how quote parts should be formatted (bold vs. italicized) can be changed without affecting HTML/RTF Generators
- Different generators (e.g., JSON Generator) can be added/substituted without affecting Quote Formatter

Exercises



Q. Which modules do you need to change to:

- Handle new RTF syntax for italics?
- Put the ask price in bold if the stock went down since open?
- Use Bloomberg Finance instead of Google?
- Add year-to-date change to the report?

Information Hiding: Takeaway

Information Hiding: Takeaway

- Information hiding can help:
  - Reduce dependency between components
  - Improve the changeability of a system by limiting the impact of a change throughout the system
  - Allow different components to change independently
- Q. But what's the catch? What's challenging about applying information hiding?
- Not all changes can be isolated to a single component; some changes will involve changing a large portion of the system
- Not all changes can be anticipated
  - But planning for possible changes during design will make your system more ready than not!

Summary

- Exit ticket!