Groupware
- 정의와 특성에 대하여 -

최 윤 경*

Ⅰ. Groupware의 정의

1976년 PC 응용 소프트웨어의 영향에 대하여 예측할 때, 현재와 같이 워드 프로세싱과 스프레드쉬트 프로그램이 사무실 문화의 주류를 이룰것은 상상도 못했다. 20년이 지난 지금 우리는 컴퓨터 기술에 있어 또하나의 혁명을 맞을 시점에 서 있다. 바로 "Groupware"이다.

"Groupware"라는 단어는 많이 사용되고 있지만, 사실 잘 이해되지 못하는 경우가 많아 종종 혼돈의 원인이 되고 있다. 이러한 현상은 다음과 같은 경향으로부터 기인되었을 것이다. 그 하나는 그룹웨어의 공급자나 사용자들 모두 그룹웨어를 단순히 그 제품(appli cation)들의 합집합으로 보며, 그룹웨어의 구현에 적합한 기술(technology)이나 하부구조에 대해서는 생각하려 하지 않는 경향이 있다는 점이다. 따라서 그룹웨어의 여러 application 들 중에서 전자 우편(e-mail)이나 group calendaring/scheduling, 양식 전달체계(forms routing), 업무흐름 자동화(workflow automation), 컴퓨터 회의, 게시판, 화상 회의 시스템 등 서로 상이한 application들로부터 어떤 일관된 그룹웨어 테크놀로지(technology)를 이해해 낸다는 것은 아주 힘든 일이었다.

또 다른 경향으로는, 그룹웨어에 대한 논의를 할 때 그 촛점이 어떤 협소한 설계 기반에 기인한 단일 테크놀로지에만 맞추어 진다는 것이다. 공급자들은 보통 그들에게 친숙한 테크놀로지에 기반한 제품들만을 생산해오고 있다. 즉, 주로 communication - 그룹 내에서 정보를 push하는 기술 - 에 기반하여 제품들을 만들고 있는 공급자들은 메세징을 그룹웨어의 핵심 기술로 여긴다. 마찬가지로, 그들의 제품이 collaboration - 정보를 공유하는 기술 - 에 기반한 공급자들은 컴퓨터 회의나 공유 데이타베이스를 그룹웨어의 핵심 으로 보고 있다. 또한, 어떤 보고서에 대한 순차적인 sign-off 작업 등의 복잡한 업무를 잘 조정함으로써 그룹 내 각 개인의 업무를 도와주는 그러한 제품들의 공급자는 작업 및 업무흐름(workflow)의 자동화를 지원하는 application 개발 환경을 그룹웨어의 핵심으로 보고 있는 것이다.

사실, 완벽한 그룹웨어의 구조는 이들 세가지 모드의 그룹 업무를 모두 지원해 주는 동

* 배화여자전문대학 사무자동화과

시에, 이들 사이의 협동 체제를 만듦으로써, 이들 각각의 합보다 더 큰 결과를 산출해낸다.

일반적으로 그룹웨어를 "그룹 업무를 지원해 주는 소프트웨어 부류"라고 정의할 수 있지만 좀 더 자세하게 언급하자면, "communication, collaboration, 그리고 coordination을 통하여 그룹 내의 각 사람들이 효율적으로 일할 수 있는 환경을 지원해 주는 툴(tools)" 이라고 정의할 수 있다.

다음 장부터는 이들 세 영역 - communication, collaboration, coordination - 에 대해, 이들의 필요성 및 이들을 지원하는 테크놀로지에 대해 살펴 본다.

II. Communication

통신 시스템의 역할은 정보를 전달하기 위한 수동적인 전자 매체로서의 역할이다. 어떤 환경에서의 시간, 공간, 참여 인원수 등에 따라 가장 적절한 형태의 통신 시스템이 결정되는데, 같은 시간, 같은 공간에서 일대일로의 대화는 가장 간단한 형태의 통신을 나타낸다. 시간대가 다르거나 공간이 멀리 떨어져 있거나 참여인원이 증가할수록 통신의 복잡성은 더해간다. 어쨌거나 가장 복잡한 환경하에서 가장 공통적인 지식이 생성되고 공유되고 전달된다.

