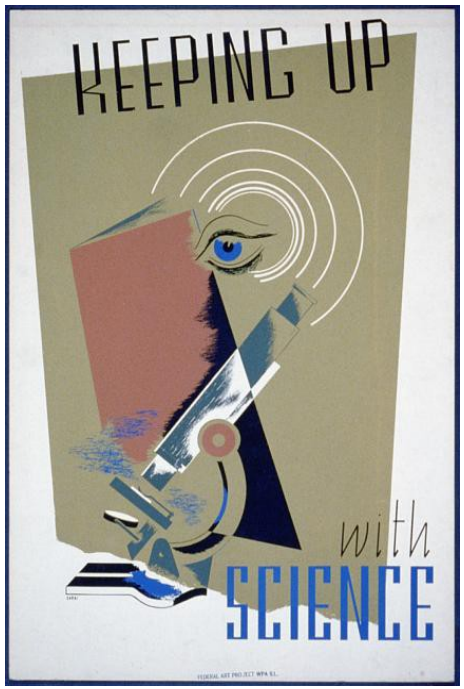
32. Different Types of Research Hypotheses, Questions, Methods, and Results in Software Engineering

Prof. Dr. Uwe Aßmann
Softwaretechnologie
Technische Universität Dresden
2016-0.6, 16-5-21
http://st.inf.tu-dresden.de/teaching/asics

1) Shaw's classification of Hypothesis and Questions
2) Types of papers



[Library of Congress WPA poster]

Obligatory Literature

- [Shaw-Research] Mary Shaw. What makes good research in software engineering? Int. Journal of Software Tools for Technology Transfer (STTT), 4(1):1-7, 2002.
- [Shaw-ETAPS02] Mary Shaw. Slide set of key note at ETAPS 2002. Good summary of [Shaw-Research]
- Mary Shaw's web site http://spoke.compose.cs.cmu.edu/shaweb/
- [Bundy] Alan Bundy. How to Write an Informatics Paper. Web page:
  - http://homepages.inf.ed.ac.uk/bundy/how-tos/writingGuide.html
- [Gonzalez] Fabio A. Gonzalez. Writing a Research Paper Depto. de Ing. de Sistemas e Industrial Universidad Nacional de Colombia, Bogota

References

- Dieter Rombach. Klaus Endres. A Handbook of Software and Systems Engineering. Addison-Wesley.
- [Xu-Nygard] Dianxiang Xu and Kendall E. Nygard. Threat-driven modeling and verification of secure software using aspect-oriented petri nets. IEEE Trans. Software Eng, 32(4):265-278, 2006.
- Fun:
  - Scientific Balloons
    - http://www.centennialofflight.gov/essay/Dictionary/Scientific_Balloons/DI72.htm

Tribute

- The web site of Mary Shaw's research course, its literature link page
  - http://spoke.compose.cs.cmu.edu/ser04/R/bib-meta.htm

Mary Shaw: "A research paper is a purposeful, designed artifact, just like a software system. Apply software design techniques to paper design:

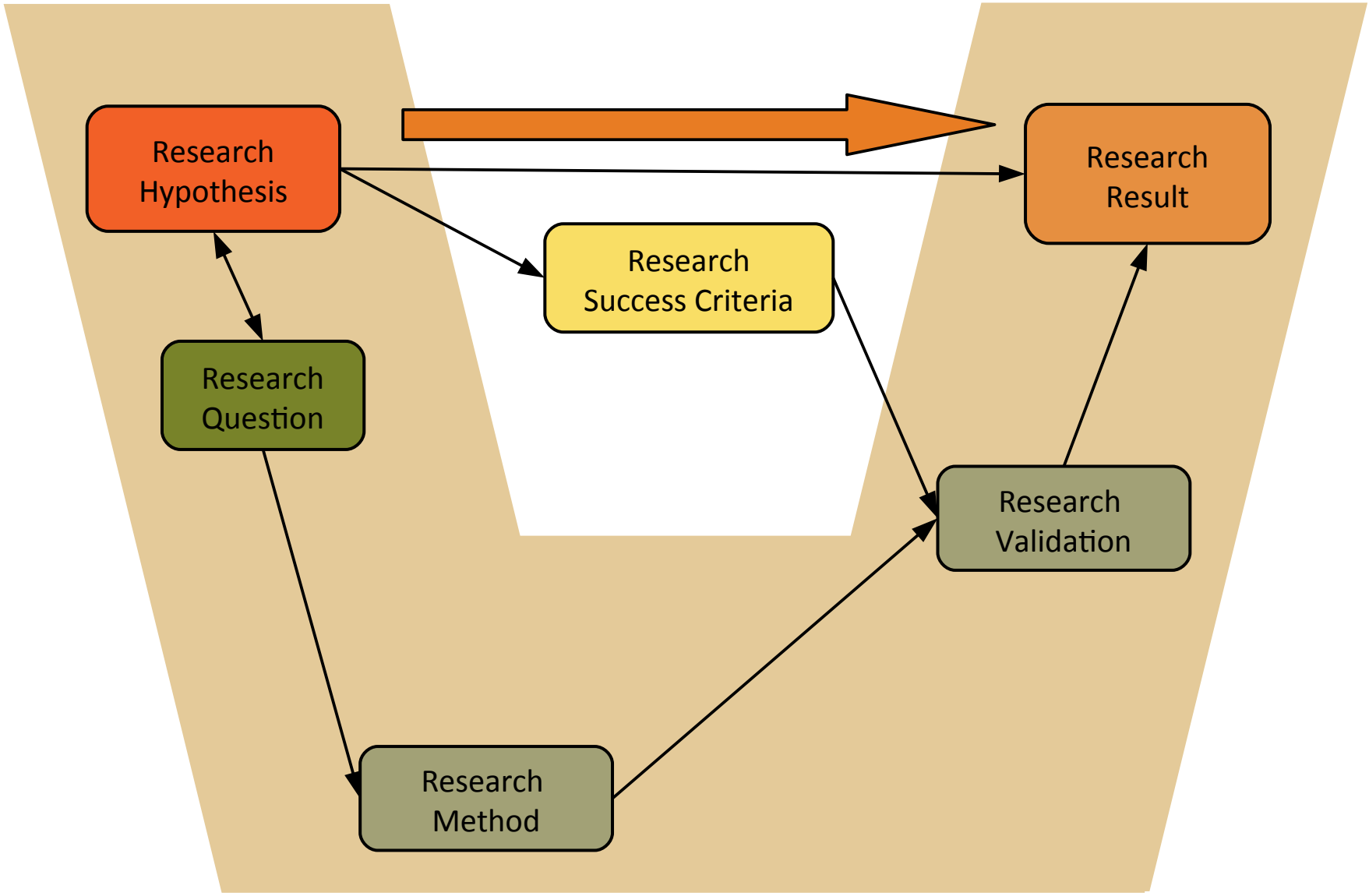
- *Start with the requirement:* read the call for papers
- *Select an architecture:* plan the sections, what they say
- *Plan a schedule:* allow time for review, revision
- *Check consistency:* type-check text like code"

http://spoke.compose.cs.cmu.edu/shaweb/images/mary-shaw.jpg

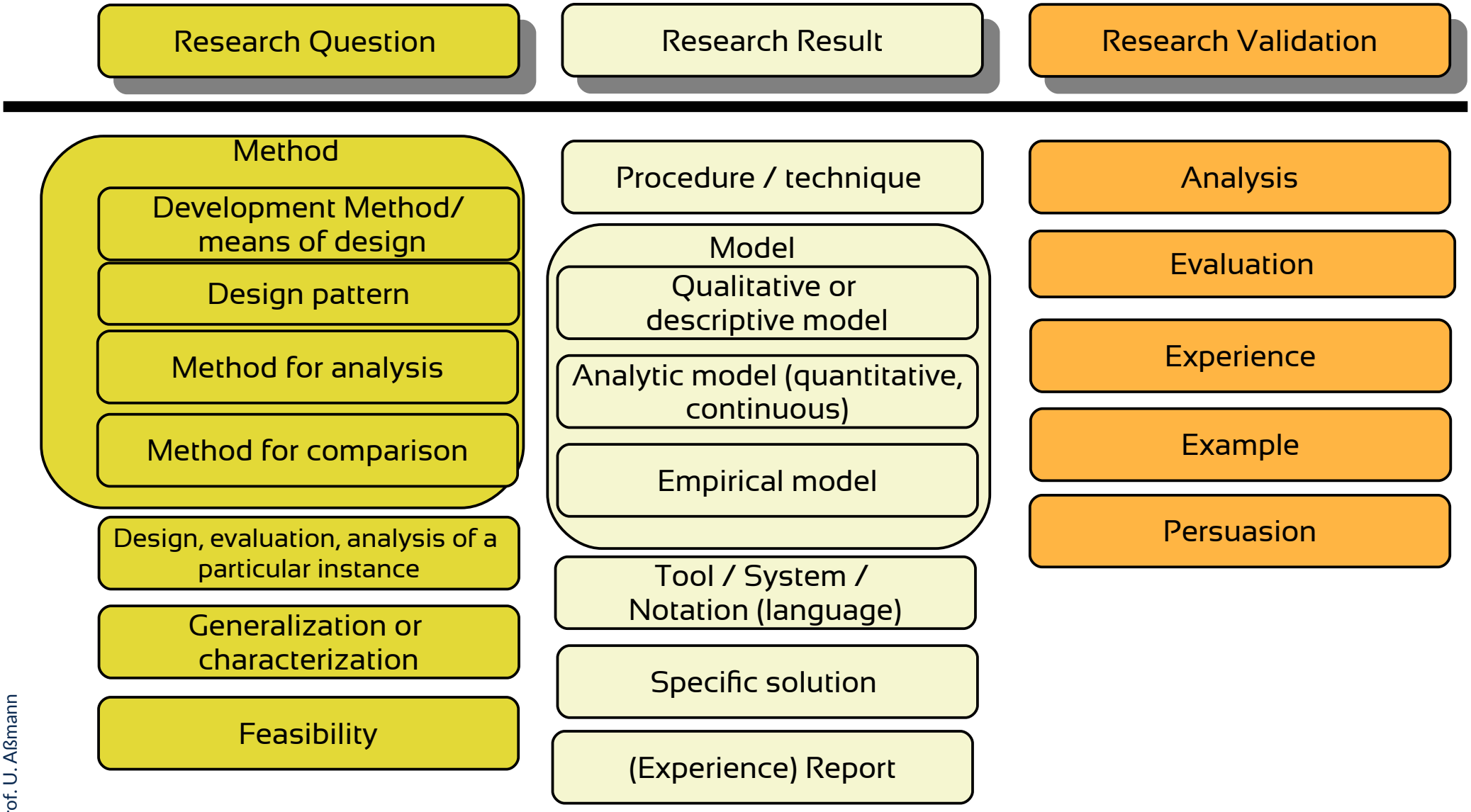
32.1 Shaw's Classification of Research Hypotheses in Software Engineering

.. and how to make more template abstracts out of the classes

The Extended Shaw Model for Research Hypothesis



Shaw's Original Facet Classification



Research Questions

Type of Question		Examples of Research Questions
New Development Method or means of development		How can we do/create (or automate doing) X? Is there a best practice how to do X? A design pattern?
	Optimized Development Method	What is a better way to do/create X?
Method for analysis		How can I evaluate the quality/efficiency/correctness of X? How do I choose between X and Y?
	Method for comparison	How do I systematically compare between X and Y? What are the criteria for comparison and contrast?
Design, evaluation, or analysis of a particular instance		What is a (better) design or implementation for application X? What is property X of artifact/method Y? How does X compare to Y? What is the current state of X / practice of Y?
Generalization or characterization		Given X, what will Y (necessarily) be? What, exactly, do we mean by X? What are the important characteristics of X? What is a good formal/empirical model for X? What are the varieties of X, how are they related?
	Advantages of classifications	Investigate the special features of all classes of a classification. Find criteria to test membership in these classes and then apply the special features. Example: AG hierarchy, XGRS classes
Feasibility		Does X even exist, and if so what is it like? Is it possible to accomplish X at all?

