University of Benghazi
Faculty of Information Technology
software Engineering Department

SE341

Software Evolution & Maintenance

Part .2

Lecturer ***: Mohammed Sultan***

Autumn (2025-2026)

University of Benghazi Faculty of Information Technology software Engineering Department
SE341 Software Evolution & Maintenance Part .2 Lecturer : Mohammed Sultan

جامعة بنغازي كلية تكنولوجيا المعلومات قسم هندسة البرمجيات SE341 تطور البرمجيات وصيانتها الجزء الثاني
المحاضر : محمد سلطان

Autumn (2025-2026)

الخريف (2026-2025)

Program evolution dynamics

- In circa emehcs noitacifissalc EPS na desoporp namheL .M rieM ‹1980 yranoitulove rieht ni yrav smargorp hcihw ni syaw eht nialpxe ot .scitsiretcarahc

Lehman observed a key difference between:

- Software developed to meet a fixed set of requirements, and
- Software developed to solve a real world problem which changes with time.

- The observation leads to the identification of 3 types of programs.
  - S (Specified)
  - P (Problem-solving)
  - E (Evolvingsmargorp (

Program evolution dynamics

ديناميات تطور البرنامج

In circa

في حوالي

scitsiretcarahc Lehman observed a key difference between:

لاحظ Scitsiretcarahc Lehman اختلافًا رئيسيًا بين:

Software developed to meet a fixed set of requirements, and Software developed to solve a real world problem which changes with time.

برمجيات تم تطويرها لتلبية مجموعة ثابتة من المتطلبات، وبرمجيات تم تطويرها لحل مشكلة حقيقية في العالم والتي تتغير مع مرور الوقت.

The observation leads to the identification of 3 types of programs.

تؤدي الملاحظة إلى تحديد 3 أنواع من البرامج.

• S (Specified(

• S (محدد)

• P (Problem-solving(

• ف (حل المشكلات)

• E (Evolving smargorp)

• E (تطور smargorp)

Software Evolution & Maintenance - SE341

تطور البرمجيات وصيانتها - SE341

Lehman system types

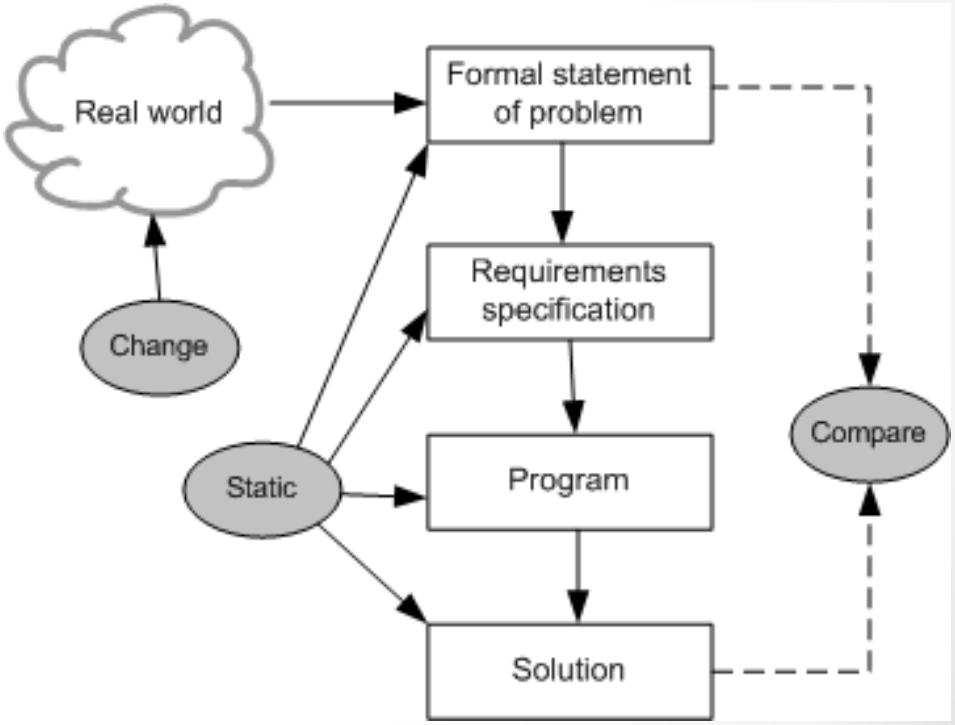
S-type (Specified) programs have the following characteristics:

