

Design Rationale

Seminar Software Evolution
SS 2005

Ausarbeitung von
David Hannappel

Universität Duisburg-Essen
Fakultät Ingenieurwissenschaften
Studiengang Angewandte Informatik
Fachbereich Software Engineering

Inhaltsverzeichnis

1. Einführung	3
2. Was ist Design Rationale?	4
3. Motivation	5
3.1 Der typische Software – "Lebenszyklus"	5
3.2 Einsatzgebiete für Design Rationale:.....	5
3.3 Zusammenfassung:.....	7
4. Design-Rationale Typen	7
5. Design Rationale-Struktur	8
6. Methoden zur Erfassung der DR	9
Vergleich der Methoden:.....	10
7. Design Rationale Darstellung	11
8. Notationen	12
8.1 QOC-Notation.....	12
8.2 IBIS-Notation:.....	13
8.3 DRL-Notation	15
9. Design Rationale Studien	17
10. Design Rationale Probleme	18

1. Einführung

Um Design Rationale zu definieren, müssen wir zunächst den Begriff Design erläutern.

Wir verstehen Design als einen Konstruktionsprozess, in dem ein Produkt Form und Gestalt erhält.

Bezogen auf Software-Entwicklung verstehen wir Design als das Entwerfen und Ausgestalten von Software aufgrund der Anforderungsdefinition.

Die Design-Theoretiker Cross (1984), Rittel (1984), Schön (1983), und Simon (1981) beschreiben Design als das kontinuierliche Suchen, Finden und Lösen von Problemen.

Das fertige Design-Produkt (artifact) ist das Ergebnis einer Reihe von Aktivitäten , Diskussionen und Verhandlungen, die im Design-Prozess stattfinden.

In diesen Verhandlungen werden wichtige Entscheidungen getroffen, z.B. warum gewisse Anforderungen existieren, warum eine bestimmte Programmiersprache verwendet werden soll und warum einige Features enthalten sind, aber andere wiederum nicht.

Diese Hintergrundinformationen sind wertvoll, oft sogar notwendig für die Weiterentwicklung, Änderung und Vermarktung der Software.

Nach dem Design-Prozess geraten diese Informationen leicht in Vergessenheit oder sind z.B. nur noch in Form von unsortierten Notizen und Emails gespeichert.

Die Rückgewinnung dieser Informationen (Stichwort Reverse-Engineering) ist unter diesen Umständen äußerst zeit- und kostenintensiv.

Aus diesem Grund ist es sinnvoll, den Design Prozess mit allen aufkommenden Streitfragen, Entscheidungen und Argumenten zu dokumentieren. In diesem Zusammenhang spricht man von Design Rationale.

Quelle: freie Übersetzung und Zusammenfassung nach [3]

2. Was ist Design Rationale?

Es existieren für Design Rationale unterschiedliche Definitionen mit unterschiedlichen Bedeutungen. Eine (Wörterbuch ähnliche) Zusammenfassung der unterschiedlichen Definitionen könnte folgendermaßen aussehen:

design rationale

1. An expression of the relationships between a designed artifact, its purpose, the designer's conceptualization, and the contextual constraints on realizing the purpose
2. The logical reasons given to justify a designed artifact.
3. A notation for the logical reasons for a designed artifact.
4. A method of designing an artifact whereby the reasons for it are made explicit.
5. Documentation of (a) the reasons for the design of an artifact, (b) the stages or steps of the design process, (c) the history of the design and its context
6. An explanation of why a designed artifact (or some feature of an artifact) is the way it is.

Die erste Definition beschreibt Design Rationale als Begründung (wie auch immer diese aussehen mag) für das Design eines Produkts.

Die zweite Definition impliziert, dass Design Rationale einem bestimmten Zweck dient (z.B. der Überzeugung eines Kunden).

Die dritte Definition bezieht sich auf die Notation des Design Rationale.

Die Notation (die formal, informal oder semiformal sein kann) ist eine Darstellungsart, um Design-Entscheidungen und -Begründungen sinnvoll zu verknüpfen.

Die vierte Definition stellt Design Rationale als eine Methode zur Konstruktion eines Produkts dar. Dabei liegt der Schwerpunkt auf einer detaillierten Begründung des Designs.

Die fünfte Definition zeigt verschiedene Aspekte der Design-Dokumentation.

Die Dokumentation kann (a) Gründe für das Produkt-Design, (b) die Einzelschritte des Design-Prozess, oder (c) den zeitlichen Verlauf des Design beinhalten.

Die sechste Definition verdeutlicht, dass Design Rationale das Schlüssel-Element für die Interpretation eines Designs darstellt.

Quelle: freie Übersetzung und Zusammenfassung nach [3]

3. Motivation

3.1 Der typische Software – "Lebenszyklus"

Das **Wasserfallmodell** bezeichnet den traditionellen Typus des Software-prozesses bzw. es ist ein Vorgehensmodell der Softwareentwicklung.

Im Wasserfallmodell hat jede Phase wohldefinierte Start- und Endpunkte mit eindeutig definierten Ergebnissen d.h. Meilensteinsitzungen mit Ergebnisdokumenten und lediglich einem Durchlauf. In der Realität ist dieses Modell selten genau so zu finden.