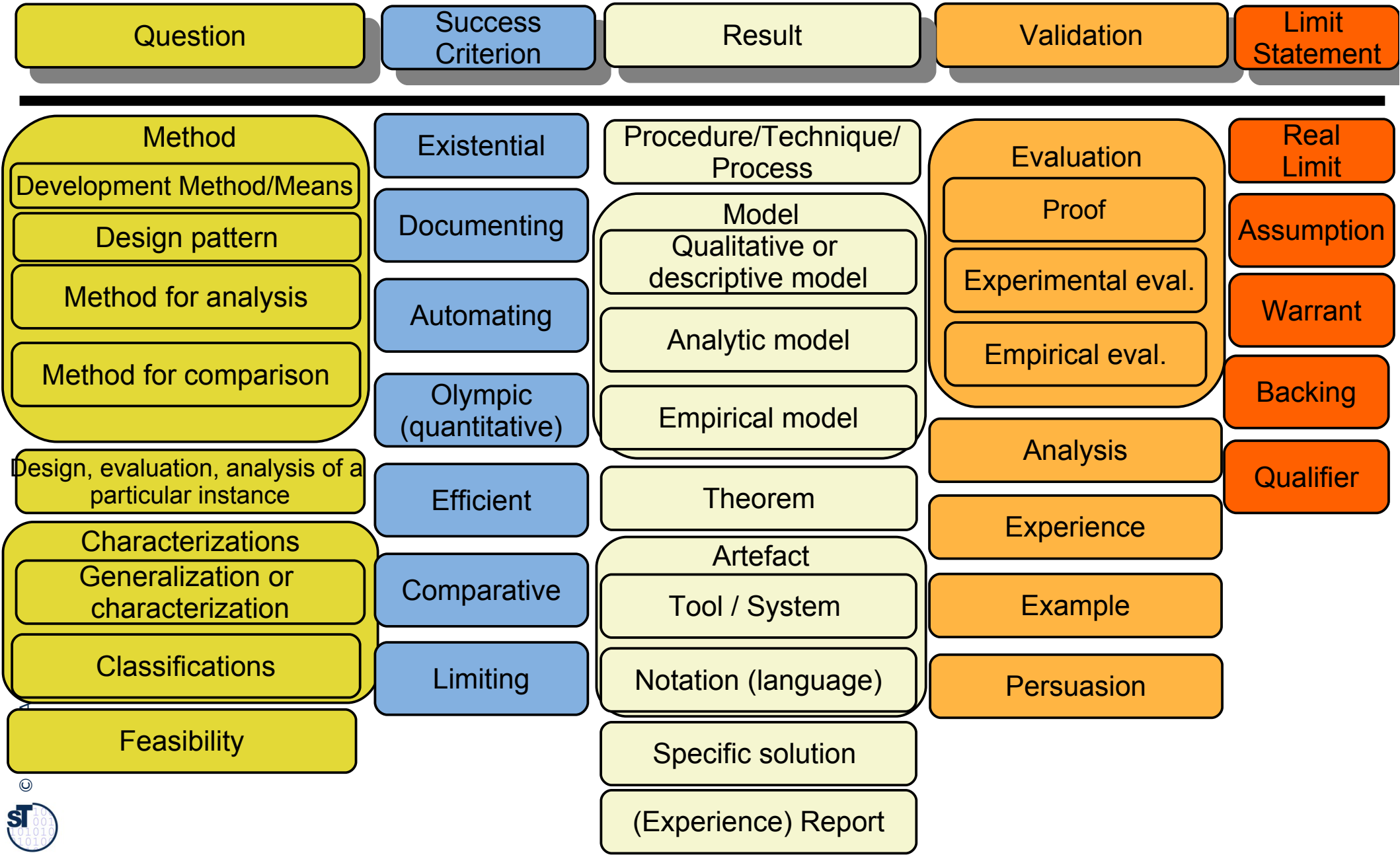
Research Results

Types of Research Results		Example of Research Result
Procedure / Technique / Process		New/better ways to do development/analysis tasks
Model	Qualitative or descriptive model	Structure/taxonomy/ontology for problem area; framework Informal guidance, informal domain analysis
	Analytic model	Structural model that permits formal analysis, automation
	Empirical model	Empirical predictive models based on real data
Tool / System		Tool that embodies model or technique
Notation (language)		New language with better X. Ex.: Gradual typing;
Specific solution		Solution to application problem applying SE principles, or result of specific analysis
(Experience) Report		Interesting observations, rules of thumb, heuristics best practices, case studies, industrial case studies
Theorem		New theorem in an existing model. Ex: Register allocation with graph cliques is polynomial (complexity), equivalence

Research Validation (Evaluation)

Type of validation		Examples of Phrases
Analysis		I have analyzed my result and find it satisfactory through … • for a empirical model: ..data on controlled use • for a controlled experiment: ...a carefully designed statistical experiment
	Experience	My result has been used on real examples by someone other than me, and the evidence of its correctness / usefulness / effectiveness is … • for a qualitative model: ….narrative • for a empirical model, tool: … some data, usually statistical, on practice • for a notation, technique: … a comparison of this with similar results in actual use
	Example	Here's an example of how it works on... • for a toy example: perhaps motivated by reality • for a slice of life: a system that I have been developing
	Evaluation	Given the stated criteria, my result... • for a descriptive model: .. adequately describes the phenomena of interest • for a qualitative model: ...accounts for the phenomena of interest... • for an empirical model: ...is able to predict ... because ..., or ... gives results that fit real data … Includes feasibility studies, pilot projects
Persuasion		I thought hard about this, and I believe that... • for a technique: ..if you do it the following way... • for a system: ... a system constructed like this would... • for a model: … this model seems reasonable... • for feasibility: … my working system is persuasive, even without analysis
Blatant assertion		No serious attempt to evaluate result

The Shaw Facet Classification, Slightly Extended with Success Criterion and Limit Statement



32.3 Types of Papers based on the Shaw Facets

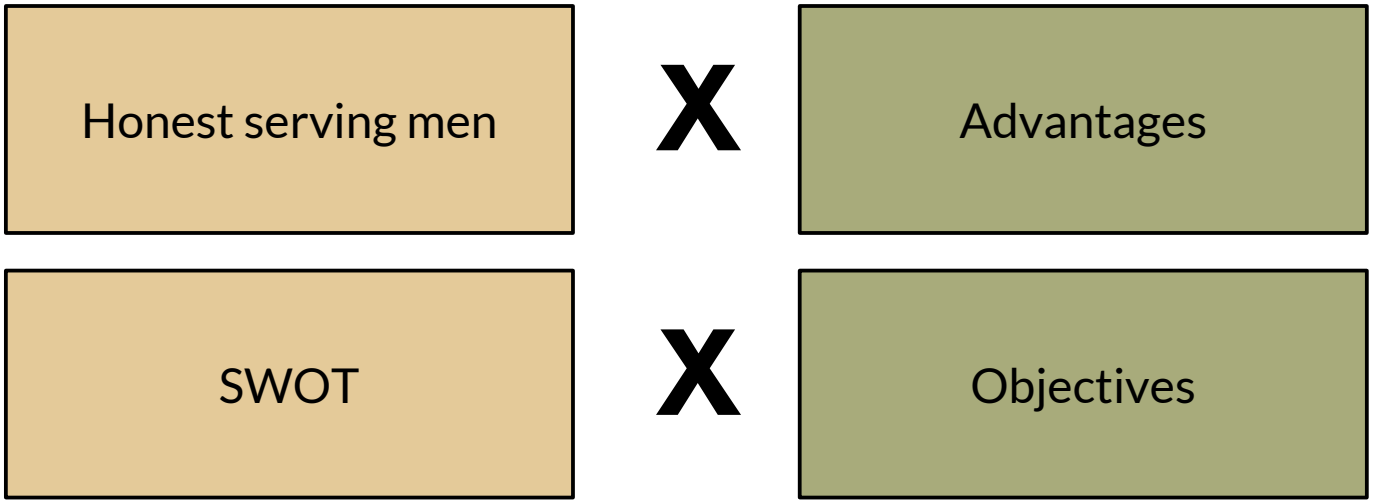
32.3.1 Problem Analysis Papers

Problem-Objective Analysis Papers

- Already done in Unit 1
- Use ZOPP, B-POPP, GQM, AO-PA, etc. to analyze the problems and goals of
  - a stakeholder
  - a domain
  - a method
- Define success factors for possible future solutions
- Indicate how solutions could look like

Aspect-Oriented Problem Analysis (AOPA) Papers

- Evaluate a SoC space to write a paper
- Fix a set of concerns (concern space)
- Fix a set of things (artifact space)
- Define a crossproduct and discuss every combination (separation of concerns)

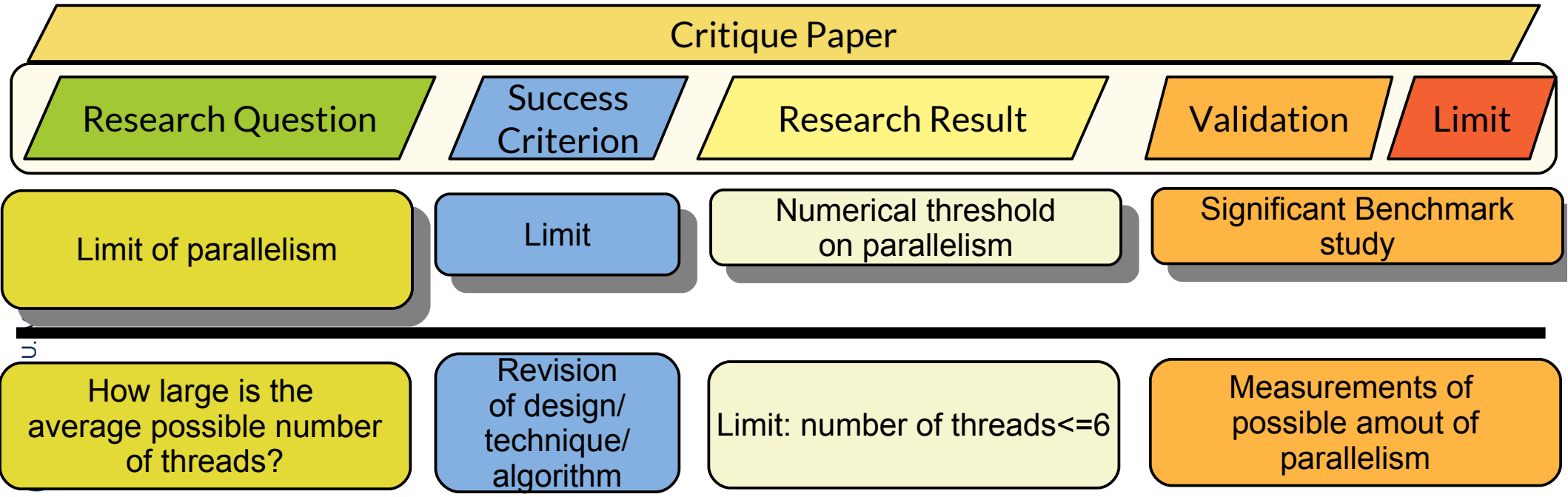


Critique Paper (Limitation Paper, Technical Problems Paper)

- A **critique paper** contains an analysis
  - why another approach is **deficient**,
  - Bug in proof found
  - why it has its **limits**,
  - limits were not mentioned
  - limits were found
  - why a paper used **unrealistic assumptions**
  - why an idealized research result does not work in practice
  - Invalid assumptions (invalid warrant)
  - why a paper should have used a qualifier, but didn't
- E. W. Dijkstra. Goto statement considered harmful. Communications of the ACM, 11:147-, 1968. Final judgement on unstructured programming in C and C++.
- **Per Brinch Hansen. Java's Insecure Parallelism.** ACM SIGPLAN Notices, 34 (4):8, April 1999. Brinch Hansen's condemnation of Java, based on his background on monitors:
  - Per Brinch Hansen. Monitors and Concurrent Pascal: a personal history. ACM SIGPLAN Notices, 28(3):1-35, March 1993.