현재 기업에서의 통신 수단으로써 점차 전자 우편(elctronic mail, e-mail)이 보편화되고 있는 실정이다. E-mail은 통신의 효율성이나 정확성을 어느 정도 증진시킨 반면, 사용자나 통신망 및 시스템 관리자에게는 새로운 난제를 던지고 있다.

전자 메세징(electronic messaging)은 사람들 사이에 또는 사람과 응용프로그램 간에, 또는 응용 프로그램들 사이에서 전자적인 개체(electronic objects)를 축적전송(store-and forwrd)하는 체계이다. 전자 메세징은 한 지역에서 다른 지역으로의 비동기적인(asyn chronous) 전송을 지원한다. 메세지는 간단하거나 복잡할 수 있고, 이것은 한 개인에게 또는 그룹에게 전송될 수 있다. 메세징은 그 축적전송 방식, 즉 전송에서의 "push" 모델 덕택에 서로 다른 시간에 서로 다른 공간에서 정보를 공유할 수 있도록 해준다.

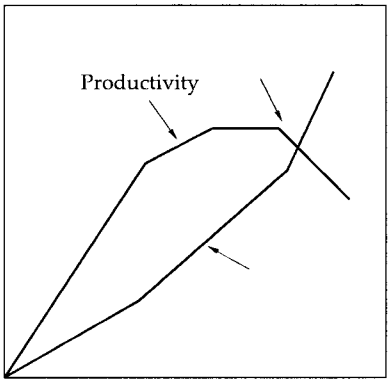
메세징은 일대다(one-to-many) 통신으로, 곧이어 다대다(many-to-many) 통신으로 진보되고 있다. E-mail을 다대다 통신으로 사용할 경우 e-mail의 용량은 기하급수적으로 증가하게 된다. 예를 들어 한 매니저가 12명의 사람들에게 프로젝트에 관한 e-mail 메세지를 전송하고, 피드백을 요청한다고 가정하자. 각 수신자가 모든 다른 수신자에게 자신의 응답 메세지를 보내는 다대다 통신 형태를 고려하면, 참여자들 간의 interaction이 증가하면 할수록 메세지의 수는 굉장히 증가하여 어느 응답이 어느 메세지에 대한 것인지를 식별하기도 어려워질 것이다.

전통적으로 메세지 저장소(message store)는 수신자에게 전달되는 메세지를 일시적으로 저장하기 위한 장소로 사용되어 왔다. 그러나 세월이 흘러 일대다, 다대다 통신으로 확장되면서 이 메세지 저장소는 공공연히 방대한 양의 공통의 정보를 담는 반영구적인 "콘테이너(container)"가 되어버렸다. 결과적으로 사용자 및 프로그램 공급자들은 이러한 정보를 관리하고, 다루고, 이들의 사용을 좀 더 자동화하

는 방안들을 탐구하게 되었다. 메세징 API (Application Programming Interfaces)들이 등장하여 메세지를 사용자로 부터 응용 프로그램으로, 응용 프로그램으로부터 사용자에게로, 그리고 응용 프로그램 간에 전달해 줌으로써, 메세징 시스템을 프로그램화하여 사용할 수 있도록 기여했다. 이러한 예제로서는 mail-enabled desktop applications, group cal en dar ing/scheduling, forms routing 등이다. 메세징 시스템은 가상적인 통신 "주체"가 될 뿐, 초기에 설계될 때 고려되지 않았던 심각한 정보 관리 문제를 초래하고 있다.

E-main은 그것이 성공하면서 오히려 자신의 희생물이 된것이다. 사용하기 용이하여 폭넓게 보급되면서 e-mail 전송량의 폭주를 유발시켰다. 이 결과 사용자들은 정보를 잃어버리기도 하고 회신받지 못한 메세지들만도 굉장히 쌓여간다. 귀중한 시간이 그리 중요하지 않은 메세지들을 읽고 가려내는데에 소비된다. 심지어 어떤 사용자들은 매일 쇄도하는 메세지들에 압도되어, 의식적으로 많은 양의 메세지들을 한꺼번에 무시해버리기도 한다. 이러한 불행한 사태는 개인적으로나 기업적인 생산성 측면에서 악 영향을 끼치게되는 것이다.