[Quelle: Wikipedia.de]

Phasen der Entwicklung:

- Anforderungsanalyse und Spezifikation
- Systemdesign und -spezifikation
- Programmierung und Modultests
- Auslieferung, Einsatz
- Wartung (Maintenance)
- Redesign

Jede einzelne Phase ist abhängig von den Outputs und dem **Verständnis der Argumentation** der vorhergehenden Phasen.

In den einzelnen Phasen sind meist unterschiedliche Personen (mit verschiedenen technischen Kompetenzen) an der Entwicklung beteiligt, was eine Kommunikation zwischen den Phasen erschwert.

Jede dieser Phasen kann effizienter durchlaufen werden, wenn die vorhergehenden Phasen mit Hilfe von Design Rationale zu einem besseren Verständnis beitragen.

Quelle: freie Übersetzung und Zusammenfassung nach [3]

3.2 Einsatzgebiete für Design Rationale:

Design Rationale besteht aus vielen unterschiedlichen Informationen wie dem zeitlichen Verlauf des Design Prozess (design history) und den Gründen für einzelne Design-Entscheidungen. Diese Informationen können in verschiedenen Hinsichten nützlich für die Entwicklung der Software sein:

- **Design Verification**
DR wird benutzt, um zu überprüfen, ob man die besten Entscheidungen getroffen hat.

- **Design evaluation**

Ähnlich Verification, beinhaltet jedoch auch eine Bewertung und die Abwägung der einzelnen Design-Entscheidungen.

- **Design Maintenance**

Es wird geschätzt, dass ca. 70% der „Lebenszyklus“-Kosten einer Software durch Pflege, Wartung und Änderungen (Maintenance-Phase) verursacht werden.

Ungefähr die Hälfte dieses Ressourcenverbrauchs lässt sich auf Reverse-Engineering zurückführen.

Das heißt die Hälfte der Maintenance-Kosten entsteht dadurch, dass sich die Entwickler mit der Wiedergewinnung von Informationen, z.B. aus Software-Quellcode, befassen .

Diese Kosten könnten minimiert werden, indem kritische Informationen und Entscheidungen aus dem Design Process in Form von Design Rationale zugänglich gemacht werden.

Der Aufwand für Reverse-Engineering würde damit automatisch geringer werden.

- **Design reuse**

Mit Hilfe von Design Rationale lässt sich leichter Herausfinden, welche Teile des Designs wiederverwendet werden können. Man kann herausfinden, welche Möglichkeiten existieren, um das Design so zu modifizieren, so dass neue Anforderungen erfüllt werden.

Es ist besonders wichtig für den Designer zu verstehen *warum* eine Entscheidung getroffen wurde, damit “kleine Änderungen“ nicht das ganze Design gefährden.

- **Design teaching**

Design Rationale bietet sich als optimale Hilfe zur Schulung von neuem Personal an.

Wie funktioniert was, warum wurden Entscheidungen getroffen? Einige DR-Systeme ermöglichen es dem Benutzer, Fragen zu stellen (Dies ist effizienter, als sich durch lange Dokumentationen durchzulesen)

Besonders hilfreich kann DR sein, wenn die Original-Designer nicht verfügbar sind, um neues Personal zu schulen oder gewisse Design-Fragen zu beantworten.

- **Design communication**

Die Kommunikation wird sowohl während als auch nach dem Design-Prozess aufgewertet, da wichtige Informationen und Hintergründe zum Design für jeden offen liegen.

Arbeiten mehrere Designer an einem Projekt zusammen, können mit Hilfe des DR Streitpunkte und Konflikte schneller gelöst werden.

- **Design assistance**

Unterstützung und Arbeitserleichterung während des Design-Prozesses.

Designer haben Möglichkeit, Entscheidungen direkt zu bewerten und zu überprüfen, dass aus allen Alternativen die Beste gewählt wurde.

- **Design documentation**

Die zeitliche Entwicklung des Designs wird dokumentiert (z.B. anhand von Screenshots oder einer Version-History) mit eigenen Detailstufen für unterschiedliche Zielgruppen (z.B.

Designer, Kunde, Händler)

3.3 Zusammenfassung:

Design Rationale ermöglicht und unterstützt das Verständnis für die Entwicklung der Software. Es identifiziert wichtige Entscheidungspunkte in der Design-Phase, wo andere Entscheidungen zu anderen Ergebnissen geführt hätten und deckt „dead-end solution paths“, also Sackgassen im Entwicklungsprozess auf.

Das durch Design Rationale gespeicherte Wissen erhöht die Entwicklungsfähigkeit und die Wiederverwendbarkeit der Software.

Quelle: http://www.sei.cmu.edu/str/descriptions/featbased_body.html

4. Design-Rationale Typen

Design Rationale kann in verschiedene Typen klassifiziert werden.

