Zendler, Andreas, "A Preliminary Software Engineering Theory as Investigated by Published Experiments", *Empirical Software Engineering*, 6(2):161-180, June 2001.

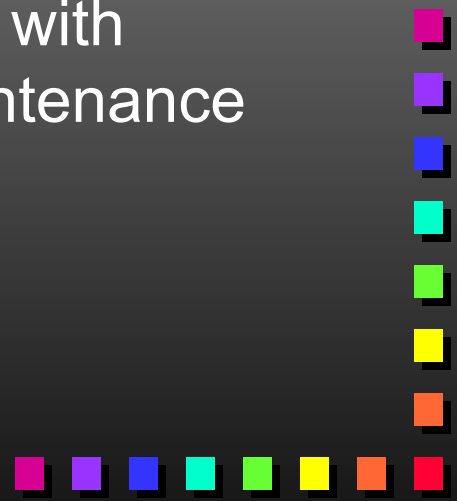
Presented by :Sara AL-Hammad

Supervised by: Ghazy Assasa, PHD

Paper Goals:

This paper is about to:

- Review the results of published experiments that studied the effectiveness and efficiency of software engineering techniques.
- Try to develop a Preliminary Software Engineering Theory by collecting these results that deal with analysis ,design, implementation ,test, maintenance and quality assurance



Empirical software engineering

Empirical software engineering is the subdiscipline of software engineering that investigate software engineering techniques by using empirical methods:

Case study, inquiry and experiments

Experimental software engineering uses the experiment to validate, improve, and select software engineering techniques which permit an efficient application development.

Experimental software engineering

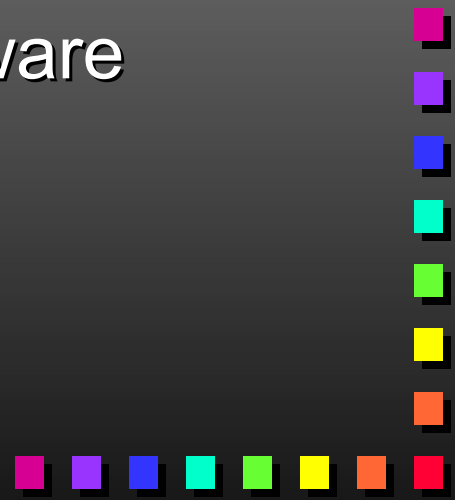
In the software engineering experiments we found that:

- comprehensive overview and results are *missing*
- *Systematic context* is not available that guides the selection of what to experiment next.

Experimental software engineering

The purpose of this paper is to develop a systematic context (Preliminary Software Engineering Theory)- for the results of software engineering experiments that will allow to:

- Ask for new software engineering problems
- Collect further data to support software engineering hypotheses
- Deduce new software engineering investigation lines



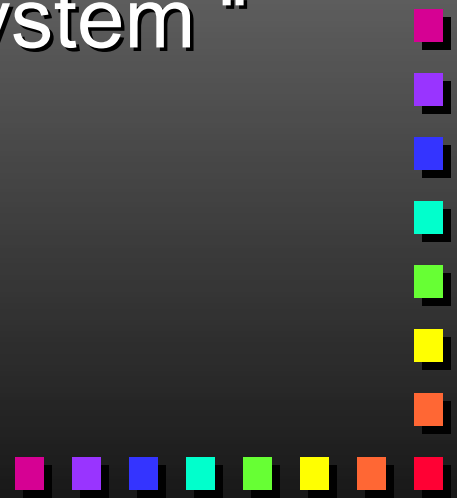
How the author organized the paper

- introduce the necessary concepts-theory and hypotheses-
- gives a brief history of experimental software engineering
- contains results from published software engineering experiments
- Presents the Preliminary Software Engineering Theory and shows how it can be used

1.Software engineering theory and hypotheses

"A scientific theory is a system of hypotheses that is supposed to give a partial and approximate account of a bit of reality .At this point we shall emphasize that a theory is a system "

(Bung,1967,p. 391)



1.Software engineering theory and hypotheses

theory is a system that:

1- It refers to a universe of discourse

2- It consists of several hypotheses

3- The hypotheses refer to the universe of discourse

4- The hypotheses are connected

5- The hypotheses is not isolated

1.1 Software engineering theory

A Software engineering theory can be understood as a theory

1- Whose universe of discourse is software engineering

2-Whose hypotheses are software engineering hypotheses

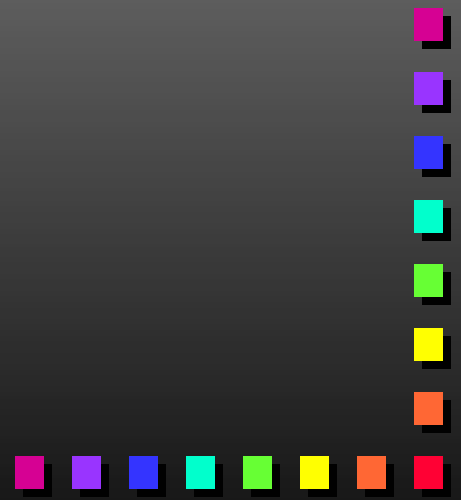
3- Whose software engineering hypotheses refer to the universe of discourse that is software engineering.

4- Whose software engineering hypotheses are connected

5-Non of the software engineering hypotheses are isolated

1.2 Software engineering hypotheses

It can be understood as assumption to explain software engineering propositions. it is used to explain a software engineering phenomenon for some information.



1.2 Software engineering hypotheses

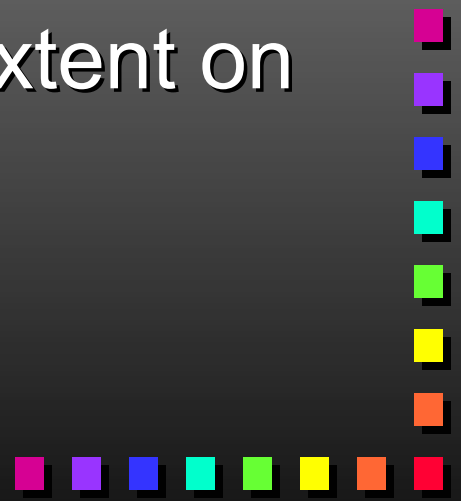
Characteristics of Software engineering hypotheses:

- They are not to be equated with fiction
- They have a greater content than the empirical propositions they cover.
- They can not be established by any single experiences

1.2 Software engineering hypotheses

Three main request for the formulation of software engineering hypotheses:

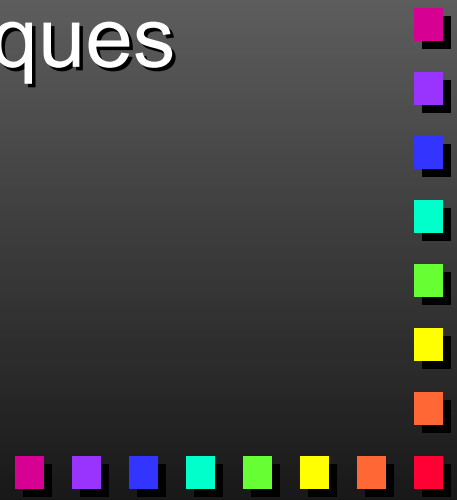
- It must be syntactically correct and meaningful
- It must be grounded to some extent on previous knowledge
- It must be empirically testable.



1.2 Software engineering hypotheses

Example of software engineering hypotheses:

- Object oriented programming techniques have advantages against structured programming techniques



1.2 Software engineering hypotheses

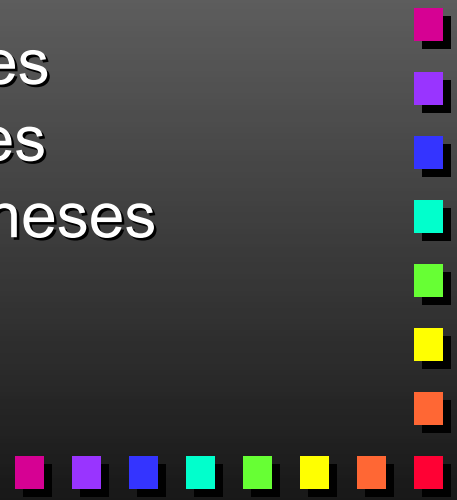
software engineering hypotheses (*h*) can be divided according to:

→antecedent knowledge (*A*)

