일관성 있는 사용자 인터랙션 설계를 위한 방법론 개발

A Methodology for Consistent Design of User Interaction

김동산, Dong San Kim*, 윤완철, Wan Chul Yoon**

요약

지난 10여 년 동안 모바일폰과 같은 인터랙티브 기기들은 새로운 버전이 나올 때마다 새로운 기능들이 덧붙여지면서 점점 더 사용하기 복잡해졌다. 다기능 복합기기의 사용 복잡도를 줄이기 위한 한 가지 방법은 그것을 일관성 있게 설계하는 것이다. 일관성을 정의하기가 쉽지 않고 때로는 일관성 있게 디자인한 것이 오히려 사용성을 낮추는 경우도 존재하지만, 일반적으로 일관성 있게 설계된 시스템은 그 사용법을 더 배우기 쉽고, 기억하기 쉬우며, 에러가 덜 발생한다. 그러나 실제적으로 다기능 복합기기의 사용자 인터랙션 또는 인터페이스를 일관성 있게 설계하는 일은 쉽지 않다. 다기능 기기의 인터랙션 설계는 매우 많은 수의 설계 변수들과 그것들 간의 관계를 고려해야 하므로 많은 시간이 소요되고 디자인 에러가 발생하기 쉽다. 따라서, 이런 복잡한 설계 과정을 지원할 수 있는 방법론의 개발이 필요하다. 본 연구에서는 CUID (Consistent Design of User Interaction)라고 명명한 효과적이고 효율적인 인터랙션 설계 방법론을 개발하였다. CUID는 일관성 있는 인터랙션 설계를 위한 디자인 프로세스와 그것을 지원하는 소프트웨어 툴을 포함한다. 이 방법론은 물리적 또는 시각적인 일관성보다는 논리적 일관성에 초점을 맞추고 있으며, 태스크 절차의 설계, 각 시스템 상태에서 가능해야 하는 조작(기능)에 대한 결정, 그리고 가능한 조작(기능)들을 어떤 인터페이스 컨트롤(control)을 통해 수행하도록 할 것인지에 대한 결정을 주요한 설계 문제로 다룬다. 또한, 본 논문에서는 사례 연구를 통해 CUID의 효용성을 검증하였다.

Abstract

Over the last decade, interactive devices such as mobile phones have become complicated drastically mainly because of feature creep, the tendency for the number of features in a product to rise with each release of the product. One of the ways to reduce the complexity of a multi-functional device is to design it consistently. Although the definition of consistency is elusive and it is sometimes beneficial to be inconsistent, in general, consistently designed systems are easier to learn, easier to remember, and causing less errors. In practice, however, it is often not easy to design the user interaction or interface of a multi-functional device consistently. Since the interaction design of a multi-functional device should deal with a large number of design variables and relations among them, solving this problem might be very time-consuming and error-prone. Therefore, there is a strong need for a well-developed methodology that supports the complex design process. This study has developed an effective and efficient methodology, called CUID (Consistent Design of User Interaction), which focuses on logical consistency rather than physical or visual consistency. CUID deals with three main problems in interaction design: procedure design for each task, decisions of available operations(or functions) for each system state, and the mapping of available operations(functions) and interface controls. It includes a process for interaction design and a software tool for supporting the process. This paper also demonstrates how CUID supports the consistent design of user interaction by presenting a case study. It shows that the logical inconsistencies of a multi-functional device can be resolved by using the CUID methodology.

핵심어: user interaction design, interaction consistency, constraint-based design, design complexity, design support

*주저자 : KAIST 산업및시스템공학과 박사과정 e-mail: kimdongsan@gmail.com

**교신저자 : KAIST 지적서비스공학과, 산업및시스템공학과 교수; e-mail: wcyoon@kaist.ac.kr

1. 서론

지난 10여 년 동안 모바일폰, mp3p와 같은 인터랙티브(interactive) 전자기기들은 이전보다 사용성(usability) 문제에 시간과 노력을 더 많이 투자했음에도 불구하고 점점 더 복잡해졌다. 이러한 현상의 주된 원인 중 하나는 "feature creep", 즉 하나의 기기로 수행할 수 있는 기능의 수가 점점 증가하는 경향 때문이다.[1-4] 사용 복잡성 등 feature creep이 가져오는 여러 가지 부정적인 영향들 때문에 일각에서는 불필요한 기능을 제거함으로써 제품의 단순성(simplicity)을 높이는 것을 강조하고 있지만[5-6], 현실에서는 제품이 제공하는 기능을 줄이는 일이 쉽지 않은 경우가 많다.[2] 이것은 소비자들이 일반적으로 구매 시점에서 많은 기능을 가진 제품을 선호하는 경향을 가진 것과 관련이 깊다.[7] 제품이 가진 기능의 수를 줄이지 않으면서도 사용의 복잡도를 줄일 수 있는 방법 중 하나가 제품을 일관성 있게 설계하는 것이다. 일반적으로 일관성 있게 설계된 시스템은 그 사용법을 더 배우기 쉽고, 기억하기 쉬우며, 사용자 에러가 덜 발생한다. 그러나 실제적으로 다기능 복합기기의 사용자 인터랙션 또는 인터페이스를 일관성 있게 설계하는 일은 쉽지 않다. 매우 많은 수의 설계 변수(design variable)들과 그것들 간의 관계를 고려해야 하기 때문에 많은 시간이 소요되고 디자인 에러가 발생하기 쉽다. 따라서, 본 연구에서는 효과적이고 효율적으로 일관성 있는 인터랙션을 설계할 수 있는 방법론인 CUID(Consistent Design of User Interaction)을 개발하였다. CUID는 일관성 있는 인터랙션 설계를 위한 디자인 프로세스와 그것을 지원하는 소프트웨어 툴을 포함한다.

2. 관련 연구: UI 디자인에서의 일관성

UI 디자인에 있어서 일관성(consistency)은 가장 많이 언급되는 원리 중 하나이다.[12-18] 잘 알려진 Shneiderman의 UI 디자인을 위한 8가지 Golden Rules 중 첫 번째가 바로 "일관성을 추구하라(Strive for consistency)"는 것이다.[14] 일관성이 무엇인지에 대한 의견이 분분하지만[8], 일반적으로 "비슷한 것은 비슷한 방식으로(do similar things in similar ways)" 하자는 원리로 정의된다.[11] 사용자가 시스템을 사용하면서 놀라는 일이 가능한 적게 만들어야 한다는 면에서 "the principle of least-astonishment"로 불리기도 한다.[13] 기존 연구들에 의하면, 일관성 있게 설계하는 것이 오히려 사용성 문제를 일으키는 경우도 있지만[8-10], 일반적으로 일관성 있게 설계된 사용자 인터페이스가 그렇지 않은 인터페이스에 비해 그 사용법을 배우기 쉽고 기억하기 쉽고, 태스크 수행 시간과 사용자 에러의 발생 빈도를 줄여 주며, 사용자의 주관적인 만족도도 높이는 것으로 밝혀졌다.[19-26] 그러나, 대부분의 연구들이 일관성이 중요하다는 것을 실험적으로 검증하거나 일관성 있는 UI 디자인을 위한 가이드라인을 제시하는 데에 그치고 있다. 디자인 가이드라인만으로는 복잡한 기기를 일관성 있게 설계하는 것이 쉽지 않다.[12] 80년대부터 TAG[27], APT[28], PROCOPE[29] 등 설계된 UI의 일관성을 평가하기 위한 정형적 모형(formal model)들이 많이 개발되었지만, 이러한 모형들은 배우기가 쉽지 않고, 개발 기간이 짧은 실제 업무 환경에서 활용하기가 어려운 것이 현실이다. 90년대에는 SHERLOCK[30], GLEAN[31]과 같이 UI의 일관성을 평가해주는 소프트웨어 툴들이 일부 개발되었다. 그러나 SHERLOCK은 색깔, 폰트, 크기, 위치, 레이블, 스타일 등 UI의 물리적 또는 시각적 일관성에만 초점을 맞추고 있고, GLEAN은 유사한 태스크 간 절차의 일관성을 체크하지만 유사한 태스크를 찾는 방법과 절차의 일관성을 체크하는 방법 모두 한계점을 갖고 있다. 최근에는 SUPPLE[32], UNIFORM[33] 등 일관성 있는 UI를 자동으로 생성해주는 툴들이 개발되었지만, 대부분 물리적인 인터페이스의 설계에 초점을 맞추고 있어 인터랙션의 논리적(logical) 부분에 대한 고려가 미흡하다. 사용자 인터랙션의 논리적인 일관성에 대해서는 아직 충분한 연구가 이루어지지 않았다.