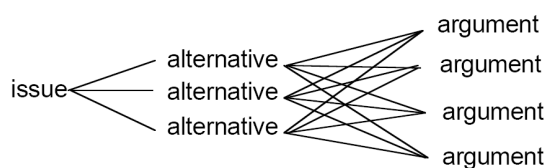
Die folgenden Typen werden hier diskutiert:

- ***Argumentation based***

Bei dieser Darstellungsform steht die Argumentation als Prozess im Vordergrund, die das Design so macht wie es ist.

Diese Argumentation besteht aus den Streitfragen (issues), welche im Entwicklungsprozess aufkommen, den möglichen Lösungen auf diese Fragen (alternatives) und den Argumenten, welche diese Alternativen entweder befürworten oder ablehnen.

Grafisch lässt sich die Argumentation based Design Rationale folgendermaßen darstellen:



- ***History-based***

Die History-Based Design Rationale konzentriert sich im wesentlichen auf die Speicherung der Design history – der Sequenz von Ereignissen, welche die Entwicklung des Designs beeinflusst haben.

Die Informationen können dabei in unterschiedlichen Formen gespeichert werden z.B. in Form von Notizen, einer Sammlung an eMails oder in Form anderer Dokumente, welche die Ereignisse im Entwicklungs-Zeitraum widerspiegeln.

- ***Device-based***

Bei der Device-Based Design Rationale wird das Design durch ein Modell von sich selbst dargestellt. So kann das Verhalten des Designs simuliert werden. Mit Hilfe des Modells kann der Anwender Fragen zum Design und Verhalten stellen.

- ***Process-based***

Die Erfassung der DR-Informationen wird in den Design-Prozess integriert. Änderungen und Verbesserungen der Zielsetzungen spiegeln sich hier in der Design Rationale wieder. Die Erfassung der Rationale-Informationen ist damit ein wichtiger Bestandteil des Entwicklungsprozesses.

- ***Active document-based***

In dieser Form kümmert sich der Designer lediglich um seine eigentliche Arbeit: das Design. Die Design Rationale wird von einer Hilfs-Software automatisch durch ein vorhandenes Wissen erstellt und gespeichert. Bei jeder Aktion des Designers vergleicht das System dann diesen Schritt mit dem vorhandenen Wissen.

Die Design-Rationale-Software ist dabei lernfähig, d.h. wenn das System eine Entscheidung des Designers nicht kennt, dann besteht die Möglichkeit, die Entscheidung zu ändern oder einige Kriterien zu modifizieren.

Quellenangabe: [4]

5. Design Rationale-Struktur

Eine allgemeine Struktur für Design Rationale wurde von Lee (1997) definiert. Diese Struktur besteht aus 3 Ebenen: Design Intent Layer, Design Artifact Layer und Design Intent Layer.

1. Design Intent Layer:

Enthält Absichten, Strategien, Ziele, Anforderungen

2. Decision Layer:

beinhaltet Informationen zu Design-Entscheidungen.

5 Sub-Ebenen:

- issue (Streitpunkte und deren Beziehungen und Abhängigkeiten zum Design)
- alternative
- argument
- evaluation (Alternativen bewerten)
- criteria (Gruppierung von evaluations und arguments)

3. Design Artifact Layer:

beinhaltet Informationen zur Komponente, die designed wird.

Quelle: freie Übersetzung und Zusammenfassung nach [4]

6. Methoden zur Erfassung der DR

Zur Erstellung und Erfassung der Design Rationale können folgende Methoden angewendet werden:

- ***reconstruction*** [Lee, 1997]

Die Erfassung der Design Rationale ist hier nicht in den Entwicklungsprozess integriert. Die Rekonstruktion der Design Rationale findet üblicherweise nach der Entwicklung des Produkts statt. Der Vorteil dieser Methode liegt darin, dass der Entwicklungsprozess nicht durch Dokumentations-Arbeiten ausgebremst wird.

Der Nachteil dabei ist, dass eventuell nicht alle Informationen exakt rekonstruiert und erfasst werden können.

- ***methodological byproduct*** [Lee, 1997]

Bei diesem Vorgehen entwickelt der Designer die Design Rationale während der Entwicklung. Dem Designer stehen dabei Methoden zur Verfügung, die ihm dabei helfen, das Design zu erstellen und gleichzeitig die Rationale zu erfassen.

Die Dokumentation ist dabei im Idealfall „nur“ ein Nebenprodukt der Arbeit des Designers.

- ***apprentice*** [Lee, 1997]

Das System überwacht die Aktionen des Designers und stellt Fragen, wenn es eine Aktion nicht „verstehet“. So erweitert sich das Wissen ständig – das System kann daher mit der Metapher eines Auszubildenden (*apprentice*) beschrieben werden.

Stimmt eine Aktion mit einer bisherigen gespeicherten Aktion überein, so kann das System dann automatisch die Begründung zur Design Rationale hinzufügen.

- ***automatic generation*** [Lee, 1997]

Die Aktionen des Design-Prozesses werden aufgezeichnet und in einer „execution history“ gespeichert. Nach dem Entwicklungsprozess kann aus dieser execution history automatisch die Design Rationale erzeugt werden.