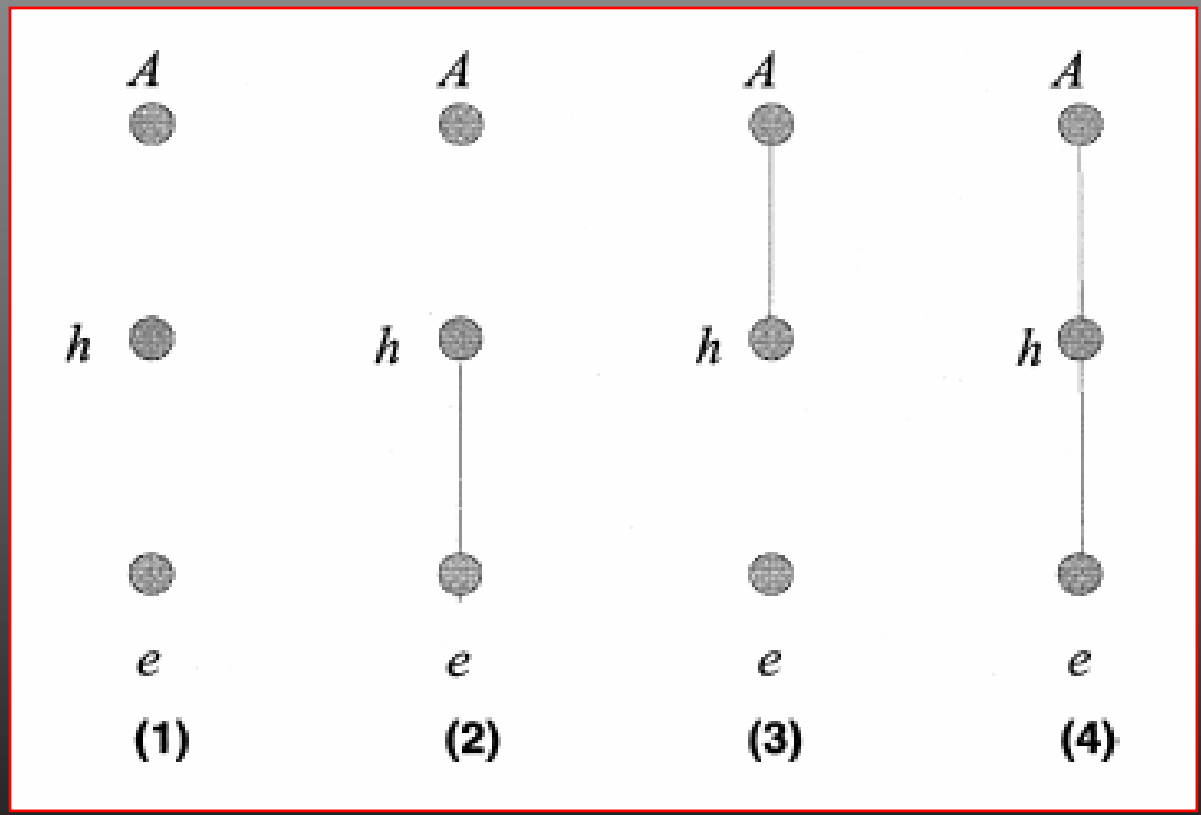
→empirical data (*e*)

Four types of software engineering hypotheses (*h*):

- software engineering guesses
- empirical software engineering hypotheses
- plausible software engineering hypotheses
- corroborated software engineering hypotheses



1.2 Software engineering hypotheses



2. History of experimental Software engineering

- first experiments goes back to the year 1967
- in 1980's only a few software engineering experiments were published, but in 1990's the number of experiments were increased.

Reasons:

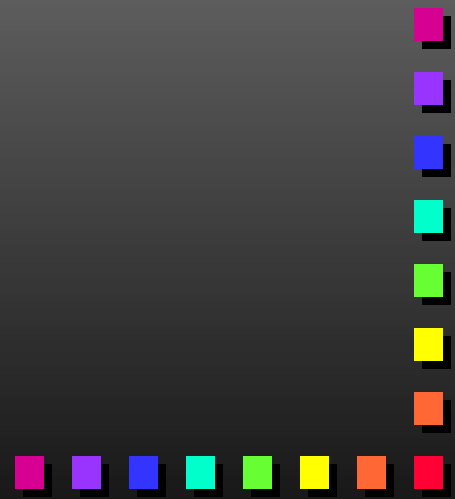
- The necessity for experimental research in the discipline of software engineering was realized
- The availability of methodological papers to conduct software engineering experiments
- Research institutes have been founded whose research programs are empirically oriented.

2. History of experimental Software engineering

- First experiments1967

→published by Grant and Sackman

→investigation for threats a software engineering problem



2. History of experimental Software engineering

- Experiments1970-1979

→investigation for threats a software engineering problem that deal with implementation and testing

→ Lucas and Kaplan(1976) investigate the advantages of structured programming

2. History of experimental Software engineering

- Experiments1970-1979

→Shneiderman (1977) analyze the use of flowcharting

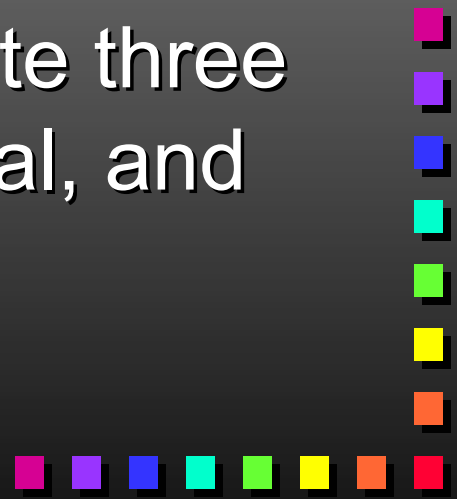
→ Myers(1978) conducts an experiment that is concerned with software testing techniques.

2. History of experimental Software engineering

- Experiments1980-1989

→ It deal with analysis, design , implementation , testing and maintenance

→ Basili and Selby(1987) evaluate three test techniques (code, functional, and structural testing)



2. History of experimental Software engineering

- Experiments1990-1999

  It deal with analysis, design , implementation , testing ,maintenance and Quality assurance

→ Porter and Votta(1994) study where and when quality assurance seems to be done

→ Moynihan(1996) analysis the appropriateness of object-oriented and structural techniques.

3. Published Software engineering experiments

To develop a Software engineering experiments theory published up to date experiments are considered that respect the following three criteria:

1- At least two software engineering techniques are studied

2- Subjects apply software engineering techniques and produce result.

3- The results are measured by software metrics

3. Published Software engineering experiments

The criteria stress:

1- The object of software engineering experiments is the comparative investigation of techniques

2- By using subjects software engineering experiments are delimited against computer-based experiments.

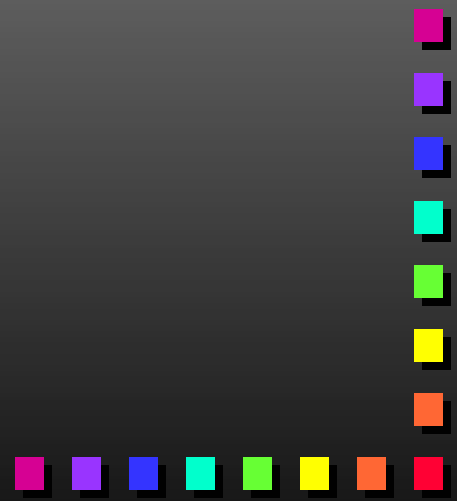
3- By emphasizing software metrics software engineering experiments are delimited against experiments with psychological variables.

3.1 Experiments on analysis techniques

Yadav(1989) and Moynihan(1996) compared two analysis techniques, the data flow diagram (DFD) and structure analysis design technique (SADT)

The result

- DFD are superior to SADT
- DFD usable more than SADT



3.2 Experiments on Design techniques

Lohse and Zweben(1984) evaluate the effect of module coupling on system modifiability

The result

- There was an interaction effect between module coupling and the type of modification

3.2 Experiments on Design techniques

Scanlan(1989) analyzes the effects of specification techniques on the understandability of algorithms.

The result

- when using flowchart in comparison with pseudocode specification more correct responses are obtained for the understanding of the algorithm
- if flowchart specification are given, the time to understand algorithms is shorter than with pseudocode specification
- flowchart specification are particularly more efficient than pseudocode specification when the complexity of the algorithms rises.

3.3 Experiments on Implementation techniques

Lucas and Kaplan(1976) study the effects of structured programming

The result

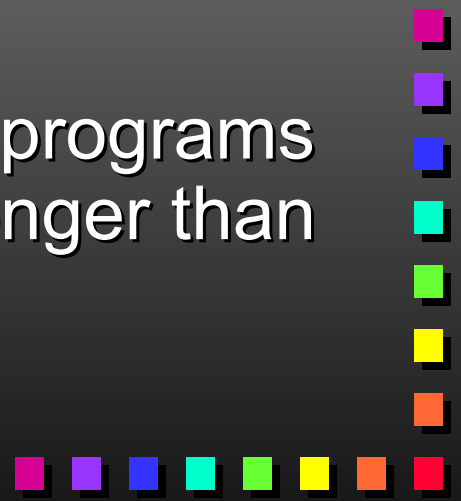
when structured programming is used the time spent modifying the program, program compile time and program storage requirements are significantly lower than other programming types

3.3 Experiments on Implementation techniques

Daly(1996) study the effects of inheritance in object oriented programs on maintainability

The result

- maintenance costs for programs with inheritance (depth of 3) are lower than costs for programs without inheritance.
- In contrast , maintenance costs for programs with inheritance (depth of 5) took longer than programs without inheritance.