Critique Paper (Limitation Paper, Technical Problems Paper)

- In a well-known approach, you have identified a **technical problem**
  - a deficiency | a limit | a prerequisite or precondition
  - In your paper, you cure the technical problem, remove the limit, generalize the preconditions
- **Limit discussion:** discuss the limits of the well-known technology.
  - D. W. Wall. Limits of instruction-level parallelism. In Conference on Architectural Support of Operating Systems IV, pages 176-188. ACM, 1991.
  - Wall's paper showed that on instruction level, many programs have only up to 6 threads, which limits parallelism



32.3.2 Teaching Papers

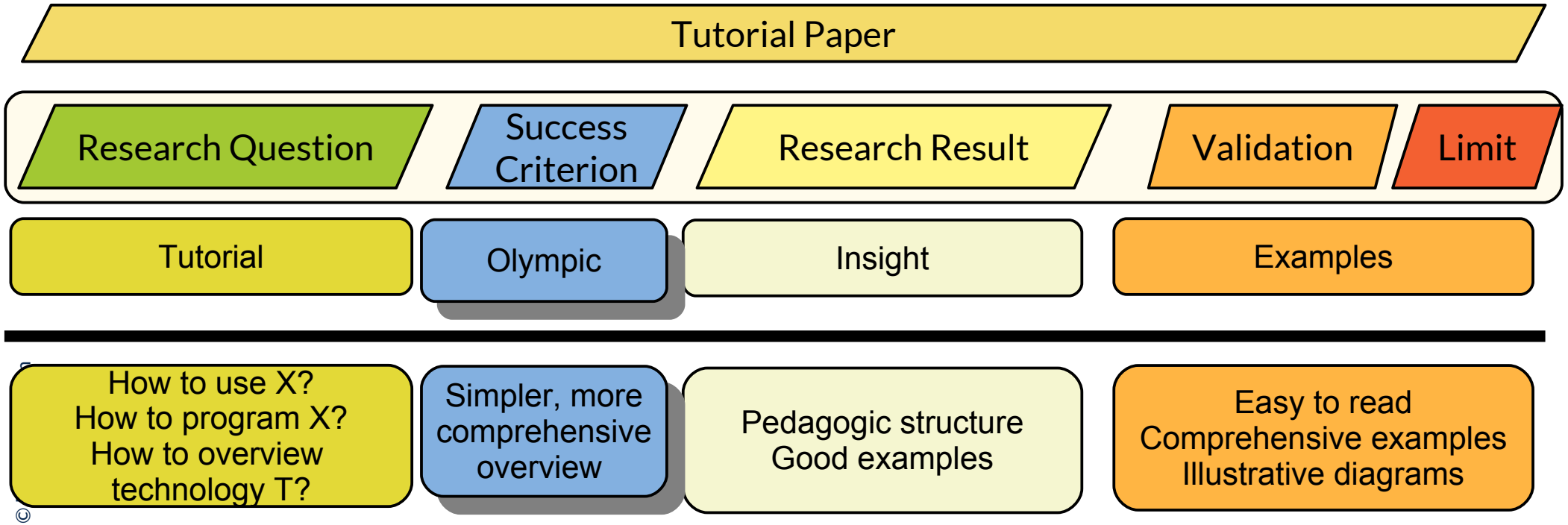
- A new language may solve some problems easier than another existing one

Tutorial Paper

- A good **tutorial paper** contains:
  - A set of running examples
  - Bottom-up explanation of concepts and ideas
  - Precise definitions of concepts
  - Classifications of concepts
  - Illustrative figures
  - Some theorems (idealistic research)
  - or case studies (practical research)
- In the SEW course, we use
  - Markus Müller-Olm, David Schmidt, Bernhard Steffen. Model-Checking. A Tutorial Introduction. Springer LNCS, Volume 1694, 1999, p 848ff
    - http://www.springerlink.com/content/l437dulbgk67jl6m/
  - [BW04] Timed Automata: Semantics, Algorithms and Tools, Johan Bengtsson and Wang Yi. In Lecture Notes on Concurrency and Petri Nets. W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, 2004
    - http://www.it.uu.se/research/group/darts/papers/texts/by-lncs04.ps
  - [BDL04] A Tutorial on Uppaal, Gerd Behrmann, Alexandre David, and Kim G. Larsen. In proceedings of the 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM-RT'04). LNCS 3185.
    - http://www.cs.auc.dk/~adavid/publications/21-tutorial.pdf

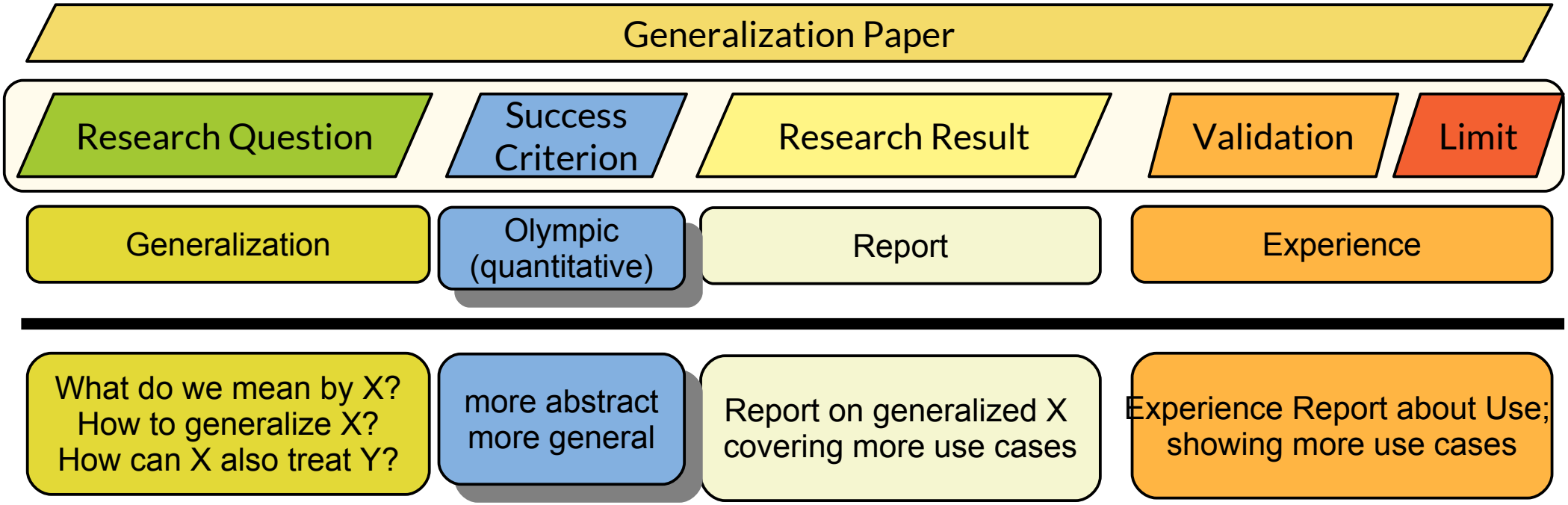
Tutorial Paper

- Speeds up comprehension



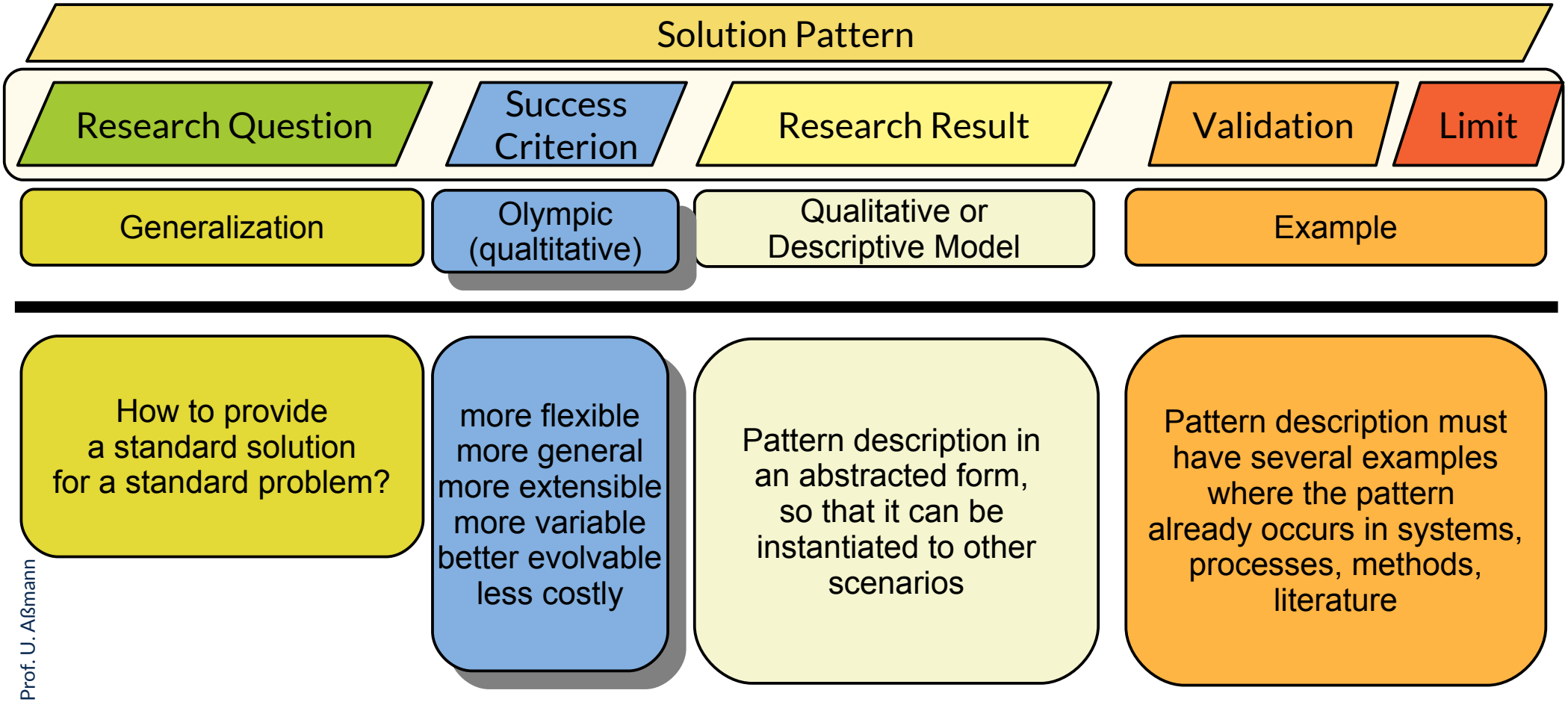
Generalization Paper, Based on Experience

- A **generalization paper** introduces a more general technique, or generalizes or abstracts several other techniques
- Difficult to write



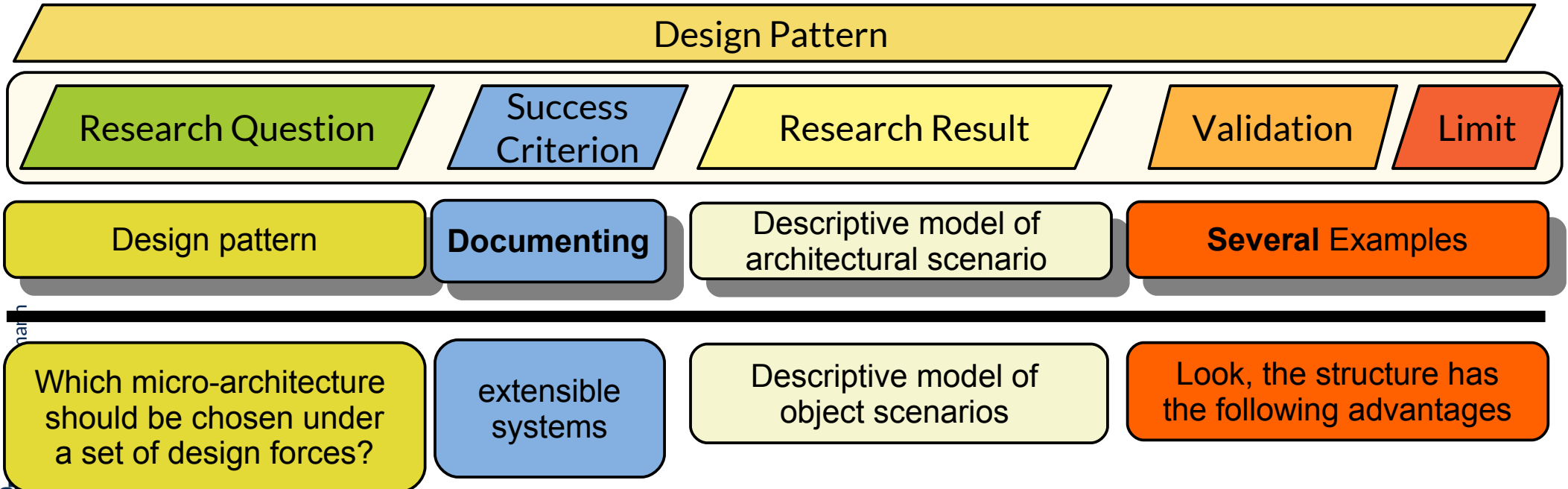
“Solution Pattern” Paper: Special form of Generalization Paper

- How can I solve a standard problem in a specific context with a standard solution?
  - Process patterns, organizational patterns, antipatterns, ...
  - See course “Design Patterns and Frameworks”



Design Pattern Papers

- Design papers need to discuss well-known *design* solutions for well-known problems
  - The criteria of a pattern catalogue (e.g., Gamma)
  - The forces under which they apply
  - Solution patterns
- The research hypothesis is "**documenting**" because a design pattern should not be new, but well-experienced
  - There must be **several examples,** because the pattern must be well-experienced



34.3.3 Typical Structures of POSE Papers

Empirical / Natural Science Outlines

- „Alle Wissenschaft ist problemlösend“ (Popper, Steinbuch)
- Structure of Empirical or Natural Science Paper:
  - Ausgangspunkt: Problem für das eine Lösung gefunden werden soll
  - Charakterisiert durch und logische Abfolge und festen Aufbau
- Abfolge
  - Wissensstand
  - Problem
  - Lösungsweg / Analyse
  - Ergebnisse
  - Lösung
  - Erweiterter Wissensstand

1. Einleitung
1.1 Forschungslage
1.2 Problemstellung

2. Untersuchungsdesign
2.1 Untersuchte Kollektive
2.2 Messmethoden
2.3 Material
2.4 Datenanalyse

3. Ergebnisse

4. Diskussion
4.1 Methoden
4.2 Ergebnisse

5. Zusammenfassung

6. Literaturverzeichnis

7. Anhang

[Gonzalez] Paper Structure (Sections)

