Vol. 8-3669

ICEMaker™: An Excel-Based Environment for Collaborative Design

Kevin L.G. Parkin , Joel C. Sercel , Michael J. Liu , Daniel P. Thunnissen
Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125
(626) 395-4785
hal9000@caltech.edu, sercel@caltech.edu, mikej_liu@hotmail.com, dthunnis@caltech.edu

Abstract—The creative process of team design can be rapid and powerful when focused, yet complex designs, such as spacecraft, can slow and quench the essential elements of this process. Concurrent Engineering techniques partially address this problem, but a fuller realization of their benefits require an approach centering on the human aspects of teamwork. ICEMaker™ is a Microsoft Excel® based software tool that facilitates closer-to-ideal collaboration within teams employing the new Integrated Concurrent Engineering (ICE) methodology. ICE is a generic approach that emphasizes focused collaborative design in a single-room context, and is now employed at several aerospace organizations to increase the productivity of design teams defining complex early development-phase products. By way of introduction, this paper describes the basic elements of ICE needed to understand ICEMaker and its application. We present the design approach, philosophy, and client-server architecture of the ICEMaker system, as well as a simplified user scenario. NASA's Jet Propulsion Laboratory (JPL) has recently adopted ICEMaker for its primary early-phase space mission and system advanced project design team, Team-X. We describe Team-X's experience with ICEMaker and report on the lessons learned, and qualitative product improvements, resulting from JPL's implementation of ICEMaker.

TABLE OF CONTENTS

[1] 0-7803-7651-X/03/$17.00 © 2003 IEEE
[2] IEEEAC paper #1564, Updated January 25, 2003

1. INTRODUCTION

The creative process of team design can be rapid and powerful when focused, yet complex designs such as spacecraft can slow and quench the essential elements of this process. Long-winded calculations and finding data become rate-determining steps for evolving and exploring design alternatives; a half-day roundtable event becomes a multi-week hierarchy of design meetings delayed by relatively isolated progression of the slowest design elements. Concurrent Engineering techniques partially address these problems, but the full potential of computers to automate calculation and centralize databases has yet to be realized because computer usage tends to disperse teams back to their offices and discourages the unimpeded verbal communication, social context, and collective judgments that make focused single-room collaborations so powerful.

ICEMaker is a software system designed to increase the productivity of multidisciplinary technical teams engaged in the design or analysis of complex products [1]. Specifically, ICEMaker has been designed to ease the transition from traditional collaborative design and analysis methods to new, more productive, Integrated Concurrent Engineering (ICE) methods. Two key features of ICE are critical to understanding ICEMaker as they dictate important requirements that ICEMaker was developed to support:

i. Team collaboration, including design and analysis, is done in real-time sessions in which conversation and quantitative computer-aided engineering are concurrent. This is in contrast to traditional methods in which teams separate meetings from quantitative engineering in a meet-work-meet mode of collaboration.

ii. Computer tools on separate team member workstations are linked together to allow the quantitative engineering process to keep pace with qualitative discussions.

Hence, ICEMaker is a system that ties computer-based design models or computer-aided design (CAD) tools together in such a way that engineers can use their tools to do quantitative analysis and design in real-time, during collaborative sessions. This real-time collaborative process is at the heart of ICE.

2. THE ICE PARADIGM

Integrated Concurrent Engineering (ICE) is a real-time collaborative process in which a multidisciplinary team discusses a design or analysis problem while concurrently conducting quantitative, computer-based calculations. Documented examples of ICE systems include TRW's Integrated Concept Design Facility (ICDF) [2], JPL's Project Design Center [3], Boeing's Concurrent Integrated Engineering Laboratory (CIEL) [4], and UTC's Integrated Concurrent Engineering Center (ICEC) [5]. In this paper we will review the basic elements of the ICE concept needed to understand the design philosophy and purpose of ICEMaker. Another paper by one of the authors (Sercel, [6]) provides a more complete description of the ICE method, gives the reader more information on how to rapidly deploy ICE, and discusses a recent case study of its application in industry.

Table 1. The Five ICE Principles Checklist

1. A well-defined set of *Standard Information Products*
2. *Network-Linked Tools*
   - Adapted to eliminate manual reformatting of inputs and outputs
   - Facilitate nearly instant quantitative engineering
3. Well-understood *Procedures for Real-time Collaboration*
   - *Concurrent* quantitative engineering and qualitative conversation
4. A standing *Multidisciplinary Team* skilled in the tools and methods
5. A *Facility* supporting the hardware, software, and human resources

By way of background, a methodology which implements ICE includes a checklist of Five ICE Principles, listed in Table 1, that define the ICE methodology and provide an objective way to assess the degree to which an organization is using ICE.

