Code Concerns

JD Kilgallin

CPSC:480

10/10/22

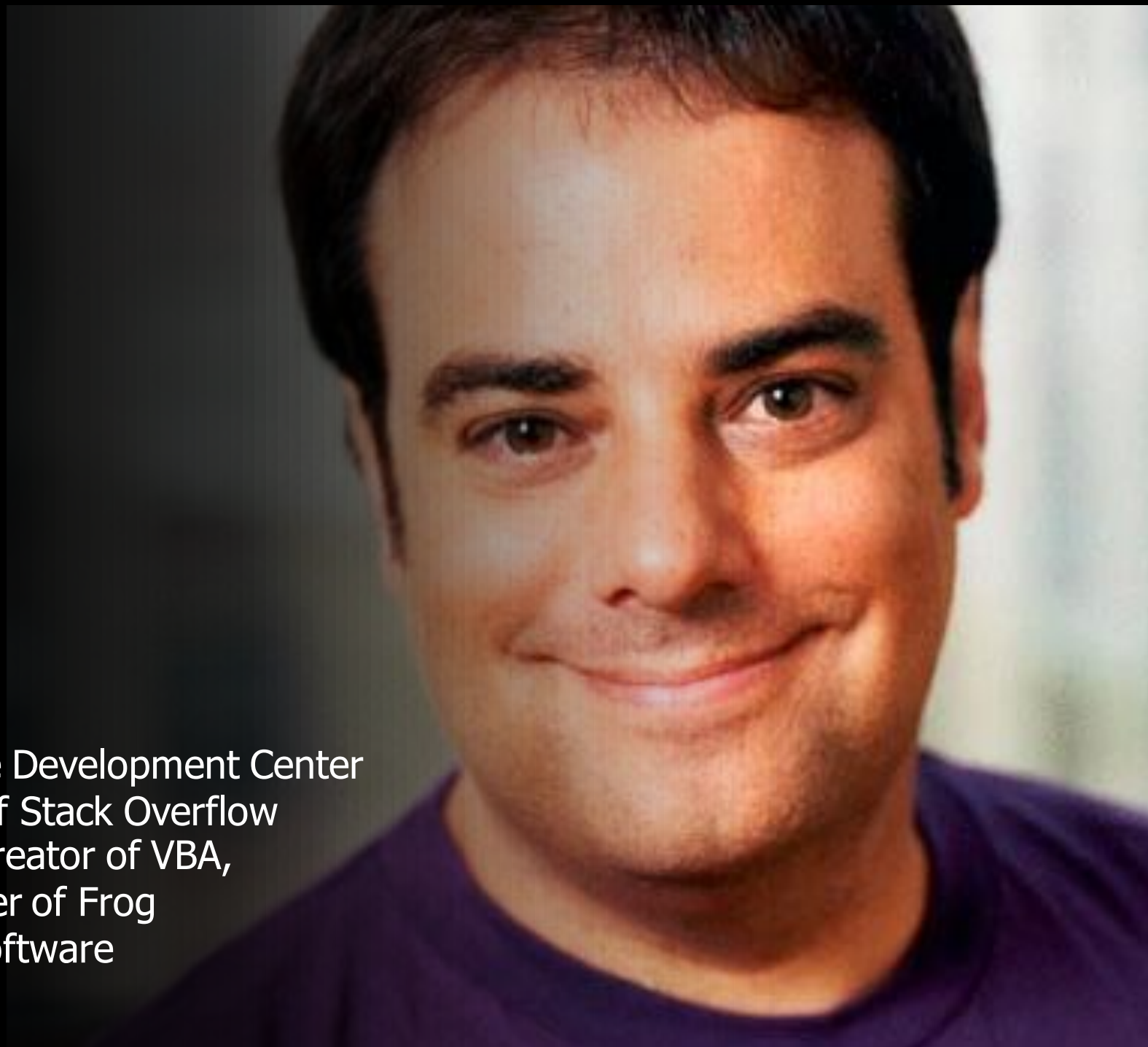

Photo: Joel Spolsky, Edison Software Development Center
Known for: Founder and Chairman of Stack Overflow and Stack Exchange, PM for Excel, creator of VBA, creator of Trello PM software, founder of Frog Creek Software, author of Joel on Software and many programming books

