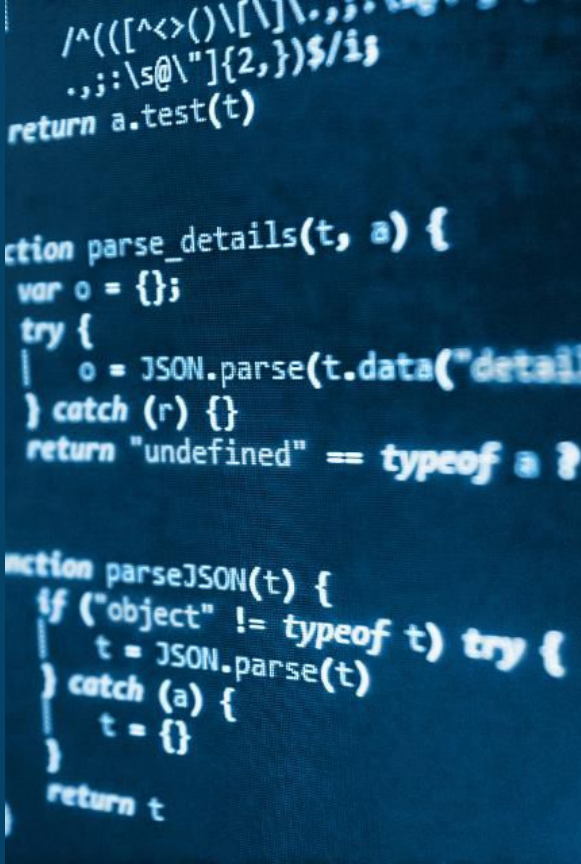
# Miscellaneous Topics

## Course on Advanced Programming Techniques

David Rebollo-Monedero
Senior Researcher in Privacy Technologies and Visiting Professor

Department of Computer Engineering and Mathematics
Rovira i Virgili University

david.rebollo@urv.cat



# Design and Analysis of Data Structures and Algorithms



- Computational complexity
  - We often encounter algorithms with counterintuitive, nonlinear complexity.
  - More precisely, algorithms often exhibit superadditive complexity $t(m+n) \geqslant t(m) + t(n)$,
  - for example quasilinear $\Theta(n \log n)$ or quadratic $\Theta(n^2)$.
  - With linear complexity $t(n) \propto n$, $t(n/2) = t(n)/2$, however, with quadratic complexity $t(n) \propto n^2$,
    - $t(n/2) = t(n)/4$, much faster than what our intuition might have led us to believe.
    - Further, if $t(n) = 1$ minute, then $t(100n) = 10000$ minutes $\approx 1$ week.
  - The "divide and conquer" principle of breaking down algorithms into subroutines, typically recursively, is an extremely powerful technique,
  - particularly with the aid of dynamic programming.
  - The master theorem enables us to analyze recursive complexities of the form

$$t(n) = at(n/b) + f(n),$$

    where $f(n)$ denotes the cost incurred by splitting and recombining the corresponding parts.

- References
  - T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed. Cambridge, MA: MIT Press, 2009.
  - "Design and analysis of algorithms", Stanford University, graduate course CS161.

# Software and Application Frameworks



- A software framework is a concrete or conceptual platform where common code with generic functionality can be selectively specialized or overridden by developers.
  - Frameworks take the form of libraries, where a well-defined application program interface (API) is reusable anywhere within the software under development.
  - A developer can extend the framework by selectively replacing default code providing generic functionality with user code satisfying application-specific needs.
- For example, the Microsoft foundation class library (MFC) is a C++ object-oriented library for developing desktop applications for Windows, particularly when you need to develop complex, Office-style user interfaces.
- References
  - P. García López, "Frameworks (toolkits)", 2018–2019 Course Slides, tap11.pdf.

# Component-Based Software Engineering



- Component-based software engineering (CBSE), also called components-based development (CBD), is a methodology that accentuates the design and development of computer-based systems with the help of reusable software components.
  - With CBSE, the focus shifts from software programming to software system composing, that is, developing software systems by choosing ideal off-the-shelf components and then assembling them using a well-defined software architecture.
  - A component could be software package, a web service, a web resource, or a module that encapsulates a set of related functions or data.
  - It aims for efficient software development, in terms of quality, productivity, and maintainability, leveraging reuse and standardization.
- References
  - P. García López, “Component software”, 2018–2019 Course Slides, tap12.pdf.

# Software Metrics (1/2)



- Quantitative analysis of software development, including effort and quality, may help anticipate the cost and schedule of most technology projects.
- Aside from computational metrics of algorithmic efficiency in terms of memory and speed, we are interested in
  - the complexity of source code with regard to development, testing, and maintenance by human programmers,
  - as well as its quality in terms of faults (bugs) and their impact in functionality.
- A simple example of software metric is the ratio faults / lines of code (LoC),
  - which implicitly but directly employs the length of the source code, be it with or without comments, as a measure of its complexity.
  - However, far more sophisticated metrics of the complexity of source code exist, most of them developed since the 70s.