- **Title**: should already contain the controlling idea (thesis)
- **Attribution**: Author list, ev. with footnotes on supporting research organizations
- **Abstract** e.g., with MOPARC or Gul Caramel
- **Introduction** should follow a ZOPP-like problem analysis
  - Paragraphs with Background, Problem, Success criteria, Research Question, Research Method, Research Result, Solution: Way how to achieve the result, Roadmap
- **Background**: Terminology, background works
- **Solution**
  - Depends on the type of research question, method
- **Validation**, e.g., Experimental evaluation: what are the findings of the experiments or analyses?
- **Discussion**: Discuss advantages, disadvantages, limits, unique features
- **Comparison** to Related Work: what is the unique feature of the result?
- **Conclusion**: Draw a conclusion
- **Acknowledgement**: Often, research funding organizations want to be acknowledged. Do also not forget helpful colleagues or your supervisor
- **References**
- **Appendices**

Shaw's Paper Structure (Sections)

- http://spoke.compose.cs.cmu.edu/write/t/d/std-otl.htm
- **Abstract**
- **Introduction** (with motivation, problem definition, research question, overview/roadmap of the paper)
- **Related work A** (Background: what is necessary to understanding the present result)
- **Meat of the paper** (the part of the structure that depends on the result; pretty different)
- **Related work B** (relations to other work that compare this work to alternatives or otherwise require the present result as a prerequisite)
- **Summary**, **conclusions**, next steps
- **Acknowledgements**, in partiular funding sources
- **Bibliography**
- **Possibly appendices** (the standard rule for appendices places them after the bibliography, which is a nuisance)

Bundy's Paper Structure

- http://homepages.inf.ed.ac.uk/bundy/how-tos/writingGuide.html
- **Title** should summarize the hypothesis (thesis, contribution) of the paper. The "controlling idea" must shine out
- **Abstract** state the contribution
- **Introduction** motivate the contribution of the paper
- **Literature Survey** allows for positioning the paper into the context
- **Background** (Background: what is necessary to understanding the present work)
- **Theory**
- **Specification**
- **Implementation**
- **Evaluation**
- **Related work** comparison with competitors
- **Further Work**
- **Conclusion**
- **Appendices**

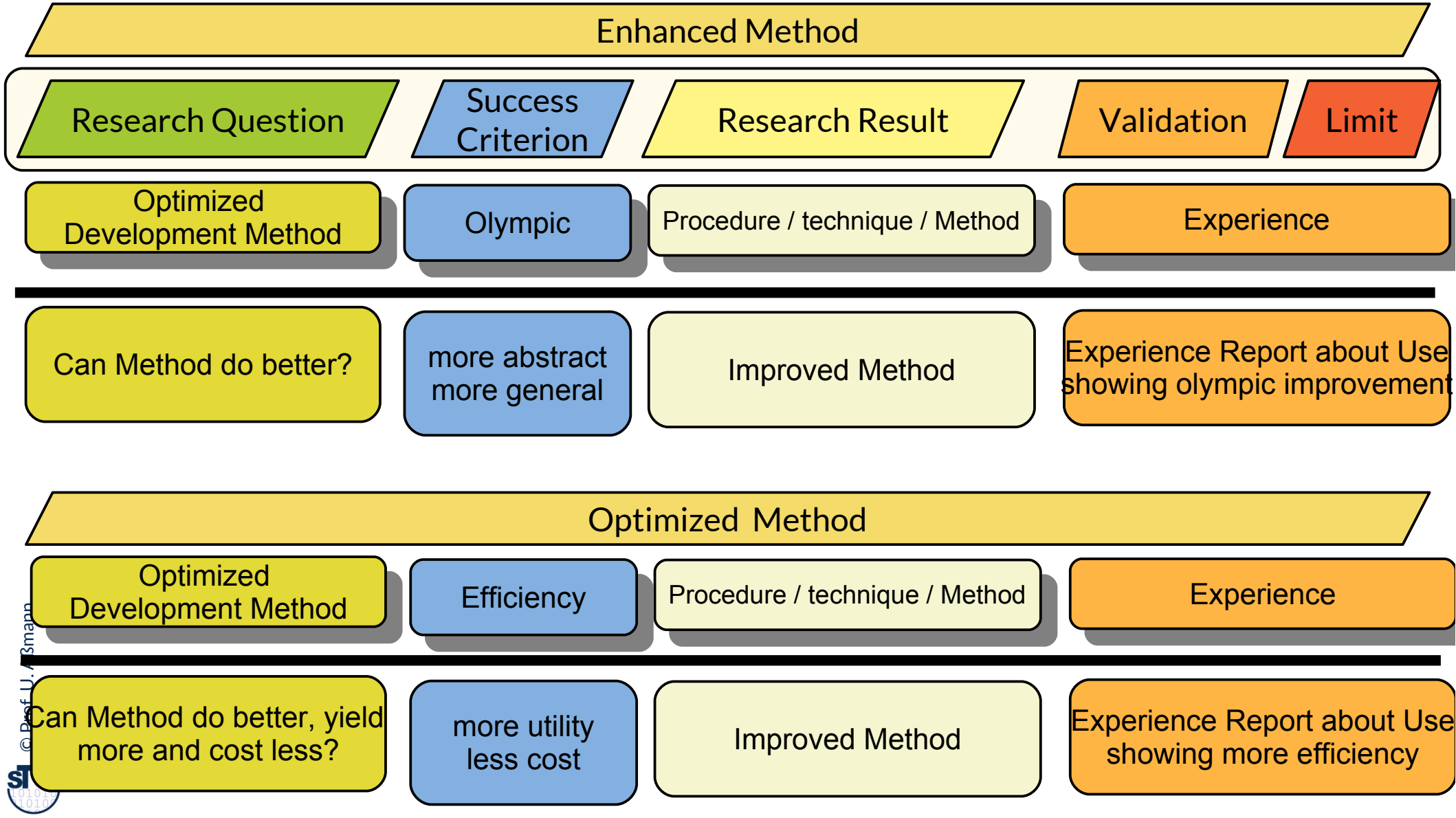
32.4 More Specific, Newman-Abstract-Like Papers

- All Newman template abstracts can be entered into the Shaw classification.

32.4.1 New Solution Paper (Enhanced Solution)

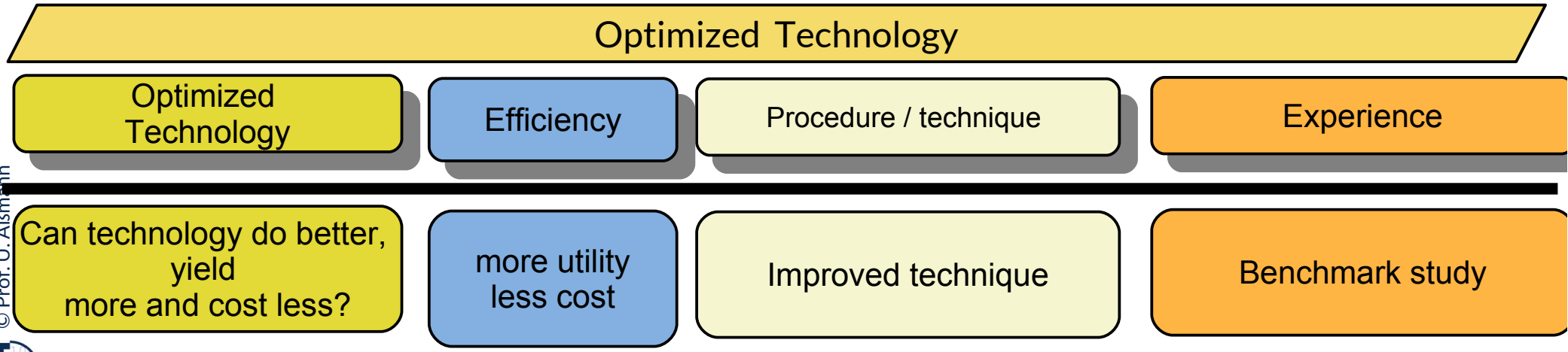
Enhanced/Improved Method (Optimization Hypothesis)

- Special subclass of “Enhanced Solution”



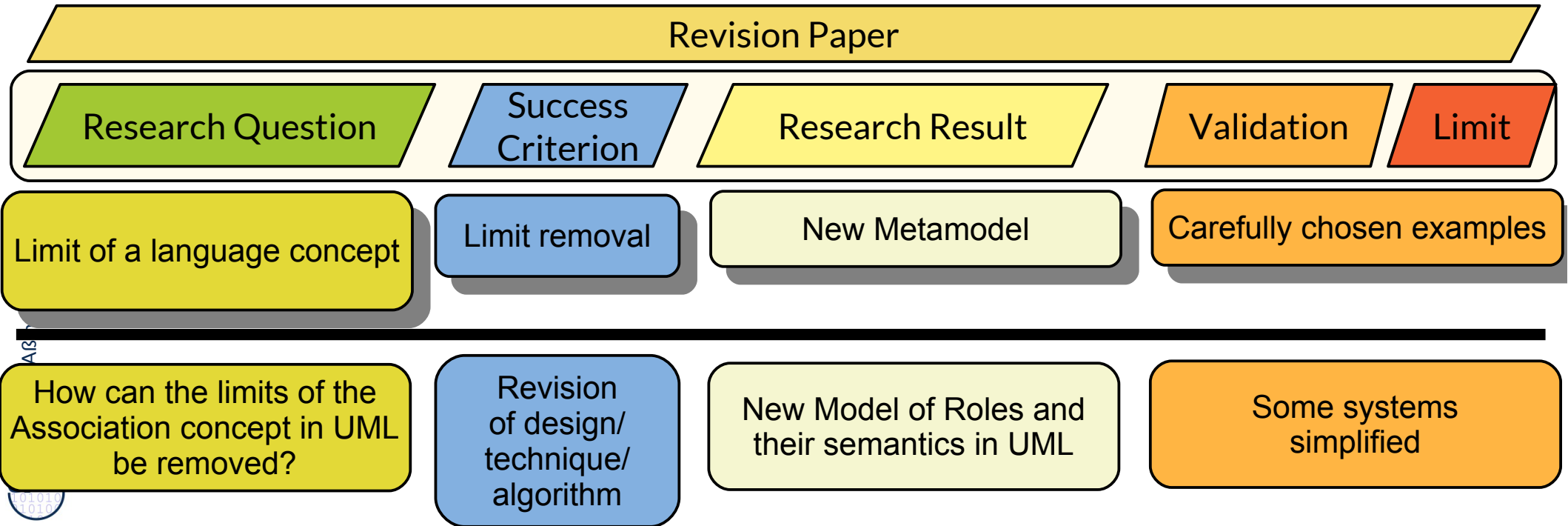
Optimization Technology Paper

- Present an optimization technology (more than an optimized algorithm)
- Show why the current technology is too slow or inefficient
- Show metamodels of optimizing technology
- Give a systems' component diagram
- Give some central algorithms
  - Prove termination
  - Analyze complexity
  - Prove quality features
- Show a case study which proves that your stuff is more efficient

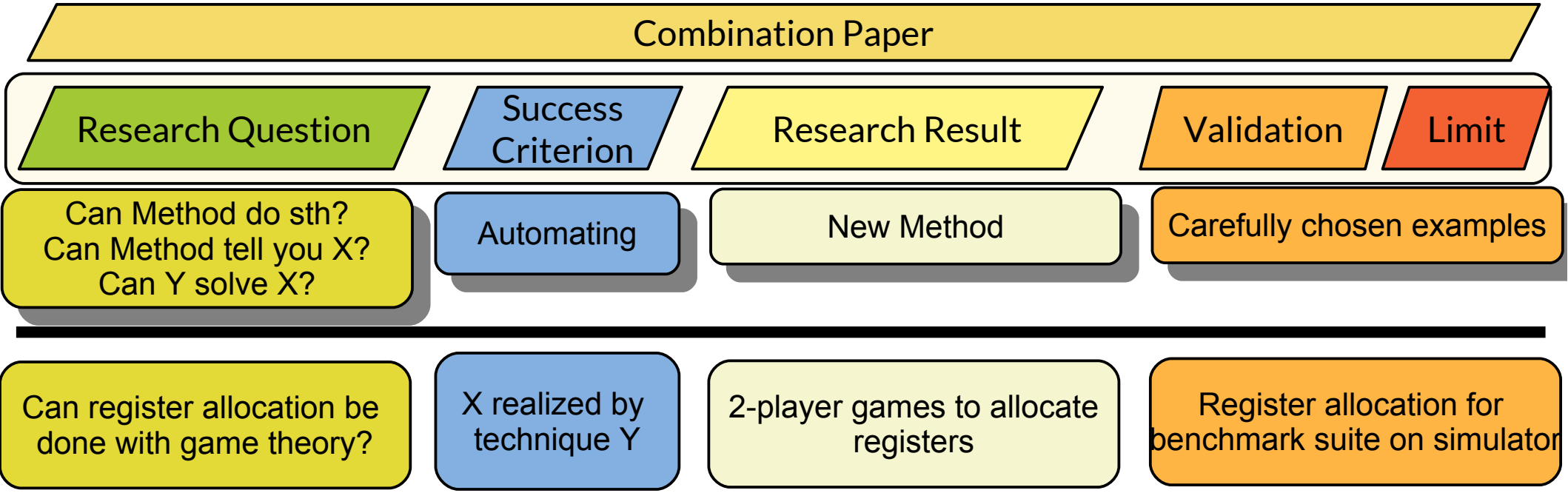


Language Revision Papers (Better Language)

- A **revision paper** extends a critique paper with a revision proposal
- Friedrich Steimann. A radical revision of UML's role concept. In Andy Evans, Stuart Kent, and Bran Selic, editors, UML 2000 - The Unified Modeling Language. Advancing the Standard. Third International Conference, York, UK, October 2000, Proceedings, volume 1939 of LNCS, pages 194-209. Springer, 2000.
- Friedrich Steimann and Thomas Kühne. A radical reduction of UML's core semantics. Lecture Notes in Computer Science, 2460:34-, 2002.