Principle 1–Standard Information Products. The first principle of ICE is to recognize that the work product of technical professionals, including engineers and scientists, is information. The job of such technical professionals can be understood as converting one type of information, for example requirements, into another type of information, for example design specifications. Organizations made up of technical professionals can be viewed as possessing competencies, whereby there exist classes of information that the professionals within the organization can predictably and consistently produce. We refer to these classes of information as *Standard Information Products.* Organizations possess unique competencies reflecting the expertise of their staff and hence possess unique sets of Standard Information Products that they can reliably produce. Examples of Standard Information Products in the domain of spacecraft systems and missions include telecommunications link budgets, spacecraft thermal analyses, structural designs, and mission delta-V estimates. Organizations working in other fields have completely different Standard Information Products.

An important criterion for ICE is that an organization systematically identifies and defines its Standard Information Products. Without defining a standard set of products, it is impossible to systematize the process and gain productivity benefits from the application of ICE. By contrast, once an organization has carefully defined a set of standard information products, it has taken the first step in defining a systematic process to best capitalize on its human and computational resources and increase productivity. This is the first step in moving from an ad-hoc process, and is in some ways analogous to the transition of hardware production from pre-industrial cottage industries, to systematic industrial process during the industrial revolution. In our case, the product is not hardware, but complex information products.

Principle 2–Network-Linked Tools. The vast majority of engineers and scientists use quantitative design or analysis tools in their day-to-day work. These tools include mathematical models, computer-aided design (CAD) systems, databases, productivity applications such as spreadsheets and mathematics packages, and many other types of software. Such software increases individual user productivity but frequently induces severe productivity bottlenecks at the points of input and output. These bottlenecks come in many forms but include tool-specific data formatting needs, extensive graphical user interface (GUI) based tool setup requirements, or data translation needs in which design parameters developed by other team members must be manually translated into a different technical language through simple but time-consuming mathematical operations. Fortunately, most of these bottlenecks are easily overcome using modern software systems that make use of standard application programming interfaces (APIs), file wrappers, and scripting languages.

ICEMaker is an example of a particularly simple such software system.

Principle 3–Procedures for Real-time Collaboration. ICE methods represent a very different way of working for technical teams. For those used to working in traditional ways, these changes can be a challenging transition. It is not obvious to most engineers how to conduct a collaboration session in such a way that the quantitative analysis is properly interspersed with qualitative discussion, or how to regulate the whole activity for maximum productivity. In addition, practical experience with ICE methods suggests that team members need to intersperse time in design sessions with time between design sessions. Finally, the ICE process has been likened to an intensive intellectual activity with similarities to those of medical operating rooms, aircraft control rooms, or spacecraft operational control centers. As with these other activities it is critical that all team members have distinct roles and know certain practiced procedures.

Principle 4–Standing Multidisciplinary Team. Given the need for clear team procedures and roles, integrated, tailored software, and well-defined Standard Information Products, it should be clear that an engineer, analyst, or scientist cannot simply step into an ICE environment and work productively on day one. Training or experience is required with the specific software tools, team procedures, and even to understand what the team is there to produce. Sercel has found that the team should be a standing group well-trained with a common perspective and approach to provide maximum productivity.

Principle 5–Available, Applicable, Facility. The team must have a place to work that provides the appropriate environment including a conference table, networked computer workstations, and a projection station capable of showing work results. This facility can be either real or virtual, as experiments and practical experience show that ICE methods can be applied remotely with the proper video conferencing system. It has also been shown that team members trying to apply ICE methods in their own offices almost inevitably fail to produce repeatable, measurable productivity gains. The significance of the facility is that it facilitates team building and team dynamics in ways that separate offices or cubicles cannot.

The ICEMaker Design Philosophy

ICEMaker is designed as a simple way to implement ICE by allowing users to easily link engineering parameters between rule-based design tools such as those instantiated in spreadsheets or parametric CAD systems. ICEMaker aids the rapid application of ICE principles to new or existing projects, fulfilling the criteria in Table 2, resulting in a quick transition and robust basis for subsequent use with complex designs.

The reader should be keenly aware that ICEMaker and the ICE process are not synonymous. The ICE process can use any of several commercial software tool-linking and database-sharing software systems. However, whatever system is chosen should support ICE concepts and therefore must meet the simple set of requirements outlined in Table 2. ICEMaker is particularly well suited to early-stage conceptual-level design processes in which design models are instantiated in spreadsheets or other tools that work with tab-delimited text files. ICE can also be implemented for later-stage design work that involves more complex data-linking and database-sharing systems.