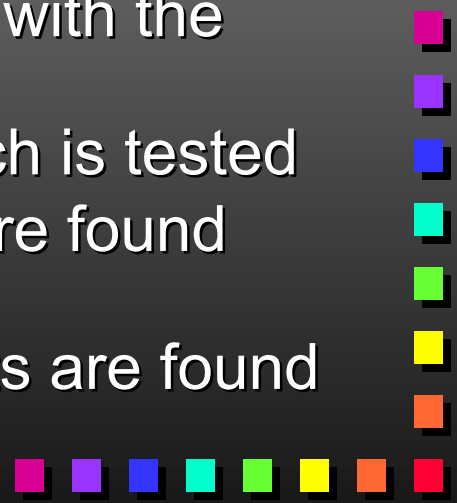


3.4 Experiments on testing techniques

Basili and Selby(1987) evaluate three test techniques (code, functional, and structural testing)

The result

- advanced reviewer obtain best results with the code testing
- intermediate reviewer obtain best results with the code testing and functional one
- error detection depends on software which is tested
- when using code testing interface error are found more easily
- using functional testing more control faults are found



3.4 Experiments on maintenance techniques

Swigger and Brazile(1991) study the effect of documentation techniques on the maintainability of an expert system

The result

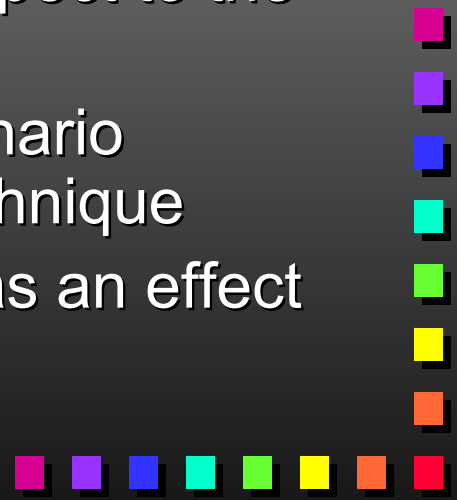
- Type of documentation (Petri net, entity relationship diagram) has an effect on the maintainability of the expert system: when documenting the software using Petri net , data oriented and procedural modifications can be processed significantly faster.

3.4 Experiments on quality assurance techniques

Porter and Votta (1994) study the effects of inspection techniques for defect detection in requirements specifications.

The result

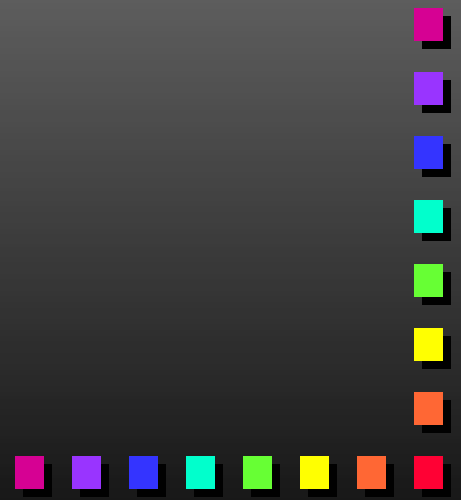
- The inspection techniques differ with respect to the team defect detection rate significantly
- The best results are obtained for the scenario technique, the worst for the check list technique
- The type of requirements specification has an effect on the team defection rate



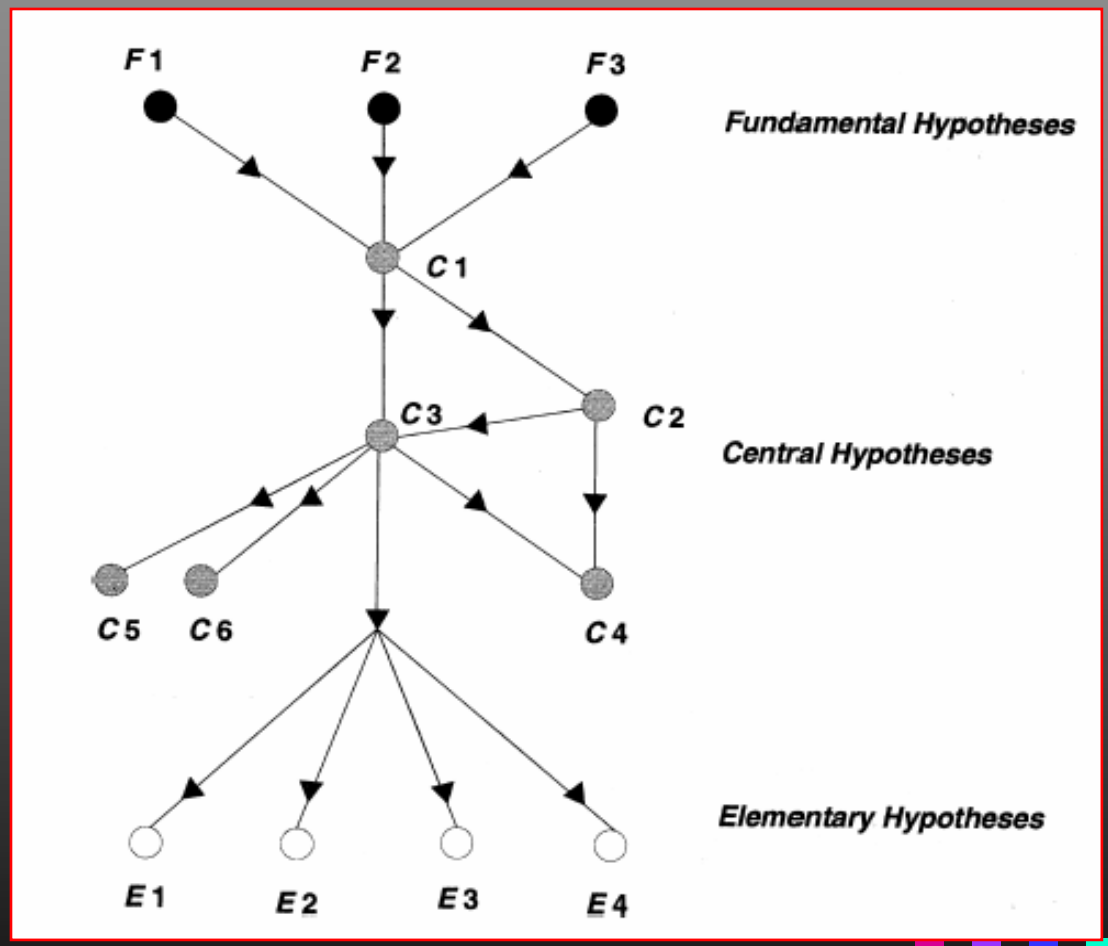
4. The preliminary Software Engineering Theory

The theory distinguish fundamental , central ,and elementary software engineering hypotheses

The glue that keeps the hypotheses together is entailment “A ► B”