Code Concerns

JD Kilgallin

CPSC:480

10/10/22

Learning objectives

- Cross-cutting concerns

Concerns

- Recall: A *concern* is a feature or behavior of the software.
- *Separation of concerns* is the principle whereby a complex program should be broken into independent, manageable components.
- *Scattering* is a *coupling* violation that occurs when code related to a concern is fragmented across multiple locations, possibly duplicated.
- *Tangling* is a *cohesion* violation that occurs when one piece of code addresses multiple concerns.
- A *core concern* is a concern designed to address a functional requirement of the program. So data access, application logic, & user interface for a search feature would be core concerns of a Pokedex.
- A *cross-cutting concern* is a concern that affects other concerns and may be scattered and/or tangled with the rest of the program.

Cross-cutting concerns

- These may address non-functional or implicit requirements, or even requirements imposed by the software *vendor* rather than the user or customer.
- There may be a primary module associated with the concern, but other code related to the concern may be found in other modules.
- Portions of cross-cutting concerns may be addressed by the Operating System, runtime, standard libraries, or 3rd-party libraries/frameworks
- Portions may also be handled by automation (e.g. inserting trace logging at the beginning of each public method), called *Aspect-Oriented Programming.*
- May be intangible and not have *any* code specifically related to it.

Cross-cutting concerns – system

- Configurability – Parameters controlling operation of the software
- Logging – Recording actions taken by the software
- Observability – Allowing insight into the operation of the software
- Performance – Latency and throughput of the software
- Memory Management – Effectively maintaining data in memory

Cross-cutting concerns – integrity

- Persistence – Permanent storage of data
- Security – Protection against malicious actors and threats
- Privacy – Confidentiality of sensitive data and prevention of disclosure
- Data Validation – Ensuring data conforms to expectations
- Error detection – Identification of data inaccuracies or inconsistencies

Cross-cutting concerns – human factors

- Aesthetics – Appearance of user interface
- Localization – Translation of I/O to user language and conventions
- Accessibility – Ability for all users to operate the software
- Help – Ability to provide guidance on usage of the software in context
- Licensing – Adherence and enforcement of code terms and conditions
- Compliance – Conformance to applicable standards and regulations

Cross-cutting concerns – architecture

- Extensibility – Ability to easily enhance/extend program functionality
- Scalability – Ability to meet high usage demand
- Testability – Ability to write and run automated test cases on program

Configurability

- Ability to control operation of software with settings and options.
- Common methods for configuration include:
  - Command-line switches – Parse arguments from command line
  - Environment variables – System-wide parameters available to program
  - Registry – Windows-specific directory of key-value pairs
  - Config file – When there are too many arguments for command-line
  - Settings menus – In-application configuration of settings, stored either into a config file, the registry, or the application database
- Usually assigns values to variables used in place of hard-coding
- Especially important for software that runs cross-platform (e.g. "\" vs "/" in file paths), communicates with other systems, is accessed by many users, or otherwise has significant complexity.

Logging

- Writing a record of software activity to an external file or system.
- One primary beneficiary is the developer - Significant help with troubleshooting bugs or issues in either a production or non-production instance of the software.
- Also helps with observability, security and performance assessment, audits and compliance checking, and more.
- There are important security and privacy considerations with logging – sensitive info such as passwords should not be logged.
- There are also important localization considerations - logs may need to be read by users of multiple languages.
- May need to integrate with multiple other systems (log *targets)* to log events.
- Frequently handled by a dedicated library or framework (e.g. NLog, log4j), which can handle integration, log *rotation* (archiving one log file and starting a new one to limit file size), timestamps and formatting, log level changes, etc