Table 2. ICE Process Requirements Placed on ICEMaker

1. Easy ties to ubiquitous and simple-to-use engineering tools such as Microsoft Excel
2. Provide a shared database that any team member can browse, search, and link to design parameters
3. Allow team members to easily send and receive design parameters directly from models or CAD tools
4. Provide an open architecture to allow other tools such as CAD systems to tie into a shared database
5. Provide a fast send-and-request process for outputting to and inputting from the shared database to/from design models or tools
6. Support but do not limit the team to a rational engineering parameter naming convention

3. The ICEMaker Software

This section describes the philosophies that guided the development of ICEMaker into a client-server system, and put into context the roles that ICEMaker and its functionality play in the design process.

Design Objectives and Approach

The ICEMaker design objectives arose from the authors' observation of the strengths and weaknesses of previous in-house software used for ICE collaboration at Caltech and JPL. Fundamentally, it was felt that the software should facilitate the digital side of ICE collaboration as invisibly as possible. Partly to avoid the drawbacks of the previous software, ICEMaker was written to:

1. Facilitate the team process of decomposing a design problem into modules. Hereon we will refer to modules as 'design subsystems', or simply 'subsystems'

2. Embody the design calculations, databases, and knowledge of a subsystem within the software in a way that can easily be written, accumulated, stored, adapted from earlier projects, and used as a tool

3. Integrate the team process of assigning tasks, design, and data responsibilities with the software process of linking modules together via the data Inputs and Outputs for each subsystem

4. Facilitate quick, easy, traceable, and understandable data exchange between subsystems

5. Provide a self-explanatory starting point to begin navigating each subsystem, for the benefit of those unfamiliar with it

6. Leave human-suited tasks to humans by refraining from automating away processes that should incorporate the personal values of the user

To fulfill these objectives a Microsoft Excel based approach was chosen, whereby a design subsystem is encompassed within a workbook file. Microsoft Excel is understood, used by, and on the desktop of most engineers. In addition, it has its own scripting language, Visual Basic for Applications (VBA), which is versatile and widely used. The spreadsheet approach is ideal for 'quick and dirty' calculations as well as more involved analysis using the built-in scientific functions and Analysis ToolPaks. For more involved calculations, Excel has commercially available add-ons to interface with scientific applications including Mathematica®, Matlab®, and LabView™. Other software, such as Visual Basic and SolidWorks®, can interface directly with Excel or read Excel workbook files. Lastly, Excel's built-in dynamic link library (DLL) calling routines allow the use of DLLs that are custom-written, for example, in C++ or Fortran.

The Client-Server Architecture and Communications

Modularity is accomplished using Excel workbooks as modules, which we call subsystems. Communication between subsystems is accomplished using a client-server architecture as shown in Figure 1, in which a server mediates data exchange between clients (subsystems). In the original version of ICEMaker, the server was also an Excel workbook, executing macros to perform communication functions; however, this has been superceded in subsequent versions by a much faster standalone Visual Basic application, discussed shortly.

Initially, blank subsystems are generated by the server, possessing just the functionality necessary for exchanging data with the server. Communication is achieved via intermediate workbook files that are exchanged via a common file system. This method has two very appealing features: First, workbooks are easily generated, opened, and saved by VBA commands. Second, the actual data exchange is handled by the operating systems of the computers involved, instead of by a custom-added communication capability for Excel. It is also worth mentioning that the on-screen buttons and VBA code of the subsystems is written in such a way that they function on any platform that runs Excel, and therefore subsystems can communicate cross-platform as long as the computers involved have read/write access to the project folder.
The ICEMaker folder structure, shown in Figure 2, can be located at any level within a file system. The root folder is named after the project (Project Chaos). When multiple revisions of a project are generated a revision number can be added to this folder name (Project Chaos – rev B). The root folder has two subfolders: Client Subsystems and Project Server. The Client Subsystems folder contains all subsystem workbooks for the project, and they are unable to communicate with the server without being in this folder. Though somewhat restrictive, this ensures that an up-to-date collection of all essential project files are maintained centrally, allowing easy archiving of design variations. Within the Client Subsystems folder the Incoming subfolder is used for sending intermediate data workbooks from the server to the clients. The Template subfolder stores a blank