- ***historian*** [Chen, 1990]

Bei dieser Methode zeichnet eine Person oder ein Computer alle Aktionen während des Design-Prozesses auf. Dieses Vorgehen ist der apprentice-Methode ähnlich, allerdings stellt das System hier keine Fragen. Es ist auch vergleichbar mit der automatic generation-Methode, jedoch wird hier die Design Rationale ausdrücklich während – und nicht nach der Entwicklung erstellt.

Vergleich der Methoden:

In den folgenden Tabellen werden die einzelnen Methoden zur Erfassung der DR miteinander verglichen.

Quellenangabe: [4]

Tabelle 1 zeigt, wann die jeweilige Methode angewendet wird (entweder während oder nach dem Design-Prozess)

<i>Capture Method</i>	<i>After Design</i>	<i>During Design</i>
Reconstruction	X	X
Methodological byproduct		X
Apprentice		X
Automatic Generation	X	
Historian		X

Bei einigen Methoden wird hauptsächlich die Design History erfasst, d.h. es werden primär die Ereignisse aus dem Design-Prozess gespeichert. Die Begründungen für diese Ereignisse und damit die Design Rationale spielen bei diesen Methoden oft nur eine untergeordnete bzw. gar keine Rolle.

In der nächsten Tabelle wird verglichen, welche Methoden die Design History - und welche auch die Design Rationale erfassen. Das Fragezeichen bedeutet, dass sich nicht eindeutig festlegen lässt, ob die Methode auch Design Rationale berücksichtigt.

Tabelle 2: Design Rationale vs. Design History

<i>Capture Method</i>	<i>Design History</i>	<i>Design Rationale</i>
Reconstruction	X	?
Methodological Byproduct	X	X
Apprentice	X	X
Automatic Generation	X	
Historian	X	?

Die verschiedenen Methoden lassen sich noch in solche unterteilen, die den Designer besonders fordern (Hohe Interaktion mit Design Rationale) und in solche, bei denen der Designer nur gering mit dem Design Rationale-System interagiert.

Die Methoden mit hoher Interaktion lenken den Designer von der Arbeit mehr ab als ein Vorgehen mit niedriger Interaktion. Die hohe Interaktion ist dann sinnvoll, wenn die Design Rationale einen deutlichen Nutzen für den Design Prozess selbst bringt.

Tabelle 3: Interaktion mit dem Designer

<i>Capture Method</i>	<i>Low Interaction</i>	<i>High Interaction</i>
Reconstruction	X	
Methodological Byproduct		X
Apprentice		X
Automatic Generation	X	
Historian	X	

7. Design Rationale Darstellung

Für Design Rationale ist die Erfassung, Speicherung und Darstellung von Informationen von hoher Relevanz. Es stellt sich die Frage, wie man diese Informationen am besten darstellen kann, um einen möglichst hohen Nutzen zu erhalten.

Design Rationale Darstellungen reichen von „informal representations“ z.B. Audio oder Video Aufzeichnungen bis hin zu formalen Darstellungsarten.

Eine Formale Darstellung eignet sich für Computer, um die Daten zu verarbeiten. Doch die formalen Daten werden nicht immer so dargestellt, dass sie auch für Menschen leicht zu verstehen sind.

Eine „informal“-Darstellung ist hingegen leicht verständlich für den Menschen, jedoch nur

schwer lesbar für Computersysteme (wie z.B. natürliche Sprache).

Eine Semi-formale Darstellung nutzt die Vorteile beider Bereiche und ist sowohl für Computer als auch den Menschen leicht zu verstehen.

Quellenangabe: [6]

8. Notationen

Im folgenden werden die 3 am meisten verbreiteten semiformalen Notationen vorgestellt: QOC, IBIS und DRL.

8.1 QOC-Notation

QOC (Questions, Options and Criteria) basiert auf der **Design Space Analysis**.

Bei der Design Space Analysis geht es hauptsächlich um das Sammeln von Informationen und Organisation sowie Strukturierung dieser Informationen um eine spätere Wiederverwendung zu gewährleisten.

Design Space Analysis hilft Software Designern (sowohl individuell als auch Gruppen) eine Repräsentation zu schaffen, die Anderen ein Verständnis ermöglicht, warum das Ergebnis genau *so* ist, *wie* es ist.

Quellenangabe: Freie Übersetzung und Zusammenfassung aus [5] und [2]

QOC, die Notation der Design Space Analysis ist eine relativ simple „argumentation based“-Notation.

Das QOC-Vokabular besteht aus

1. **Questions:** Design-Fragen, die es zu lösen gilt.
2. **Options:** Alternativen, Möglichkeiten die diese Fragen beantworten könnten.
3. **Criterion:** Argumente und Kriterien, die für oder gegen die Möglichkeiten sprechen.

Die folgende Grafik zeigt, wie Questions, Options & Criteria in dieser Notation verknüpft werden (nächste Seite).