Log levels

- Logging is usually implemented by passing a string to a "log" function, which appends the string to the log, possibly with metadata like the time, class/stack trace, etc. Log functions usually also take a *level*.
- Not all information needs to be logged at all times - excessive logging consumes disk space, and can make it *harder* to find important info.
- Log level can be changed at runtime, and all logs below configured level will be recorded. May also set *verbosity,* or amount in a message
- Common levels include the following.
  - Error – Incorrect behavior or state detected, such as an unhandled exception.
  - Warn – Undesired behavior that isn't an outright error.
  - Info (default) – Events of interest to an admin user during normal operation.
  - Debug – Additional detail about program state and function input/output
  - Trace – Maximum detail showing control flow between and within functions

Observability

- The ability to understand the operation of a program based on external output and interactions.
- Observable output includes responses to input, log files (or other targets like a system event log), system insights (e.g. CPU/memory usage and network traffic volume), and intercepted communications.
- *Health check* – Monitoring that a service is functioning and responding to requests as intended.
- *Fault monitoring* – Many software systems encounter routine errors *constantly*. Fault monitoring identifies spikes and trends in error rates that may require developer attention.
- *Performance counter* – An event registered with the system indicating that an action has been performed, giving visibility into usage patterns and throughput speed.

Performance

- The response time, measured via latency and throughput, to perform an operation as well as the resources required to do so.
- Significantly impacts user perception of software quality.
- May not have code that specifically relates to it, but is a measurable aspect of many pieces of code, especially code related to:
  - Algorithms – Choice of data structures and indexes plus algorithms for search, insertion, transformation, cross-reference, aggregation, etc.
  - Database access – Queries joining multiple records or accessing large datasets
  - Graphics – Sensitive to very small changes in amount of memory per-object
  - Network – High latency compared to local access. Sensitive to number of round-trips needed. Allowing batched data retrieval can be significantly faster.
  - Devices – Hard disks are slow, and good memory architecture can help greatly

Memory management

- The control of data storage and access by a program.
- Tasks include memory allocation (assigning and tracking a specific location in memory for a particular piece of data), caching (storing data in quickly-accessible locations), and garbage collection (unassigning memory when it is no longer needed).
- Heavily tied to programming language and runtime. Most modern languages do memory allocation and garbage collection automatically and abstract much of the management.
- The ability to explicitly cache data in memory can speed up an application by several orders of magnitude. In-memory databases like memcached and redis facilitate this. Web applications may use an *Application Delivery Controller* to handle data caching.

Persistence

- Storage of data in a manner that can be accessed after an application or system restart (aka "non-volatile data storage").
- Data that must be persisted includes information in the problem domain (application data), information in the infrastructure domain (program configuration), application logs, and sometimes snapshots of application state.
- Accomplished through file storage, relational databases like MySQL or Microsoft SQL Server (or "no-SQL" databases like MongoDB), cloud storage, or sometimes system storage like the Windows registry.

Security

- Precautions taken to limit the impact of a malicious actor on program operation, data storage and access, and ability to use a system.
- An important consideration for any multi-user application, with several aspects to consider. Most of them come down to preventing unintended data *access*, unintended data *manipulation*, or unintended *program execution*.
- Most enterprise applications will use "Role-Based Access Control" (RBAC), where users will be assigned one or more *roles* within the program, and a given role is mapped to permissions to perform certain actions.
- An application security model specifying the permissions that need to exist and when they will be checked is an important part of a product design document.

Access Control

- *Authentication* ("AuthN") is the process of establishing the identity of a user or process accessing a system or resource.
- *Authorization* ("AuthZ") is the process of establishing that the authenticated user has permission to take the requested action.
- For local system access, authentication is done at logon and authorization takes place in the context of the logged-in user (e.g. files on the file system allow different users to read and/or write)
- For remote system access, authentication is usually done with a set of HTTP headers using "Basic" (username+password), "Token" (a machine-readable format that represents a user for a limited duration), or "Certificate" (a digital file that uses cryptographic math to prove that a user has a key issued by a trusted party to that particular user).
  - HTTP 401 "Unauthorized" response code means *unauthenticated*
  - HTTP 403 "Forbidden" response code means *unauthorized*

Privacy

- The ability to prevent data from being accessed by unauthorized parties (aka data confidentiality)
- Requiring proper authentication and authorization before a program provides access to its data is the primary mechanism.
- Program must also take care that persisted data cannot be accessed by other unauthorized systems. *Encryption at rest* is the practice of encrypting persisted data in storage with a key that is only accessible by the program.
- Program must also take care that data communicated to another party cannot be accessed by intermediate or adjacent parties that may participate in the communication. *Encryption in transit* is the practice of encrypting communicated data so that it can only be accessed by the intended recipient.