- All the non-functional and functional program properties, that are important to its stakeholders, are formally and completely defined.
- Correctness of the program with respect to its formal specification is the only criterion of the acceptability of the solution to its stakeholders.

Examples of S-type programs:

(i) Calculation of the lowest common multiple of two integers.

(ii) Perform matrix addition, multiplication, and inversion.



S-type programs

Lehman system types

أنواع نظام ليمان

S-type (Specified) programs have the following characteristics:

تتميز برامج النوع S (المحددة) بالخصائص التالية:

• All the non-functional and functional program properties, that are important to its stakeholders, are formally and completely defined.

• جميع خصائص البرنامج غير الوظيفية والوظيفية، التي تهم أصحاب المصلحة، محددة بشكل رسمي وكامل.

• Correctness of the program with respect to its formal specification is the only criterion of the acceptability of the solution to its stakeholders.

• صحة البرنامج فيما يتعلق بمواصفاته الرسمية هي المعيار الوحيد لمقبولية الحل لدى أصحاب المصلحة.

Examples of S-type programs:

أمثلة على برامج النوع S:

(i) Calculation of the lowest common multiple of two integers.

(ط) حساب المضاعف المشترك الأصغر لعددين صحيحين.

(ii) Perform matrix addition, multiplication, and inversion.

(2) إجراء عمليات جمع المصفوفات والضرب والعكس.