3. CUID 접근방법 (approach)

CUID(Consistent Design of User Interaction)는 그림 1과 같이 3가지 인터랙션 디자인 문제를 다룬다.

1) 태스크 절차의 설계 (task-procedure design)
2) 시스템 상태에서 가능해야 하는 조작*(기능)의 설계 (state-operation design)
3) 가능한 조작(기능)들과 인터페이스 컨트롤 간의 매핑 (operation-control mapping)

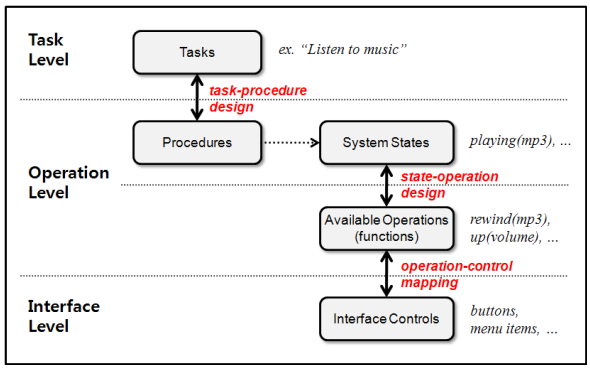

그림 1. CUID에서 다루는 3가지 디자인 문제

각각의 디자인 문제를 해결하는 데 있어 '일관성'을 핵심

* 여기서 '조작(operation)'은 특정한 버튼을 누르는 것과 같은 물리적인 행위가 아니라 구체적인 수행 방법이 결정되지 않은 논리적 행위를 의미한다. 시스템이 제공하는 '기능(function)'이라고도 볼 수 있다. (예. '음악 재생', '볼륨 조절', '슬라이드쇼 실행' 등)

적인 원리로 사용하는데, 구체적으로 "유사한 태스크끼리는 그 수행 절차가 비슷하도록 설계하자", "유사한 시스템 상태에서는 가능한 조작(기능)이 비슷하도록 설계하자", "유사한 조작(기능)은 같은 또는 유사한 인터페이스 컨트롤(버튼 등)로 수행하도록 설계하자"는 원리이다. 여기서 태스크-절차, 상태-조작, 조작-컨트롤은 차례대로 Why-What, Where-What, What-How의 관계에 있다. 이 3가지 디자인 문제들은 UI의 물리적 또는 시각적인 일관성보다는 논리적인 일관성에 관한 것이다. 이 중에서 태스크 절차 간의 일관성 문제는 HCI 분야에서 비교적 많이 다뤄졌지만[27,34], 나머지 2개의 일관성 문제들을 다룬 연구는 아직 드물다. 이들 문제는 기능의 수가 적은 비교적 단순한 시스템에서는 중요한 문제가 아닐 수 있지만, 기능의 수가 많은 복합기기 또는 컨버전스 기기들에서는 매우 중요한 문제가 된다.

일관성 있는 인터랙션 설계를 위해 CUID에서는 두 가지 접근방법, 제약기반 설계(constraint-based design)와 하이웨이 기반 설계(highway-based design) 방법을 사용하였다.

3.1 제약기반 설계 (constraint-based design)

제약기반 설계는 공학 설계(engineering design) 분야에서 효과적으로 활용되는 디자인 방법이다.[35] 여기서 설계 문제(design problem)는 일련의 제약조건들을 만족시키면서 주어진 설계 변수들의 값을 결정하는 문제로 간주되며, 제약조건은 특정 설계변수가 가질 수 있는 값에 대한 제한 또는 설계변수 간의 관계로 정의된다. 제약조건들과 그것들에 의해 연결된 설계 변수들이 모여 제약 네트워크(constraint network)를 형성하는데, 특정 설계변수에 값이 할당되면 이 디자인 결정(design decision)이 제약 네트워크를 통해 전파(propagation)되어 연결된 설계변수들의 값을 결정하는 데에 영향을 미친다.[36] 제약기반 설계 방법에서는 제약조건을 디자인을 제한하는 부정적인 요소로 인식하기보다 디자인 탐색 과정을 이끄는 설계 동인(design driver)으로 사용한다.[37] UI 디자인 또한 제약기반 설계로 볼 수 있지만, HCI 분야에서는 그 동안 제약기반 설계를 효과적으로 활용한 연구[32,38]가 많지 않았다. CUID가 제약기반 설계 방법을 취한 것은 UI 설계자의 인지적 과정이 이러한 특성을 갖고 있기 때문이다.[39] 그림 2는 이러한 제약기반 설계 방법을 도식화한 것이다. 각 계층(abstraction level) 별로 설계 변수(task, state, operation, control)들이 관계(제약조건)에 따라 네트워크를 형성하고 있는데, CUID에서 다루는 설계변수 간의 관계는 크게 2가지로 나눌 수 있다. 하나는 "의미적으로 유사한(semantically similar)" 관계이고, 다른 하나는 "쌍(pair)" 관계이다. 여기서 "쌍" 관계는 "추가하기" - "삭제하기"와 같이 대부분 의미적 또는 기능적으로 반대되는 것끼리의 관계를 의미한다. 하나의 설계 변수에 대한 값이 결정되면 (그림 2의 실선 화살표), 이 디자인 결정이 제약조건이 되어 이 설계 변수와 연결된 다른 설계 변수들의 값을 결정하는 데에 영향을 미친다 (그림 2의 점선 화살표).

