Larry Wall：語言多義性、後現代實用主義與社群文化

Larry Wall: Linguistic Polysemy, Postmodern Pragmatism, and Community Culture

v1.0 / 2026-07-30 / Neo.K / 公開版／第三部設計師個案正式研究

https://thisoneisneok.com/html/papers/pldst-019.html

英文名稱： Larry Wall: Linguistic Polysemy, Postmodern Pragmatism, and Community Culture

系列： Programming Language Designer Style Taxonomy（PLDST）

文件編號： PLDST-019

版本： v1.0

日期： 2026-07-30

作者： Neo.K

文件狀態： 公開版／第三部設計師個案正式研究

SECTION

摘要

Larry Wall 常被描述為 Perl 的創造者、自然語言式程式設計的倡議者，以及「There’s more than one way to do it」的代表人物。這些描述正確，卻也容易把 Perl 寫成一門只追求短碼、容許混亂，甚至故意反對一致性的語言。

Perl 的起點其實極為具體。Wall 在 1987 年從事系統管理與跨機器報告工作時，需要整合：

- Shell 的程序與管線能力；
- awk 的欄位及文字處理；

- sed 的替換；
- grep 的模式匹配；
- C 的系統介面和效率；
- Unix 工具鏈的可組合性。

他沒有嘗試建立一個純粹、最小、形式上完全正交的新世界，而是把多種已證實有效的語言慣用法組合成一門能快速完成真實工作的語言。Perl 官方歷史將 Perl 1 的公開發布記為 1987 年 12 月，後續版本持續增加正規表示式、二進位資料、Module、Reference、Object、Unicode 等能力。[R1][R2]

Wall 對語言的理解強烈受到語言學影響。他反覆把 Perl 與自然語言比較：自然語言不是因為完全無歧義才有力量，而是因為具有 Context、冗餘、不同表達層級、短常用形式和長精確形式，能映射異質現實。他甚至把 Perl 稱為「第一門後現代電腦語言」，並批評某些語言設計者把現實複雜性掃到程式設計者的地毯下。[R3][R4]

本文將 Wall 的設計生涯分成六個相位：

- Unix 問題整合期：從 Shell、awk、sed、C 與文字處理建立 Perl；
- 自然語言式實用核心期：Context、Sigil、Regex、預設變數與多種表達方法；
- Perl 5 模組化與社群擴張期：Reference、Package、Object、CPAN 與 Pumpkin maintainer；
- 文化自我描述期：TIMTOWTDI、三大美德、Postmodern language 和 State of the Onion；
- Perl 6／Raku 重寫期：RFC、Apocalypse、Grammar、Role、多 Dispatch 與社群共同重設；
- 後創始者治理與語族分流期：Perl Core Team／Steering Council、Raku 改名及各自制度。

本文核心判斷為：

> 【Wall 的多義性不是拒絕規則，
>
> 而是拒絕假設所有問題與所有人都只有一種自然表示。】

其深層配置可以表示為：

> 【Context-sensitive expression
> +
> Common-case compression

這種設計把複雜度配置到不同位置。對熟悉 Perl 的使用者而言，常見文字處理、資料轉換及系統工作可以高度壓縮；Interpreter 承擔 Context、Dynamic conversion、Regex、Default variable 及大量語法分析。然而，讀者和工具必須理解：

- Scalar／List context；
- Sigil；
- Implicit $_；
- Operator precedence；
- Regex；
- Dynamic scope 歷史；
- 多種等價慣用法；
- 特殊變數；
- Module 和版本差異。

  因此：

> 【作者表達自由
> not⇒
> 讀者理解成本自動降低】

Wall 的設計並非不知道這項代價。他以文件、文化格言、use strict、Warning、Module、社群慣例和後來 Perl 6 的重新設計來治理自由。問題在於，文化治理不像型別規則那樣可由 Compiler 一致強制；它依賴共同體對「好 Perl」的學習。