S-type programs

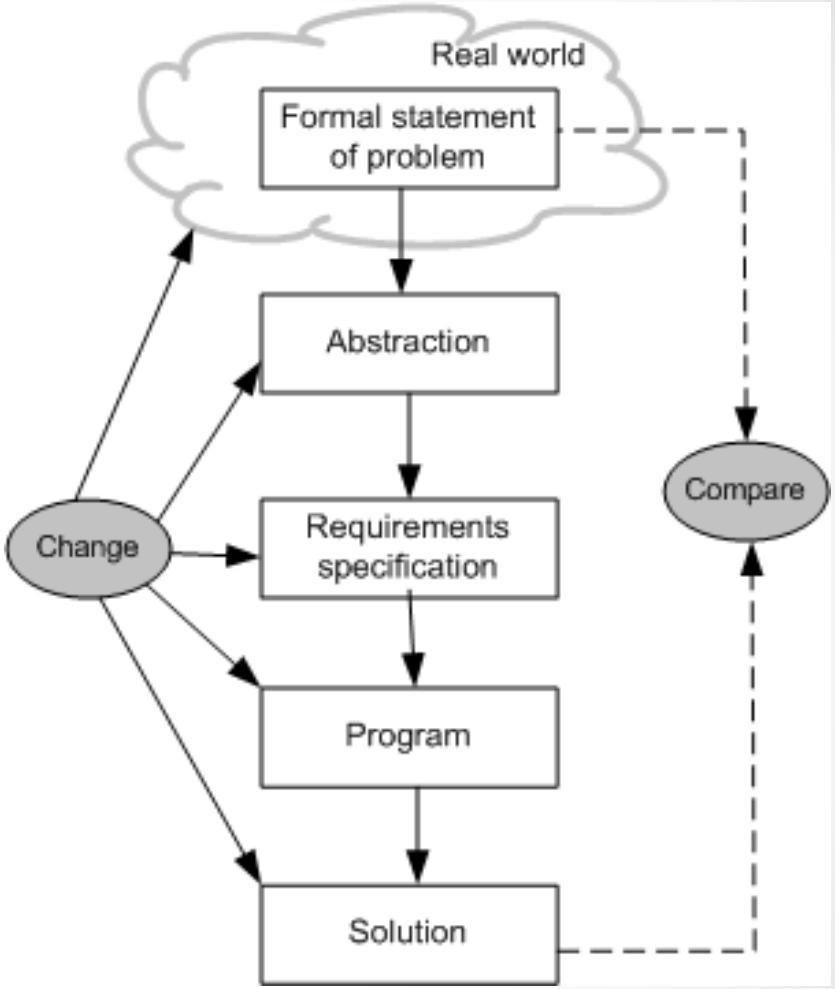
برامج من النوع S

Software Evolution & Maintenance - SE341 3

تطور البرمجيات وصيانتها - SE341 3

Lehman system types

- P-type (Problem-solving) program is based on a practical abstraction of the problem, instead of relying on a completely defined specification.
- Example: A program That play chess.
- The P-type program resulting from the changes cannot be considered a new solution to a new problem. Rather, it is a modification of the old solution to better fit the existing problem.
- In addition, the real world may change, hence the problem changes.



P-type programs

Lehman system types

أنواع نظام ليمان

• P-type (Problem-solving) program is based on a practical abstraction of the problem, instead of relying on a completely defined specification.

• يعتمد برنامج النوع P (حل المشكلات) على التجريد العملي للمشكلة، بدلاً من الاعتماد على مواصفات محددة بالكامل.

• Example: A program That play chess.

• مثال: برنامج يلعب الشطرنج.

• The P-type program resulting from the changes cannot be considered a new solution to a new problem.

• لا يمكن اعتبار البرنامج من النوع P الناتج عن التغييرات حلاً جديدًا لمشكلة جديدة.

Rather, it is a modification of the old solution to better fit the existing problem.

بل هو تعديل للحل القديم ليناسب المشكلة الحالية بشكل أفضل.

• In addition, the real world may change, hence the problem changes.

• وبالإضافة إلى ذلك، فإن العالم الحقيقي قد يتغير، وبالتالي تتغير المشكلة.

P-type programs

برامج من النوع P

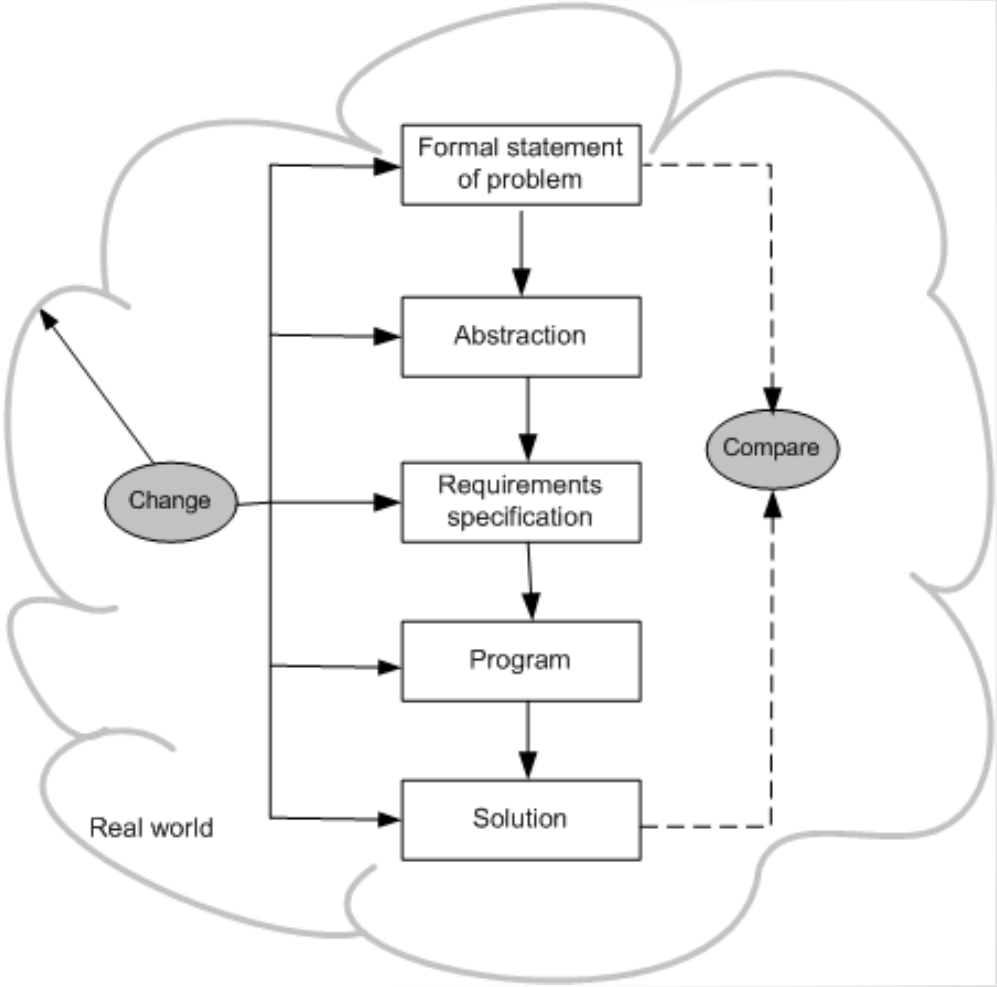
Software Evolution & Maintenance - SE341 4