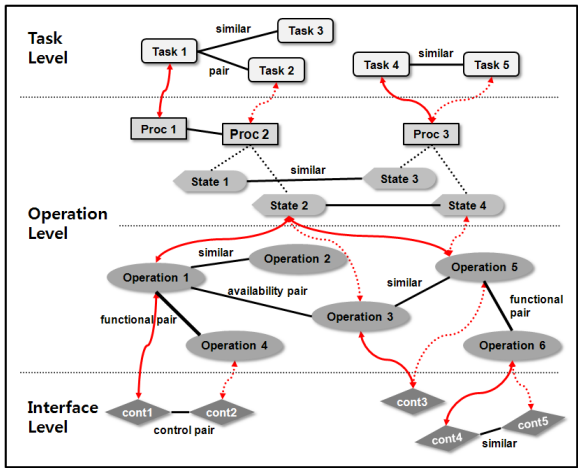

그림 2. CUID의 제약기반 설계방법 예시

여기서 제약조건은 특정 계층에 존재하는 설계 변수들 간의 수평적 관계에 해당하는 수평적(horizontal) 제약조건과 서로 다른 계층에 있는 설계변수-값 사이의 수직적 관계에 해당하는 수직적(vertical) 제약조건으로 나뉨을 알 수 있다. 표 1은 CUID에서 다루는 3가지 디자인 문제별로 사용되는 제약조건들을 정리한 것이다.

표 1. CUID에서 디자인 문제별로 사용되는 제약조건

디자인 문제	수직적 제약조건 (between-level)	수평적 제약조건 (within-level)
태스크-절차 설계	태스크 유형별 절차 템플릿 결정된 태스크-절차 관계	태스크 유형에 따른 태스크 그룹, 태스크 쌍(pair)
상태-조작 설계	필수 조작 결정된 상태-조작 관계	상태 유형에 따른 상태 그룹, 상태 쌍
		이용가능성에 따른 조작 쌍
조작-컨트롤 매핑	필수 매핑 결정된 조작-컨트롤 관계	조작 유형에 따른 조작 그룹, 기능적 관계에 따른 조작 쌍
		유사도에 따른 컨트롤 그룹, 컨트롤 쌍

수직적 제약조건에는 디자인 과정에서 결정된 변수-값 관계(design decision) 뿐 아니라, 기존 기기들 또는 사용자들이 가진 일반적 멘탈모델(mental model)과의 일관성을 지키기 위한 외적(external) 제약조건도 포함된다. 태스크 유형에 따른 수행절차 템플릿, (거의) 모든 시스템 상태에서 가능해야 하는 필수 조작(기능), "재생하기" - "▶ 버튼" 과 같이 반드시 지켜져야 하는 필수 매핑이 이에 해당한다. 이러한 외적 제약조건은 사회적 표준(social standards)[39] 또는 문화적 관습(cultural conventions)[1]으로 볼 수 있다. 이런 측면에서 CUID는 외적 일관성과 내적 일관성을 모두 고려한다고 볼 수 있다. 수평적 제약조건에는 기본적으로 의미적 유사성에 따른 "그룹"과 "쌍(pair)" 관계가 포함된다. 각 디자인 문제에서 설계변수 간 의미적 유사성을 판단하기 위해 태스크 유형, 시스템 상태 유형, 조작 유형을 개발하였

다. 상태-조작 설계 문제에서는 또 하나의 수평적 제약조건인 "이용가능성(availability)에 따른 조작 쌍(operation pair)" 이 존재한다. 이 조작 쌍은 조작-컨트롤 매핑 문제에서 사용되는 "기능적 관계에 따른 조작 쌍" 과는 달리 시스템 상태에서 자주 함께 이용 가능한 조작들의 쌍을 의미한다. 이것은 "특정 상태에서 조작 A가 가능하면, 일반적으로 조작 B도 가능하다" 는 식의 규칙으로 사용될 수 있다.

이러한 제약기반 설계 과정은 매우 많은 수의 설계변수들과 그것들 간의 관계를 고려해야 하는 실제 UI 디자인 작업의 효율성을 훨씬 더 높여줄 수 있을 뿐 아니라, 디자이너가 놓치고 넘어갈 수 있는 부분들을 줄여줌으로써 완성되는 UI의 사용성 또한 높일 수 있다.

3.2 하이웨이 기반 설계 (highway-based design)

제약기반 설계 과정에서 주어진 설계변수들 중 어떤 변수의 값을 먼저 결정하느냐에 따라 디자인 결과와 디자인에 소요되는 시간이 달라질 수 있다. 이 문제를 해결하기 위해 CUID에서는 설계변수들 중에서 대표성을 띄는 변수들을 하이웨이 변수(highway variable)로 선정하여 이 변수들의 값을 먼저 결정하고 이 결정을 제약(highway constraint)으로 삼아 해당 하이웨이 변수와 유사한 나머지 변수들의 값을 결정하는 방법을 사용하였다. 하이웨이 태스크는 태스크의 수행 빈도(frequency)와 일반성(generality)*을 고려하여 선정하며, 하이웨이 상태는 하이웨이 태스크에 속해 있는 상태 중에서 선정한다. 다기능 복합기기에 대해서 대부분의 사용자들은 전문가(expert user)인 동시에 초보자(novice user)이다. 즉, 자주 수행하는 태스크에 대해서는 전문가이지만, 수행해 본적이 없거나 거의 수행하지 않는 태스크에 대해서는 초보자에 해당한다. 이러한 사용자들은 새로운 또는 수행 빈도가 낮은 태스크를 수행할 때 이미 알고 있는 것, 즉 자주 수행해본 태스크(highway task)에 대한 지식을 적용한다.[40,41] 따라서 하이웨이 기반의 설계를 통해 이러한 사용자들의 유추적 추론(analogical reasoning) 과정을 효과적으로 지원할 수 있다.

CUID는 기본적으로 일관성을 추구하지만, 일관성만 고집하는 것은 위험한 일이다.[1,8,9,33] 따라서 CUID에서는 일관성의 상위 목표인 사용성(usability)을 높이기 위해 일관성 외에 다른 속성들(효율성, 안전성 등)도 고려한다. 예를 들어, 수행 빈도가 높은 태스크(frequent task)는 보다 간략한 절차(shortcut)를 제공해야 한다는 규칙, 잘못 수행한 결과가 치명적일 수 있는 태스크는 "확인(confirmation)" 절차를 반드시 포함시켜야 한다는 규칙, 필요하다면 특정 태스크를 수행하는 절차와 특정 조작을 수행하는 컨트롤을 2개 이상 제공해야 한다는 규칙 등이 사용된다.

* 여기서 태스크의 일반성(generality)은 그 수행 빈도와 관계없이 해당 태스크가 그것과 유사한 다른 태스크들을 포괄할 수 있을 정도로 일반적 것인지를 나타낸다.

4. 일관성 있는 인터랙션 설계(CUID) 방법론

그림 3은 CUID 디자인 프로세스를 나타낸다. 크게 태스크-절차 설계, 상태-조작 설계, 조작-컨트롤 매핑의 3단계로 구성되며, 각 단계(phase)는 다시 3~4개의 세부 단계(step)로 이루어져 있다. 3개의 설계 단계 간에, 그리고 각 단계 내의 세부 단계 간에는 피드백 루프(feedback loop)가 존재하여 전체 디자인 과정은 반복적으로(iteratively) 진행된다. 각 단계의 시작은 해당 설계변수들에 대한 분석에 해당하는데, 여기서 설계 제약조건(design constraints), 즉 설계변수들 간의 관계가 결정된다. 태스크 분석(task analysis)을 통해서는 태스크 간의 관계(유사성에 기반을 둔 태스크 그룹 및 태스크 쌍)가, 시스템 상태 분석(state analysis)을 통해서는 상태 간의 관계(유사성에 기반을 둔 상태 그룹 및 상태 쌍)가, 그리고 조작/컨트롤 분석(op/control analysis)을 통해서는 조작 간의 관계(유사성에 기반을 둔 조작 그룹 및 조작 쌍)와 컨트롤 간의 관계(유사성에 기반을 둔 컨트롤 그룹 및 컨트롤 쌍)가 도출된다. 각 단계에서 사용될 설계 제약조건들이 정해지면 본격적으로 설계변수들에 대해 값을 결정하고, 각 단계의 마지막에는 결정한 설계해(design solutions)들을 수정/보완(refinement)하는 과정을 거친다.