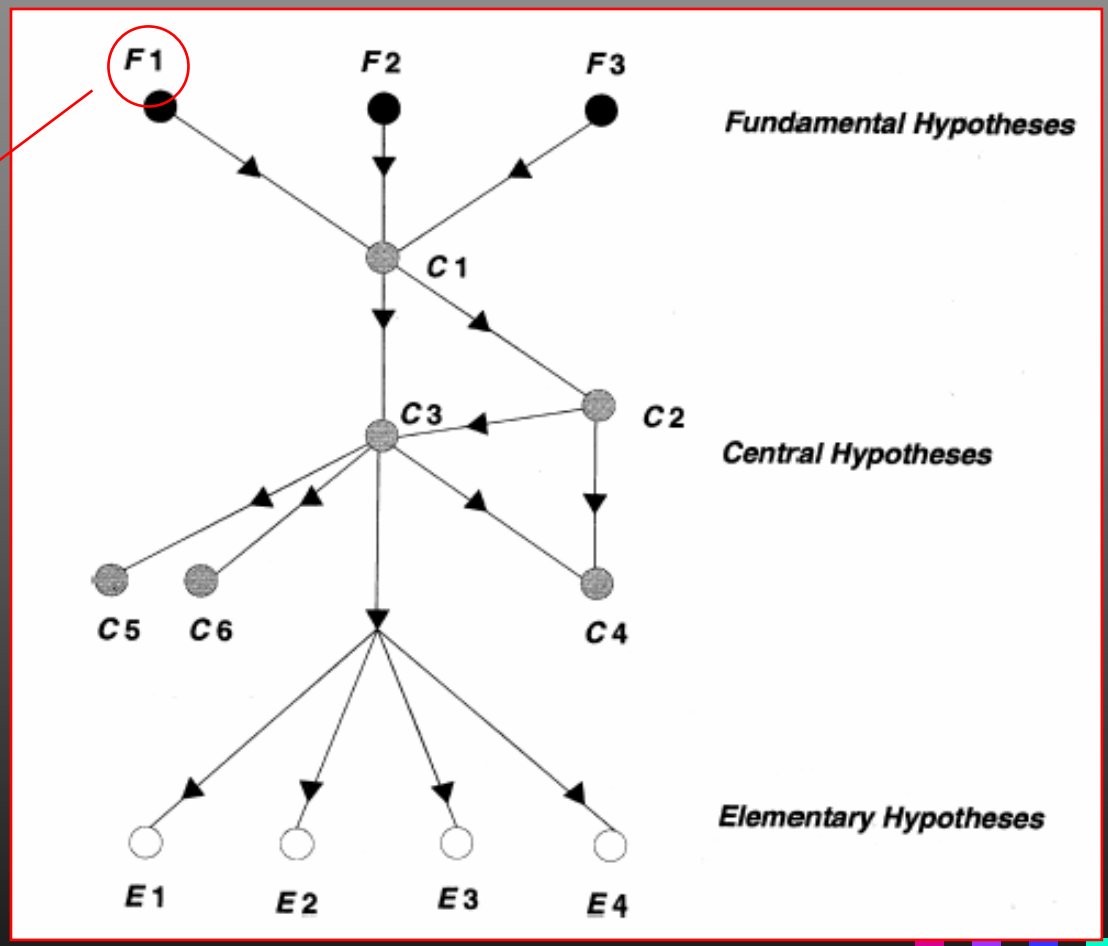


4. The preliminary Software Engineering Theory



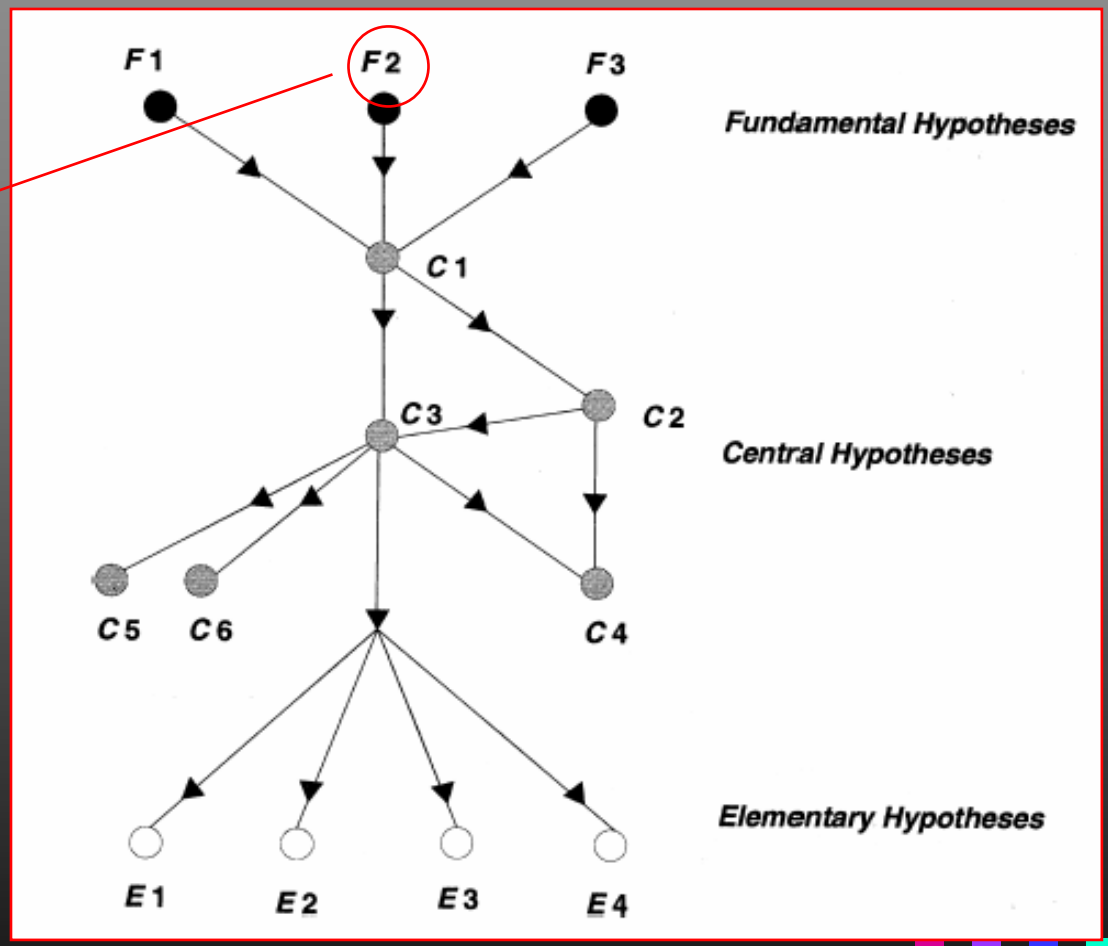
4. The preliminary Software Engineering Theory

team composition



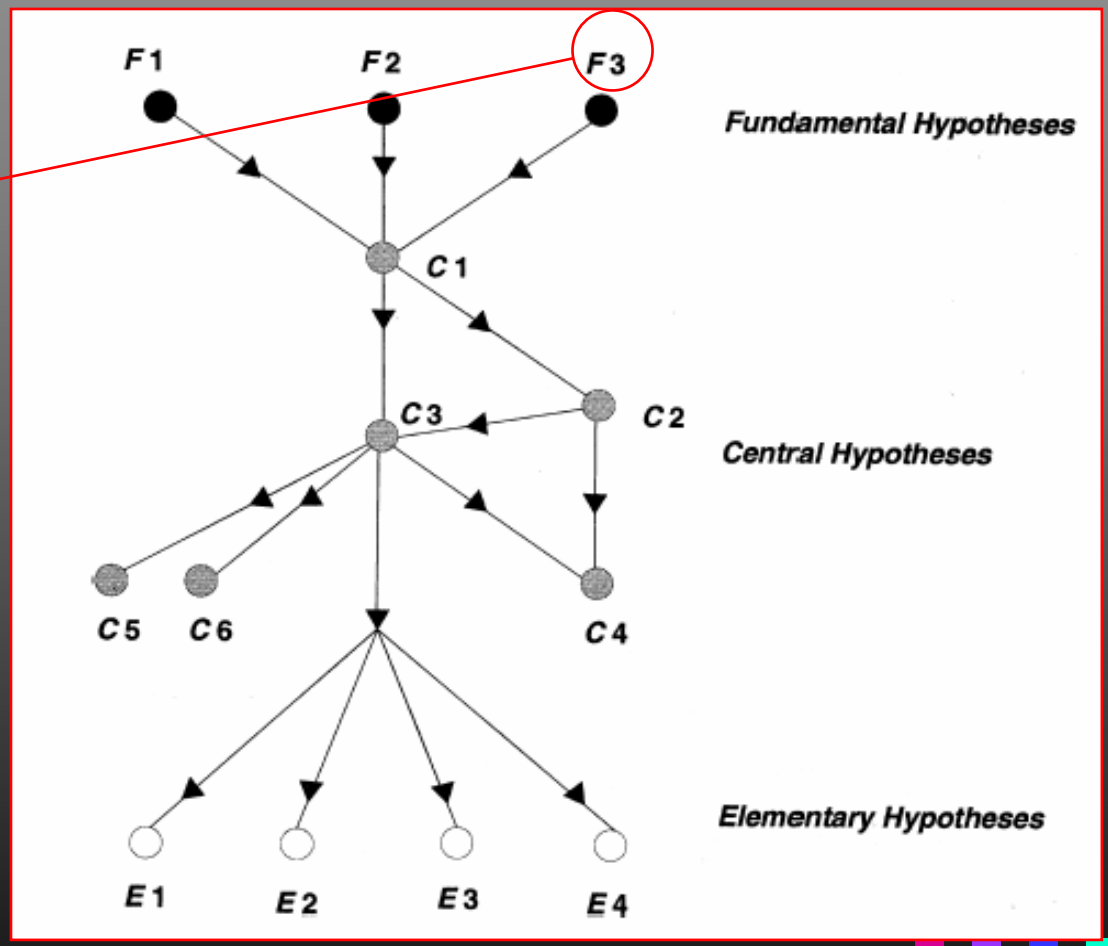
4. The preliminary Software Engineering Theory