Perl 6／Raku 又展示其風格的第二層。Wall 在 2000 年宣布不再只修補 Perl 5，而要由社群重寫 Perl 和社群本身；數百份 RFC 被收集，再由 Wall 以 Apocalypse 系列重新綜合。[R5][R6] 這是一場比 Perl 5 更系統、更語義化的設計，但長期開發、相容斷裂與命名混淆也形成巨大成本。2019 年 Perl 6 正式更名為 Raku，使兩個語言及社群得以分開演化。

因此，Wall 的風格不是「越亂越好」。更精確的判定是：

> 他把語言視為文化、生態與人類表達系統，願意用 Interpreter 和社群承擔複雜性，換取使用者對異質問題的高表達自由；當早期自由累積成歷史負擔時，他又願意發起一次幾乎重寫語言本體的共同體實驗。

關鍵詞： Larry Wall、Perl、TIMTOWTDI、自然語言、後現代程式語言、Context、Regex、CPAN、Perl 6、Raku、PLDST

SECTION

一、本文研究範圍

本文主要分析：

- 1987 年 Perl 問題形成；
- Perl 1–4 的文字和系統腳本核心；
- Perl 5 的 Reference、Module、Object 和 CPAN 生態；
- Wall 的語言學、後現代及文化論述；
- Perl 6／Raku 的設計與制度；
- Perl 後創始者治理。

本文不把以下成果全部歸入 Wall：

- Henry Spencer 的 Regex package；
- Perl 5 Core 的全部演化；
- CPAN 所有 Module；
- use strict、Moose 等所有社群工具；
- Parrot、Pugs、Rakudo 的完整實作；
- Raku 所有語言特徵；
- 現代 Perl Steering Council 的決策。

SECTION

二、Wall 的創始權重

Wall 可直接歸因的工作包括：

- Perl 原始問題設定；
- 初代 Interpreter；
- 語法與 Context 模型；
- 正規表示式整合；
- Perl 5 的總體重設；
- 多項文化格言；
- Perl 6 初始方向、Apocalypse 和長期語言裁決。

  稱其為 Perl 創造者完全合理。

SECTION

三、Perl 很快成為多人系統

官方 perlhist 記錄大量 Release maintainer 和貢獻者。Perl 5 的長期實作、維護及發布由 Perl 5 Porters 共同完成；不同時期的「Pumpkin holder」承擔版本整合。[R1]

重要共同主體包括：

- Randal Schwartz、Tom Christiansen、Jon Orwant、brian d foy 等文件作者；
- Henry Spencer 的 Regex 工作；
- Malcolm Beattie、Chip Salzenberg、Tim Bunce 等 Core 貢獻者；
- CPAN 作者；
- Perl 5 Porters；

- Audrey Tang 與 Pugs 社群；
- Parrot 團隊；
- Rakudo／Raku 團隊；
- 現代 Core Team 與 Steering Council。

因此：

Perl 原始語言：Wall 極高
Perl 5 方向：Wall 高，Core 實作多人
CPAN 生態：社群
Perl 6 方向與綜合：Wall 高
Raku 實作及後期演化：多團隊
現代 Perl 治理：Core Team／Steering Council

SECTION

四、Perl 的原始任務

Wall 的工作需要：

- 從大量檔案抽取資訊；
- 在不同機器和網路間生成報告；
- 處理不規則文字；
- 快速建立一次性及長期腳本；
- 存取 Unix 系統；
- 避免反覆使用多個工具和臨時檔案。

這不是語言學實驗先行，而是系統管理摩擦先行。

SECTION

五、為何不只用 Shell

Shell 擅長：

- 啟動程序；
- 管線；
- 檔案；
- 環境。

但大型資料結構、複雜文字分析及錯誤處理會變得困難。

SECTION

六、為何不只用 awk、sed、grep

這些工具對特定文字工作極強，但：

- 每個工具有自己的語言；
- 狀態和資料結構有限；
- 複雜流程需外部 Shell；
- 系統呼叫與一般程式能力不足。

SECTION

七、為何不直接用 C

C 可以完成所有工作，但：

- 開發慢；
- 字串和 Regex 成本高；
- 記憶體管理繁瑣；
- 一次性管理任務不值得建立完整程式。

SECTION

八、Perl 的第一個設計方程

> Perl
> ≈
> ShellProcess
> +
> AwkData
> +
> SedTransformation
> +
> GrepPattern
> +
> CSystemAccess