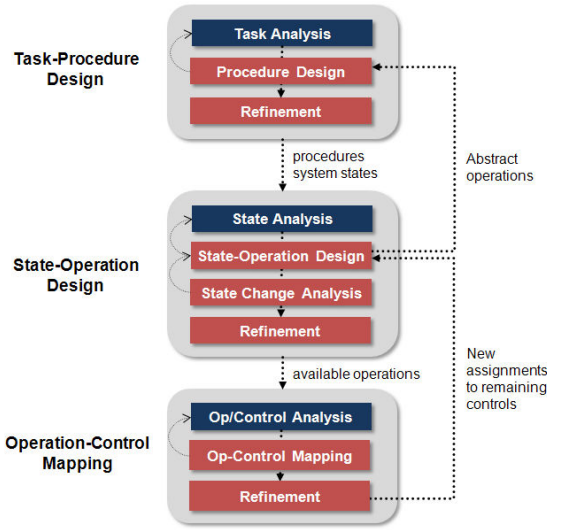

그림 3. CUID 디자인 프로세스

이러한 프로세스가 잘 개발된다 하더라도 다기능 복합기기의 사용자 인터랙션 설계를 수작업으로 하기란 매우 어려운 일이다. 따라서 이 프로세스를 효과적으로 지원할 수 있는 소프트웨어 툴**을 함께 개발하였다.

4.1 태스크-절차 설계 (task-procedure design)

** 현재(2009년 1월 5일) 이 툴은 100% 완성된 S/W가 아닌 CUID의 핵심적인 프로세스들을 위주로 한 프로토타입으로 개발되었다.

4.1.1 태스크 분석 (task analysis)

태스크 절차의 설계는 태스크 분석으로 시작한다. 설계할 제품으로 수행되어야 할 태스크들을 선정하고, 태스크 별로 몇 가지 속성을 파악한다. 태스크 속성에는 "action(object)" 형태로 이루어진 태스크 이름, 태스크 객체(object)의 유형(text, picture, audio, video 등), 데이터흐름(dataflow) 유형, 사용 모드(visual, auditory, both), 빈도(frequency), 긴급도(urgency)*, 치명도(criticality)**가 포함된다. 여기서 데이터흐름 유형이 무엇인지에 따라 표 2와 같이 태스크 유형이 결정된다. 그림 4는 사용자와 기기 간 상호작용에서의 데이터흐름을 나타내고 있다. 여기서 "Outside(외부)"는 통신이 가능한 다른 기기, 위성, 기지국 등이 포함된다. 태스크 유형을 데이터흐름에 따라 분류한 것은 태스크 수행 절차의 많은 부분이 데이터흐름에 의해 결정되기 때문이다. 태스크 유형에 따라 결정되는 태스크 그룹(계층 1의 4개 그룹, 계층 2의 7개 그룹)에서 디자이너는 그룹별로 하나 또는 둘 이상의 하이웨이 태스크를 선정한다. 이 때 태스크 속성 중 "빈도"에 대한 정보가 사용된다. 마지막으로 주어진 태스크 중에서 의미적 또는 기능적으로 반대되는 태스크끼리는 태스크 쌍(pair)으로 선정한다. (예. "파일전송" – "파일받기")

표 2. 태스크 유형

일반 유형 (계층1)	데이터흐름 유형 (계층2)	태스크 예시
1. 생성/기록	1.1. User 〉 Device	메모하기, 일정추가
	1.2. Outside 〉 Device	음성녹음, 사진촬영
2. 편집/설정	2.1. Device 〉 Device	사진편집, 파일삭제, 알람설정, 메모잠금
3. 전송/출력	3.1. User 〉 Outside	메시지/메일 전송
	3.2. Device 〉 Outside	파일전송, 사진출력
4. 재생/확인	4.1. Device 〉 User	음악재생, 사진보기 게임하기, 단어검색
	4.2. Outside 〉 User	라디오듣기, TV보기

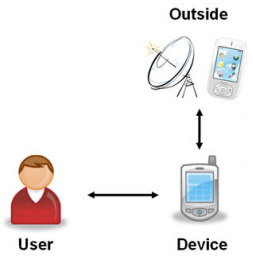

그림 4. Dataflow in HCI

4.1.2 절차 설계 (procedure design)

태스크 절차의 설계는 앞에서 결정된 태스크 그룹별로 이루어진다. 그림 5와 같이 먼저 태스크 그룹을 선택하고, 해당 그룹에서의 하이웨이 태스크 먼저 절차를 설계한다. 여기서 사용되는 제약조건은 크게 두 가지로 구분할 수 있는데, 현재 선택한 태스크의 유형에 따라 제공되는 절차 템플릿(procedure template)과 현재 선택한 태스크와 관련된 태스크들의 수행절차가 그것이다. 표 2의 7가지 데이터흐름 유형별로 1개 이상의 절차 템플릿이 제공된다. 가령, "음악재생" 태스크와 같이 데이터흐름이 "Device 〉 User"인 경우에는 다음과 같은 절차 템플릿을 사용할 수 있다.

~menu(main) → select_menu(x)→ ~list(obj) → select(obj) → ~processing(x)

여기서 "~" 표시는 시스템 상태를 나타내고, "~" 표시가 없는 부분은 사용자의 조작(operation)에 해당한다. 주어진 템플릿을 사용하지 않고 현재 태스크와 관련된 태스크들의 절차를 활용할 수도 있는데, 여기서 관련된 태스크는 같은 태스크 그룹으로 묶여진 유사한 태스크들과 쌍 관계에 있는 태스크 모두 포함한다. 관련된 태스크 리스트에는 기본적으로 현재 선택된 태스크 그룹에 속하는 태스크들 중 절차 설계가 완성된 것들이 나타지만, 추상화 계층(Level 0, Level 1, Level 2)에 따라 리스트에 나타나는 태스크의 범위를 확대/축소해가며 적절한 태스크를 고를 수 있도록 하였다. 계층 1은 일반 유형, 계층 2는 데이터흐름 유형이 같은 태스크 범위를 나타내며, 계층 0은 전체 태스크 범위를 의미한다.

