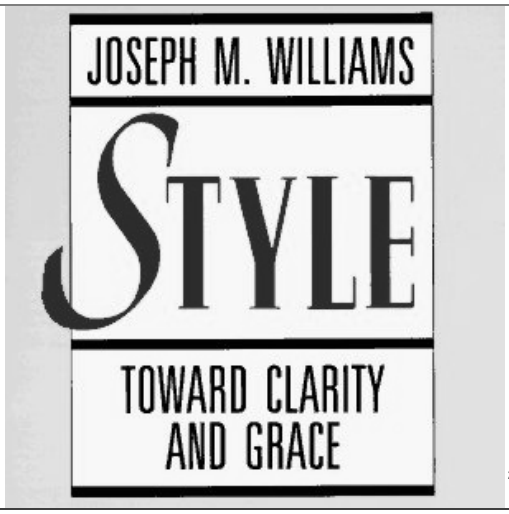
On Writing Well

William Cook
University of Texas at Austin

based on
Style: Toward Clarity and Grace
by Joseph Williams



- Subjects of sentences name cast of characters
- Verbs name actions involving characters

Missing Subjects

"Termination occurred after 23 iterations"

"The program terminated after 23 iterations"

Weak Verbs

"The algorithm supports effective garbage collection in distributed systems"

Stronger Verbs

"The algorithm collects garbage effectively in distributed systems"

Verb Nominalization

Verb	*Nominalization*
discover	discovery
move	movement
collaborate	collaboration

Adjective Nominalization

Adjective	*Nominalization*
difficult	difficulty
applicable	applicability
different	difference

empty verb + NOM

"The police conducted an investigation of the matter"

9

Verb=Action

"The police investigated the matter"

10

"there is" + NOM

"There is a need for further study of this program"

11

Name the Actor

"The engineering staff must study this program futher"

12

Using "how"

"She reviewed how the technique evolved"

13

NOM + verb + NOM

"Extensive rust damage to the hull prevented repairs to the ship"

14

"Because rust had damaged the hull, we could not repair the ship"

15

Useful Nominalizations

16

Name a an action

"I do not understand her meaning or his intention"

(what she means what he intends)

17

Common concepts

"Taxation without representation was not the central cause of the revolution"

18

compilation

dependency

inheritance

implementation

19

Noun+Noun+..

Early childhood thought disorder misdiagnosis often occurs in state-funded labs.

Noun+Noun+..

State-funded labs often misdiagnosis disordered thought in young children.

Concision

- It is imperative to think over in a punctilious manner each and every suggestion that is offered to us.

Concision

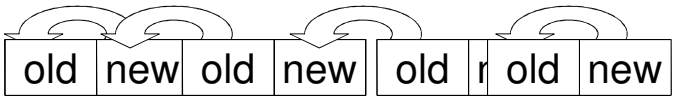
- Consider each suggestion carefully.

Sentences

subject	• ideas already mentioned • familiar ideas
verb	•action
object	•new ideas

24

Topics form a logical sequence of ideas



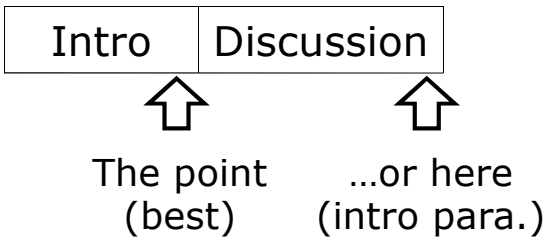
Active

"Our partners were old friends… but they let us down. The partners broke the agreement."

Passive

"We thought we had a good agreement. Then we found out who killed it. The agreement was broken by the partners."

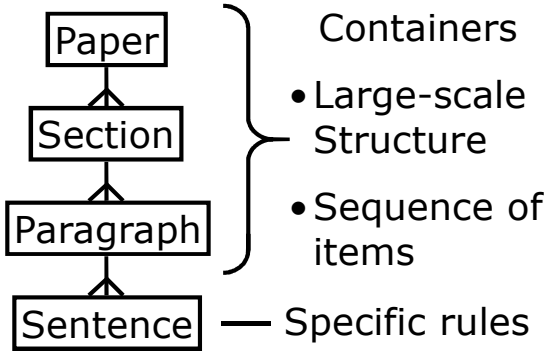
The Point



Emphasis

Put important things at the end

sentence	final words
paragraph	last sent.
section	last para.

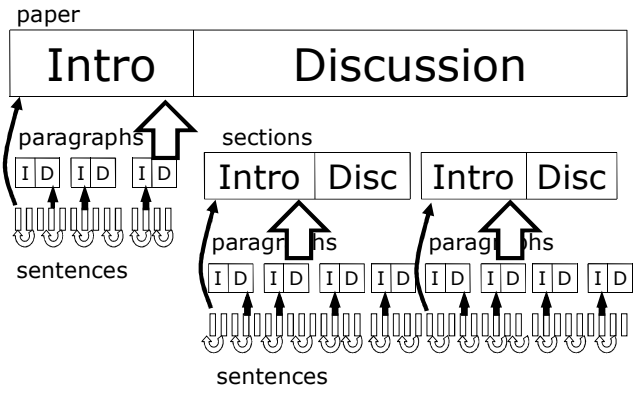


Section Titles

First sentence (or the point)

m**ust include all words in section title**

(except intro/conclusion)