這不是精確語法來源清單，而是問題能力組合。

SECTION

九、自然語言不是仿英文

Wall 的語言學觀不是把程式寫成普通英文，而是吸收自然語言的深層特性：

- Context；
- 省略；
- 冗餘；
- 不同詞類；
- 常見形式較短；
- 可從簡單逐步學到複雜；
- 同一意思有多種語氣和結構。[R4]

SECTION

十、Context

Perl 表達式可在：

- Scalar context；
- List context；
- Boolean context；
- Void context；

  中產生不同結果。

  例如同一輸入操作在 Scalar context 可能取得一項，在 List context 可能取得整體集合。

  這提高表達壓縮，也要求讀者理解周圍語法。

SECTION

十一、Sigil 作為詞類線索

Perl 使用：

- $ Scalar；
- @ Array；
- % Hash；
- & Subroutine；
- * Typeglob。

  官方資料甚至把 $ 類比為英文的 “the”，表示正在取得單一值；@ 類似 “these／those”，表示多值。[R7]

  Sigil 的作用不是裝飾，而是將資料角色放到名稱表面。

SECTION

十二、預設主題 $_

許多 Perl 操作可以省略明顯主題：

```
while (<>) {
  chomp;
  print if /error/;
}
```

這類寫法像自然語言省略已知主詞。

優勢：

- 短；
- 突出操作；
- 適合管線。

風險：

- 非局部依賴；
- Nested scope 可能混淆；
- 新讀者不知道操作對象。

SECTION

十三、Regex 作為第一級語言

Perl 將 Pattern matching 與 substitution 放入語言核心，而不只作外部 Library。

這讓文字工作高度直接：

- Match；
- Capture；
- Substitute；
- Transliterate；
- Split。

但 Regex 自己是一個密集 DSL，形成「語言中的語言」。

SECTION

十四、Huffman coding 式表面

常見操作應較短，罕見操作可以較長。

可表示為：

> SyntaxLength(operation)
> ∝
> (1)/(Frequency(operation))

這提高熟練使用者效率，但使語言表面包含大量特殊短形式。

SECTION

十五、不是所有寫法都一樣好

“There’s more than one way to do it” 表示：

- 不同問題適合不同慣用法；
- 不同使用者具有不同背景；
- 語言不應過早封閉表達空間；
- 可先寫簡單方法，再逐步學習精緻方法。

它不表示：

- 任何程式都同樣可讀；
- 社群不需要 Style；
- Compiler 不需要規則；
- 每項功能都應有十種語法。

SECTION

十六、作者自由與讀者成本

令某任務可用寫法集合為：

> $W(T)=\{w_1,w_2,\ldots,w_n\}$

作者可以選擇最貼近問題的 w_i。

但讀者必須能辨認更多 w_i。

因此：

> $\text{AuthorChoice} \uparrow$
> $\Rightarrow$
> $\text{ReaderVocabulary} \uparrow$

SECTION

十七、多義性與領域語言

Perl 可讓：

- 系統管理者；
- 文字處理者；
- Web 開發者；
- Bioinformatics 使用者；
- Network 工程師；

  形成各自慣用語。

  這使語言能滲透異質領域，也會造成不同群體的 Perl 看起來像不同方言。

SECTION

十八、文化治理

Perl 依靠：

- perldoc；
- Camel Book；
- Perl Best Practices；
- use strict；
- Warning；
- Test culture；
- CPAN convention；
- Code review；
- 社群幽默與格言；

  塑造「好 Perl」。

  這是一種文化層，而非全部進入語法規格。

SECTION

十九、Laziness

所謂懶惰不是少做必要工作，而是：

- 自動化；
- 重用；
- 文件；
- 寫工具避免重複；

- 建立 Module。

SECTION

二十、Impatience

不耐煩表示：

- 系統應快速回應；
- 錯誤應及早發現；
- 工具應減少等待；
- 使用者不應反覆做機械工作。

SECTION

二十一、Hubris

自負表示：

- 寫出值得別人批評不了的程式；
- 對品質有責任；
- 不把短期 Hack 當永久合理。

這些格言帶有幽默，卻是一套工程文化。

SECTION

二十二、「Getting the job done」

Programming Perl 把 Perl 定位為完成工作的語言，而非證明單一範式純度。[R8]