더우기 e-mail의 용량이 증가하면서 (예를 들어, 10명 이상의 수신자에게 50 메가바이트 크기의 화일을 보낼 수 있다면) 문제는 더욱 심각해 진다. 다음의 그래프는 메세지 용량과 생산성 사이의 관계를 나타내고 있다. 초기에는 사람들이 필요한 정보를 빠르게 받아볼 수 있으므로 생산성에 좋은 영향을 미치게 되지만, 정보의 양이 너무 많아지고 복잡해짐에 따라 결국 생산성의 체감점에 도달하게 되고, 그 후부터는 생산성에의 악 영향을 나타내고 있다. 이러한 현상의 원인으로는 우리가 필요로 하는 정보와 우리가 필요할 것이라고 다른 사람들이 생각하는 정보들이 서로 섞여 버리기 때문인 것이다.



(그림 1) 정보 용량의 증가와 생산성 사이의 관계

결론적으로, E-mail은 일대일 또는 일대다 통신 형태에는 효과적인 매체이다. 그러나 "push" 모델에 의한 비조직적인 통신 형태로는 점점 증가하는 정보의 양을 관리할 수가 없기 때문에 다대다 통신의 경우에는 많은 제약점을 가지게 된다.

이러한 문제점들을 해결하기 위한 기술을 개발하는데 있어 정보의 분배(delivery)와 정보의 관리(management)를 구별할 필요가 있다. 메세징은 축적전송 방식을 사용함으로써 정보의 분배에는 효과적이다. 그러나 정보의 관리 문제를 해결하기 위해서는 다른 테크놀로지를 생각해보아야 한다.

III. Collaboration

연계(Collaboration)는 공유(shared) space에 의존한다. 즉 사무실, 칠판, 네프킨 또는 온라

인 공유 장소 등이 모두 공유 스페이스에 해당될 수 있다. 이것은 연계 행위의 touchstone으로서 기여하며, 사람 사이의 관계에 내재되어 있는 모호성을 제거할수 있는 매체로써 아주 중요한 역할을 한다. 사실 이 공유 스페이스는 상호관계를 통한 전체의 융합이 각 개인의 기술을 합한 것보다 더 강하다는 사실을 뒷받침해 주는 연계적 도구(collaborative tools)인 것이다.

연계는 두 사람사이에서 이루어질 수도 있고, 다대다(many-to-many) 형태의 정보 공유 형태를 취할 수도 있다. Problem solving, 브레인스토밍(brainstorming), 다른 사람이 만들어 낸 데이타를 식별해 내고 찾아내는 행위 등이 모두 연계 행위이다.

통신에서와 마찬가지로, 연계 측면에서의 가장 중요한 기술적 공헌은 바로 시간과 공간의 제약을 제거한 것이다. 대면(face-to-face) 미팅은 그룹의 멤버들이 같은 때 같은 장소에 있을 때 만 가능하다. 전화는 공간적 제약을 해소시켰고 음성 메일(voice mail)은 시간의 제약까지도 제거시켰다. 사실 경험적으로 보건대 그룹 멤버들이 연계적 기술을 그들의 환경에 도입하기만 하면 그들간에 정보 및 아이디어를 교환하기 위해 필요한 대면 미팅의 회수를 아주 효과적으로 줄일 수 있다. 어떤 논점에 수렴하기 위해서나 의견 일치를 필요로 할 때에만 대면 미팅이 필요할 것이다.

앞 장에서 메세징은 편리한 범용의 통신 매체이며 mailing-list를 이용하면 그룹간에도 사용이 가능하나, 다대다 통신의 경우는 아직 미흡한 점이 많이 있다고 지적한 바 있다. 그러나 메세징의 이러한 범용성과 비동기성(asynchronous na ture) 때문에 메세징은 연계의 매체로서 사용될 수 있게 된다.

메세징 시스템은 근본적으로 화일을 송신자와 수신자 사이에서 메세지로서 전달하는 체계이다. 따라서 사용자들이 정보를 주제별로 추적하기란 쉽지가 않다. 더우기 누가 어떤 메세지에 대해 어떤 순서로 대답했는지를 추적하기가 어렵기 때문에, 일련의 e-mail 메세지를 통해 이루어지는 토의(discussion)에 있어 그 맥락을 재대로 따라가며 관리하기는 쉽지가 않다. 이러한 여러 이유에서 e-mail 시스템은 온라인 공유 스페이스를 지원하도록 확장될 필요가 생긴 것이다.