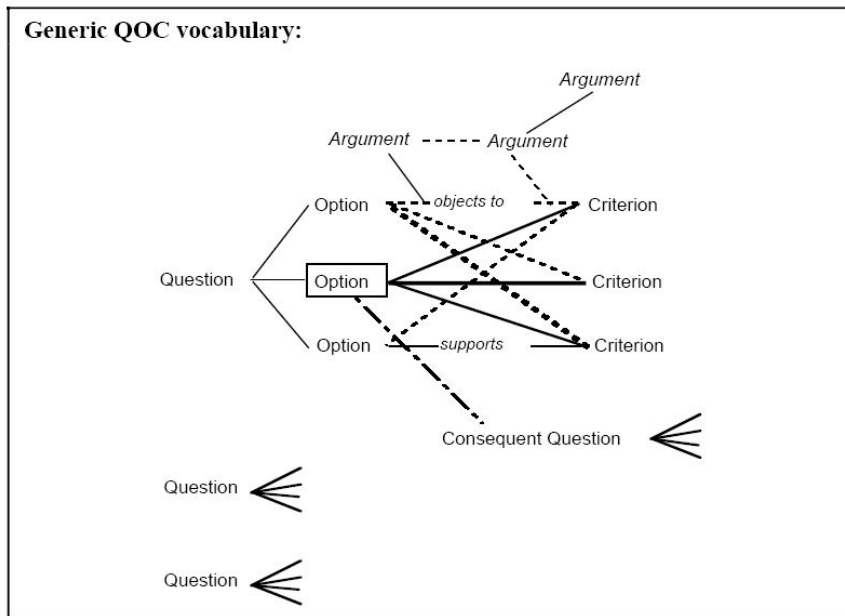
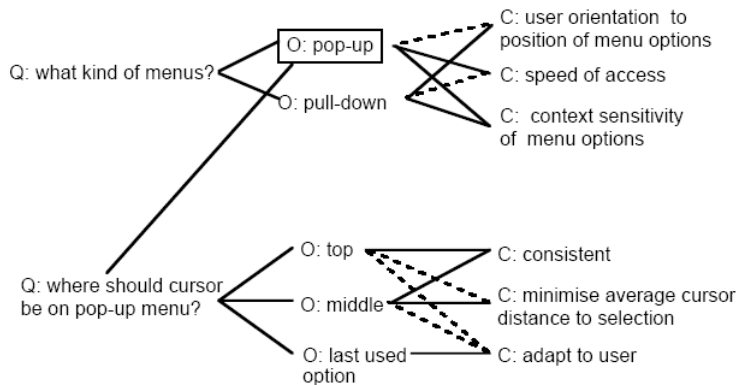


FIGURE 3a: The generic QOC notation as used in Design Space Analysis. Note the use of link-thickness to indicate relative weights of Assessment.

Quellenangabe: [1]

QOC Beispiel:

User interface QOC:



Quellenangabe: [1]

8.2 IBIS-Notation:

Das Issue-Based Information System wurde entwickelt von Horst Rittel.

Issues: Streitfragen, die das Design oder ein Argument aufwirft.

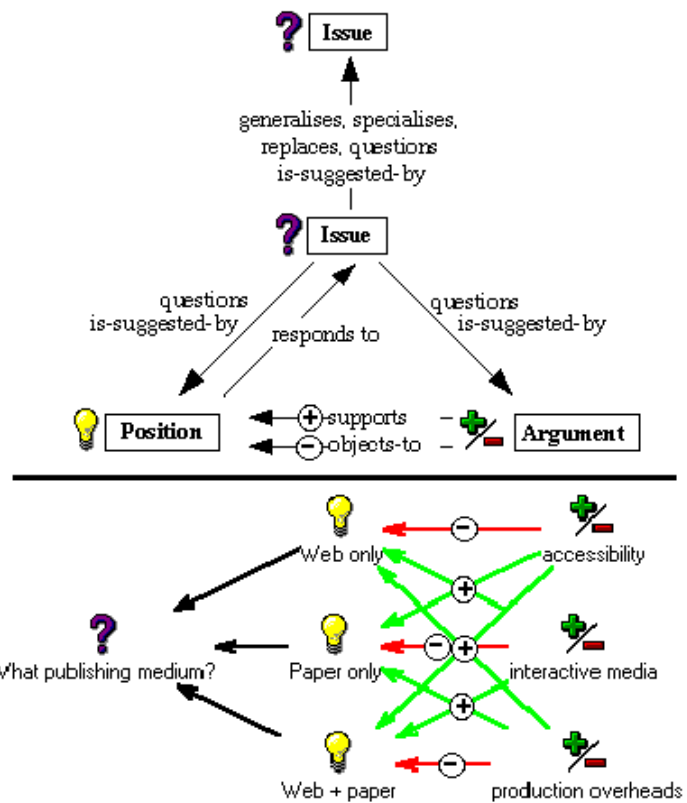
Positions: mögliche Lösungen auf eine Streitfrage.

Arguments: dienen der Unterstützung oder Ablehnung von Positionen.

Die IBIS Notation ist der QOC- Notation relativ ähnlich.

IBIS legt den Schwerpunkt deutlicher auf den Design Prozess (wie er sich z.B. in Meetings abspielt). QOC hingegen arbeitet dagegen eher rückblickend mit dem Design Prozess.