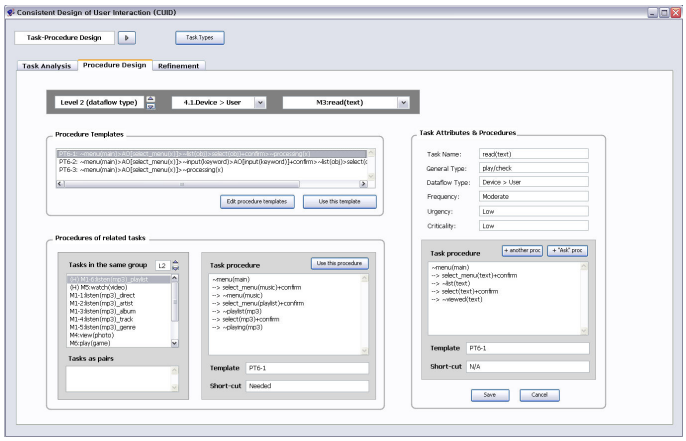

그림 5. CUID 태스크-절차 설계 화면

4.1.3 태스크 절차 수정/보완 (refinement)

모든 태스크의 절차 설계가 완성되면, 태스크 그룹별로 속해 있는 태스크들의 수행 절차를 비교해가면서 서로 얼마나 유사하게 설계되었는지를 평가하여 필요한 경우 절차를 수정 · 보완한다.

4.2 상태-조작 설계 (state-operation design)

4.2.1 시스템 상태 분석 (state analysis)

* 여기서 긴급도(urgency)는 해당 태스크를 얼마나 재빠르게 시작할 필요가 있는지를 의미한다. 가령, '음성 녹음하기' 태스크가 긴급한 정도가 높은 태스크에 해당한다.

** 치명도(criticality)는 해당 태스크를 잘못 수행한 결과가 사용자에게 얼마나 치명적일 수 있는지를 나타낸다. 예를 들어, '파일 삭제하기' 태스크가 치명도가 높은 태스크에 속한다.

앞서 설계된 태스크 절차로부터 시스템 상태의 집합이 도출된다. CUID에서 시스템 상태(system state)는 사용자에게 뚜렷이 인식되는 화면(screen)의 내용과 보이지 않거나 뚜렷이 인식되지 않는 기기 내부의 상태(internal state)로 구성된다. 먼저 시스템 상태별로 화면의 내용과 화면의 유형[표 3], 내부 상태가 무엇인지를 분석한다. 화면 유형에 따라 결정되는 상태 그룹(계층 1의 6개 그룹, 계층 2의 14개 그룹, 계층 3의 20개 그룹)에서 디자이너는 그룹별로 하나 또는 둘 이상의 하이웨이 상태를 선정한다. 이 때 각 상태가 속해 있는 태스크가 하이웨이 태스크에 해당하는지에 대한 정보가 주어진다. 상태 쌍(pair) 관계는 내부 상태가 무엇인지에 따라 결정된다. 즉, 화면의 내용(예. "비디오재생 화면")이 동일하고 내부 상태가 서로 반대되는(예. "재생중" - "일시정지중") 상태들이 상태 쌍 관계를 형성한다.

표 3. 화면(screen) 유형

계층(level) 1	계층(level) 2	계층(level) 3
1. menu/list	1.1. menu	1.1.1. generalMenu
		1.1.2. contextualMenu
		1.1.3. setMenu
	1.2. list	1.2.1. fileList
		1.2.2. checkList
		1.2.3. searchList
	1.3. other_menu/list	1.3.1. other_menu/list
2. edit/set	2.1. edit_text	2.1.1. input_text
		2.1.2. input_number
		2.1.3. input_keyword
	2.2. edit_picture	2.2.1. edit_picture
	2.3. other_edit/set	2.3.1. other_edit/set
3. ready	3.1. ready	3.1.1. ready
4. processing	4.1. playing	4.1.1. playing
	4.2. recording	4.2.1. recording
	4.3. searching	4.3.1. searching
	4.4. transmitting	4.4.1. transmitting
	4.5. other_processing	4.5.1. other_processing
5. content	5.1. content	5.1. content
6. confirm	6.1. confirm	6.1. confirm

4.2.2 상태-조작 설계 (state-operation design)

상태-조작 설계도 태스크 절차의 설계와 마찬가지로 앞에서 결정된 상태 그룹별로 이루어진다. 그림 6과 같이 먼저 상태 그룹을 선택하고, 해당 그룹에서의 하이웨이 상태 먼저 가능해야 하는 조작(기능)을 결정한다. 여기서 사용되는 제약조건 또한 두 가지로 구분할 수 있는데, 거의 모든 시스템 상태에서 가능해야 하는 필수 조작들(essential operations)과 현재 선택한 상태와 유사하거나 쌍(pair) 관계에 있는 상태에서 가능한 조작들이다. "전원 끄기", "이전(또는 상위) 화면으로 돌아가기" 등이 필수 조작에 해당한다. 이 세부 단계를 시작할 때 설계할 시스템에서의 필수 조작들을 먼저 입력하고, 상태별로 가능한 조작을 결정할 때 사용하도록 한다. 관련된 상태 리스트에는 기본적으로 현재 선택된 상태 그룹에 속하는 상태들이 나타나지만, 태스크와 마찬가지로 추상화 계층(Level 0, Level 1, Level 2, Level 3)에 따라 리스트에 나타나는 상태의 범위를 확대/축소해가며 적절한 상태를 고를 수 있도록 하였다. 계층 1, 계층 2, 계층 3은 화면 유형의 계층[표 3]을 나타내고, 계층 0을 선택하면 모든 시스템 상태가 리스트에 주어진다.

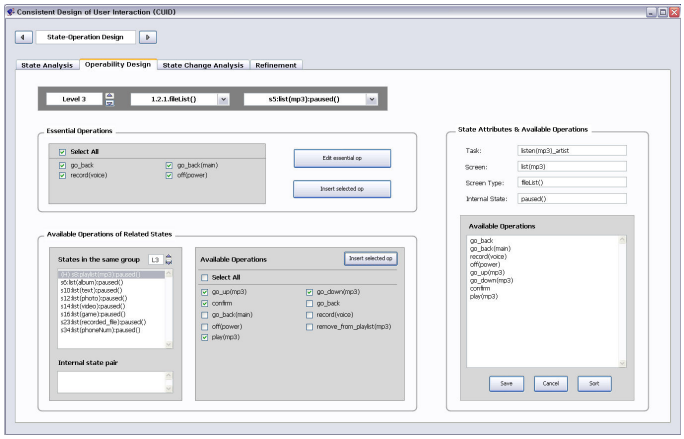

그림 6. CUID 상태-조작 설계 화면

특정 시스템 상태에서 가능한 조작으로 둘 이상의 조작으로 구성되는 추상 조작(abstract operation)이 포함되는 경우에는, 팝업창이 나타나 포함된 추상 조작 각각에 대해 절차 설계가 이루어진다. 여기서 새로운 시스템 상태들이 생겨나면, 생겨난 상태들에 대해서도 상태-조작 설계가 진행된다.

4.2.3 상태 변화 분석 (state change analysis)

앞 단계에서 결정된 상태별 가능한 조작 하나하나에 대해 해당 조작이 실행되면 시스템 상태에 어떤 변화가 생기는지를 분석한다. 화면의 내용 또는 내부 상태가 달라지는 조작이 있으면 또다시 새로운 시스템 상태가 도출되고, 앞 단계로 돌아가 이 상태들에 대해서 상태-조작 설계를 수행한다.