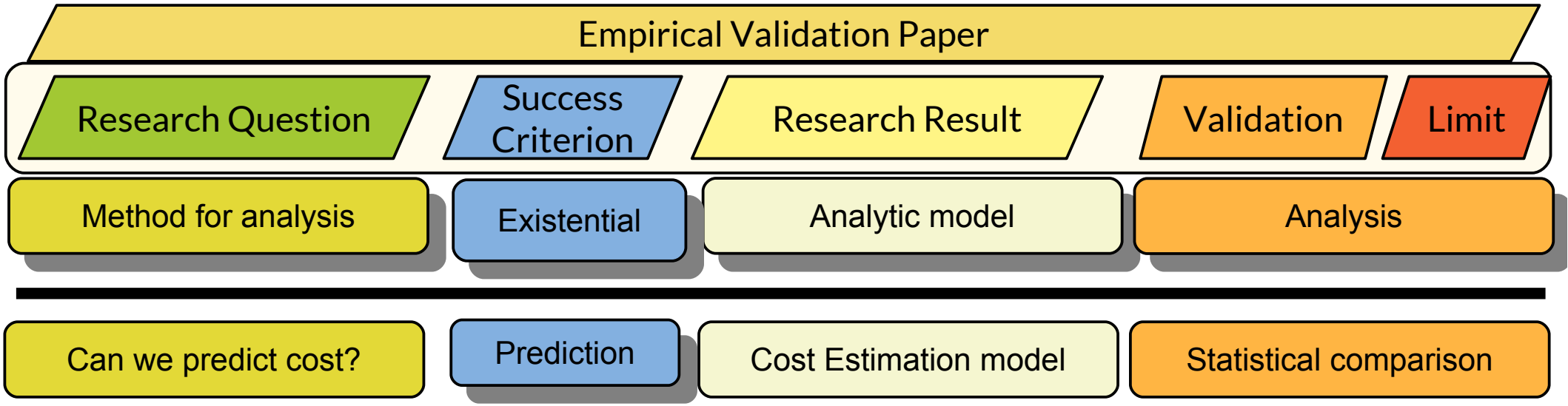


New Method (with Automation Hypothesis), Validated with Examples



- A **combination result** shows that a so far uncorrelated method from another branch in science can solve problem X
  - Ex.: Graph rewrite systems can describe program optimizations
  - How to use Datalog to solve traffic problems

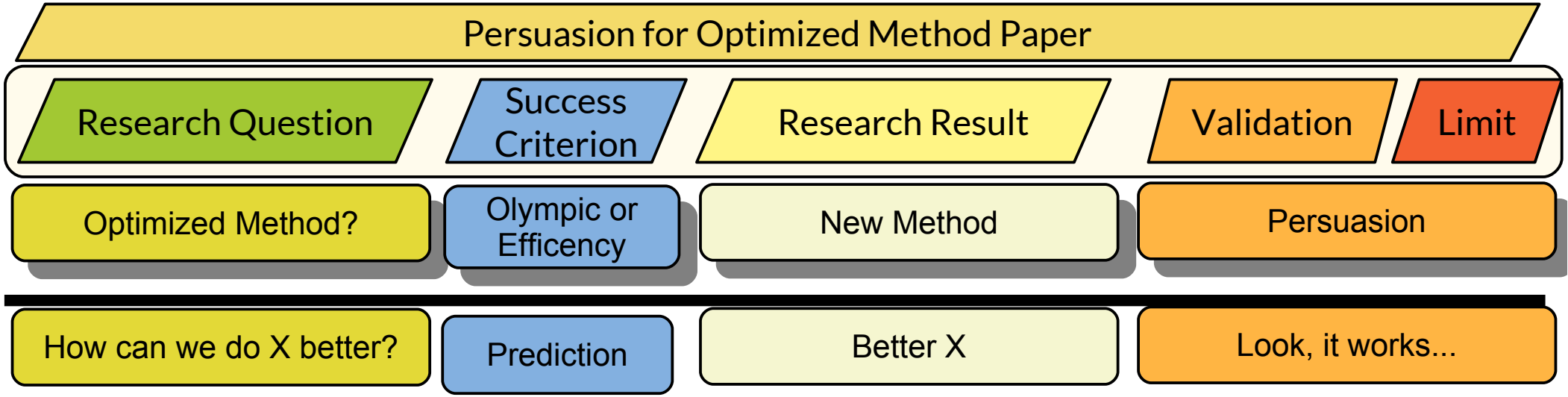
Empirical Validation by Statistics



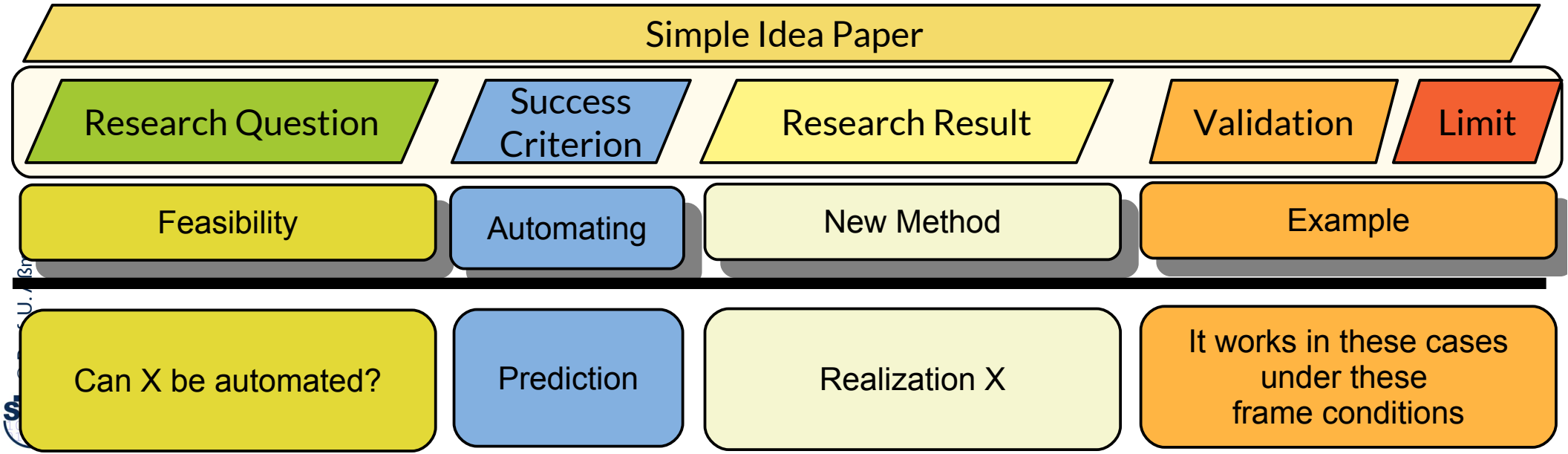
- Empirical validation is possible by
  - statistics
  - controlled experiments with user groups
  - field studies
- Example: [Xu-Nygard] reduces attack trees to aspect-oriented PetriNets and verifies absence of intrusions: first time automating intrusion checking

Not Easy to Publish: Persuasion for Optimized Method

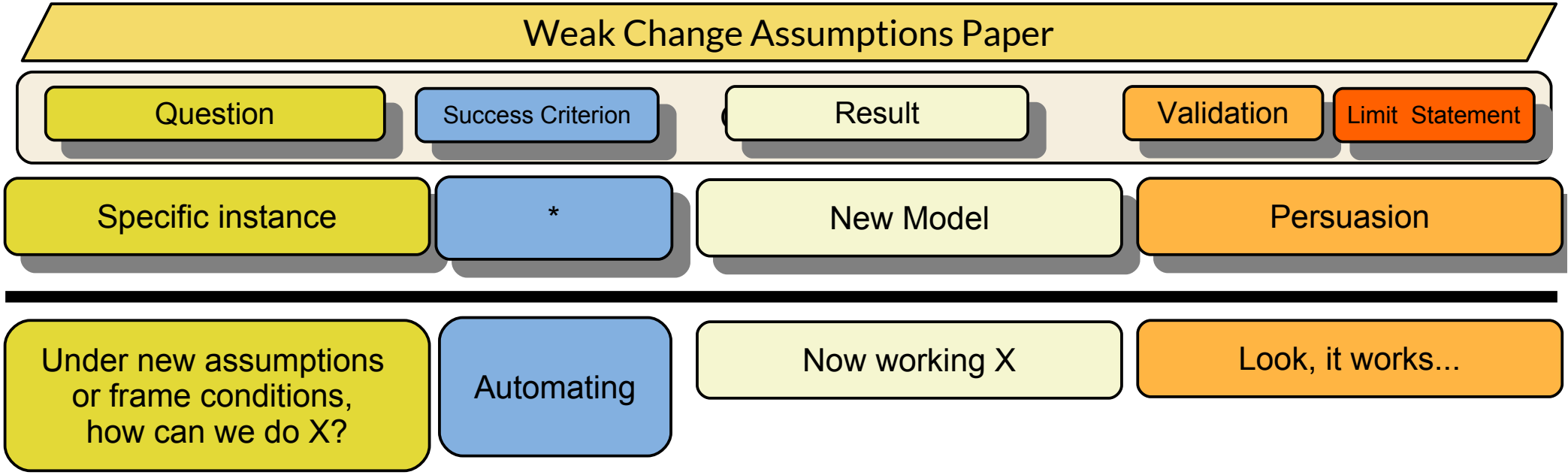
- Hard to Publish:

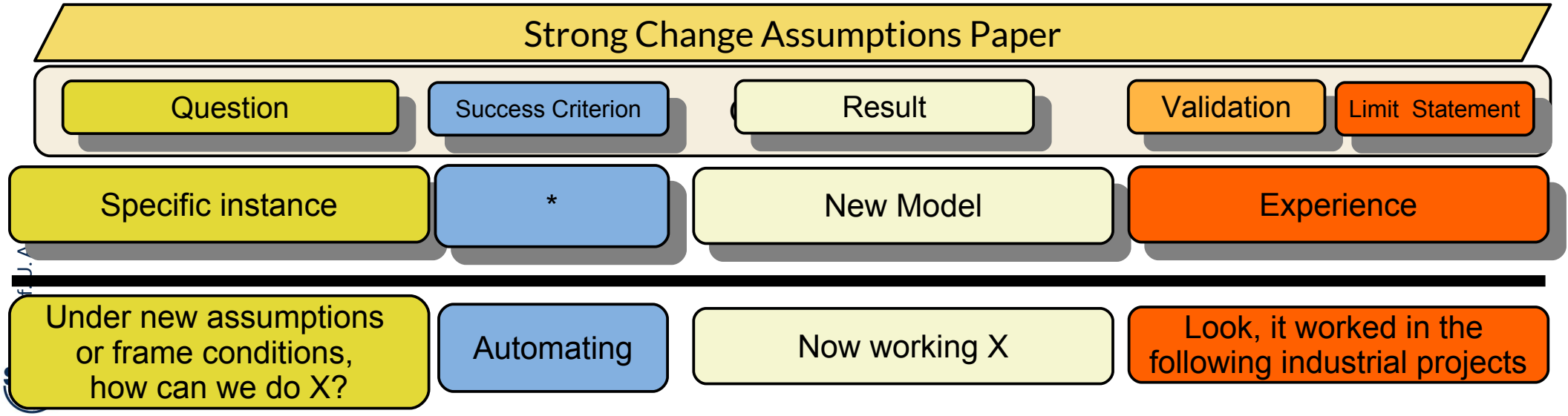


- Simple Idea paper, is more interesting and sometimes published:



Change Assumptions Paper ("..dennoch..")