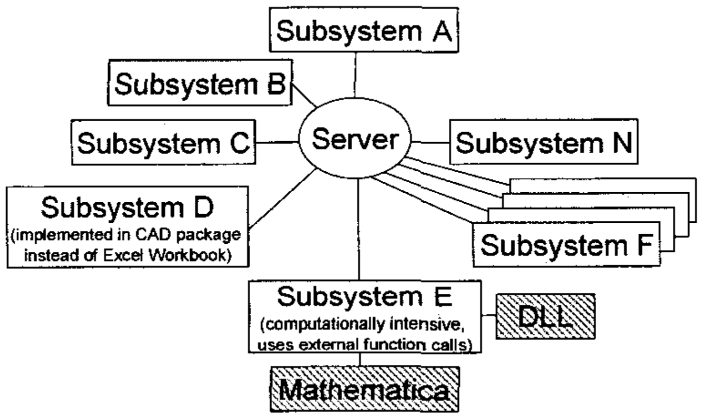

Subsystem A
Subsystem B
Subsystem C
Server
Subsystem N
Subsystem D
(implemented in CAD package instead of Excel Workbook)
Subsystem F
Subsystem E
(computationally intensive, uses external function calls)
DLL
Mathematica

Figure 1 - The Client-Server Architecture

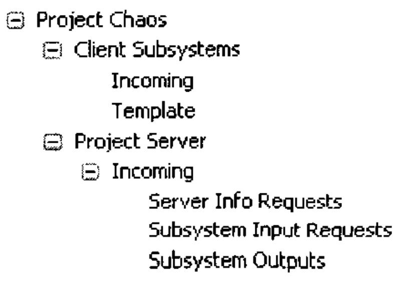
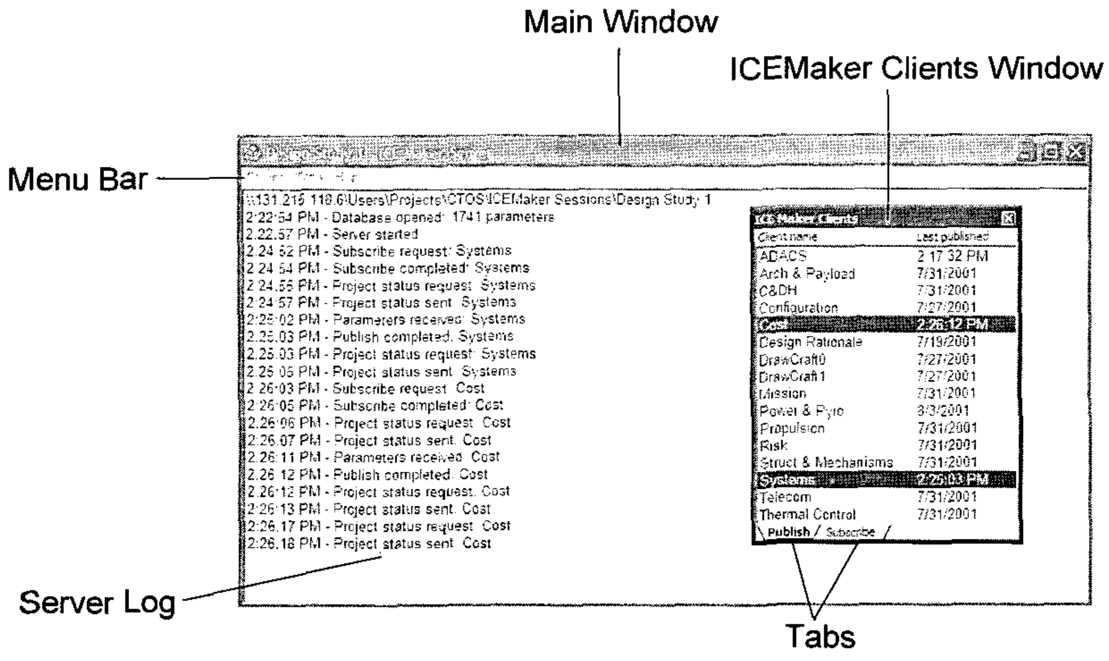

⊟ Project Chaos
  ⊟ Client Subsystems
    Incoming
    Template
  ⊟ Project Server
    ⊟ Incoming
      Server Info Requests
      Subsystem Input Requests
      Subsystem Outputs

Figure 2 - The ICEMaker Project Folder Structure



```
\\131.215.118.6\Users\Projects\CTOS\ICEMaker Sessions\Design Study 1
2:22:54 PM - Database opened: 1741 parameters
2:22:57 PM - Server started
2:24:52 PM - Subscribe request: Systems
2:24:54 PM - Subscribe completed: Systems
2:24:55 PM - Project status request: Systems
2:24:57 PM - Project status sent: Systems
2:25:02 PM - Parameters received: Systems
2:25:03 PM - Publish completed. Systems
2:25:03 PM - Project status request: Systems
2:25:05 PM - Project status sent: Systems
2:26:03 PM - Subscribe request: Cost
2:26:05 PM - Subscribe completed: Cost
2:26:06 PM - Project status request: Cost
2:26:07 PM - Project status sent. Cost
2:26:11 PM - Parameters received: Cost
2:26:12 PM - Publish completed. Cost
2:26:12 PM - Project status request. Cost
2:26:13 PM - Project status sent. Cost
2:26:17 PM - Project status request: Cost
2:26:18 PM - Project status sent: Cost
```