4.2.4 상태-조작 관계 수정/보완 (refinement)

모든 시스템 상태에 대해 가능한 조작들이 결정되면, 그 결과를 평가하여 필요한 경우 절차를 수정·보완한다. CUID에서는 상태-조작 관계를 평가하기 위해 상태-조작 행렬(state-operation matrix)과 연관규칙 탐사(association rule mining) 기법이 활용된다. 표 4는 상태-조작 행렬의 예이다. 행은 동일한 그룹에 속한 상태들을, 열은 가능한 조작들을 나타내며, "√" 표시가 된 셀은 해당 상태에서 해당 조작이 가능하도록 설계되었음 의미한다. 설계자는 이러한 행렬을 살펴봄으로써 상태-조작 관계의 일관성을 평가할 수 있다. 예를 들어, 표 4에서 op1과 op10는 하나의 상태만 제외하고 나머지 모든 상태에서 가능한데, 이런 경우 st7과 st2에서도 각각 op1과 op10이 가능하도록 수정할 수 있다. 연관규칙 탐사는 데이터마이닝 분야에서 널리 활용되는 기법인데[42], 상태-조작 설계 문제에서도 이것을 활용할 수 있다. CUID에서는 다수의 시스템 상태에서 함께 가능한 조작들의 쌍

(availability pair)을 찾아주고, 이 중 한 가지만 가능한 상태들을 리스트로 보여주어 디자이너가 나머지 한 개의 조작도 가능하도록 설계할 지를 고려하도록 하였다.

표 4. 상태-조작 행렬 예시

	op1	op2	op3	op4	op5	op6	op7	op8	op9	op10
st1	√	√				√	√			√
st2	√	√			√	√	√			
st3	√		√		√	√	√			√
st4	√		√		√	√	√	√		√
st5	√	√		√		√	√		√	√
st6	√		√		√	√	√	√		√
st7		√	√			√	√			√

4.3 조작-컨트롤 매핑 (operation-control mapping)

4.3.1 조작 및 컨트롤 분석 (operation and control analysis)

먼저 가능한 모든 조작에 대해 각각이 어떤 유형에 속하는지를 파악한다. 기본적으로 조작의 이름(action 부분)을 기반으로 적합한 유형이 주어지면, 디자이너가 확인하여 수정한다. 결정되는 조작 유형에 따라 조작 그룹이 형성된다. 또한, 가능한 조작들 중 의미적/기능적으로 상반되는 것들을 조작 쌍(pair)로 선정한다. 예를 들어, "add(x)-delete(x)", "go_prev(x)-go_next(x)", "start(x)-stop(x)" 등이 조작 쌍에 해당한다. 인터페이스 컨트롤에 대해서는 먼저 가능한 컨트롤들(H/W 또는 S/W 버튼, 다양한 터치 동작 등)을 정의하고, 의미적/기능적으로 서로 유사한 관계에 있는 컨트롤 그룹(예. "왼쪽 방향키" 와 "back 버튼")과 서로 상반되는 관계에 있는 컨트롤 쌍(예. "왼쪽 방향키" 와 "오른쪽 방향키")을 결정한다. 마지막으로 주어진 조작들과 컨트롤 중에서 반드시 지켜져야 하는, 즉 사회적 규범에 해당하는 필수 매핑(required mapping)들을 결정한다. 가령, "볼륨 높이기" 조작은 "+ 버튼" 또는 "위쪽 방향키" 와 매핑시켜야 한다. 이러한 분석이 끝나면, 그 결과로서 가능한 조작들 간의 제약 네트워크와 컨트롤 간의 제약 네트워크가 형성된다.

4.3.2 조작-컨트롤 매핑 (operation-control mapping)

조작-컨트롤 간 매핑 문제는 주로 인공지능(AI) 분야에서 다루는 제약만족 문제(constraint satisfaction problem, CSP)로 볼 수 있다. 즉, 가능한 조작(operation)들을 변수(variable)로, 가능한 인터페이스 컨트롤(control)들을 변수들이 취할 수 있는 값(value)들로, 조작들 간의 관계와 인터페이스 컨트롤들 간의 관계들을 제약조건(constraint)으로 놓고 각 변수에 적당한 값을 할당하는 문제가 된다. 따라서 이 문제를 해결하기 위한 절차는 일반적인 제약만족 알고리즘[43]을 기반으로 하였다. 그림 7은 CUID에서의 조작-컨트롤 매핑 과정을 도식화한 것이다.

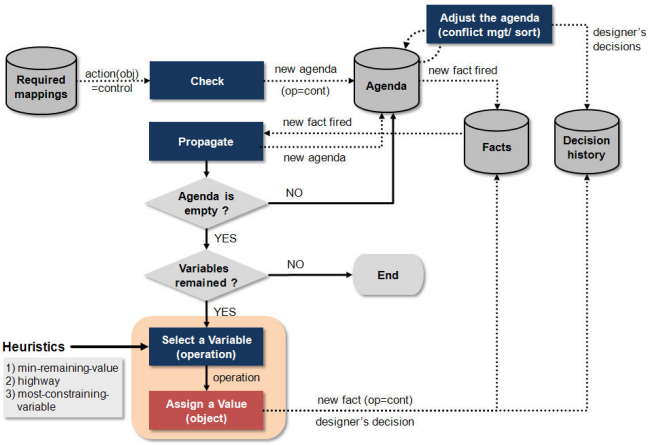

그림 7. CUID 조작-컨트롤 매핑 프로세스

이 절차는 많은 부분을 프로그램이 자동으로 수행하고, 필요할 때마다 설계자에게 질문을 던져 직접 의사결정하도록 한다. 먼저, 필수 매핑들을 체크하여 해당되는 조작들에 컨트롤을 할당한다. 여기서 할당된 조작-컨트롤 관계는 사실(fact)로 확정되기 전에 아젠다(agenda)로 추가된다. 그리고 나서, 아젠다 리스트를 조정하는데 서로 상충되는 조작-컨트롤 관계가 있는 경우(예. 동일한 상태에서 가능한 서로 다른 조작들에 동일한 컨트롤이 할당된 경우, 동일한 상태에서 가능한 동일한 조작에 서로 다른 2개 이상의 컨트롤이 할당된 경우)에는 설계자에게 질문을 하여 직접 의사결정을 하도록 한다. 아젠다 리스트의 조정이 끝나면, 아젠다를 하나씩 차례대로 사실(fact)로 발생(fire)시키고 그림 8과 같이 발생한 사실, 즉 디자인 결정(실선 화살표로 연결된 관계)을 새로운 제약조건으로 삼아 조작들 간의 제약 네트워크와 컨트롤 간의 제약 네트워크에 전파(propagation)시킨다.