Wall 的實用主義可以表示為：

> PracticalSuccess
> >
> ParadigmPurity

但不等於：

> PracticalSuccess
> =
> NoDesignPrinciple

SECTION

二十三、反對單一大敘事

Wall 所稱 Postmodern，包含：

- 不假設存在唯一最佳範式；
- 接受不同歷史語言的有效機制；
- 允許局部真理；
- 將語言視為文化及文本；
- 使用引文、拼接和再解釋；
- 承認設計者本身有立場。[R3]

SECTION

二十四、「混亂」的精確邊界

Wall 說英文有用，部分因為它能映射複雜現實；Perl 也被設計為「mess」，但這不是鼓勵無規則。

它更接近：

> RepresentationalDiversity
> ≈
> ProblemDiversity

而非：

> SyntaxChaos
> as an end

SECTION

二十五、後現代的工程風險

如果每個歷史功能都被保存：

- 核心膨脹；
- 特徵交互；
- 教學困難；
- Tooling 負擔；
- 新舊 Style 分裂；
- 相容性化石。

  Perl 5 後期正面臨此問題。

SECTION

二十六、Perl 5 的重大轉換

Perl 5 帶來：

- Reference；
- Complex data structure；
- Module；
- Object support；
- Lexical variable；
- Extensibility；

- 更完整 Interpreter。

  它使 Perl 從強力腳本語言成為一般用途平台。

SECTION

二十七、Module 優先於核心功能

Perl 大量能力透過 CPAN 提供，而不是持續寫入核心。

這是：

> SmallCoreEvolution
> +
> LargeCommunityLibrary

但核心仍需提供足夠動態能力讓 Module 擴展語言。

SECTION

二十八、CPAN 的制度作用

CPAN 讓：

- 作者獨立發布；
- Dependency 可重用；
- 文件和測試共同分發；
- 社群以 Module 先驗證功能；
- 不必由 Wall 核准所有領域能力。

  因此 Perl 的實際設計權部分移至生態。

SECTION

二十九、Pumpkin 與維護權轉移

Perl Release 歷史中的 Pumpkin holder 象徵整合權由不同維護者承擔。[R1]

Wall 保有創始文化和語言影響，但日常 Core 已是共同體工作。

SECTION

三十、為何不是 Perl 5.6 的普通升級

2000 年，Wall 宣布需要更根本的重新設計：

- 清理歷史 Warts；
- 重新思考物件和型別；
- 統一 Grammar；
- 改善並行；
- 建立可擴展語法；
- 讓困難事情更容易；
- 更新社群設計方式。[R5]

SECTION

三十一、RFC 開放

社群提交數百份 RFC。

這表示：

> FounderVision
> +
> CommunityProposalSpace

但不是所有建議直接表決加入；Wall 負責分類、綜合和裁決。

SECTION

三十二、Apocalypse 與 Exegesis

Wall 以 Apocalypse 文件「揭示」各領域新設計，再由社群以 Exegesis、Synopsis、實作和測試展開。

這是介於：

- BDFL；
- 公開 RFC；
- 文學式設計文件；
- 多實作實驗；

之間的治理方式。

SECTION

三十三、語言重寫的野心

Perl 6／Raku 包含：

- Grammar；
- Role；
- Multiple dispatch；
- Gradual typing；
- Junction；
- Lazy list；
- Metaobject；
- Unicode-first；
- Native concurrency；

- Programmable syntax。

  它更系統，也更龐大。

SECTION

三十四、長期開發的代價

- 規格反覆；
- 實作延遲；
- Parrot、Pugs、Rakudo 路線變化；
- Perl 5 使用者不確定未來；
- 名稱造成「Perl 5 已過時」誤解；
- 人才和注意力分散。

SECTION

三十五、Raku 改名

2019 年 Perl 6 正式更名 Raku。

其制度意義是：

- 承認它不是 Perl 5 的普通後繼版本；
- 兩個社群可以建立不同品牌；
- Perl 不再被版本號置於等待淘汰位置；
- Raku 可獨立演化。[R9]