Before

Problems such as the design of distributed controllers are characterized by modularity and symmetry. However, the symmetries useful for solving them are often difficult to determine analytically. This paper presents a nature-inspired approach called Evolution of Network Symmetry and mOdularity (ENSO) to solve such problems. It abstracts properties of generative and developmental systems, and utilizes group theory to represent symmetry and search for it systematically, making it more evolvable than randomly mutating symmetry. This approach is evaluated by evolving controllers for a quadruped robot in physically realistic simulations. On flat ground, the resulting controllers are as effective as those having hand-designed symmetries. However, they are significantly faster when evolved on inclined ground, where the appropriate symmetries are difficult to determine manually. The group-theoretic symmetry mutations of ENSO were also significantly more effective at evolving such controllers than random symmetry mutations. Thus, ENSO is a promising approach for evolving modular and symmetric solutions to distributed control problems, as well as multiagent systems in general.

33

After

The design of distributed controllers to coordinate functional units often involves symmetric relationships between modular components. For example, the movement of joints of a quadruped robot on an unknown terrain exhibit many degrees of symmetry. However, the symmetries useful is such controllers are often difficult to determine analytically. Inspired by nature, we have developed Evolution of Network Symmetry and mOdularity (ENSO) to generate controllers automatically. ENSO abstracts properties of generative and developmental systems and utilizes group theory to represent symmetry and search for it systematically. We evaluate ENSO by evolving controllers for a quadruped robot in physically realistic simulations. On flat ground, the resulting controllers are as effective as those having hand-designed symmetries. However, they are significantly faster when evolved on inclined ground, where the appropriate symmetries are difficult to determine manually. The group-theoretic symmetry mutations of ENSO were also significantly more effective at evolving such controllers than random symmetry mutations. Thus, ENSO is a promising approach for evolving modular and symmetric solutions to distributed control problems, as well as multiagent systems in general.

34

Before

The main theme underlying my research is how to build a seamless computing environment so that users can access their data and perform “computing” from any network-connected device at any location. Personal and mobile environments are characterized by heterogeneity and changes in device resources, network availability, bandwidths, location, user preference, and user mobility. In order to be applicable in such diverse environments, it is imperative for software systems to be adaptable. My work has focused on building flexible infrastructures that make it easier to build adaptable systems.

In doing so, I have explored the fields of distributed systems, software architecture, data replication, and declarative languages.

35

After

The focus of my research is on changing the way that large distributed systems are developed. Rather than code each component in low-level procedural language using primitive concurrency and communication libraries, my research demonstrates that complete systems can be generated from high-level descriptions of the overall system behavior, which are automatically compiled into efficient code for individual distributed components. My approach reduces a 30,000 line program to 500 lines of code, while preserving the same level of performance. The techniques I have developed focus on the hard problem of fault tolerant data replication, which is fundamental to the emerging paradigm of cloud computing, in which users access heterogeneous services via a wide range of mobile devices over unreliable networks with dynamic bandwidth and latency constraints.

36

Before

The quality of error reporting of modern software has decreased as its complexity has increased. End-users take the cryptic error messages given to them by programs and struggle to fix their problems using search engines and support websites. Developers cannot improve their error messages when they receive an ambiguous or otherwise insufficient error indicator from a black-box software component.

37

After

An error occurs when software cannot complete a requested action as a result of some problem with its input, configuration, or environment. A high-quality error report allows a user to understand and correct the problem. But the quality of error reports has been decreasing as software becomes more complex and layered. End-users take the cryptic error messages given to them by programs and struggle to fix their problems using search engines and support websites. Developers cannot improve their error messages when they receive an ambiguous or otherwise insufficient error indicator from a black-box software component.

38

Before

Virtual classes are class valued attributes of objects, and they can be refined to subclasses in context of an instance of a subclass, in a similar manner as mainstream OO languages enable (virtual) methods to be redefined in an instance of a subclass. This paper formalizes the notion of virtual classes which is at the core of the semantics of languages supporting higher-order hierarchies and family polymorphism such as Caesar and gbeta. The main contribution of this work is the extraction and presentation of a simple calculus, vc, which faithfully reproduces the core structure and properties of virtual classes in the abovementioned full-fledged languages, along with a proof of its soundness.

39

After

Virtual classes are class-valued attributes of objects. Like virtual methods, virtual classes are defined in an object’s class and may be redefined within subclasses. They resemble inner classes, which are also defined within a class, but virtual classes are accessed through object instances, not as static components of a class. When used as types, virtual classes depend upon object identity – each object instance introduces a new family of virtual class types. Virtual classes support large-scale program composition techniques, including higher-order hierarchies and family polymorphism. The original definition of virtual classes in BETA left open the question of static type safety, since some type errors were not caught until runtime. Later the languages Caesar and gbeta have used a more strict static analysis in order to ensure static type safety. However, the existence of a sound, statically typed model for virtual classes has been a long-standing open question. This paper presents a virtual class calculus, vc, that captures the essence of virtual classes in these full-fledged programming languages. The key contributions of the paper are a formalization of the dynamic and static semantics of vc and a proof of the soundness of vc.

40