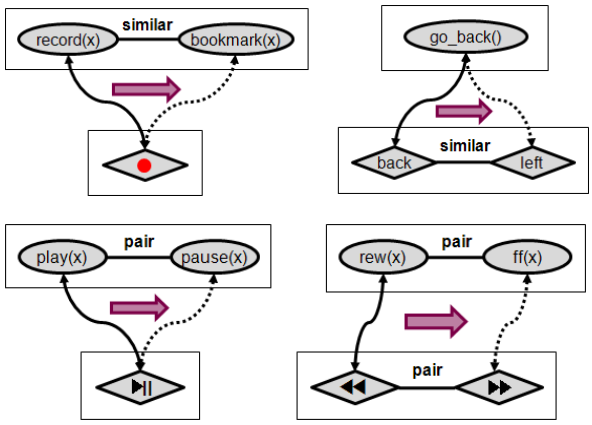

그림 8. 조작-컨트롤 매핑에서의 제약조건 전파 예시

전파 과정에서 생겨난 새로운 조작-컨트롤 관계들(그림 8에서 점선 화살표로 연결된 관계)은 다시 새로운 아젠다로 추가되고, 그 때마다 다시 아젠다 리스트를 조정한다. 이러한 과정은 아젠다 리스트가 텅 빌 때까지 반복된다. 더 이상 고려할 아젠다가 없는 상태에서 아직 값(컨트롤)이 정해지

지 않은 변수(가능한 조작)가 남아 있으면, 몇 가지 휴리스틱을 사용하여 남은 변수들 중 값을 먼저 할당해야 할 변수와 그것이 취할 수 있는 값의 범위(domain)를 디자이너에게 제시한다. 여기서 사용되는 휴리스틱은 제약만족 문제에서 일반적으로 사용되는 minimum-remaining-value (MRV) 휴리스틱과 most-constraining-variable 휴리스틱[44] 외에도 하이웨이(highway) 휴리스틱이 포함되는데, 이것은 하이웨이 상태에서 가능한 조작부터 컨트롤을 할당하기 위함이다. 이 역시 하이웨이 기반 설계방법의 일환이라고 할 수 있다. 디자이너가 선택된 변수(조작)에 적절한 값(컨트롤)을 할당하면, 이 결정(fact)을 새로운 제약조건으로 삼아 전파시킨다. 이러한 전체 과정은 모든 가능한 조작에 컨트롤이 할당될 때까지 반복된다. 조작-컨트롤 매핑 과정에서 디자이너가 직접 내린 결정들은 decision history에 저장하여 디자이너가 언제라도 특정 의사결정 시점으로 다시 돌아갈 수 있도록 하였다.

4.3.3 조작-컨트롤 관계 수정/보완 (refinement)

이 단계에서는 크게 두 가지 작업이 이루어진다. 먼저, 특정 시스템 상태에서 아무런 조작(기능)도 할당되지 않고 남은 컨트롤들을 설계자에게 알려주면, 설계자는 남은 컨트롤을 어떻게 처리할 지에 대한 결정을 내린다. 해당 컨트롤이 다른 상태들에서 어떤 조작(기능)에 할당되었는지를 참고하여 적절한 조작(기능)을 추가할 수도 있고, 해당 상태에서 이미 다른 컨트롤에 할당된 조작(기능)을 중복 배치함으로써 하나의 조작(기능)을 수행하는 컨트롤이 2개 이상이 되도록 할 수도 있고, 아무 조작(기능)이 할당되지 않은 채로 둘 수도 있다. 두 번째 작업은 인터페이스 컨트롤마다 할당된 조작(기능)들을 모아 놓고, 사용자들이 해당 컨트롤에 대한 일관된 이미지를 형성하는 데 방해가 될 수 있는 것을 찾는 것이다. 해당되는 것이 있으면, 그것을 제거하거나 인터페이스(화면)에 해당 조작(기능)에 대한 힌트를 제공하는 등의 조치를 취한다.

5. 사례 연구: CUID 방법론 검증

CUID 프로세스 및 소프트웨어 툴의 효용성을 검증하기 위해 다음의 절차를 거쳤다.

1) CUID를 사용하여 실험기기의 인터랙션 재설계
2) 사용자들로부터 실험기기의 비일관성 문제 수집
3) 재설계한 인터랙션 결과가 비일관성 문제들을 얼마나 해결했는지 파악

먼저 다기능 복합기기 중 하나인 최신 mp3 player (음악 재생, 동영상 재생, FM 라디오, 텍스트 뷰어, 사진 뷰어, 블루투스, 게임 등의 기능을 포함)의 사용자 인터랙션 부분을 CUID를 사용하여 재설계하였다. 즉, 실험기기가 제공하는 모든 태스크의 수행 절차, 각 시스템 상태에서 가능해야 하는 조작(기능)을 설계하였고, 가능한 모든 조작(기능)과 주어진 인터페이스 컨트롤(여기서는 물리적 버튼만 해당) 간의 매핑을 수행하였다. 그리고 나서, 6명의 사용자들을 대상으로 실험기기를 3~4일씩 사용해 보도록 한 뒤 일관성이 지켜지지 않은 부분들을 기록하도록 하였다. 사용자는 실험기기 및 이와 유사한 기기에 대한 사용 경험이 없으며, UI 디자인에 대한 과목을 1개 이상 수강하여 일관성을 비롯한 UI 디자인 원리를 제대로 이해하고 있는 대학원생들로 구성하였다. 6명의 사용들로부터 총 21개의 논리적 비일관성 문제들을 수집(여기서 물리적/시각적 비일관성 문제는 제외)하였는데, 몇 가지 대표적인 예는 다음과 같다.

- 대부분의 메뉴 화면에서는 "음악 재생 또는 정지" 조작이 가능하지만, 일부 메뉴 화면에서는 불가능하다. (해당 버튼을 눌러도 아무런 반응이 없음)
- 대부분의 메뉴 화면에서는 "재생(▶‖) 버튼"을 누르면 음악이 재생 또는 정지되지만, 파일리스트 화면에서는 선택된 파일이 재생된다.
- 대부분의 메뉴/리스트 화면에서는 "back 버튼"과 "왼쪽 방향키" 둘 다 같은 기능(이전 화면으로의 이동)을 수행하지만, 메인메뉴 화면과 재생 화면 등에서는 서로 다른 기능을 수행한다.
- 동영상재생 화면에서는 볼륨조절 조작이 가로보기를 기준으로 상하 방향키로 실행되지만, 같은 가로보기 상태인 사진보기 화면에서는 볼륨조절 조작이 좌우 방향키로 실행된다.

마지막으로, CUID를 사용해 재설계된 결과가 사용자들이 기록한 실험기기의 비일관성 문제들 중 얼마나 많은 부분을 해결하였는지를 파악하였다. 사용자들이 기록한 총 21개의 논리적인 비일관성 문제들 중에서 18개의 문제들이 CUID를 사용한 인터랙션 설계를 통해 해결되었다.

6. 결론

본 연구에서는 사용자 인터랙션을 일관성 있게 설계하도록 돕는 방법론(프로세스와 S/W 툴 포함)인 CUID를 개발하였고, 사례 연구를 통해 CUID의 효용성을 검증하였다. 앞으로 보다 많은 다양한 종류의 다기능 복합기기들을 재설계해 봄으로써 CUID 방법론을 수정·보완할 계획이다. 또한, 설계변수 간의 유사성을 파악하는 데에 활용되는 태스크 유형, 상태 유형, 조작 유형들을 보완하는 일이 필요하다.

CUID는 UI 설계자에게는 기능의 수가 매우 많은 인터랙티브 기기의 인터랙션 부분을 설계하는 문제의 복잡도를 줄여주고, 사용자에게는 많은 기능을 가지면서도 일관성이 높아 배우기 쉽고 사용하기 쉬운 제품을 제공하는 데에 기여할 수 있을 것으로 기대된다.

참고문헌

[1] D. A. Norman, The Psychology of Everyday Things, New York: Basic Books, 1988.