Client name	Last published
ADACS	2:17:32 PM
Arch & Payload	7/31/2001
C&DH	7/31/2001
Configuration	7/27/2001
Cost	2:26:12 PM
Design Rationale	7/19/2001
DrawCraft0	7/27/2001
DrawCraft1	7/27/2001
Mission	7/31/2001
Power & Pyro	8/3/2001
Propulsion	7/31/2001
Risk	7/31/2001
Struct & Mechanisms	7/31/2001
Systems	2:25:03 PM
Telecom	7/31/2001
Thermal Control	7/31/2001

Figure 3 - The ICEMaker Server User Interface

template subsystem workbook used by the server to create new subsystem workbooks. The project server and associated data files are stored in the Project Server folder. The Incoming subfolder has further subfolders that deal with different types of data transfer from the subsystems to the server. Collectively, the Incoming subfolder of the Client Subsystems and Project Server folders handles both directions of client-server communication.

The Server

ICEMaker Server, developed using Visual Basic and Visual C++, is a standalone application that runs on any 32-bit version of Microsoft Windows®. Each running copy of the server is associated with a single ICEMaker project and processes all data sent by that project's client workbooks. An ICEMaker project cannot be associated simultaneously with more than one running copy of the server. The server GUI consists of two primary windows: the main window, titled ICEMaker Server, and the ICEMaker Clients window, both shown in Figure 3. The main window displays a log of server events; warnings and error messages are highlighted in the log to distinguish them from less significant informational messages.

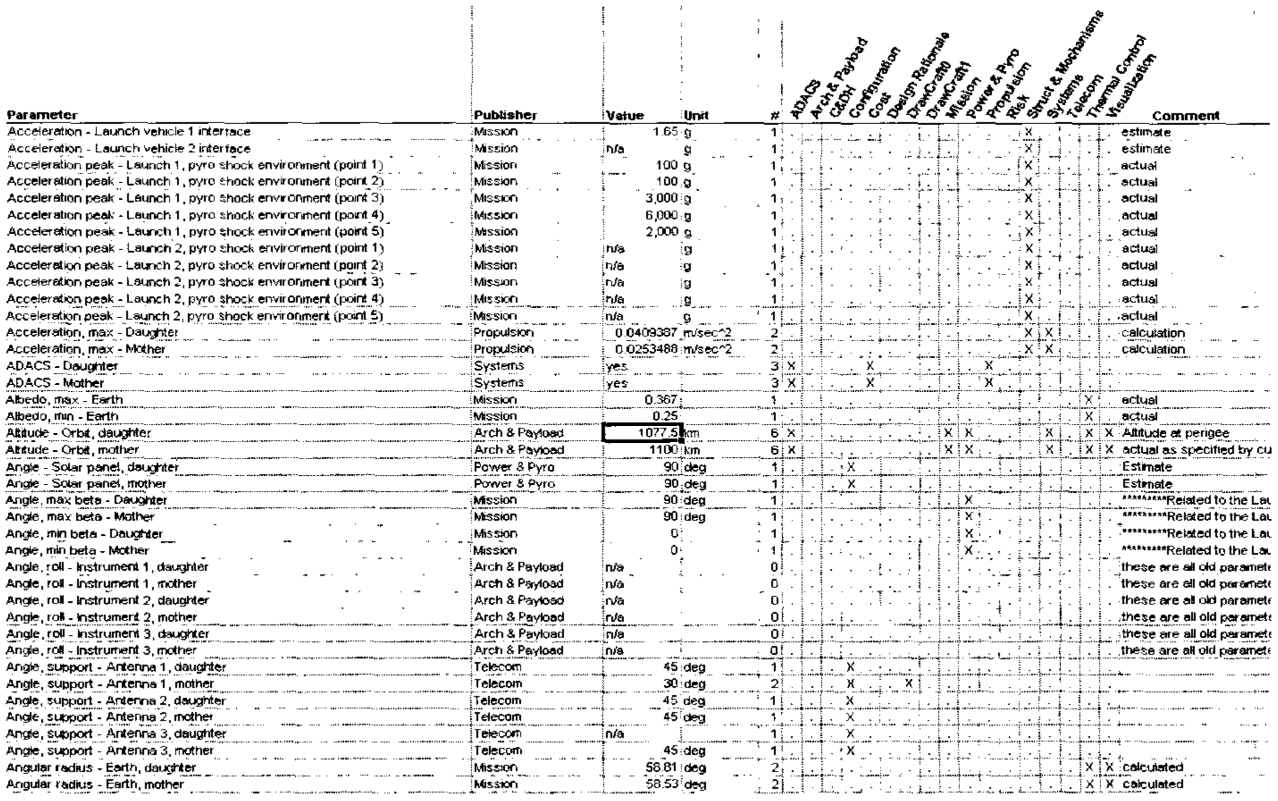
The ICEMaker Clients window, also shown in Figure 3, contains panes labeled Publish and Subscribe displaying the time that each subsystem last sent or requested parameters, respectively. The user switches between panes by clicking the tabs at the bottom of the window. Whenever a subsystem sends or requests parameters, the time shown is updated and the subsystem name is highlighted; highlighting makes it easier for the user to determine which subsystems have recently sent or requested parameters. By right-clicking within the window, the user can access a shortcut menu containing commands to remove all highlighting and to rename or remove a client.