Paradigm



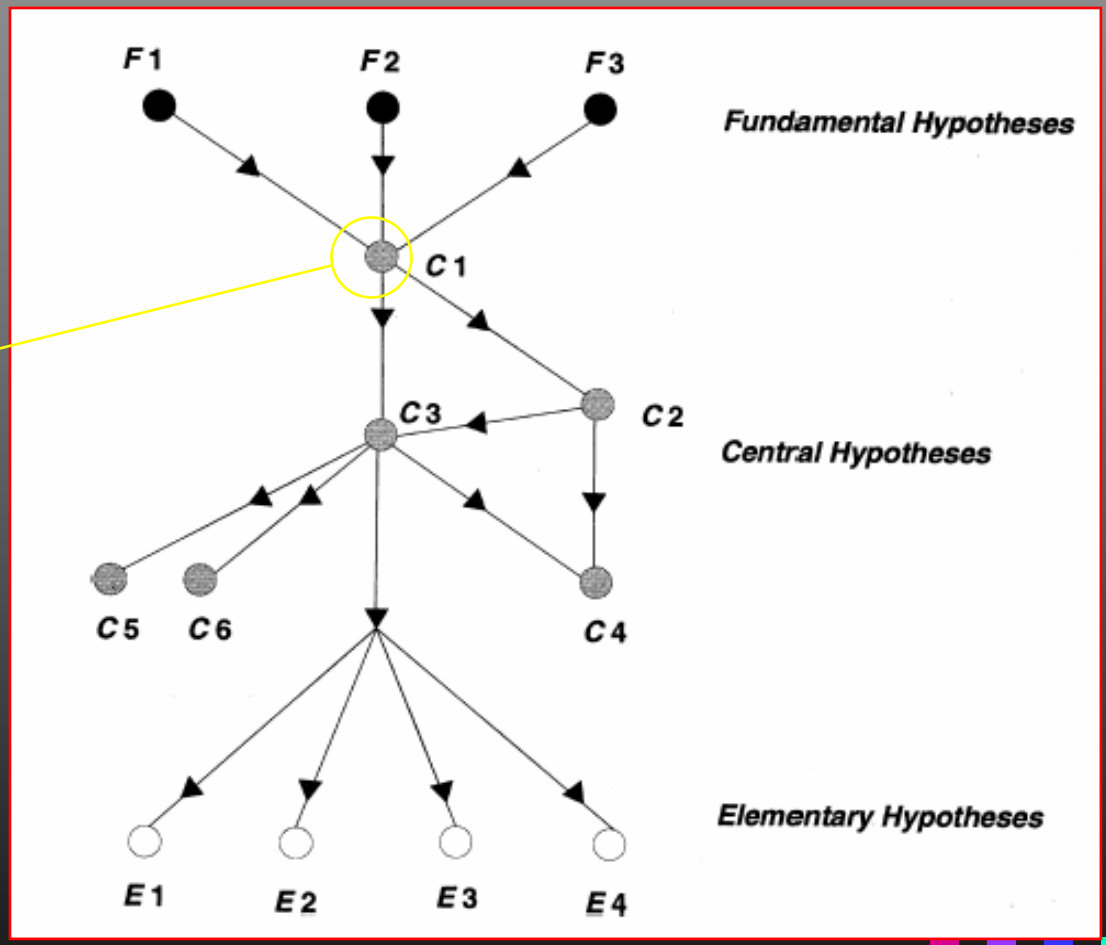
4. The preliminary Software Engineering Theory

experience



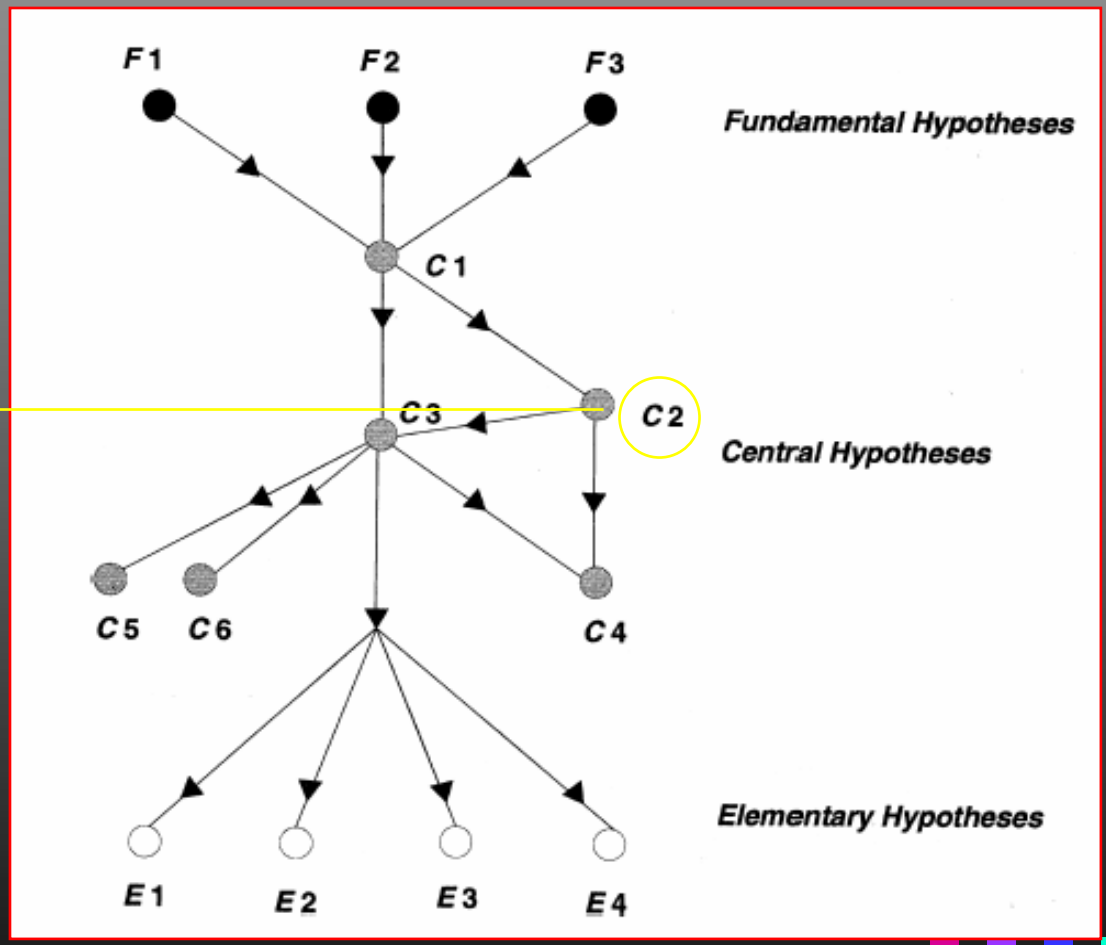
4. The preliminary Software Engineering Theory

one paradigm



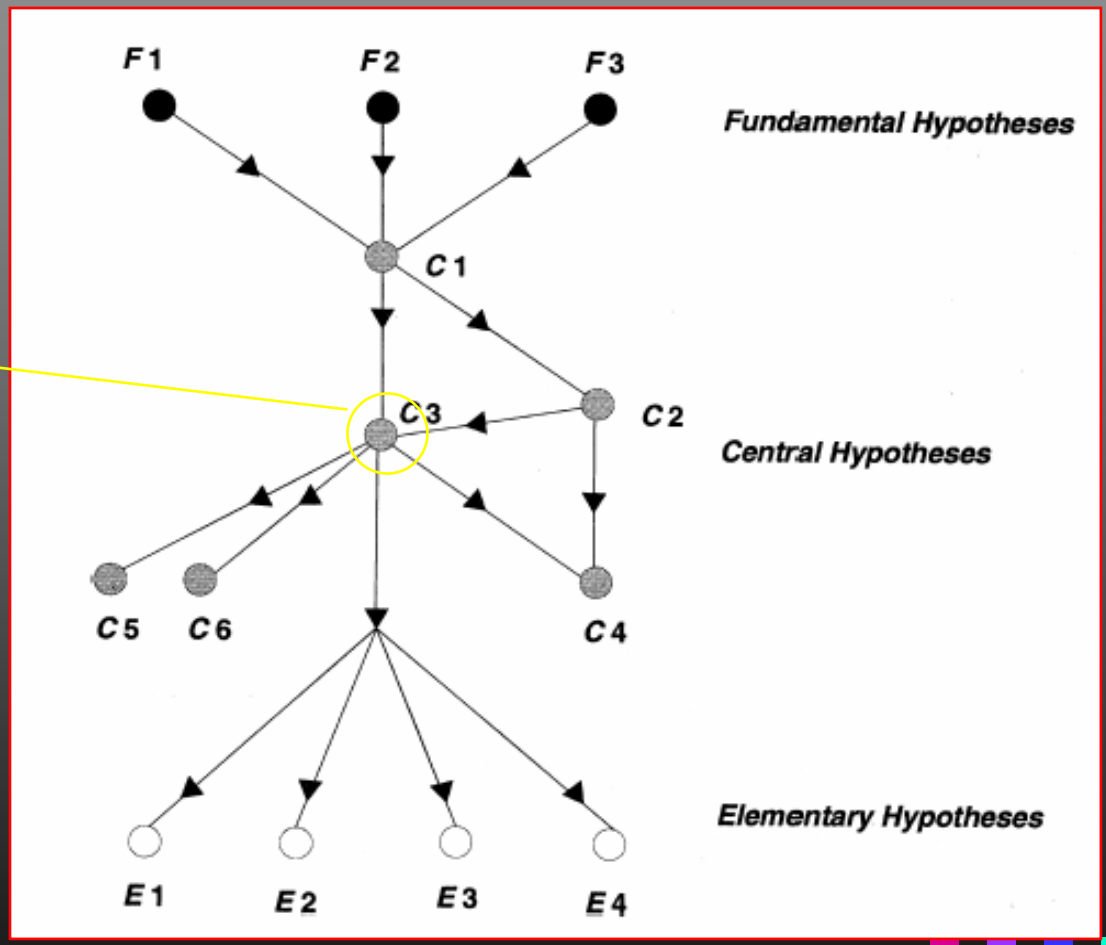
4. The preliminary Software Engineering Theory