IBIS wurde zu einem Grafischen Tool mit der Bezeichnung gIBIS erweitert.



Ein kommerzielle Tool zur Verknüpfung der Issues, Positionen und Arguments ist „QuestMap“.

Die folgende Abbildung zeigt einen Screenshot von QuestMap:

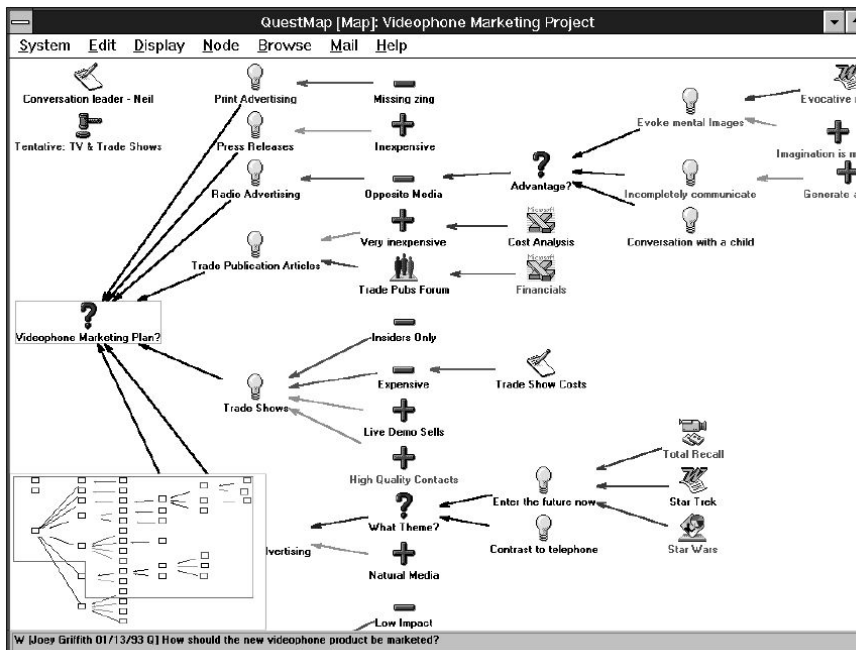


FIGURE 4: A screen from the QuestMap argumentation-based design rationale system [35]. Based on the gIBIS research tool [26], this is a commercial groupware hypertext system, providing design projects with a way to conduct extended, public debates about issues, make decisions, and capture the rationale behind them.

Quelle: [1]

8.3 DRL-Notation

Die Decision Representation Language (DRL) erweitert die Notationen QOC und gIBIS.

„Decision Representation Language (DRL) was developed to explore the potential for computational support within the semiformal paradigm“.

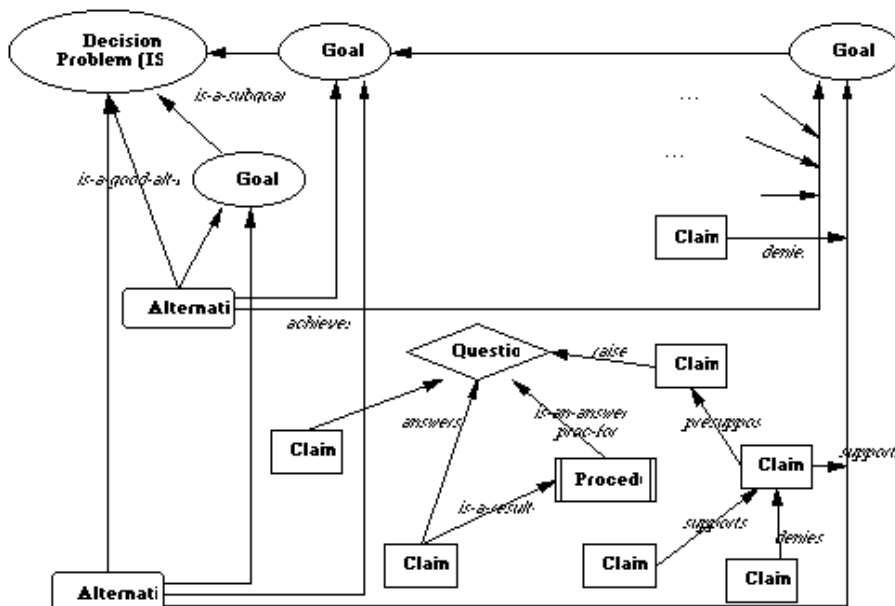
Quelle: [1]

DRL ist die wahrscheinlich vollständigste Design Rationale Notation, aber auch die komplizierteste.