  這不是否定共同歷史，而是修正版本命名造成的制度誤導。

SECTION

三十六、現代 Perl Governance

現行 perlgov 定義：

- Core Team；
- Steering Council；
- Vote Administrator。

  治理目標包括：

- Functional；
- Trusted；
- Sustainable；
- Transparent；
- Respectful。

  Core Team 選舉及可移除 Steering Council，Steering Council 承擔重大決策；制度只治理 Perl 語言和 Interpreter，不直接統治 CPAN、基金會及全部生態。[R10]

SECTION

三十七、創始文化與正式權力分離

Wall 的語言觀仍深刻影響 Perl，但：

> CulturalAuthority_(Wall)
> ≠
> FormalGovernanceAuthority

現代 Perl 已不是由 Wall 日常裁決的 BDFL 語言。

SECTION

三十八、Raku 的獨立共同體

Raku 具有：

- 自己的規格；
- Rakudo；
- 文檔；
- Release；
- 社群；
- Steering／Foundation 關係。

  Wall 的 Apocalypse 是歷史核心，但當代 Raku 同樣不能全歸於 Wall。

SECTION

三十九、Unix 整合期

問題：多個工具無法方便處理完整報告任務
策略：把文字、程序、Regex 和系統能力合成 Perl

SECTION

四十、自然語言核心期

問題：固定單一語法無法映射異質問題
策略：Context、Sigil、Default、TIMTOWTDI

SECTION

四十一、Perl 5 平台期

問題：腳本語言需支撐大型資料與生態
策略：Reference、Module、Object、CPAN

SECTION

四十二、文化自我描述期

問題：語言自由需共同體判斷
策略：格言、書籍、State of the Onion、幽默

SECTION

四十三、Perl 6／Raku 重寫期

問題：歷史 Warts 阻礙長期演化
策略：RFC、Apocalypse、重新設計

SECTION

四十四、分流治理期

問題：創始者和單一名稱無法治理兩個語言
策略：Raku 改名、Perl 正式治理制度

SECTION

四十五、問題 framing

Wall 的核心問題是：

> 如何讓語言容納真實工作中不整齊、跨領域、文字密集且充滿例外的結構，而不是要求問題先變成語言偏好的形狀？

SECTION

四十六、價值優先序

```
V_(Wall)
≈
(
Expressiveness,
Practicality,
LinguisticNaturalness,
Context,
Freedom,
CommunityCulture,
Evolvability
)
```

SECTION

四十七、核心—擴張偏好

Perl 5：

- 動態核心；
- Regex；
- Context；
- Module；
- CPAN 擴張。

Raku：

- 更明確 Meta model；
- 可程式 Grammar；
- 多 Dispatch；
- 更系統的可延展核心。

SECTION

四十八、顯式—推導偏好

Wall 接受：

- Default variable；
- Implicit context；
- Dynamic conversion；
- Contextual parsing。

同時以 Sigil 和操作符提供表面線索。

SECTION

四十九、效率—可讀性偏好

優先：

- 作者生產力；
- 文字和系統工作；
- 常見操作短。

讀者可讀性依賴：

- 慣用法；
- 文件；
- strict；
- Style；
- 社群共識。

SECTION

五十、安全—自由偏好

Perl Runtime 提供 Memory management，但保留：

- eval；
- Dynamic symbol；
- Native extension；
- Implicit conversion；
- Powerful Regex；
- Metaprogramming。