type of application



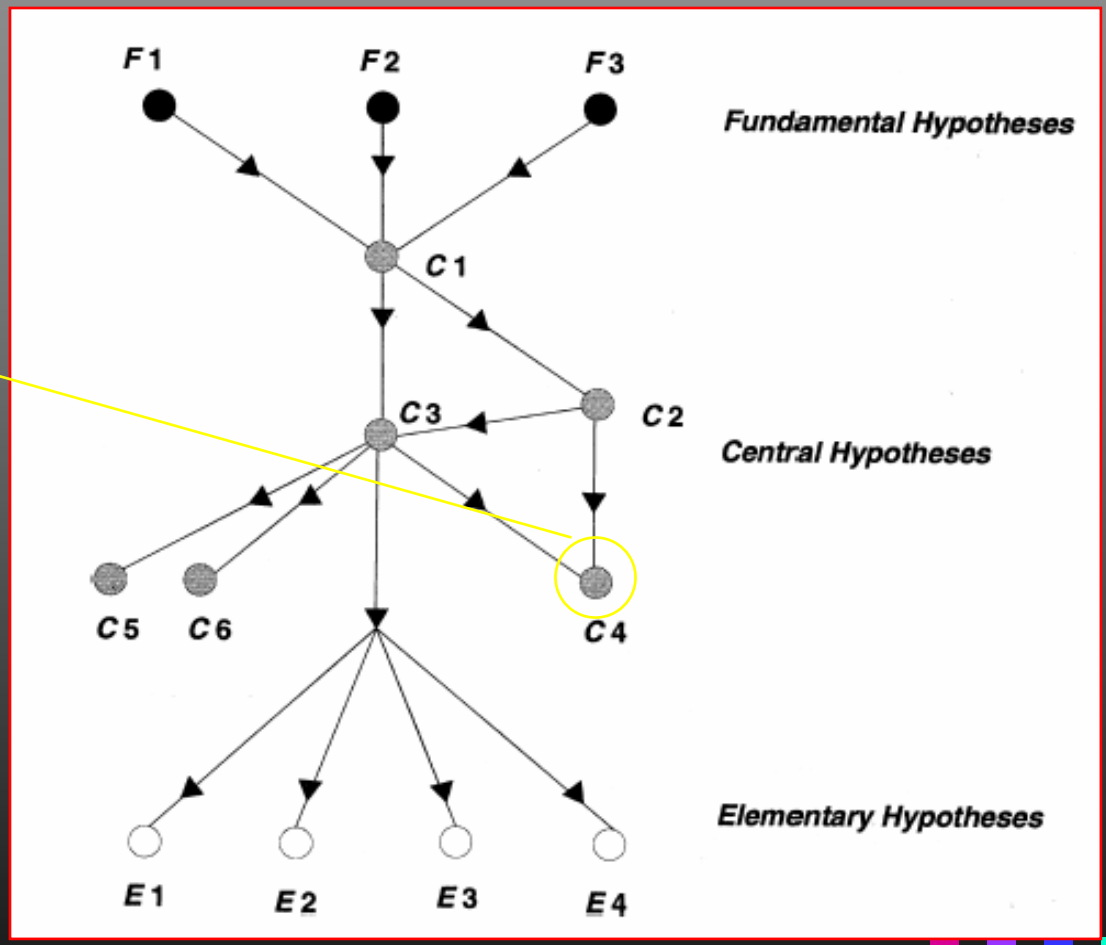
4. The preliminary Software Engineering Theory

one activity



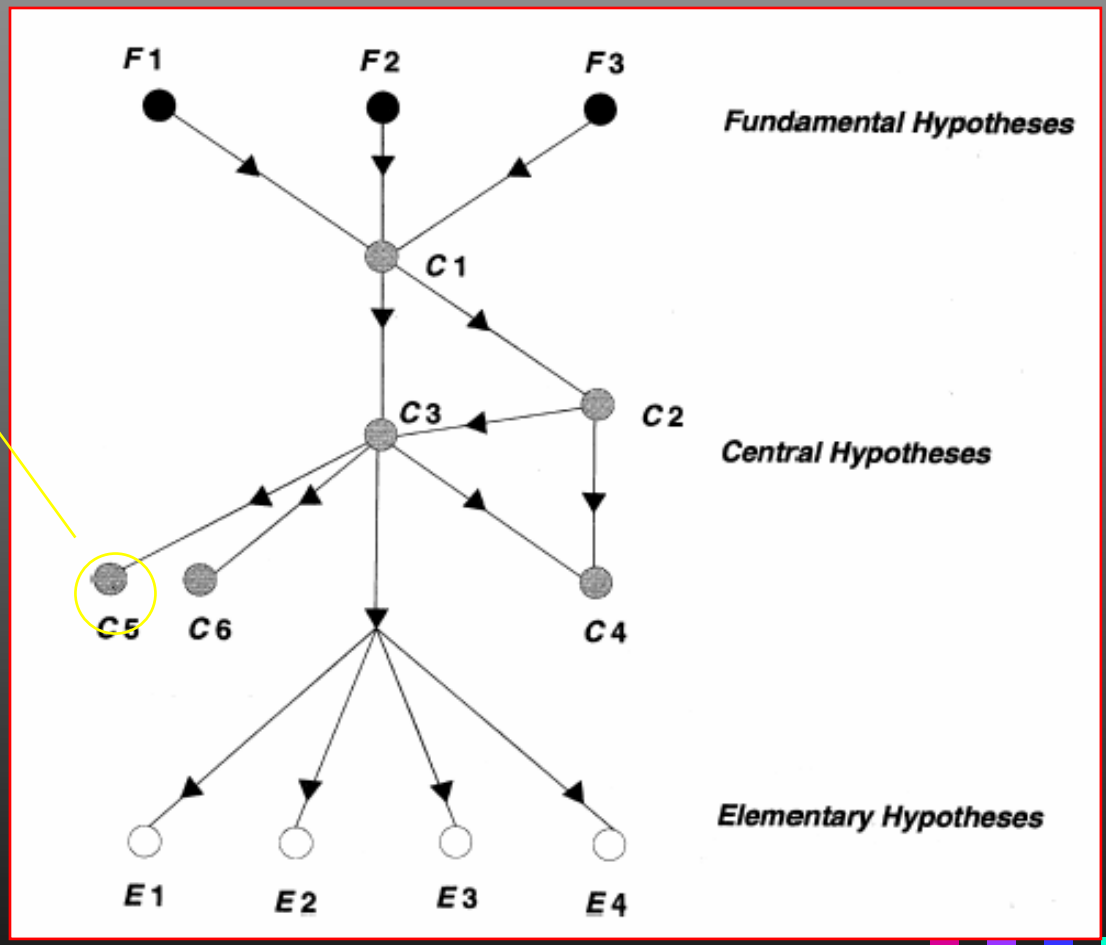
4. The preliminary Software Engineering Theory