Localization

- The process of making a program usable across multiple languages and cultures (aka "globalization")
- The biggest task is translating text to another language. It is common to have all static strings defined in one location to facilitate this, but extracting the strings that require translation can be automated.
- Software must be tested in each language – text of different lengths or using different character sets may cause display issues that do not occur in the original language.
- There are security considerations with languages that read right-to-left or that support Unicode (non-ASCII) characters.
- Displaying times in the local timezone can be a significant challenge that requires additional testing. Conventions for date formats (M/D/Y vs D/M/Y) also vary based on location.
- Important to note that keyboards in different locations (or even different vendors) have different keys, which complicates keyboard shortcuts.

Accessibility

- The practice of making software usable regardless of impairments.
- An important consideration for perception of product maturity, and in some cases a legal requirement (Section 508 of the U.S. Rehabilitation Act requires all federal agencies to make systems fully accessible).
- In many cases, accessibility is handled by other software.
- Important considerations for applications include:
  - Color-blindness (color shouldn't be the **only** factor that conveys information)
  - Blindness (all images should have an accompanying alt-text description)
  - Deafness (all audio should have a written transcription or description)
  - Limited mobility (applications should not require precise mouse movements and should offer navigation through standard keyboard shortcuts)
  - Slow reading (should be able to proceed through program at user's own pace)

Licensing

- Requirements placed on the use and modification of source code or compiled software, and the compliance and enforcement of a license
- License terms may make some software unusable for a given purpose:
  - May require written consent and/or payment for commercial use. Under U.S. copyright law, this is the "default" case for material on the web that does not specify other terms.
  - May require open-sourcing any software that uses it. This is called a "copyleft" license, and the primary example is the GNU Public License (GPL).
  - May require that any software that uses it has similar license terms. This is the case for the Creative Commons Share-Alike license.
- License terms may require attribution of the original source, and/or may require distribution of license terms along with the software that uses it. This is good practice for all software regardless of requirement
- Code you write and distribute should generally have a license that at least disclaims warranty for correctness and suitability, to limit liability.

Compliance

- Adherence to standards and regulations that apply to a program
- Enforceable license terms are one class of compliance requirements.
- Many government agencies, contractors, and other trade groups require compliance with one or more sets of guidelines that covers software product standards. "Statement of Controls" (SOC) and ISO 27001 are two standards that may apply. Payment Card Industry (PCI) sets standards for programs that accept credit card payments.
- Companies, teams, and individual products may be certified as compliant with one or more applicable standards.
- Most standards ensure software is secure, usable, and interoperable.

Architecture

- Extensibility, Scalability, and Testability are all related concerns that amount to a modular software design with well-defined interfaces between modules.
- Extensibility amounts to the ability to register a new implementation of one module's interface that performs new functionality.
- Scalability amounts to the ability to launch multiple instances of one module, and have other modules select an instance of that module that is ready to handle a request.
- Testability amounts to the ability to test one module in isolation by replacing the modules it interfaces to with *mock* components that use pre-defined data and behavior to simulate the real components.

References

- Joel on Software. Joel Spolsky. 1999-present.
- On the Role of Scientific Thought. Edsger Dijkstra. Selected writings on Computing: A Personal Perspective. Springer.
- The Lost Art of Seperating Concerns. Mark Baker. Dec 2006. InfoQ.
- Cross-cutting concern. Wikipedia.
- About Performance Counters. Karl Bridge et al. June 2022. Microsoft.
- Coding Standards and Techniques. JD Kilgallin et al. June 2019. Keyfactor.

- *Bring a laptop to class Wednesday*