32.4.2 “New Concepts” Paper (Enhanced Model)

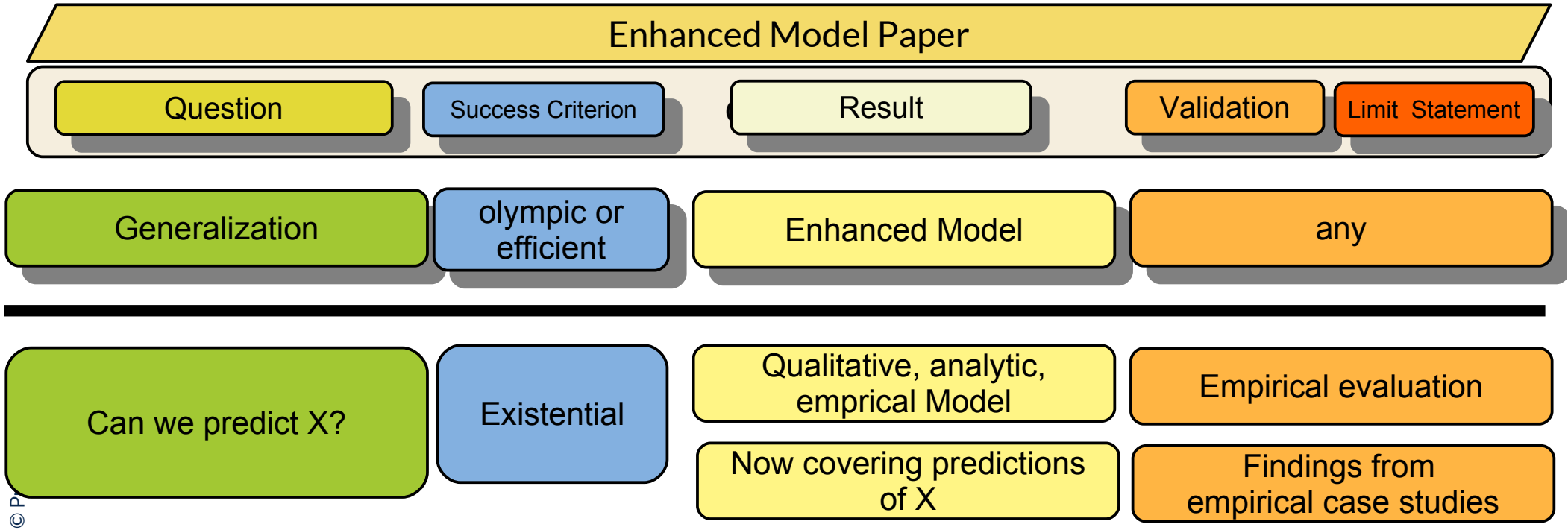
Enhanced Model (EM) (Generalized model)

- Enhanced Model:

***Problem**:* Existing <model-type> models are deficient in dealing with <properties> of <solution strategy>.

***Result and Solution**:* An enhanced <model-type> is described, capable of providing more accurate analyses / predictions of <properties> in <solution strategy> designs.

Validation: The model has been tested by comparing analyses / predictions with empirically measured values of <properties>.



Kiczales, Lamping, et.al. “Open Implementation”: Definition Essay

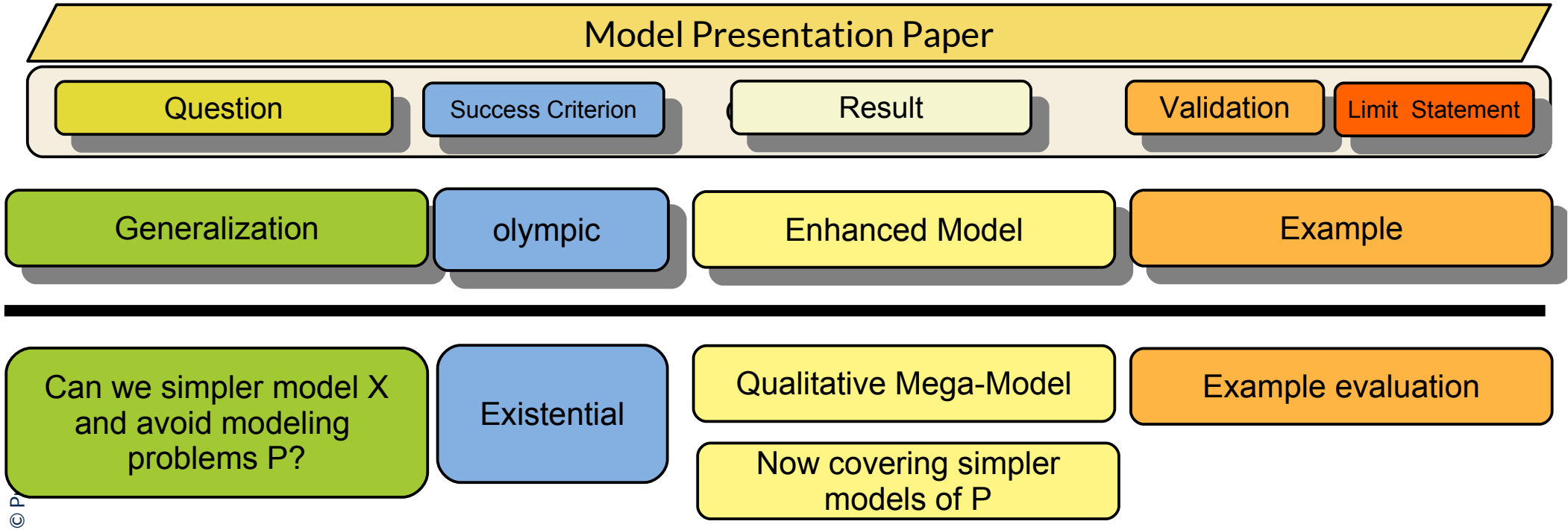
- [OpenImp] Outline:
  - 1. Introduction
  - 2. A Base Case
  - 3. Separation of Use from Implementation Strategy Control
  - 4. Scope Control
    - Choosing the scope control
  - 5. Subject Matter
    - Tradeoffs
  - 6. Style of the ISC code
  - 7. The Design space
- Why does this outline work? problem-solution paper (“enhanced model”):

Abstract: “An examination of existing software systems shows that an increasingly important technique for handling this problem is to design the module’s interface in such a way that the client can assist or participate in the selection of the module’s implementation strategy. We call this approach *open implementation*.

When designing the interface to a module that allows its clients some control over its implementation strategy, it is important to retain, as much as possible, the advantages of traditional closed implementation modules. This paper explores issues in the design of interfaces to open implementation modules. We identify key design choices, and present guidelines for deciding which choices are likely to work best in particular situations.”

Model Presentation Paper

- [Atkinson/Kühne 2003, A Foundation for Metamodeling] presents a 2-dimensional metamodeling scheme for metamodeling.
  - Classification in 2 dimensions, different instance-of-relationships



32.4.3 “New Language” Paper (Enhanced Model)

- A new language may solve some problems easier than another existing one

Simple Language Journal Paper

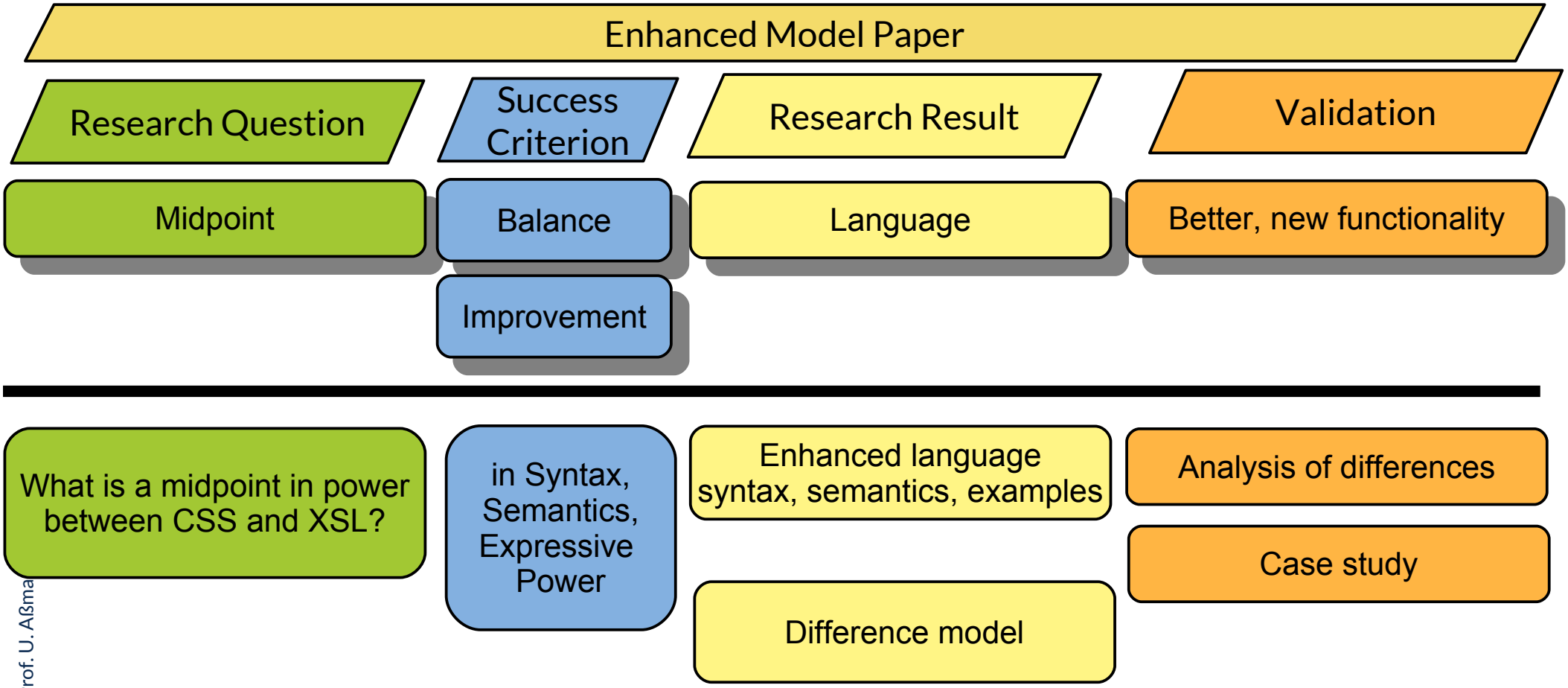
Philip M. Marden, Jr., Ethan V. Munson. PSL: An Alternate Approach to Style Sheet Langauges for the World Wide Web. Journal of Universal Computer Science, vol. 4, no 10(1998), Springer

- Why does this outline work? "enhanced model" paper:

Abstract: "Style sheets, which are used to specifiy the appearance of documents, we rapidly growing in their importance for the World Wide Web. Cascading Style Sheets are now in widespread use and work on a future Web Standard, the Extensible Style Sheet Language (XSL) is proceeding at a rapid pace. In this paper, we show how a different style sheet language, PSL, represents an attractive midpoint between CSS and XSL in complexity and power. PSL is based on general language design principles that give its simple syntax, easily-described semantics, and considerable expressive power. Our testbed MPMosaic uses Proteus, an portable style sheet system, to support PSL."

- Outline:
- 1. Introduction
- 2. CSS, XSL, and DSSL
- 3. The PSL Language
  - 3.1. Properties and Rules
  - 3.2. Tree Elaboration
  - 3.3. Box Layout
  - 3.4. Other Features
  - 3.5 Combining PSL's Services
- 4. Comparing PSL and CSS
  - 4.1. Syntactic Complexity
  - 4.2. Semantic Consistency
  - 4.3. Expressive Power
- 5. Experience with MPMosaic (a PSL-based browser)
  - This shows some functionality, views, which are not available in a classical CSS-based browser

PSL Style Sheet Language



Language Paper

- Maribel Fern§ndez, H®l¯ ne Kirchner, Olivier Namet. A Strategy Language for Graph Rewriting. In G. Vidal (ed.), Logic-Based Program Synthesis and Transformation (LOPSTR). 21st International Symposium, 2011, Springer
- Why does this outline work? "enhanced model" paper:

Abstract: "We give a formal semantics for a graph-based programming language, where a program consists of a collection of graph rewriting rules, a user-defined strategy to control the application of rules, and an initial graph to be rewritten. The traditional operators found in strategy languages for term rewriting have been adapted to deal with the more general setting of graph rewriting, and some new constructs have been included in the language to deal with graph traversal and management of rewriting positions in the graph. This language is part of the graph transformation and visualisation environment PORGY."