메세징 시스템은 push 모델을 사용하는 반면에 공유 데이타베이스 기술은 정보 공유에 있어 pull 모델을 지원한다. Pull 모델은 사용자로 하여금 필요할 때마다 정보를 검색해 낼 수 있도록 한다. 사용자들은 스스로가 다양한 그룹 토의에 대해 언제 가담할 것인지를 잘 조절해야 한다. 사용자들은 이제 더이상 e-mail 스케쥴에 따라가야만 하는 것이 아니라 그들 자유자재로 정보를 검색해 낼 (또는 무시할 수 있는) 책임이 부여된 것이다.

공유 데이타베이스는 전체 메세지를 저장하고 있을 뿐만 아니라, 토의 아이템이나 도움될 만한 도큐먼트, 정보 등을 한 곳에 모아 놓았으며, 이들은 공통된 구조를 가지고 있어 보기에 편하고, 도중에 발생되는 일들을 일관성있게 기록해 놓는다는 점에서 메세징 시스템과는 다르다. 이를 통해 연계 작업자들은 상호이해를 쉽게 할 수 있고, 이들이 key concept와 issue를 포착하는데 도움이 된다. 더우기 공유된 view가 제공되는 곳에서는 여러 형태의

프리젠테이션이 가능하여 각 개인이 자신의 특정 작업에 맞는 view를 제공받을 수가 있다. 이러한 다중 표현성(multiple representation)은 각 개인에게 정보를 여러 측면으로, 즉, 날짜별로, 저자별로, 도큐먼트 양식별로 등등 살펴볼 수 있도록 해준다. 또한 정보를 여러 방면의 사람들에게, 예를 들어, 마케팅 그룹의 사람들에게는 고객이름이나 마켓 이름으로, 엔지니어링 부서의 사람들에게는 부서 번호나 제품 이름 등으로, 서로 다르게 보여질 수 있도록 지원해 준다. 각각의 표현양식은 서로 다른 렌즈를 가진 것이며, 이 렌즈를 통해 여러 상황에서 정보를 다루는 연계적 작업을 바라보게 되는 것이다. 이러한 이유로, 그리고 작업들은 시간에 따라 자꾸 변하기 때문에, end-user 용 설계 도구(design tool)를 사용해야 할 필요성이 생긴다. 이를 사용하면 연계 작업자들은 정보를 자신의 필요에 맞게 양식화할 수 있고 수정할 수도 있는 반면, 그렇지 않으면 해당 시스템은 점점 도태되고 사용되지 않게 되는 것이다.

설계 도구를 사용할 수 있느냐, 그리고 이를 통해 application들이 쉽게 개발되고 필요에 따라 양식화될 수 있느냐에 따라 시스템 개발의 장기적 성공여부가 결정된다. 시스템에 관련된 많은 변수를 정의할 수 있는 능력은 물론, 사용자 인터페이스를 제작할 수 있는 능력, 그리고 데이타와 정보를 어떤 유형으로 보여줄 것이며 어떻게 다룰것인가 등이 성공여부의 중요한 요인이 된다. 하부의 데이타베이스 플랫폼을 액세스하는 기능이 제공되지 않는 application을 산출해내는 제품들은 (예를 들어, group conferencing) 이러한 측면에서 한계가 있게 된다. 이러한 시스템들은 단기적인 상황에서는 사용가능할 지 모르나, 광범위한 연계적 문제를 해결하기에는 융통성이나 양식화 기술등이 결여되어 있다. 하부의 데이타베이스 플랫폼이 노출되어 있고 필요에 따라 양식화가 가능한 도구 및 application을 제공하는 제품들은 훨씬 더 융통성있게 사용될 것이다. 예를 들어 설명하자면, 위의 예에서 마케팅 그룹이 고객의 정보를 분석하는데 있어 마켓 이름은 물론 지리적인 정보가 필요하다는 것을 깨달았다고 하자. 이를 실현하려면, 지역에 대한 정보를 나타내는 새로운 필드가 기존의 데이타베이스에 추가되어야 할것이다. 하부의 데이타베이스를 액세스할 수 있음으로써 필드에서 발견된 추가 정보들이 쉽게 수정될 수 있는 그러한 연계적 tool 만이 이러한 그룹의 요구사항들을 지원할 수 있는 것이다.