安全主要靠工具、模組和組織。

SECTION

五十一、相容性偏好

Perl 5 長期高度保護相容；Perl 6 則是另開重設路線。

這是一種雙軌：

> LegacyContinuity
> +
> RadicalParallelExperiment

SECTION

五十二、治理偏好

早期：

- 創始者設計。

  中期：

- 創始者文化＋Porters／Pumpkin。

  Perl 6：

- 公開 RFC＋Wall 綜合。

  後期：

- Perl／Raku 各自制度。

SECTION

五十三、自然語言類比不能過度延伸

程式必須具有可執行精確語義。

Perl 可以使用 Context 和省略，但不能像普通語言無限依靠世界知識消解歧義。

SECTION

五十四、TIMTOWTDI 會放大維護成本

不同作者可能：

- 使用不同 Regex；
- 使用不同 Object system；
- 使用不同 Loop；
- 使用不同 Error style。

團隊需要選擇子文化。

SECTION

五十五、短碼不等於好 Perl

One-liner 對一次性文字工作很強，但大型系統需要：

- 名稱；
- Module；
- Test；
- Strict；
- Documentation；
- Stable interface。

SECTION

五十六、Perl 6 的野心不能只寫成進步

它也帶來：

- 時間；
- 分裂；
- 採用不確定；
- 品牌混亂；
- 實作負擔。

Raku 的技術完整性與 Perl 生態成本必須同時評價。

SECTION

五十七、Wall 的演講是修辭性材料

State of the Onion 和 Postmodern 演講包含：

- 幽默；
- 宗教；
- 語言學；
- 自嘲；
- 社群表演。

它們提供直接價值排序，但不能把每個比喻當成形式規格。

時期 問題 決策 複雜度去向 風格

1987 Unix 工具碎片化 Perl 1 Interpreter 實用整合

1988–91 文字及二進位需求擴大 Regex、Binary、Camel Book 語言／文件 常用壓縮

1994 腳本需成為平台 Perl 5、Reference、Module Runtime／CPAN 生態擴張

1990s 多種寫法需文化治理 TIMTOWTDI、三美德 社群 語言文化

2000 歷史 Warts 阻礙演化 Perl 6 RFC 創始者＋社群 平行重寫

2000s–15　規格及實作巨大　Apocalypse、Rakudo 等　多團隊　語義重建

2019+　Perl／Perl 6 名稱衝突　Raku 改名　雙社群　語族分流

2020s　創始者不再日常治理　perlgov　Core Team／Council　後創始者制度

SECTION

五十八、主要原型

Larry Wall 同時屬於：

- 問題異質性整合者；
- 自然語言式程式語言設計者；
- 後現代實用主義者；
- 文化型語言治理者；
- 平行語言重寫發起者。

SECTION

五十九、不適合的簡單標籤

不應只稱：

短碼語言設計者
語法混亂倡議者
最小驚訝反對者
Perl 5 永久獨裁者
Raku 單一作者

較精確的描述是：

> 把現實的異質性和人類表達差異視為不可刪除條件，以 Context、多種慣用法和文化制度換取高度實用表達力的設計者。

SECTION

六十、最重要的連續性

Perl 1 到 Raku 的共同方向是：

> 【讓容易的事情保持容易
>
> ∧
> 讓困難的事情仍然可能】

SECTION

六十一、最重要的轉換

Perl 5：

在現有語言中累積多種解法
Perl 6／Raku：

重新建立更一般的語義和 Meta mechanism

SECTION

六十二、最重要的制度修正

由：

Larry 作為語言創始文化中心
轉為：

Perl 和 Raku 各自的正式共同體治理
Larry Wall 的設計不能被「There’s more than one way to do it」一句話完全代表。

他的完整方法包括：

- 從真實工作而非理想分類開始；
- 吸收已有工具最有效的語言習慣；
- 讓常用操作短而高頻；
- 使用 Context 和省略降低重複；
- 允許不同領域形成不同慣用法；

- 用文化、文件和工具治理自由；
- 用 CPAN 把擴展權交給社群；
- 當歷史負擔過大時，另開一條根本重寫路線；
- 最終允許創始者文化與正式治理分離。

  本文對 Wall 的 PLDST 判定為：

> 【Heterogeneous-Problem Integrator
> →
> Linguistic Pragmatist
> →
> Postmodern Community Language Architect】

其核心優勢是：

- 能把多個 Unix 工具工作流壓縮為一門語言；
- Regex、Context 和資料類型高度適合文字及系統工作；
- 語言允許漸進學習及多領域慣用法；
- CPAN 展示社群擴展的巨大力量；
- Wall 對文化、文件和人類差異具有少見敏感度；
- Raku 展示從歷史語言中重新抽象一般機制的勇氣。