Das Basisvokabular besteht, ähnlich wie bei den vorigen Notationen aus:

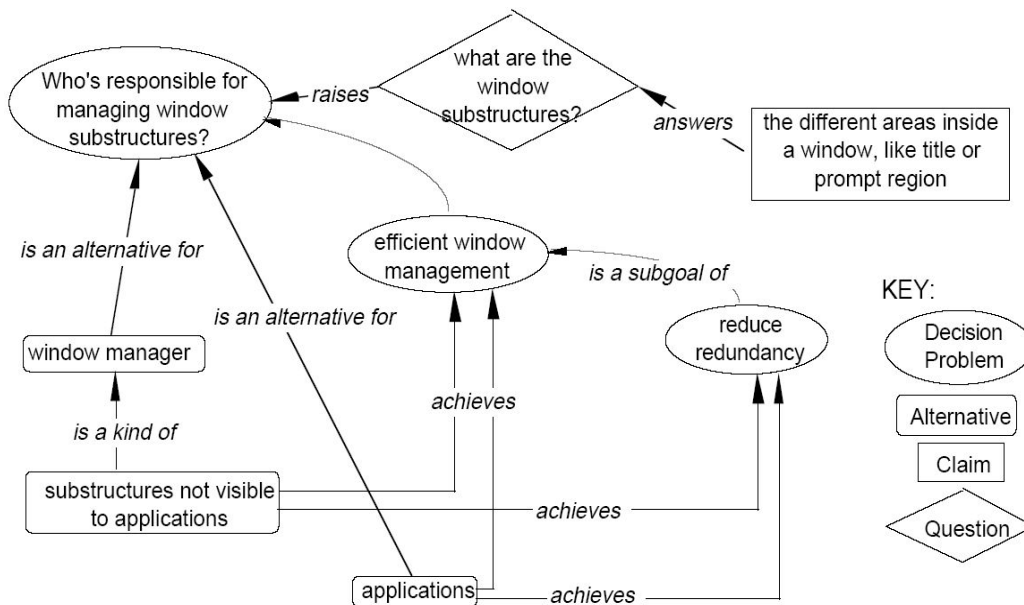
- **Alternatives**
- **Goals**
- **Claims**

Die folgende Abbildung veranschaulicht das Vokabular und die Verknüpfungen der Decision Representation Language:



Quelle: [1]

Ein Beispiel für ein Entscheidungsproblem in DRL könnte folgendermaßen aussehen:



Quelle: [1]

Mit **SYBIL**, einem kommerziellen Tool der DRL Notation, lassen sich Abhängigkeiten zwischen Entscheidungen darstellen.

SYBIL bietet folgende Features:

- **dependency management:**
Entscheidungen und deren Abhängigkeiten verwalten
- **precedence management:**
Andere Entscheidungen auflisten, welche vergleichbare Ziele erreichen.
- **viewpoint management:**
Argumente verwalten, welche vergleichbare Annahmen tätigen.
- **plausibility management:**
Glaubwürdigkeit der Begründung einer Alternative unterstreichen.

Sybil-Screenshot:

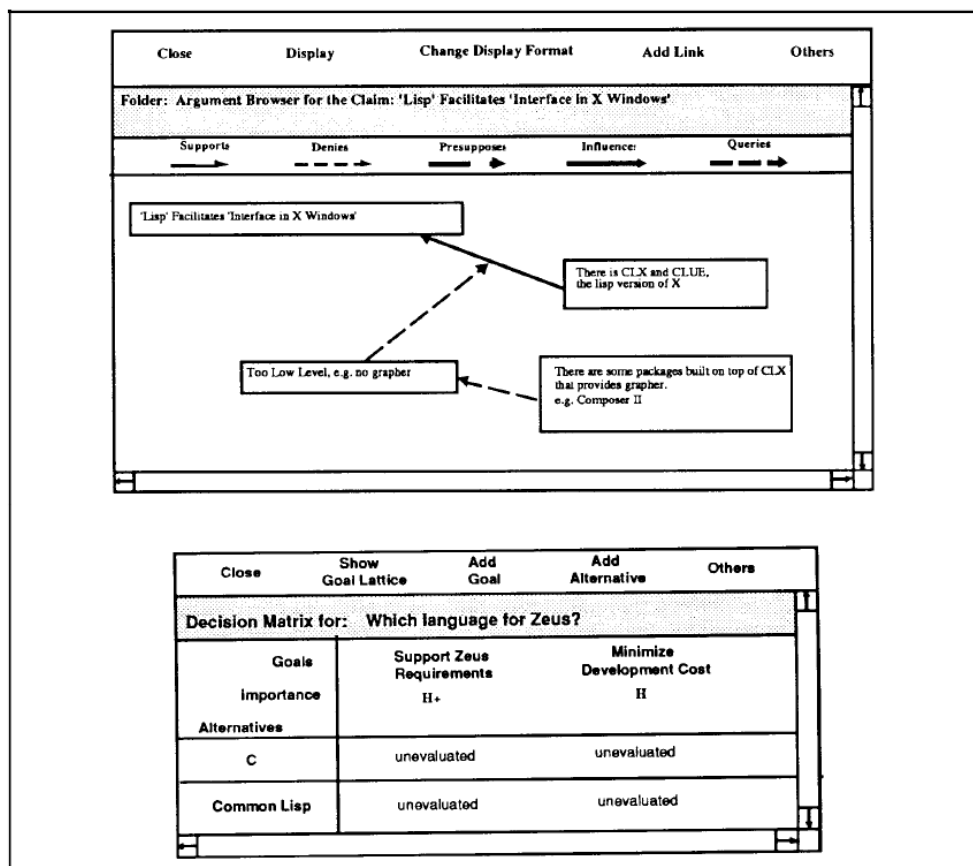


FIGURE 5b: Two views of a DRL network (an Argument browser and a Decision matrix) using SIBYL, its support tool (from [37]).