The main window also provides menu commands to create new client workbooks (by copying and customizing a generic "template" workbook for specific subsystems), and to export all parameters to an Excel worksheet or tab-delimited text file. The exported file (Figure 4) lists the values of all parameter fields (Name, Value, Unit, and Comment) and indicates which subsystems, if any, currently send or request each parameter.

The Client

The ICEMaker client encapsulates subsystems as Excel workbooks, capable of communicating with each other via the server. The client workbook is generated by the server, and is 'born' with the necessary worksheets, GUI, and communication routines already in place. The communication routines ⑤,⑥,⑦, shown in Figure 5, are written in VBA, initiated via buttons on the GUI, and indirectly communicate with the server via intermediate worksheet files. The workbook initially consists of four worksheets: *Main*, *Inputs*, *Outputs*, and *Project Status*.

The *Main* sheet ①, in Figure 5, is initially blank and is a starting point from which a user unfamiliar with the subsystem can begin to navigate its calculations. For simple subsystems this might become a summary sheet of main



Parameter	Publisher	Value	Unit	#	ADACS	Arch & Payload	C&DH	Configuration	Cost	Design Rationale	DrawCraft0	DrawCraft1	Mission	Power & Pyro	Propulsion	Risk	Struct & Mechanisms	Systems	Telecom	Thermal Control	Visualization	Comment
Acceleration - Launch vehicle 1 interface	Mission	1.65	g	1													X					estimate
Acceleration - Launch vehicle 2 interface	Mission	n/a	g	1													X					estimate
Acceleration peak - Launch 1, pyro shock environment (point 1)	Mission	100	g	1													X					actual
Acceleration peak - Launch 1, pyro shock environment (point 2)	Mission	100	g	1													X					actual
Acceleration peak - Launch 1, pyro shock environment (point 3)	Mission	3,000	g	1													X					actual
Acceleration peak - Launch 1, pyro shock environment (point 4)	Mission	6,000	g	1													X					actual
Acceleration peak - Launch 1, pyro shock environment (point 5)	Mission	2,000	g	1													X					actual
Acceleration peak - Launch 2, pyro shock environment (point 1)	Mission	n/a	g	1													X					actual
Acceleration peak - Launch 2, pyro shock environment (point 2)	Mission	n/a	g	1													X					actual
Acceleration peak - Launch 2, pyro shock environment (point 3)	Mission	n/a	g	1													X					actual
Acceleration peak - Launch 2, pyro shock environment (point 4)	Mission	n/a	g	1													X					actual
Acceleration peak - Launch 2, pyro shock environment (point 5)	Mission	n/a	g	1													X					actual
Acceleration, max - Daughter	Propulsion	0.0409387	m/sec^2	2													X	X				calculation
Acceleration, max - Mother	Propulsion	0.0253488	m/sec^2	2													X	X				calculation
ADACS - Daughter	Systems	yes		3	X				X						X							
ADACS - Mother	Systems	yes		3	X				X						X							
Albedo, max - Earth	Mission	0.367		1																X		actual
Albedo, min - Earth	Mission	0.25		1																X		actual
Altitude - Orbit, daughter	Arch & Payload	1077.5	km	6	X								X	X				X		X	X	Altitude at perigee
Altitude - Orbit, mother	Arch & Payload	1100	km	6	X								X	X				X		X	X	actual as specified by cu
Angle - Solar panel, daughter	Power & Pyro	90	deg	1				X														Estimate
Angle - Solar panel, mother	Power & Pyro	90	deg	1				X														Estimate
Angle, max beta - Daughter	Mission	90	deg	1										X								*********Related to the Lau
Angle, max beta - Mother	Mission	90	deg	1										X								*********Related to the Lau
Angle, min beta - Daughter	Mission	0		1										X								*********Related to the Lau
Angle, min beta - Mother	Mission	0		1										X								*********Related to the Lau
Angle, roll - Instrument 1, daughter	Arch & Payload	n/a		0																		these are all old paramete
Angle, roll - Instrument 1, mother	Arch & Payload	n/a		0																		these are all old paramete
Angle, roll - Instrument 2, daughter	Arch & Payload	n/a		0																		these are all old paramete
Angle, roll - Instrument 2, mother	Arch & Payload	n/a		0																		these are all old paramete
Angle, roll - Instrument 3, daughter	Arch & Payload	n/a		0																		these are all old paramete
Angle, roll - Instrument 3, mother	Arch & Payload	n/a		0																		these are all old paramete
Angle, support - Antenna 1, daughter	Telecom	45	deg	1				X														
Angle, support - Antenna 1, mother	Telecom	30	deg	2				X			X											
Angle, support - Antenna 2, daughter	Telecom	45	deg	1				X														
Angle, support - Antenna 2, mother	Telecom	45	deg	1				X														
Angle, support - Antenna 3, daughter	Telecom	n/a		1				X														
Angle, support - Antenna 3, mother	Telecom	45	deg	1				X														
Angular radius - Earth, daughter	Mission	58.81	deg	2																X	X	calculated
Angular radius - Earth, mother	Mission	58.53	deg	2																X	X	calculated