  其核心代價是：

- 語言詞彙及特殊規則龐大；
- Context 與省略增加非局部理解；
- TIMTOWTDI 增加團隊 Style 差異；
- 強大 Regex 和 Metaprogramming 可形成不可讀程式；
- 文化治理難以替代靜態約束；

- Perl 6／Raku 的重寫週期及品牌分裂代價極高。

  最終原則為：

> 【表達自由不是沒有語法
>
> ∧
> 實用主義不是拒絕理論
>
> ∧
> 語言文化不是裝飾】

Wall 真正提出的問題是：

> 若現實本身充滿例外、語境和多種人類表達，程式語言是否應強迫所有人使用單一純粹模型，還是應建立一套足夠有規則、又能容納差異的人工文化？

Perl 的歷史回答是後者；而它的成就和維護成本，正好共同證明這個答案有多強，也有多昂貴。

人物：Larry Wall
主要語言／制度：Perl、Perl 5、Perl 6／Raku、Perl culture
核心時期：1987–2010s
主要問題：Unix 工具無法自然整合異質文字和系統工作
主要策略：Context、Regex、Sigil、Default、TIMTOWTDI、Module
複雜度去向：Interpreter、讀者詞彙、文件、文化、社群
責任去向：作者獲得自由，團隊需建立慣用法
主要保護對象：系統管理者、文字處理者、實用程式設計者
主要限制：可讀性、工具、特徵交互、重寫與社群分流
歸因信心：高

[R1] Perl official documentation, perlhist.

— Perl 發布歷史、初始版本、Pumpkin maintainer 和多作者演化。

[R2] Larry Wall et al., Programming Perl, various editions.

— Perl 的實用定位、語言文化、三大美德和公共說明；不同版具有不同共同作者。

[R3] Larry Wall, “Perl, the First Postmodern Computer Language,” 1999.

— 後現代語言、現實複雜性、拼接與多元模型。

[R4] Larry Wall, “Present Continuous, Future Perfect” and related linguistic talks.

— Perl 與自然語言、漸進學習、表達力和語言學設計原則。

[R5] Larry Wall, “State of the Onion 2000.”

— Perl 6 宣布、長期演化、重寫 Perl 和社群的方向。

[R6] Raku official community documentation and historical design materials.

— 「社群重寫」、RFC、Apocalypse 和多實作歷史。

[R7] Perl official documentation, perldata, perlintro, perlop.

— Sigil、Context、預設變數、運算和實際語義。

[R8] Programming Perl Prefaces and Glossary.

— Getting the job done、TIMTOWTDI 及 Programmer virtues。

[R9] Raku／Perl community records on the 2019 rename.

— Perl 6 改名 Raku、品牌及社群分流。

[R10] Perl official documentation, perlgov and perlpolicy.

— Core Team、Steering Council、信任、可持續、透明及治理邊界。

[T-U] Unix integration phase
[T-L] Linguistic core phase
[T-5] Perl 5／CPAN phase
[T-C] Cultural self-description phase
[T-6] Perl 6／Raku rewrite phase
[T-G] Post-founder governance phase
[S-H] Heterogeneous problem mapping
[S-N] Natural-language analogy
[S-C] Context sensitivity
[S-M] Multiple idioms
[S-P] Postmodern pragmatism
[S-K] Community culture

附錄 D　第二輪史實與歸因校對紀錄
D.1 Perl 的起源與發布時間
第二輪重新核對官方 `perlhist`：
- Perl 1 的公開發布日期為 1987 年 12 月；
- Perl 0 先在 Wall 的同事間使用；
- Perl 2 引入 Henry Spencer 的正規表示式套件；

- Perl 3 支援包含 Null 的二進位資料；
- Perl 4 與第一版 Camel Book 的公共傳播密切相關；
- Perl 5 則是語言和 Interpreter 的重大重寫。