تطور البرمجيات وصيانتها - SE341 4

Lehman system types

An E-type (Evolving) program is one that is embedded in the real world and it changes as the world does.

- These programs mechanize a human or society activity, make simplifying assumptions, and interface with the external world by requiring or providing services.
- The acceptance of an E-type program entirely depends upon the stakeholders' opinion and judgment of the solution.



E-type programs

Lehman system types

أنواع نظام ليمان

An E-type (Evolving) program is one that is embedded in the real world and it changes as the world does.

برنامج النوع E (المتطور) هو برنامج مضمن في العالم الحقيقي ويتغير مع تغير العالم.

• These programs mechanize a human or society activity, make simplifying assumptions, and interface with the external world by requiring or providing services.

• تعمل هذه البرامج على ميكنة النشاط البشري أو المجتمعي، ووضع افتراضات مبسطة، والتفاعل مع العالم الخارجي من خلال طلب الخدمات أو تقديمها.

• The acceptance of an E-type program entirely depends upon the stakeholders' opinion and judgment of the solution.

• يعتمد قبول برنامج النوع الإلكتروني كليًا على رأي أصحاب المصلحة وحكمهم على الحل.

E-type programs

برامج النوع الإلكتروني

Laws of Software Evolution

- Lehman formulated a set of observations that he called laws of evolution.
- The laws themselves have evolved from three in 1974 to eight by 1997.
- These laws are the results of studies of the evolution of large-scale proprietary or closed source system (CSS).
- The laws concern what Lehman called E-type systems:

”Monolithic systems produced by a team within an organization that solve a real world problem and have human users.”

- Lehman’s laws were not meant to be used in a mathematical sense, as, say, Newton’s laws are used in physics.
- The term “laws” was used because the observed phenomena were beyond the influence of managers and developers.
- The laws were an attempt to study the nature of software evolutions and the evolutionary trajectory likely taken by software.

Laws of Software Evolution

قوانين تطور البرمجيات

• Lehman formulated a set of observations that he called laws of evolution.

• صاغ ليمان مجموعة من الملاحظات أطلق عليها اسم قوانين التطور.

• The laws themselves have evolved from three in 1974 to eight by 1997.

• تطورت القوانين نفسها من ثلاثة في عام 1974 إلى ثمانية بحلول عام 1997.

• These laws are the results of studies of the evolution of large-scale proprietary or closed source system (CSS).

• هذه القوانين هي نتائج دراسات تطور نظام الملكية أو نظام المصدر المغلق (CSS) على نطاق واسع.

The laws concern what Lehman called E-type systems:

تتعلق القوانين بما أسماه ليمان أنظمة النوع E:

”Monolithic systems produced by a team within an organization that solve a real world problem and have human users.”