Figure 4 - A worksheet produced by the Export All Parameters menu command

calculation results. More involved subsystems might construct this page as an index to other worksheets that contain major sub-calculations or component databases. As subsystem calculations grow, the owner adds extra worksheets as needed for sub-calculations, databases, design knowledge, and so on. The ease of adapting and expanding subsystem calculations with project needs is one of the most powerful aspects of ICEMaker.

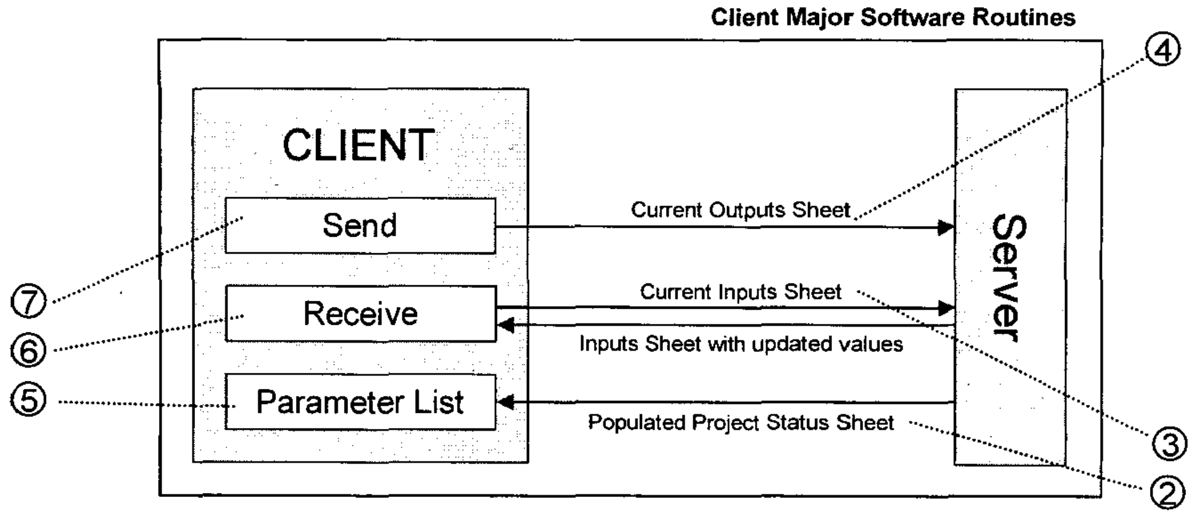
The *Inputs* sheet ② requires a subsystem to explicitly declare data from other subsystems as parameters, and standardizes the location and way in which parameters are declared. It contains columns labeled *From Subsystem*, *Name*, *Value*, *Units*, and *Comments*; hence the data in a row constitutes a parameter. The parameter name is the key field and must be unique. In projects with many parameters this naturally leads to standardization of the parameter name structure to include context information, clarifying what the parameter is, and making the name easier to guess by someone who needs it. A parameter value can be requested from the server by entering just the name on the first blank row of the *Inputs* sheet and pressing a *Request* button. If a parameter of that name has already been sent to the server from another subsystem, the server fills in the remaining fields; otherwise, the server registers the parameter as wanted. Importantly, supplementary information about where a value is calculated and what it represents is conveyed by the *Subsystem*, *Units*, and *Comments* fields. The latter of these can be used to indicate the method that generated the value, such as 'educated guess' or 'calculated via method X', or a degree of confidence such as 'take with a pinch of salt'.