本文因此沒有把後來 Perl 5 的全部能力回寫成 1987 年已完成的固定設計。

D.2 Unix 工具整合是分析模型

本文以：

Shell + awk + sed + grep + C

表示 Perl 原始能力組合，屬於 PLDST 的問題建模，不是 Wall 提出的形式方程式。

它表示 Perl 同時承擔：

- 程序控制；
- 欄位及文字處理；
- 模式匹配；
- 系統介面；
- 一般程式結構。

不表示 Perl 每項語法都能逐一追溯至單一 Unix 工具。

D.3 Context 與 Sigil

第二輪核對 `perldata`、`perlintro` 與 `perlop`：

- Perl 的 Sigil 確實標示所取值的資料角色；
- `$` 在官方文件中被類比為英文 “the”，表示單一值；
- `@` 類比 “these／those”，表示多值；
- Perl 的 Scalar／List context 能改變部分操作的結果；
- `$_` 是大量操作的預設主題；
- 這些機制提高常見程式的壓縮能力，也增加 Context 推理義務。

本文沒有把自然語言比喻寫成 Perl Parser 可接受一般自然語言歧義。

D.4 Postmodern language 的來源邊界

第二輪直接核對 Wall 1999 年〈Perl, the First Postmodern Computer Language〉：

- Wall 確實以 Postmodern 描述 Perl；
- 他批評語言設計把現實複雜性掃給程式設計者；
- 他使用拼接、文化、文本及多元觀點等修辭；
- 演講是價值與文化的一手資料，不是 Perl 的規範語義文件。

本文的「後現代實用主義者」是依據此演講及語言決策建立的 PLDST 原型，不把每個文學比喻技術化。

D.5 TIMTOWTDI 與三大美德

TIMTOWTDI 及 Laziness、Impatience、Hubris 主要透過 Programming Perl、Glossary、演講及社群文化流傳。

第二輪保留以下界線：

- 這些格言是工程文化與價值壓縮；
- 不是 Compiler 強制規則；
- 不能用來合理化不可讀程式；
- Programming Perl 各版具有 Wall 以外的共同作者。

D.6 Perl 6 的 RFC 與 Wall 的角色

第二輪核對 State of the Onion 2000、Apocalypse 保存頁與 Raku 社群文件：
- 2000 年正式宣布 Perl 6 重設方向；
- 社群提交數百份 RFC；
- Wall 承擔 Language designer 和最終綜合工作；
- Apocalypse 文件本身在後來設計演化中可能過時，保存頁明確提醒應參考更新 Synopsis；
- Perl 6 的設計不是社群提案直接投票相加，也不是 Wall 單人閉門完成。
本文因此使用「公開提案空間＋創始者綜合」描述治理。

D.7 Perl 6 與 Raku
第二輪校對確認：
- Perl 6 的設計自 2000 年開始；
- 2015 年左右形成第一個正式可用語言版本里程碑；
- 2019 年正式採用 Raku 名稱；
- 改名的重要動機包括減少與 Perl 5 的版本後繼混淆，並建立獨立身份；
- Wall 同意改名，但決策和實際制度屬於更大的 Raku 共同體。
本文沒有把 Raku 寫成 Perl 5 的直接相容升級，也沒有把改名寫成否定 Perl 家族歷史。

D.8 現代 Perl Governance
截至本文日期，官方 `perlgov` 仍定義：
- Core Team；
- Steering Council；
- Vote Administrator。
治理的明示目標包括：
- Functional；
- Trusted；
- Sustainable；
- Transparent；
- Respectful。
Core Team 選舉及可移除 Steering Council；治理範圍限於 Perl 語言、Interpreter、測試、文件及開發政策，不直接統治 CPAN、基金會和所有社群組織。
因此：

> Wall cultural authority
> ≠
> current formal authority

D.9 多作者歸因
本文已分開：
Larry Wall：原始語言、核心哲學、Perl 5／6 方向

Perl 5 Porters：Core 實作和維護

CPAN 作者：生態功能

Parrot／Pugs／Rakudo 團隊：不同 Perl 6／Raku 實作

現代 Council：Perl 正式治理

個別 Module、Interpreter 功能和 Raku 語義仍需逐案歸因。

D.10 PLDST 推論邊界

下列名稱是本文分析原型，不是 Wall 自稱的正式學派：

問題異質性整合者

自然語言式程式語言設計者

後現代實用主義者

文化型語言治理者

「多義性」表示 Context 及多種慣用表示，不表示 Perl 允許沒有確定執行語義的無限歧義。