- Outline:
  - 1. Introduction
  - 2. Background: Port Graph Rewriting
  - 3. The Strategy Language
    - 3.1. Syntax and Informal Description
    - 3.2. Semantics
  - 4. Examples
  - 5. Properties
    - Proofs about semantic features
  - 6. Implementation
  - 7. Related Work and Conclusion

Language Paper "The TXL Language"

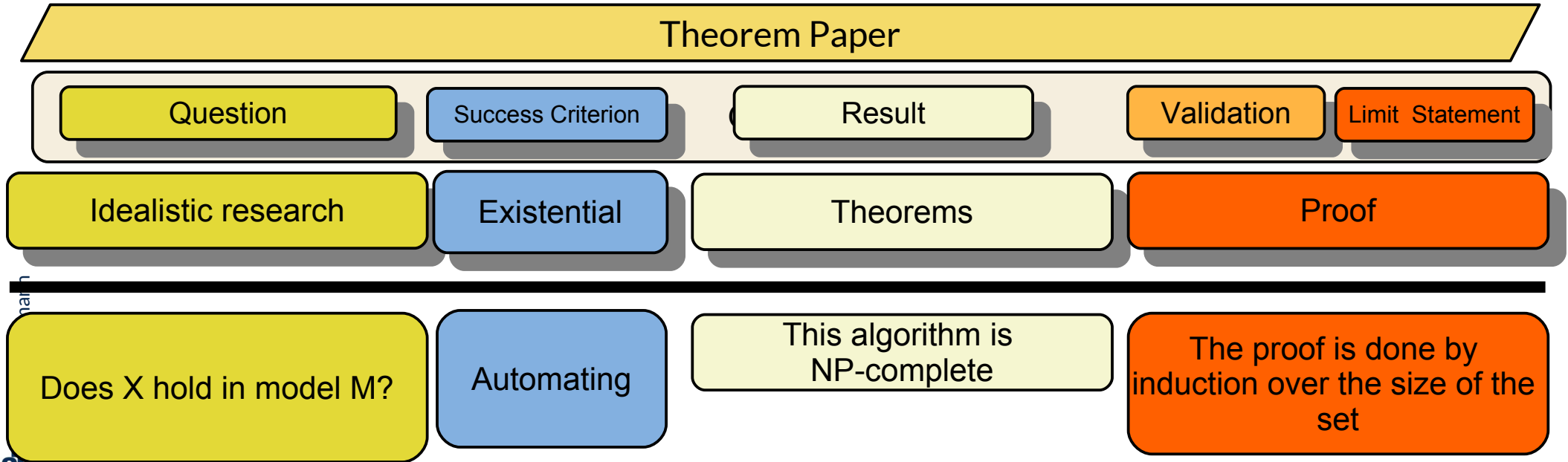
- James R. Cordy. The TXL source transformation language. Sci. Comput. Programming, 61(3), 2006.
- "enhanced tool" paper covering many more nice applications:

Abstract: "TXL is a special-purpose programming language designed for creating, manipulat- ing and rapidly prototyping language descriptions, tools and applications. TXL is designed to allow explicit programmer control over the interpretation, application, order and backtracking of both parsing and rewriting rules. Using first order func- tional programming at the higher level and term rewriting at the lower level, TXL provides for flexible programming of traversals, guards, scope of application and parameterized context. This flexibility has allowed TXL users to express and exper- iment with both new ideas in parsing, such as robust, island and agile parsing, and new paradigms in rewriting, such as XML markup, rewriting strategies and contex- tualized rules, without any change to TXL itself. This paper outlines the history, evolution and concepts of TXL with emphasis on its distinctive style and philoso- phy, and gives examples of its use in expressing and applying recent new paradigms in language processing.

32.4.3 New Knowledge Paper (Enhanced Idealized Model)

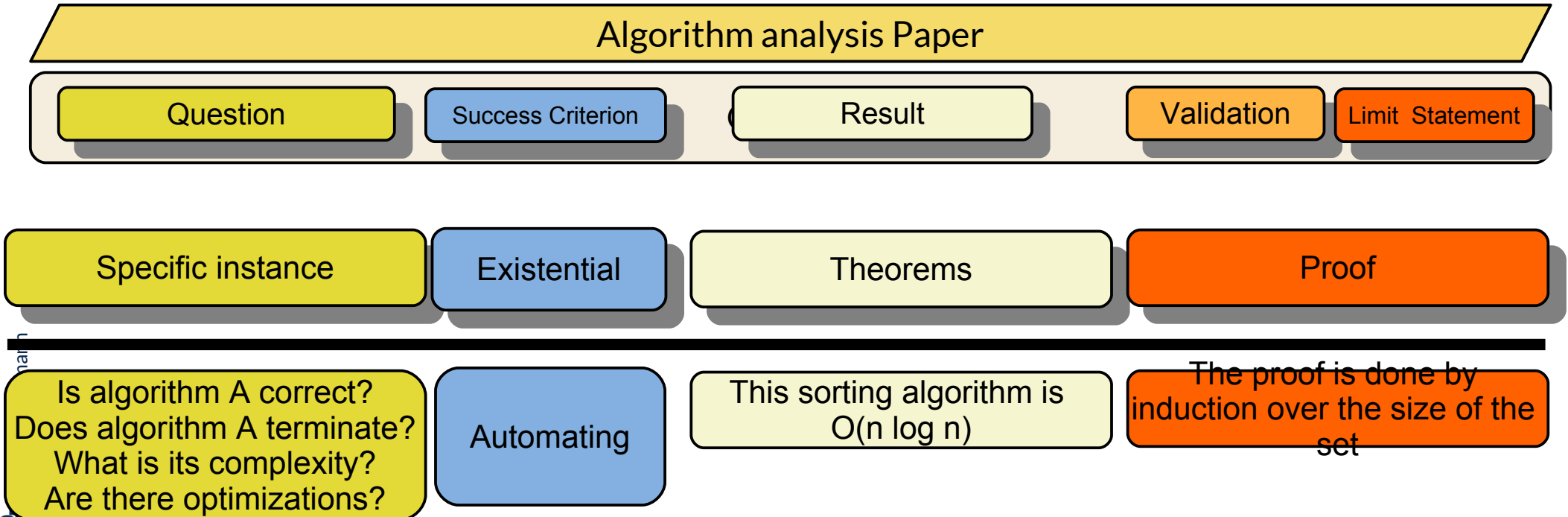
Theorem Paper

- A theorem paper is always working on an idealized research result, based on a model of reality
- LogP Papers of Löwe, Zimmermann, Eisenbiegler discuss the LogP-model of distributing data and computations on distributed machines
  - Much better than the usual PRAM model, because parallel distrubted machine is modeled more realistically (L – latency, o - overhead, g - gap)
- Wolf Zimmermann and Welf Löwe. Foundations for the integration of scheduling techniques into compilers for parallel languages. IJCSE, 1(2/ 3/4):99-109, 2005.



Algorithm Analysis/Design Paper

- Papers presenting a new or optimized algorithm need to discuss:
  - Correctness
  - Termination
  - Complexity on a RAM, PRAM or on a logp-machine
    - NP-completeness, decidability
    - for practical algorithms: linearity, n log n, quadratic, cubic
- Prove quality features, such as memory consumption, energy consumption

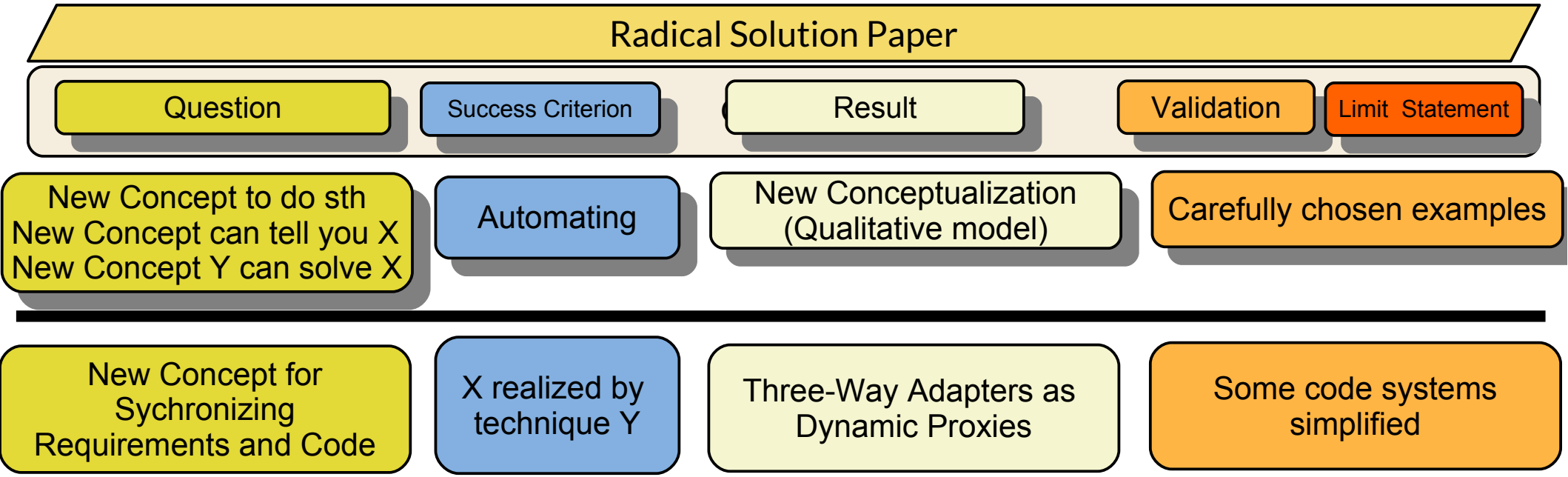


32.4.4. Radical Solution

1)

Groundbreaking Idea Paper (Radical Solution)

- In recent years, these are harder to publish
- Contains basically a *conceptualization* of an unknown field (white space)



- Ex.: Uwe Aßmann. Automatic Roundtrip Engineering. In U. Aßmann, E. Pulvermüller, P. Cointe, N. Bouraquadi, and I. Cointe, editors, Proceedings of Software Composition (SC) - Workshop at ETAPS 2003, volume 82 of Electronic Notes in Theoretical Computer Science (ENTCS), Warshaw, April 2003. Elsevier.
- Defines different classes of round-trip systems, such as “bidirectional weaving systems”, “partitionable round-trip systems”, etc.
- Validation by examples (weak): explains the difference of TeX and Word
- Nevertheless, 30 citations

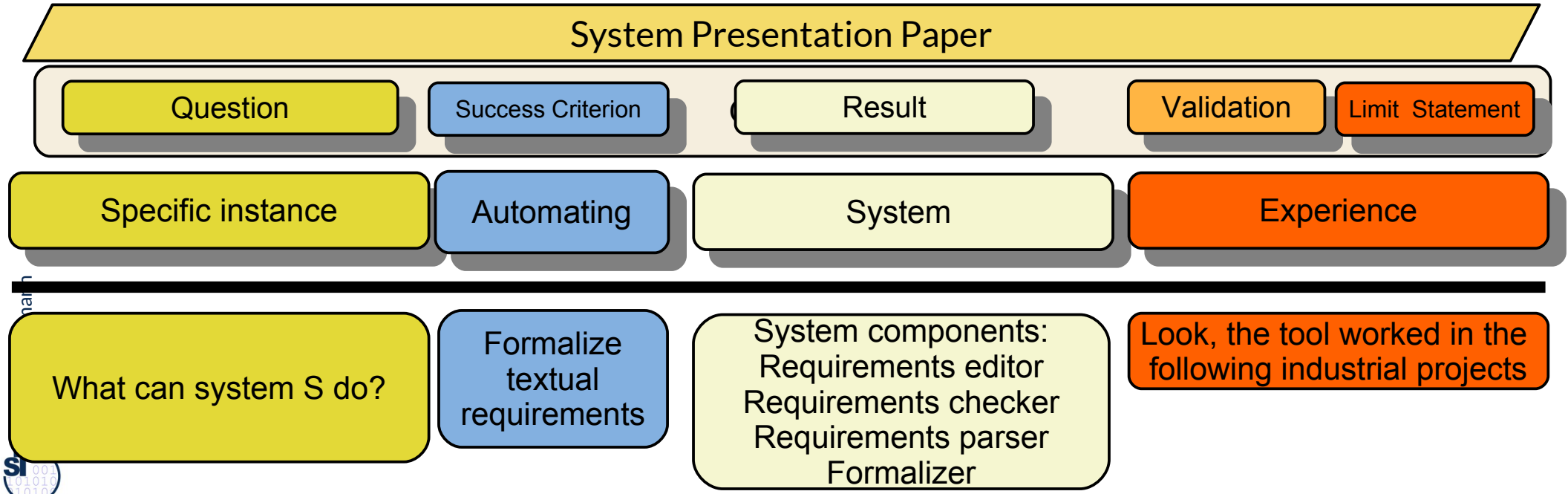
32.4.5. Writing a Systems Paper (Enhanced Tool)

Obligatory Literature

- Roy Levin and David D. Redell. An Evaluation of the Ninth SOSP Submissions or How (and How Not) to Write a Good Systems Paper. ACM SIGOPS Operating Systems Review, Vol. 17, No. 3 (July, 1983), pages 35-40
- http://infolab.stanford.edu/~widom/paper-writing.html.