[2] D. -S. Lee, D. D. Woods, D. Kidwell. "Escape from Designers' Dilemma on Creeping Featurism", In Proceedings of Human Factors and Ergonomics Society Annual Meeting, Test and Evaluation, pp. 2562-2566, 2006.

[3] J. Kapica. "My Cellphone, My Enemy: Feature Creep and the Cellphone Dilemma", Digtial Journal, January 3, 2006.

[4] J. Surowiecki. "Feature Presentation", The New Yorker, May 28, 2007.

[5] Philips. Our Brand Promise "Sense and Simplicity", Available from http://www.philips.com/about/brand/brandpromise/

[6] Physorg. "Vodafone Simply: the more the better?", May 21, 2005. Available from: http://www.physorg.com/news4197.html

[7] R. T. Rust, D. V. Thompson, R. W. Hamilton. "Defeating Feature Fatigue", Harvard Business Review, Vol. 84, No. 2, pp. 98~107, 2006.

[8] J. Grudin. "The Case Against User Interface Consistency", Communications of the ACM, Vol. 1, No. 1, pp. 59~71, 1989.

[9] J. Grudin. "Consistency, standards, and formal approaches to interface development and evaluation", ACM Transactions on Information Systems, Vol. 10, No. 1, pp. 103~111, 1992.

[10] P. Ketola, H. Hjelmeroos, K. J. Raiha. "Coping with consistency under multiple design constraints: the case of the Nokia 9000 WWW browser", Personal Technologies, Vol. 4, pp. 86~95, 2000.

[11] P. Reisner. "What is consistency?", In INTERACT '90, pp. 175~181, 1990.

[12] J. Nielsen. Usability Engineering, Academic Press, Boston, MA, 1993.

[13] D. Hix, H. R. Hartson, Developing User Interfaces: ensuring usability through product and process, Wiley, New York, 1993.

[14] B. Shneiderman. Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd Ed., Reading Mass.: Addison-Wesley, 1998.

[15] A. Cooper, R. Reimann, D. Cronin. About Face 3.0: The Essential of Interaction Design, John Wiley & Sons, New York, 2007.

[16] A. Dix, J. Finlay, G. D. Abowd, R. Beale. Human Computer Interaction, Prentice-Hall, 2004.

[17] H. Sharp, Y. Rogers, J. Preece. Interaction Design: Beyond Human-Computer Interaction, 2nd Ed., John Wiley & Sons, 2007.

[18] Apple. iPhone Human Interface Guidelines, Apple Inc., 2007.

[19] P. J. Barnard, N. V. Hammond, J. Morton, J. B. Long. "Consistency and Compatibility in Human-Computer Dialogue". International Journal of Human-Computer Studies, Vol. 15, pp. 87~134, 1981.

[20] W. A. Kellogg. "Conceptual Consistency in the User Interface: Effects on User Performance", In H. J. Bulinger, B. Shackel (Ed.), INTERACT '87 IFIP Conference on Human-Computer Interaction, pp. 389~394, 1987.

[21] P. G. Polson. "The Consequences of Consistent and Inconsistent User Interfaces", In R. Guindon (Ed.), Cognitive Science and Its Applications for Human-Computer Interaction, pp. 59~108, 1988.

[22] J. Nielsen, et al. Coordinating User Interfaces for Consistency, Academic Press, San Diego, 1989.

[23] T. Tanaka, R. E. Eberts, G. Salvendy. "Consistency of Human-Computer Interface Design: Quantification and Validation", Human Factors, Vol. 33, pp. 653~676, 1991.

[24] A. A. Ozok, G. Salvendy. "Measuring Consistency of Web Page Design and Its Effects on Performance and Satisfaction", Ergonomics, Vol. 43, No. 4, pp. 443~460, 2000.

[25] A. A. Ozok, G. Salvendy. "The Effects of Language Inconsistency on Performance and Satisfaction on the World Wide Web: results from four experiments", Behaviour and Information Technology, Vol. 22, No. 3, pp. 155~163, 2003.

[26] A. A. Ozok, G. Salvendy. "Twenty Guidelines for the Design of Web-based Interfaces with Consistent Language", Computers in Human Behavior, Vol. 20, pp. 149-161, 2004.

[27] S. J. Payne, T. R. G. Green. "Task-Action Grammar: a model of the mental representation of task languages", Human-Computer Interaction, Vol. 2, pp. 93~133, 1986.

[28] P. Reisner. "APT: A description of user interface inconsistency", International Journal of Man-Machine Studies, Vol. 39, pp. 215~236, 1993.

[29] S. Poitrenaud. "The PROCOPE semantic network: An alternative to action grammars", International Journal of Human-Computer Studies, Vol. 42, pp. 31~69, 1995.

[30] R. Mahajan, B. Shneiderman. "Visual Consistency Checking Tools for Graphical User Interfaces", IEEE Transactions on Software Engineering, Vol. 23, No. 11, pp.722-735, 1997.

[31] D. E. Kieras, S. D. Wood, K. Abotel, A. Hornof. "GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs", In Proceedings of UIST, pp. 91~100, 1995

[32] K. Gajos, D. S. Weld. "SUPPLE: automatically generating user interfaces", In Proceedings of IUI, pp. 93~100, 2004.

[33] J. Nichols, B. A. Myers, B. Rothrock, "UNIFORM: automatically generating consisten remote control user interface", In Proceedings of CHI, pp. 611~620, 2006.

[34] B. E. John, D. E. Kieras. "Using GOMS for User Interface Design and Evaluation: Which technique?", ACM Transactions on Computer-Human Interaction, Vol. 3, No. 4, pp. 287~319, 1996.

[35] L. Lin, L. -C. Chen. "Constraints Modelling in Product Design", Journal of Engineering Design, Vol. 13, No. 3, pp. 205~214, 2002.

[36] M. Hashemian, P. Gu. "A Constraint-Based System for Product Design", Concurrent Engineering: Research and Applications, Vol. 3, No. 3, pp. 177~186, 1995.

[37] A. Kilian. "Design Innovation Through Constraint Modeling ", International Journal of Architectural Computing, Vol. 4, No. 1, pp. 87~105, 2006.

[38] A. Borning, R. Duisberg. "Constraint-based tools for building user interfaces", ACM Transactions on Graphics, Vol. 5, No. 4, pp. 345~374, 1986.

[39] W. C. Yoon. "Task-Interface Matching: How we may design user interface", In Proceedings of the 15th International Ergonomics Association Triennial Congress, 2003.

[40] J. M. Carroll, M. B. Rosson. "Paradox of the active user", In J. M. Carroll (Ed.) Interfacing thought: cognitive aspects of human-computer interaction, MIT Press, Cambridge, MA, 1987

[41] J. Rieman, C. Lewis, R. M. Young, P. G. Polson. "Why is a raven like a writing desk? Lessons in interface consistency and analogical reasoning from two cognitive architectures", In Proceedings of CHI '94, pp. 438~444, 1994.

[42] R. Agrawal, T. Imielinski, A. N. Swami. "Mining association rules between sets of items in large databases", In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, pp. 207~216, 1993.

[43] E. Rich, K. Knight. Artificial Intelligence. 2nd Ed., McGraw-Hill, New York, 1991.

[44] S. Russel, P. Norvig. Artificial Intelligence: A Modern Approach, 2nd Ed., Prentice-Hall, 2003.