온라인 정보를 저장하고 관리하는 공유 데이타베이스 모델은 메세징 모델에 비해 여러 가지 장점을 가지고 있다. 정보는 수요자의 필요에 따라 끌어다 쓰면 된다. 그러나 이 기술이 어떤 통고(noti fi ca tion)나 메세징의 지원이 없이 사용된다면 이것도 한계가 있다. 공유 데이타베이스는 그 정보가 사용되는지의 여부가 오로지 정보 수요자에게 달려 있기 때문에 아주 소극적인 성향을 띠게 된다는 점이다. 공유 데이타베이스에서 빠진 기능은 바로, 정보의 추가나 수정이 발생했을 때 이 사실을 사용자에게 즉시 통고해 줄 수 있는 기능이다.

E-mail에서 정보의 홍수가 일어나듯, 수백 수천개의 공유 데이타베이스의 증가 또한 사용자에게 어떤 마비 현상을 초래할 수 있다. 비슷한 현상이 매달 수천개의 home page들이 등장하는 Wor ld Wide Web에서도 발생할 수 있다. 결과적으로 굉장히 많은 양의 정보가 데이타베이스에 저장되게 되며, 반면 중요하거나

자주 변경되는 정보등이 이에 관심있는 사용자에게 제때에 잘 전달될지는 보장할 수가 없는 것이다.

결론적으로 말하면, Collaboration과 Communication의 요구사항은 서로 다르므로 전자적 메세징 만으로는 연계적 작업들을 충분히 수행해 낼 수 없다. 데이타베이스 기술이 정보의 분배에 있어 "pull" 모델을 사용함으로써 연계적 작업을 가능하게 하는 것이다.

Collaboration은 이러한 push 및 pull 모델을 결합한 시스템을 요구한다. 이들을 결합하는 한 방법으로는 메세징과 공유 데이타베이스 기술을 적절히 조절하여(coordinated) 사용할 수 있도록 지원해 주는 tool을 사용하는 것이다.

IV. Coordination

지금까지 우리는 그룹 멤버들이 그들의 일을 좀 더 효율적으로 수행하기 위해 도움이 되는 정보들을 공유하기 위해 어떻게 통신하고 연계하는지를 살펴보았다. 이러한 방법에 있어 특징적인 성질은 바로 임기응변적인 비구조적 방법이라는 것이다. 즉, 사람들은 e-mail 메세지를 그들 마음대로 전송하고, 그들이 필요로 할 때마다 공유 자원들을 참조한다.

그러나 기업에서의 많은 작업들은 훨씬 구조적이다. 예를 들어 기업은 사람들이 지출보고서를 가지고 프로세싱 하는데에 있어 "연계적"으로 작업하기를 기대하기 보다는, 지출보고서가 조직내에서 어떤 경로로 전송되어서 승인되고, 감사 및 보증 받을지를 미리 정의해 놓고 있다. 많은 사람들이 관여되어 있지만, 기업의 이러한 정책은 정해진 목표에 도달하기 위해 사람들 사이에 "조정(co or di nation)"이 이루어질 것을 명시하고 있다. 미리 정해진 이러한 프로세스에 있어 이런 업무가 성공적으로 완성되는가 아닌가는 정해진 시간내에 일련의 구조적 작업들을 특정한 순서대로 완료할 수 있도록 사람들을 어떻게 조정하는가에 달려있다. 이것은 어느정도 업무흐름 자동화(workflow automation) 시스템의 영역에 해당한다. 시스템의 관점에서 collaboration은 매우 소극적인 반면(공동의 workspace를 만들기는 하지만 그 스페이스의 사용방법에 대해서는 언급하지 않는다), coordination은 매우 적극적인 방법이다(작업이 어떻게 진행될 것인가를 명시해 준다).