System and Tool Papers

- System papers need to discuss:
  - Deficiencies or limits of other systems
    - Market data or studies of economical need
  - Success factors and requirements for the system
  - Unique features not available in other systems
    - Components of the system that contribute to the unique features
    - why is automation with a tool important?
  - Important use cases, limits of the system, empirical evaluation
- Tools are special systems which automate things that should otherwise be done by hand
  - Aching factors: what aches if the tool is not available?



R. Vuduc , J.W. Demmel, K.A. Yelick. OSKI: A Library of Automatically Tuned Sparse Matrix Kernels. SciDAC 2005 (Journal of Physics), UCRL-CONF-213753

Outline:

- 1. Goals and Motivation: Interesting, explicit list of motivations
- 2. An Introduction to the Tuning Interface by Example
  - 2.1. Basic usage: globally migrating applications
  - 2.2. Providing explicit tuning hints
  - 2.3. Tuning based on implicit profiling
- 3. Saving and restoring tuning transformations
- 4. Other features
- 5. Related work
- 6. Conclusions and future work

- Why does this outline work? constructive hypothesis (automation hypothesis):

“Abstract. The Optimized Sparse Kernel Interface (OSKI) is a collection of low-level primitives that provide *automatically tuned* computational kernels on sparse matrices, for use by solver libraries and applications. These kernels include sparse matrix-vector multiply and sparse triangular solve, among others. The primary aim of this interface is to hide the complex decision- making process needed to tune the performance of a kernel implementation for a particular user’s sparse matrix and machine, while also exposing the steps and potentially non-trivial costs of tuning at run-time. This paper provides an overview of OSKI, which is based on our research on automatically tuned sparse kernels for modern cache-based superscalar machines.”

Combined Paper on New Method based on New Language and New Tool

- Thomas R. Dean and James R. Cordy and Andrew J. Malton and Kevin A. Schneider. Agile Parsing in TXL. Autom. Softw. Eng. 10 (4), 2003
- Outline:
  - 1. Introduction: background, research question
  - 2. Agile parsing – the concept
  - 3. TXL, the tool for agile parsing (Fig. 2 is a concept map of agile parsing with TXL)
    - 3.1 TXL language: introduction
    - 3.2 TXL Support for Agile Parsing
    - 3.3 An example
  - 4. Agile parsing idioms (patterns)
    - 4.1 Rule Abstraction
    - 4.2 Grammar Specialization
    - 4.3 Grammar Categorization
    - 4.4. Union Grammars for Translation
    - 4.5. Markup
    - 4.6 Semi-parsing
    - 4.7 Data-structure grammars
  - 5. Experience with use cases
  - 6. Related work
- 7. Conclusions

Agile Parsing ctd. - Unification of Technologies

- Why does this outline work?
  - constructive hypothesis (automation hypothesis): Aigile parsing based on TXL features can automate several important use cases others can't automate yet.
- The paper unifies several best practices in grammar engineering by generalization to the new technique of “agile parsing”.

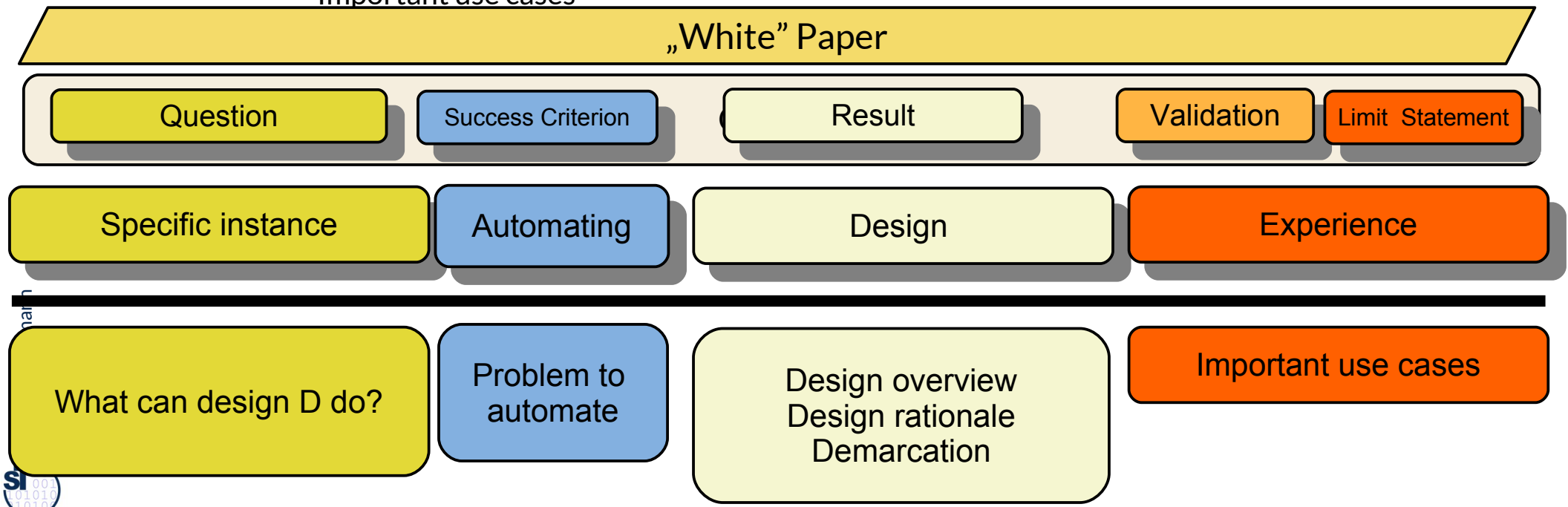
“Abstract. Syntactic analysis forms a foundation of many source analysis and reverse engineering tools. However, a single standard grammar is not always appropriate for all source analysis and manipulation tasks. Small custom modifications to the grammar can make the programs used to implement these tasks simpler, clearer and more efficient. This leads to a new paradigm for programming these tools: agile parsing. In **agile parsing** the effective grammar used by a particular tool is a combination of two parts: the standard base grammar for the input language, and a set of explicit grammar overrides that modify the parse to support the task at hand. This paper introduces the basic techniques of agile parsing in TXL and discusses several industry proven techniques for exploiting agile parsing in software source analysis and transformation.

32.4.5. Experience and Heuristics

1) Writing

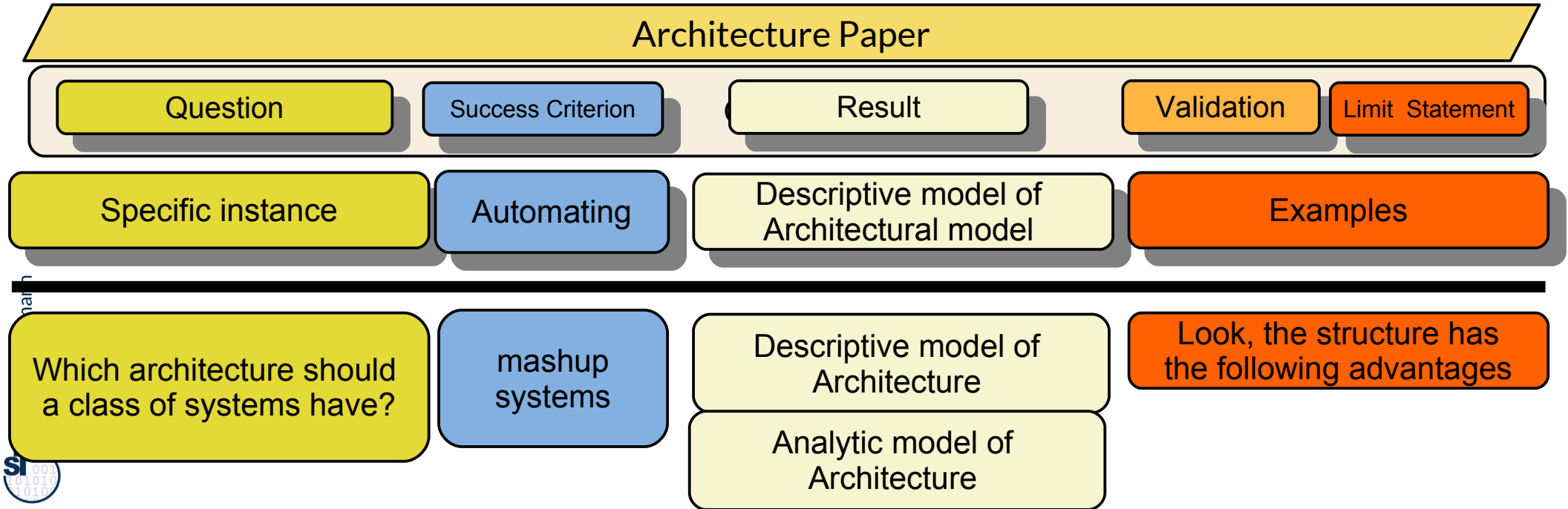
Design Papers ("White Paper", "Red Book")

- [Hermann Kopetz, Astrit Ademaj, Petr Grillinger, Klaus Steinhammer. The Time-Triggered Ethernet (TTE) Design.]
- Design papers describing the design of a new technology can describe:
  - Basic concepts of the domain
  - Success factors and requirements for the design
  - Deficiencies or limits of other designs
  - Overview of the design
  - Design rationale (why was the design chosen like that? Which other solutions were rejected?)
  - Unique features not available in other designs
  - Important use cases



Architecture Papers

- Architecture papers need to discuss
  - Deficiencies or limits of other systems
    - Market data or studies of economical need
  - Success factors and requirements for the system
  - Unique features not available in other systems
    - Components of the system that contribute to the unique features
    - why is automation with a tool important?
  - Important use cases, limits of the system, empirical evaluation



Experiment Papers

- Experimental papers measure with benchmarks olympic or efficiency features of programs, processes, techniques
- Benchmark suites, such as:
- Java Grande Benchmark
- Spec benchmark
- Java Qualitas Corpus
  - Ewan D. Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. The Qualitas Corpus: A curated collection of java code for empirical studies. In Jun Han and Tran Dan Thu, editors, APSEC, pages 336-345. IEEE Computer Society, 2010.
  - Roberto Tonelli, Giulio Concas, Michele Marchesi, and Alessandro Murgia. An analysis of SNA metrics on the Java Qualitas Corpus. In Arun Bahulkar, K. Kesavasamy, T. V. Prabhakar, and Gautam Shroff, editors, ISEC, pages 205-213. ACM, 2011.

Interface Specification Paper

- Formal specification of interfaces (in Object-Z)

Statistics on Types of Papers

- Shaw's findings on papers submitted to ICSE 2002

Question	Result	Validation	Count
Development method	Procedure	Analysis	3
		Experience	4
		Example	7
	Qualitative model	Experience	2
		Persuasion	1
	Analytic model	Experience	3
	Notation/tool	Analysis	1
		Experience	1
		Example	2
Analysis method	Procedure	Analysis	1
		Experience	3
		Example	2
	Analytic model	Analysis	1
		Experience	1
		Example	2
	Tool	Example	1
Evaluation of instance	Specific analysis	Analysis	3
		Example	1
	Answer	Analysis	1

32.5 Different Kind of Research Results

What You Can Expect from a SE Researcher

- Remember the difference of engineers and technical scientists:
  - An engineer works out systems to solve problems
  - a technical scientist works out methods and techiques for engineers
- Papers (examples):
  - Problem papers
  - Literature analysis studies
  - SWOT analyses (strategic analyses)
  - Solution Pattern descriptions/papers
  - HOWTO-Papers (methods, process patterns)
  - Design pattern papers
- Artefacts (demonstrators often in 1st, 2nd and 3rd generation, most often not for industrial use):
  - Code Libraries and Frameworks helping other people doing work
  - Model frameworks
  - Tools for automation, for specific languages
  - Composition systems and reuse langauges
  - Interpreters and compilers for languages
  - Books overviewing a subject area or method

The Daily Schmidt

- Read “Ohne den Euro ist alles nichts” - a plea for the support of Greece and other poor countries
- http://german.irib.ir/component/k2/item/119164-helmut-schmidtwiwo-ohne-den-euro-ist-alles-nichts?tmpl=component&print=1
- What is the hypothesis of the paper?
- How does the paper prove the hypothesis? What are the arguments?
- Why is the paper a bitter pill for Germans, but shows a way to the future?
- What do you think in hindsight about the Greek crisis in 2014, four years later?

The End

- Explain the elements of a research paper.

The End

Mary Shaw: “A research paper is a purposeful, designed artifact, just like a software system. Apply software design techniques to paper design:

- *Start with the requirement:* read the call for papers
- *Select an architecture:* plan the sections, what they say
- *Plan a schedule:* allow time for review, revision
- *Check consistency:* type-check text like code”

The Original Shaw Model of Research in Software Engineering