# Software Metrics (2/2)



- Purdue University professor Maurice Howard Halstead introduced several empirical laws with intuitive justification, such as the estimated length of program in terms of the number of distinct operators and operands it contains.
  - This may be thought of as the problem of estimating the relationship between the length of a given text and that of its vocabulary, where some words will be repeated more often than others.
  - Let $\eta_1$ and $\eta_2$ denote the number of distinct operators and distinct operands in a program. Halstead proposed the following estimate $\widehat{N}$ for the total length $N$ of the program, in terms of repeated operators and operands:

$$\widehat{N} = \widehat{N}_1 + \widehat{N}_2, \text{ where } \widehat{N}_1 = \eta_1 \log_2 \eta_1 \text{ and } \widehat{N}_2 = \eta_2 \log_2 \eta_2 .$$

  - From those sizes, he proposed certain measures of difficulty and effort of a program.
- Thomas J. McCabe considered the representation of the control flow and decision points of a program as a graph, and proposed to measure its complexity, which he termed *cyclomatic* complexity, according to the topology of the corresponding graph, and its number of nodes and edges.
- References
  - P. García López, “Software metrics”, 2018–2019 Course Slides, tap_metrics.pdf.

# Zipf's Law



- Halstead's metrics immediately raise the following questions:
  - What is the reason behind the estimate $\widehat{N}_i \approx \eta_i \log \eta_i$ for both operators and operands?
  - More generally, how are words in a vocabulary statistically distributed in a piece of text?
  - In particular, what is the quantitative relationship between the size of a given text and the size of its vocabulary?

- Zipf's experimental law was originally formulated in terms of quantitative linguistics,
  - stating that the frequency of a word is inversely proportional to its rank in the frequency table.
  - Thus, the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, and so on.

- In the Brown Corpus of American English text,
  - the word "the" is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million).
  - True to Zipf's Law, the second-place word "of" accounts for slightly over 3.5% of words (36,411 occurrences),
  - followed by "and" (28,852).
  - Only 135 vocabulary items are needed to account for half the Brown Corpus.

# Interpretation of a Halstead Metric from Zipf's Law (1/2)



- Let $H_m$ represent the $m^{\text{th}}$ harmonic number

$$H_m = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{m} = \sum_{i=1}^{m} 1/i\,.$$

  - The approximation of the harmonic sum by an integral, with $\ln x$ as the primitive integral of $1/x$, leads to the limit

$$\lim_{m\to\infty} H_m - \ln m = \gamma_0$$

  where $\gamma_0 \approx 0.5772$ is the Euler-Mascheroni constant. We may thus approximate

$$H_m \approx \gamma_0 + \ln m \approx \ln m\,.$$

- Consider a vocabulary of (distinct) words indexed by $i = 1, \ldots, m$ represented by outcomes of a random variable (r.v.), and a text of (typically repeated) words indexed by $j = 1, \ldots, n$, modeled as independent, identically distributed drawings of said r.v.
  - Suppose that the distribution of indexed words follows Zipf's experimental law with exponent parameter 1, i.e., it is distributed according to

$$p_i = \frac{1/i}{\sum_{i=1}^{m} 1/i} = \frac{1/i}{H_m}.$$

  - If the absolute frequency $k_m$ of the least common word $i = m$ is known and small, say $k_m = 1$, and its relative frequency $p_m$ follows the above law, since $p_i = k_i/n$, then $n = 1 \,/\, p_m$ and consequently,

$$n = mH_m \approx m(\ln m + \gamma_0) \approx m \ln m\,.$$

# Interpretation of a Halstead Metric from Zipf's Law (2/2)



- Incidentally, the Shannon entropy of such Zipfian r.v. with parameter 1 is

$$S = \frac{1}{H_m} \sum_{i=1}^{m} \frac{\ln i}{i} + \ln H_m \text{ nats,}$$

  - which can be expressed in terms of the Stieltjes constant $\gamma_1 \approx -0.0728$, and where the summation can be roughly approximated by $(\ln m)^2/2 + \gamma_1$,
  - and the entropy, by

$$S \approx \frac{1}{\ln m + \gamma_0} \left( \frac{\ln^2 m}{2} + \gamma_1 \right) + \ln(\ln m + \gamma_0) \approx \frac{\ln m}{2} + \ln \ln m \,.$$

  - This means that a text of size $n$ would have a compressed length $nS = \Theta(n \log m)$, roughly half of that required if words were uniformly distributed.

# Thank you

David Rebollo-Monedero
Senior Researcher in Privacy Technologies and Visiting Professor

Department of Computer Engineering and Mathematics
Rovira i Virgili University

david.rebollo@urv.cat