어떤 특정 문제에 대해 collaboration이 아닌 coordination에 의한 해결을 원할 경우, 필요한 양식(form)들을 정의하고, 그 양식에 대해 수행되어질 기능들을 명시하고, 그 양식의 전달 방식(rou ting)에 대한 법칙을 명시하고, 외부의 데이타가 어떻게 엑세스되고 수정될 것인가, 어떤 상황이 되면 어떤 일이 일어날 수 있는가 등을 명시하는 workflow 시스템을 개발하여야 한다. 즉, workflow application을 개발하는 것이며, 이를 위하여 application 개발 툴(tool)이 필요해진다. 즉 Coordination은, 일반적으로 workflow라 불리워지는 application들에 대한 개발 툴의 사용을 의미한다.

Workflow application에 있어 조직화, 구조화하는 것을 중요시 하는 반면, 미리 정의되지 않은 작업에 관련된 상당 부분은 간과해버리게 된다. 사실, 대부분의 실제 업무는 매우 구조화된 작업과, 그렇지 않은 작업, 즉 프로세스 자체가 fuzzy하고 규칙이나 전달방향, 역할 등이 일이 진행됨에 따라 유동적으로 정의되

는 그러한 작업들로 섞여 있게 된다. 이것이 바로 workflow 시스템이 "soft"한 교류를 지원하는 collaborative, communication 작업이 없이는 그 단독으로는 성공하지 못하고 "너무 경직되었다"고 여겨지는 이유인 것이다.

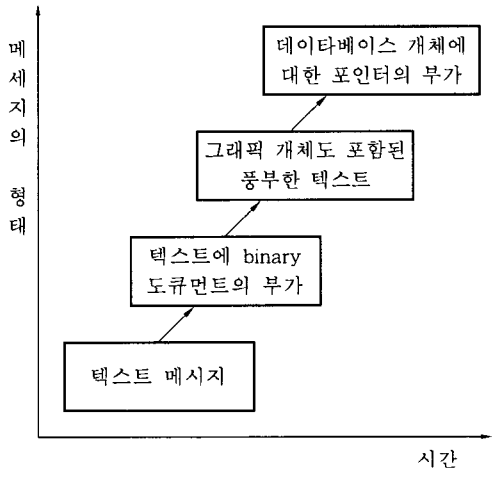
예를 들어 소프트웨어 bug 교정 어플리케이션의 경우를 고려하자. 이것은 다음과 같은 구조적인 과정을 포함한다: 초기의 bug 기재 과정, 프로젝트 매니저에게 전달 과정, 프로그래머/분석가에게 전달, 품질 보증단(quality assurance)에게 전달, 구성 관리(configuration management) 전문가에게 전달, 그리고 고객에게 다운로딩하기 위하여 공동 참조 라이브러리(예를 들어, World Wide Web 또는 게시판)에 우송하는 과정 등이다. 그러나 이러한 과정 중에 다음과 같이 미리 예측할 수 없는 또는 자동화되지 못하는 비구조적인 작업이 여럿 존재할 수도 있다: 유사한 bug를 발견하거나 교정하기 위하여 trouble tracking database의 도움을 받거나, 여러 부서에서 더 많은 정보를 얻기 위하여 e-mail 문답을 사용하는 작업등이 그것이다.

Coordination은 사람들에게 필요에 따라 작업을 부여하고 해제함으로써 일련의 구조적인 작업들을 자동화하는, 그런 과정 이상의 것이다. 실제로 작업이 어떻게 진행되는가를 지켜보면, 어떤 과정을 완성하는데에 기본적으로 필요한 정보는 그 과정 자체에서가 아니라 그 외부에서, 여러 부서 사이의 관계를 통해서 얻어진다는 사실을 알 수 있는 것이다. 완벽한 coordination은, 그 시스템이 사람들로 하여금 그들의 작업을 완수하는데 있어 필요로 하는 정보를 얻을 수 있도록 도와주는 비공식적인 대화(e-mail을 통한, 또는 토의 데이타베이스나 reference publishing 시스템을 이용한)까지도 지원해 줄때 가능한 것이다.

앞 장에서 우리는 정보를 각각 push하고 pull하는 형태에 기반을 둔 communication과 collaboration 모델이 각각 개별적으로 사용될 때에 얼마나 그 기능이 불충분한가를 살펴보았다. 이들을 통합한 솔류션, 즉 어떤 사실을 통고(notification)하기 위해서는 메세징을, 그리고 collaboration을 위해서는 공유 데이타베이스를 함께 사용하도록 잘 조정(coordination)하는 기술은 구조화된 작업 및 비구조화된 작업을 모두 지원해 주기 위한 좀더 균형있고 포괄적인 접근방법이 될 것이다.