Quelle: [1]

9. Design Rationale Studien

Es gibt bislang nur wenige Studien, die Design Rationale auf Anwendbarkeit und Nutzbarkeit überprüfen.

Im folgenden werden 2 Feldversuche erläutert, in denen Design Rationale auf Nutzen und Erfolg geprüft wurde.

In einem Experiment des NCR (NCR, Conklin and Burgess-Yakemovic, 1995) wurde die Software-Entwicklung eines kommerziellen Produkts mit Unterstützung durch IBIS / gIBIS beobachtet und dokumentiert.

Es wurde festgestellt, dass IBIS sowohl bei der Anforderungsanalyse als auch im Design-Prozess eine Hilfe war, um Probleme zu lösen und zwischen Entscheidungen abzuwägen.

Fehler, die normalerweise erst später aufgefallen wären, wurden schneller entdeckt und konnten mit Design Rationale besser gelöst werden.

Bei den Meetings wurde eine verbesserte Team-Kommunikation beobachtet – die Meetings waren produktiver als sonst.

Es wurde allerdings auch bemängelt, dass sich die Design-Rationale-Dokumentation nicht dynamisch genug auf aktuelle Probleme anwenden ließe.

„The DR documentation must become 'living'...“.

Quellenangabe: freie Übersetzung und Zusammenfassung nach [7]

In einem weiteren Experiment (Karsenty, 1996) wurde ein Meeting mit Unterstützung einer bereits vorhandenen QOC-Dokumentation durchgeführt.

50% der aufkommenden Fragen bezogen sich auf Gründe für / gegen Design-Entscheidungen. 41% dieser Fragen konnten mit der vorhandenen QOC- Design Rationale sofort beantwortet werden.

Der Hauptgrund, warum bisher nur so wenige Studien zum Thema Design Rationale existieren, ist vermutlich, dass sich Design Rationale in den letzten Jahren nicht durchgesetzt hat.

Design Rationale bietet nicht nur Vorzüge sondern hat auch einige Schwächen und Probleme, die im folgenden Abschnitt diskutiert werden.

10. Design Rationale Probleme

Das sammeln und speichern aller Informationen für Design Rationale ist eine Investition in den frühen Entwicklungsprozess („Upstream“). Design Rationale verspricht große Ersparnisse im „downstream“-Bereich z.B. in der Wartung und Pflege des Produkts.

Dabei wird aber oft nicht berücksichtigt, dass viele Projekte keine „downstream“-Aktivitäten besitzen – weil sie nicht fertiggestellt sind. Selbst wenn ein Projekt fertig ist, sind oft andere Unternehmen für die Wartung und Pflege zuständig. Damit lohnt sich die „Upstream“-Investition nicht für die Entwickler.

Es besteht zudem das Risiko, zu viele Ressourcen für die Dokumentation als für die Programmierung zu investieren, was ein Scheitern des Projekts zur Folge haben kann.

Quelle: freie Übersetzung und Zusammenfassung nach [8]

Es ist relativ mühevoll, Design Rationale „up to date“ zu halten, denn getroffene Entscheidungen können sich später oft als falsch, irreführend oder unwichtig herausstellen.

Fast alle Software-Projekte stehen unter enormen Zeitdruck. Doch Design Rationale ist relativ zeitaufwendig und verlangsamt den frühen Entwicklungsprozess.

Quellenangaben:

1. Simon Buckingham Shum 1996. Design Argumentation as Design Rationale. In The Encyclopaedia of Computer Science and Technology, Marcel Dekker Inc: NY, Vol 35 Supp. 20, 95-128.
2. www.usabilityfirst.com Glossary
3. „Overview of Design Rationale“ Thomas P. Moran and John M. Carroll (erschieden in dem Buch „Design Rationale: Concepts, Techniques, and Use“ (Thomas P. Moran, John M. Carroll), 1996)
4. Design Rationale Types and Tools (Janet E. Burge & David C. Brown), 1998
<http://www.cs.wpi.edu/Research/aidg/DR-Rpt98.html>
5. Bellotti, V. & MacLean, A. Design Space Analysis (DSA). Viewed at: <http://www.mrc-cbu.cam.ac.uk/amodeus/summaries/DSAsummary.html>
6. J. BURGE, D. C. BROWN – Reasoning With Design Rationale
7. E. Jeffrey Conklin - A Process Oriented Approach to Design Rationale (erschieden in dem Buch „Design Rationale: Concepts, Techniques, and Use“ (Thomas P. Moran, John M. Carroll), 1996)
8. Jonathan Grudin - Evaluating Opportunities for Design Capture (erschieden in dem Buch „Design Rationale: Concepts, Techniques, and Use“ (Thomas P. Moran, John M. Carroll), 1996)