task to be solved



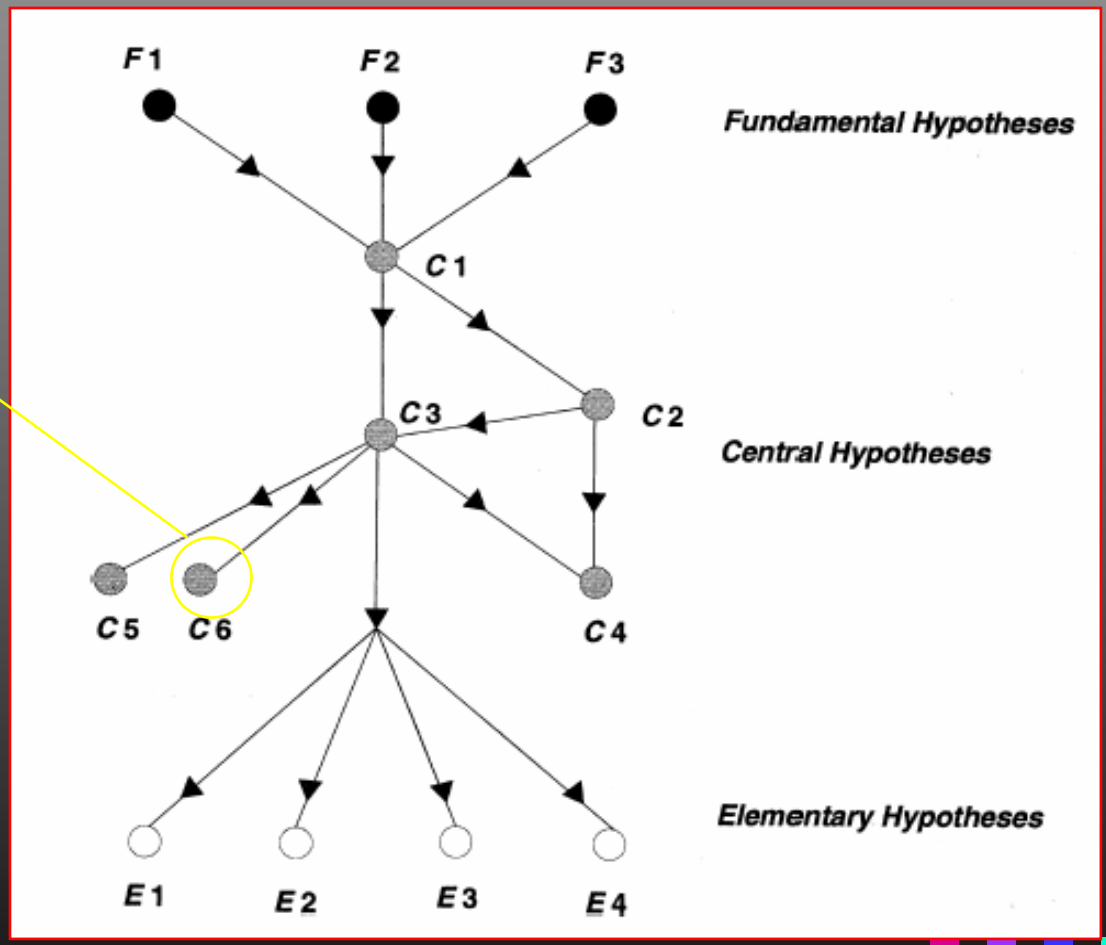
4. The preliminary Software Engineering Theory