제 4세대 메세징 시스템은 바로 메세징과 공유 데이타베이스가 잘 조정되어 사용되는 그런 시스템이다. 메세징 시스템의 발전과정을 처음부터 살펴보면, 제 1세대 시스템은 단순히 텍스트 메세지만을 전달할 수 있었고, 제 2세대 시스템은 이 텍스트 메세지에 binary 도큐먼트를 부가하여 전달할 수 있었다. 제 3세대 시스템은 이들 메세지 및 binary 도큐먼트에 부가하여 풍부한 텍스트(예를 들어 color 색상, 여러 폰트, 다양한 문자의 크기 등)와 개체의 삽입까지도 지원해 준다. 마지막으로 제 4세대 시스템은 도큐먼트 내에 공유 데이타베이스와 화일 시스템에 대한 하이퍼텍스트 링크까지도 지원해 줌으로써 메세징에 있어 심오한 진보를 보이고 있다. 메세지에 어떤 개체를 부가하여 전달하는 대신, 공유 데이타베이스에 있는 그 개체에 대한 전자적 포인터("doclink")만을 메세지에 부가하여 전달한다. 사용자가 그 doclink를 더블 클릭하면, 그 개체는 자동적으로 곧바로 검색되어 사용자에게 나타난다. 메세징 측면에서는 사용자로 하여금 메세지에 개체들을 첨부하여 보내지 않아도 되도록 했

기 때문에 진보된 것이며, 공유 데이타베이스 측면에서도 또한 중요한 정보가 새로 발표되었다는 것을 상대방에게 알릴 수 있는 수단이 생겼다는 점에서 진보된 것이다. 우리는 이것을 메세징과 그룹웨어의 통합(inte gra tion)이라 부른다.



(그림 2) 메세징의 4세대

예를 들어 다음과 같은 작업을 고려해 보자. 광고제작팀의 관리자가 제작된 광고의 드래프트(텍스트와 그림으로 구성된)를 정기적으로 편집자들에게 수정을 위해 전송한다고 가정하자. 각각의 광고에 대해 관리자는 이들을 수정팀에게 배부하기 위해 e-mail을 사용할 것이다. 이들은 e-mail 메세지에 binary 정보가 부가되어 전달된다. 여러 그룹 멤버로부터의 응답 - 그 광고에 대한 실제 수정본이거나 제안 또는 코멘트 - 이 전달되면서부터는 버전 컨트롤 문제나 도큐먼트 관리 문제가 대두된다. 동시에 커다란 메세지들이 이쪽 저쪽으로 push됨으로 인해 네트워크의 트래픽이 엄청나게 증가되는 결과가 초래된다.

반면에 공유 데이타베이스를 사용한다면 이 관리자는 많은 문제점들을 피해갈 수 있게 된다. 관리자와 수정팀은 가장 최근 버전의 도큐먼트만을 참고하게 될 것이며, 그동안 여러가지 수정을 유도했던 논의의 내용들을 따라가며 추적할 수 있을 것이다. 한편 이 솔류션은 새로운 광고가 데이타베이스에 기록되어 수정을 요할 경우에 이 사실을 알려줄수 있는 방법은 결여되어 있다.

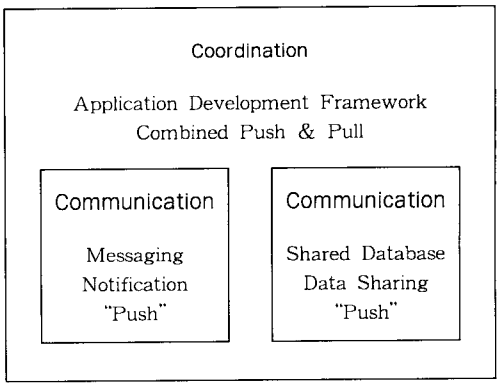
메세징과 공유 데이타베이스를 잘 조절하여 사용하는 제 4세대 e-mail 시스템은 이들 두 문제를 모두 해결하고 있다. 광고팀의 관리자가 새로운 광고를 공유 데이타베이스에 보낼 때, 그 도큐먼트에 대한 하이퍼텍스트 링크, 즉, "포인터"를 포함하는 e-mail 메세지를 함께 생성하는 것이다. E-mail은 수정팀에게 새 광고의 존재를 알려줄 뿐, 그 광고가 실제적으로 메세지에 부가되어 전달되는 것은 아니다. 이 방법은 광고물들이 중앙집중적으로 관리되고 유지될 수 있도록 함으로써 버전 콘트롤을 용이하게 하며 공유 스페이스에 대한 collaboration을 지원해 주는 것이다.