"أنظمة متجانسة ينتجها فريق داخل مؤسسة تحل مشكلة حقيقية في العالم ولها مستخدمون بشريون."

• Lehman’s laws were not meant to be used in a mathematical sense, as, say, Newton’s laws are used in physics.

لم يكن المقصود من قوانين ليمان أن تستخدم بالمعنى الرياضي، كما تستخدم قوانين نيوتن على سبيل المثال في الفيزياء.

• The term “laws” was used because the observed phenomena were beyond the influence of managers and developers.

• تم استخدام مصطلح "القوانين" لأن الظواهر المرصودة كانت خارجة عن تأثير المديرين والمطورين.

• The laws were an attempt to study the nature of software evolutions and the evolutionary trajectory likely taken by software.

• كانت القوانين محاولة لدراسة طبيعة تطورات البرمجيات والمسار التطوري الذي من المحتمل أن تتخذه البرمجيات.

Laws of Software Evolution

Names of the laws	Brief descriptions
I. Continuing change (1974)	E-type programs must be continually adapted, else they become progressively less satisfactory.
II. Increasing complexity (1974)	As an E-type program evolves, its complexity increases unless work is done to maintain or reduce it.
III. Self regulation (1974)	The evolution process of E-type programs is self regulating, with the time distribution of measures of processes and products being close to normal.
IV. Conservation of organizational stability (1978)	The average effective global activity rate in an evolving E-type program is invariant over the product's lifetime.
V. Conservation of familiarity (1978)	The average content of successive releases is constant during the life-cycle of an evolving E-type program.
VI. Continuing growth (1991)	To maintain user satisfaction with the program over its lifetime, the functional content of an E-type program must be continually increased.
VII. Declining quality (1996)	An E-type program is perceived by its stakeholders to have declining quality if it is not maintained and adapted to its environment.
VIII. Feedback system (1971-1996)	The evolution processes in E-type programs constitute multi-agent, multi-level, multi-loop feedback systems.

Table 2.5: Laws of software evolution, adapted from Lehman et al. [35] (©[1997] IEEE).

Laws of Software Evolution

قوانين تطور البرمجيات

Software Evolution & Maintenance - SE341 7

تطور البرمجيات وصيانتها - SE341 7

Laws of Software Evolution

القانون	الوصف
1- التغيير المستمر	يجب تغيير البرمجيات باستمرار لتتكيف مع البيئة أو أنها ستفقد فائدتها تدريجيا
2- التعقيد المتزايد	مع تغيير البرمجية المستمر سيصبح هيكلها أكثر تعقيدا ما لم تخصص موارد للحد من ذلك
3- التنظيم الذاتي	البرمجيات الكبيرة تتطور بشكل منظم ذاتيا يحافظ على توزيع يقارب الطبيعي للحجم والوقت وعدد العيوب بين الإصدارات
4- المحافظة على ثبات المؤسسة	على مدى عمر البرمجية وبشكل عام معدل العمل الفعلي المؤثر في تطورها هو في المتوسط ثابت بغض النظر عن الموارد المخصصة لذلك
5- المحافظة على الألفة	على مدى عمر البرمجية يكون كل محتوى إصدار منها تقريبا متقارب ليحافظ على ألفتها لمستخدميها ومطوريها مستقبلا
6- النمو المستمر	لكي نحافظ على رضا الزبون على البرمجية خلال فترة حياتها يجب أن ننمي ونزيد من وظائف البرمجية باستمرار
7- انحدار الجودة	مع التغيير المستمر للبرمجية فان جودتها ستكون في انحدار ما لم يتم تكييف البرمجية مع هذه التغيرات
8- نظام التغذية الراجعة	عمليات تطور البرمجيات يجب أن نتعامل معها على أنها نظم تغذية راجعة يساهم في تحسين تطورها بشكل بارز

Laws of Software Evolution

قوانين تطور البرمجيات

Software Evolution & Maintenance - SE341 8

تطور البرمجيات وصيانتها - SE341 8