client

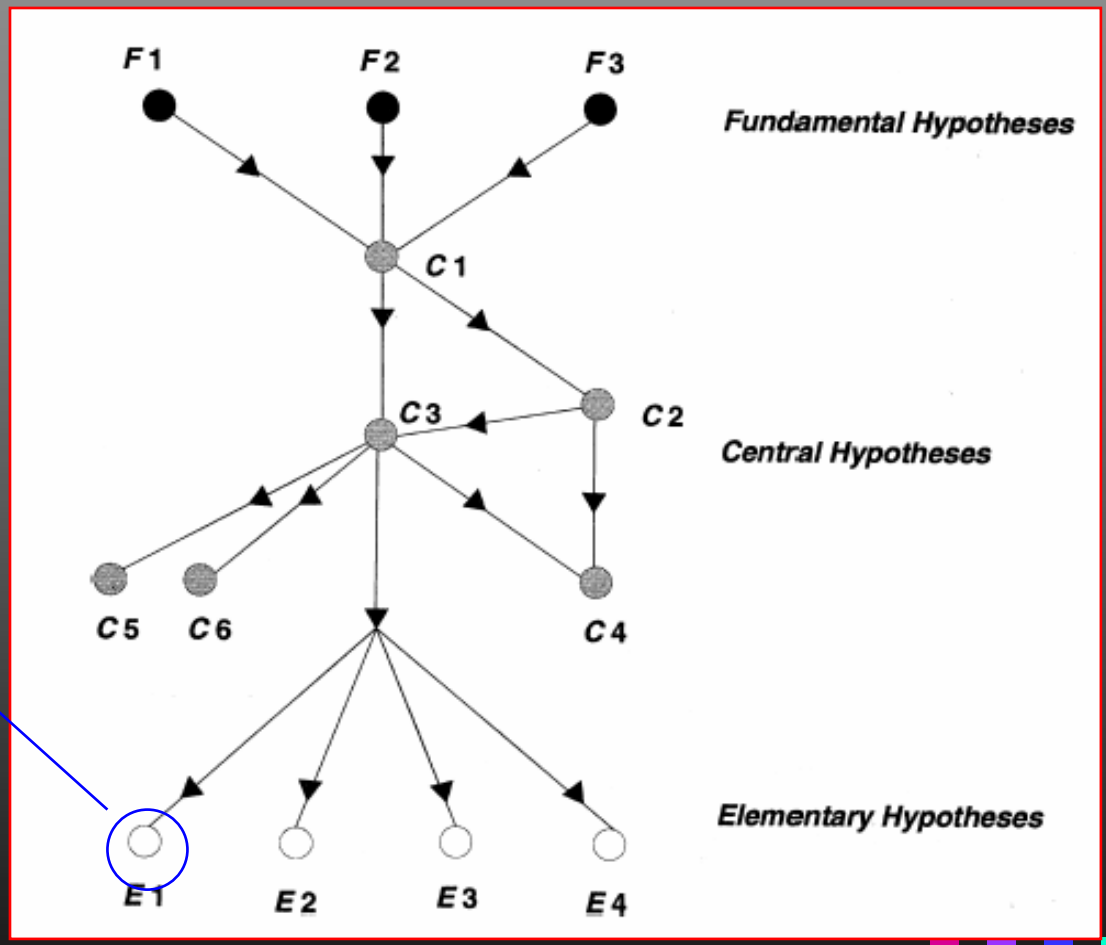


4. The preliminary Software Engineering Theory

concept

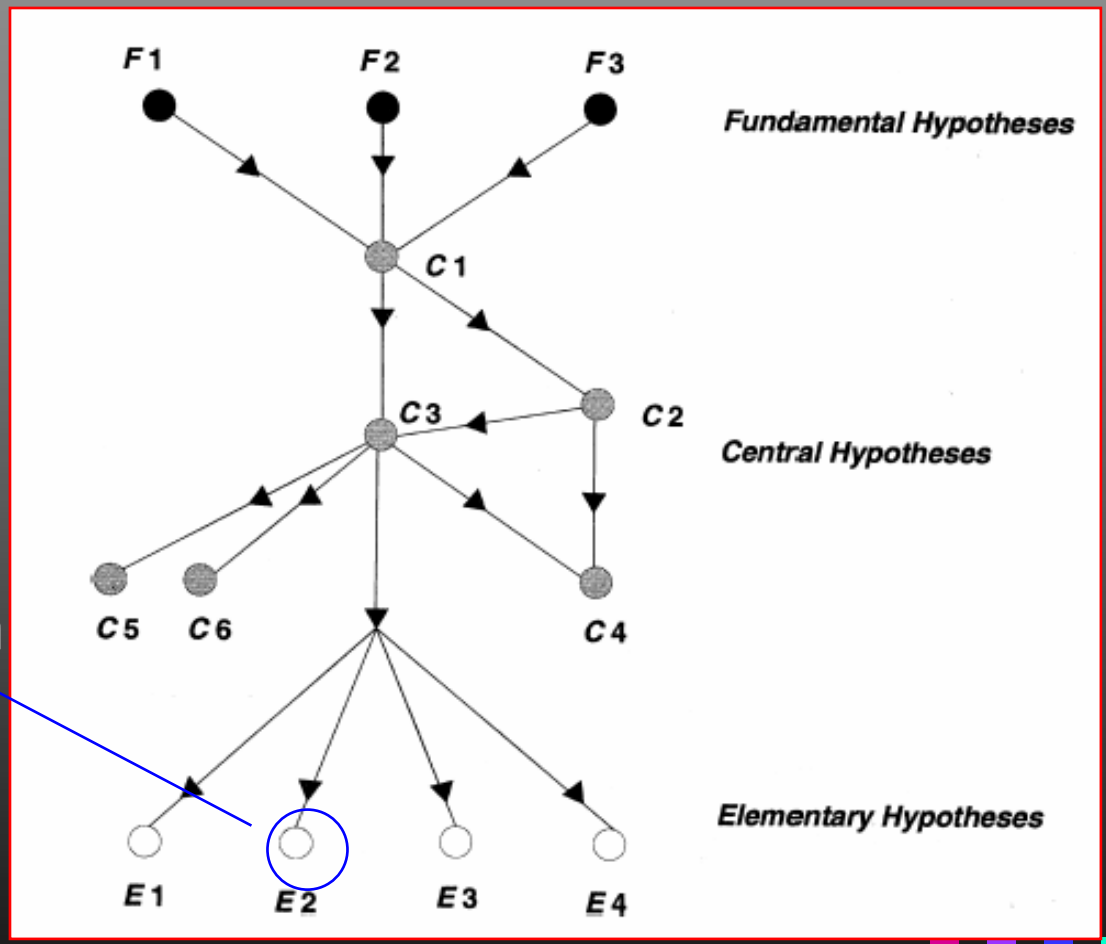


4. The preliminary Software Engineering Theory



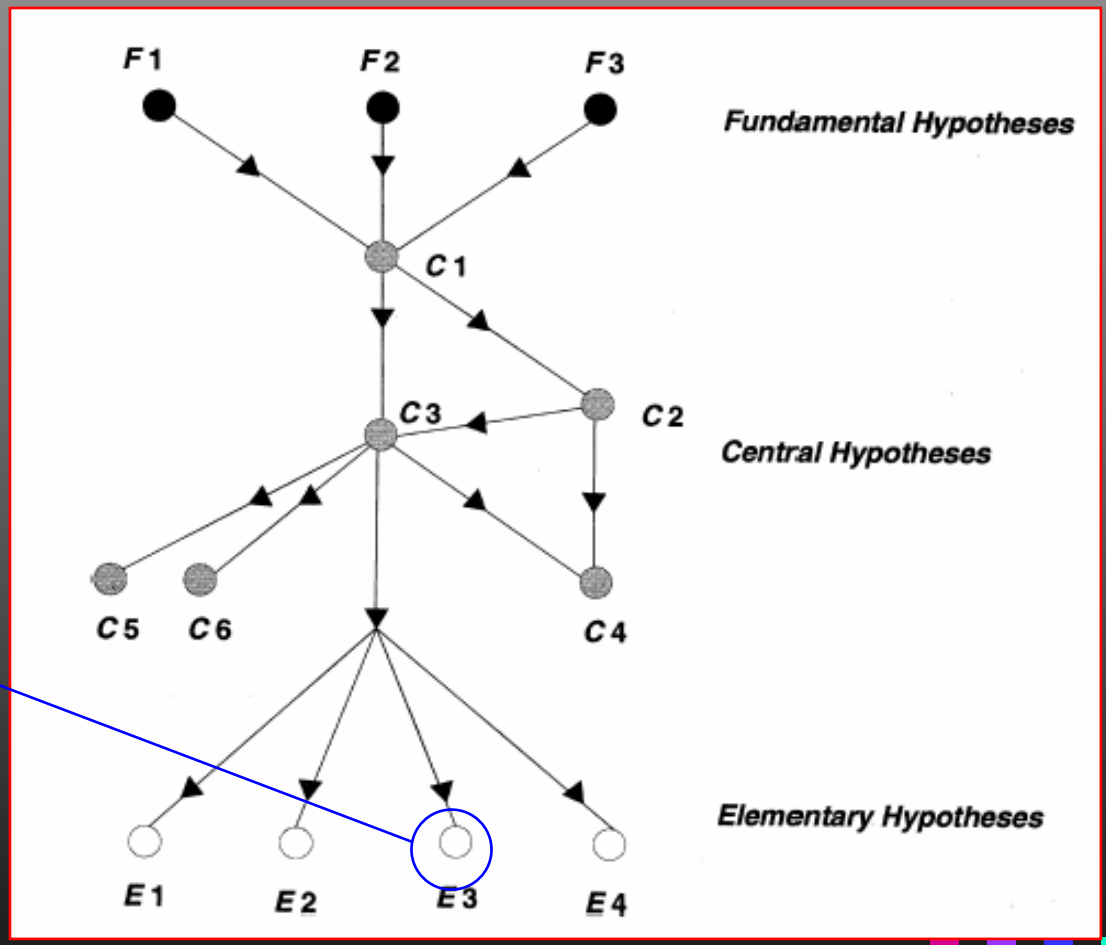
Design

4. The preliminary Software Engineering Theory



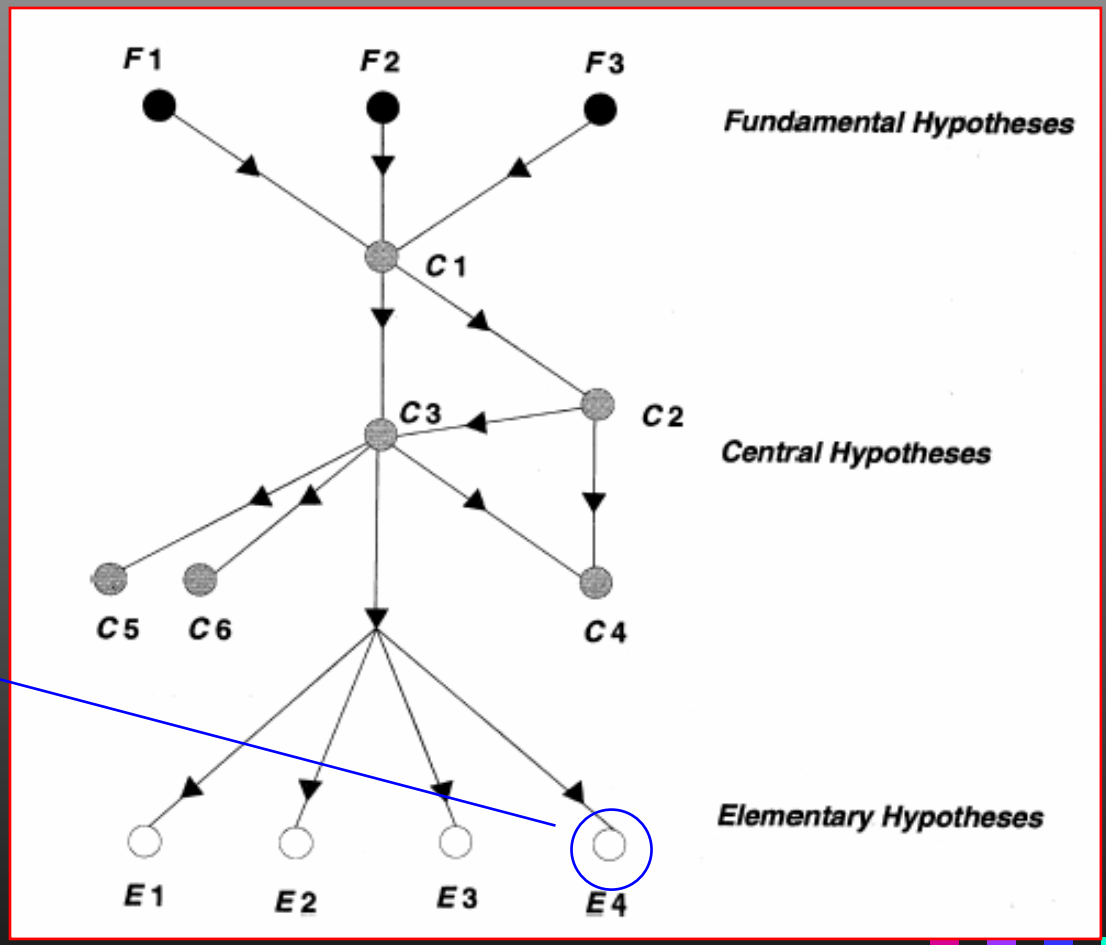
implementation

4. The preliminary Software Engineering Theory



testing

4. The preliminary Software Engineering Theory



Quality assurance

4. The preliminary Software Engineering Theory

The theory used for:

1- Posing and Reformulating fruitful problems (C2,C4,C5)

2- Suggesting the gathering of new data (E1,E2,E3,E4)

3- Suggesting Entirely new lines of investigation-

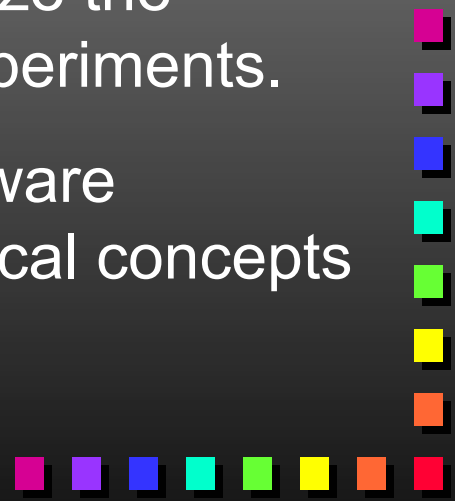
(All hypothesis)

5. Conclusion

New problems and new investigation lines were derived from the developed preliminary software engineering theory.

The submitted preliminary software engineering theory is a first attempt to systematically summarize the present results of software engineering experiments.

Next step→→ the formalization of the software engineering theory by algebraic or theoretical concepts



References

1-Daly, J.,Brooks,A., Miller, J., Roper,M. , and Wood M.1996. Evaluating inheritance depth on the maintainability of object oriented software 1(2): 109-132

2-Moynihan,T.1996. An Experimental comparison of object-oriented and functional decomposition as paradigms for communication system functionality to users. Journal of systems and software 33(2): 163-169

3- Porter,A. A. ,Votta, L. G. , and Basali, V. R. 1995.Comparing detection methods for software requirements inspections: A Replicated experiments. IEEE Transaction on software engineering SE-21(6):563-575

References

4- Swigger,K. M., and Brazil,R. P. 1991.An empirical study of the effects of design/documentation format on expert system modifiability. In: C. R. Cook, J. C. Schultz and J. Z. Spohrer: empirical studies of programmers: fifth work shop. Norway, apple puplishing,210-226.