Workflow automation에 대한 예를 하나 더 살펴보자. 어떤 매매 계약서에 대해 고객과 협상하려 할 때, 이 작업을 자동화하는 과정에 대해 고려해보자. 우선 이 계약서가 승인되기 위해, 회사 내부에서 몇몇 사람들에게 서명받아야 한다. Mail에 기반을 둔 workflow 시스템이라면, 이 계약서를 각 사람들에게 차례로 전송하여 승인 또는 거부를 또는 코멘트를 요청할 것이다. 만일 고객이 이 계약서의 현재 승인 상태를 알고 싶다거나, 또는 현재 승인이 진행되는 도중에 계약서의 내용을 일부 수정하려 한다고 가정해보자. 현 시점에 누가 그

계약서를 가지고 있는지 알 수 있겠는가? 어떤 시점에서 계약서를 수정해야 겠는가? 도중에 수정이 될 경우 이 승인 체계에 어떤 일이 발생하겠는가?

두번째로 공유 데이타베이스를 이용하는 workflow 모델을 고려해보자. 이 모델에서는 각 사용자가 tracking 데이타베이스를 참조함으로써 이 보고서의 현재 상태를 체크한다. 이 경우 앞에서 언급되었던 mail에 기초한 모델에서의 단점을 극복할 수 있다. 즉, 데이타베이스가 중앙집중적으로 관리될 수 있음으로써 workflow가 진행되는 도중에도 다른 사람에게 이 도큐먼트의 현재 상황을 보여줄 수 있으며, 이것에 대한 관리가 용이하게 된다. 그러나 이 모델의 단점으로는 사용자들에 대한 event-driven notification이 없다는 것이다. 즉, 데이타베이스를 체크해야 할 책임은 오로지 사용자에게 있다는 점이다.

이들 두가지가 통합된 모델을 고려해보자. 매매 계약서가 작성되면 공유 데이타베이스에 저장된다. 이것이 저장되는 순간, mail 메세지가 첫번째 승인자에게 날라간다. 이 메세지는 계약서 자체가 아니며 더우기 이 계약서가 어디에 저장되어 있다는 정보를 전달해 주는 것도 아니다. 이 메세지는 단지 그 계약서에 대한 하이퍼텍스트 링크만을 가지고 있어서, 이것이 활성화되면, 공유 데이타베이스에 있는 계약서의 내용이 승인자에게 나타날 것이다. 승인과 전달과정이 계속 진행될 동안, 가장 최근 버젼의 계약서가 항상 이 데이타베이스로부터 검색 가능하며, 현 상황에 대한 문의사항에 대해서도 이 데이타베이스를 엑세스함으로써 답변할 수 있는 것이다.



(그림 3) 그룹웨어의 Building Block

V. 결 론

지금까지 그룹웨어의 중요한 세 기술 - communication, collaboration, 그리고, coordination - 에 대해 살펴보았다. 결국 메세징 서비스와 공유 데이타베이스 시스템 서비스를 모두 지원하는 통합된 application 개발 환경하에서 폭넓은 그룹업무의 자동화가 실현될 수 있으며, 이러한 환경이 바로 그룹웨어 시스템 환경인 것이다.

□ 著者紹介

최 윤 경(崔 允 景)

1982년 3월 - 1986년 2월 서울대학교 계산통계학과 학사
1986년 3월 - 1988년 2월 한국과학기술원 전산학과 석사
1988년 3월 - 1994년 2월 한국과학기술원 전산학과 박사수료
1990년 3월 - 1993년 2월 부산공업대학 전자계산학과 전임강사
1993년 9월 - 현재 배화여자전문대학 사무자동화과 전임강사

※ 관심 분야 : Groupware