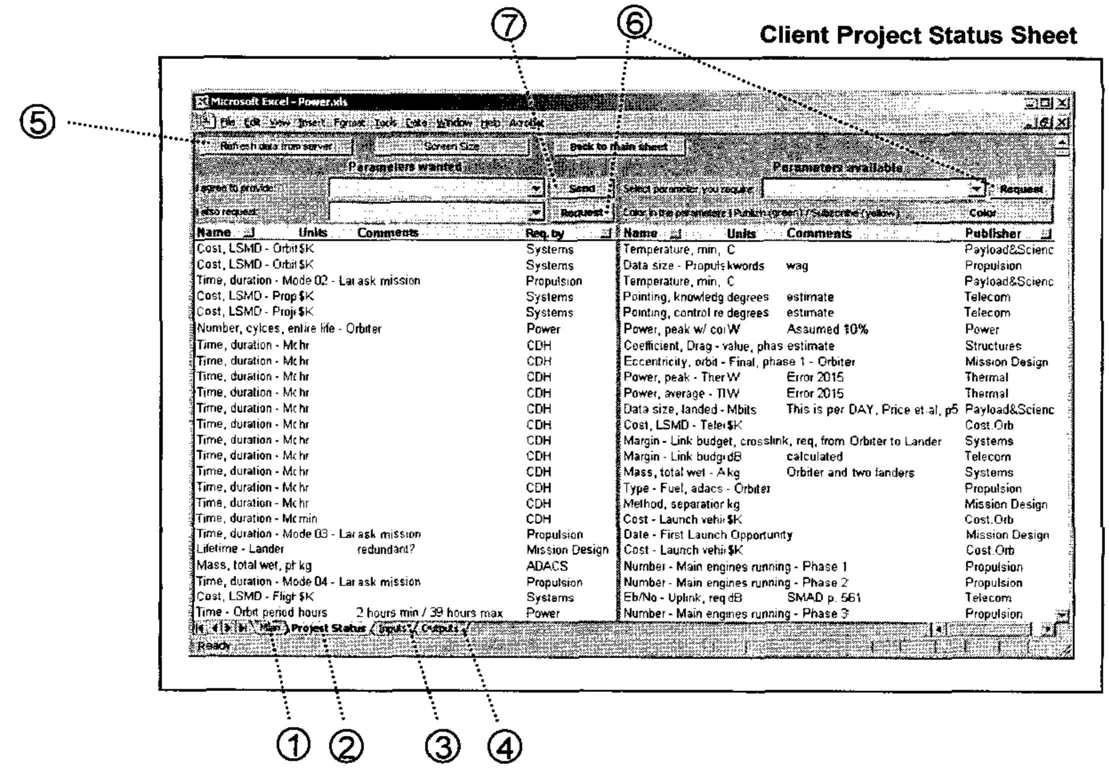
Similarly to the *Inputs* sheet, the *Outputs* sheet ③ requires a subsystem to explicitly declare data calculated within the subsystem workbook that is used by other subsystems. It contains columns labeled *Name*, *Value*, *Units*, and *Comments*. A parameter can be sent to the server by entering a name and any data in any other fields, and pressing a *Send* button. In combination, the *Inputs* and *Outputs* sheets represent the connectivity of the tool with the other subsystems via the server. Within the workbook, calculations reference input values from the *Inputs* sheet, and output values are referenced by the *Outputs* sheet.

The *Project Status* sheet ② provides an alternative menu-based method for requesting and agreeing to provide new parameters, instead of entering parameter names directly onto *Inputs* and *Outputs*. It is used mostly at the parameter auctioning stage, and initially has no data. Pressing the *Refresh Data From Server* button ⑤ populates the sheet with all the parameters held by the server. These parameters are categorized into two columns: *Parameters Wanted* and *Parameters Available*. When someone requests a parameter that is not provided by anyone, the server registers the parameter as wanted, and it appears on the left. Parameters that are provided by any subsystem are categorized as available, on the right. Drop-down lists allow the selection of a parameter, either to provide or request. Pressing button ⑥ or ⑦ adds that parameter to the corresponding *Inputs* or *Outputs* worksheet, and then communicates that change to the server. For example, if the *Power* subsystem (shown) agrees to provide the parameter named *'Number, cycles, entire life – Orbiter'* then the parameter moves to the *Parameters Available* category, on the right. By agreeing to provide this parameter, *Power* is obligated to calculate values, specify units and comments, and *Send* them. It has exclusive control of those fields until the parameter is released by removing the corresponding row from *Outputs* and pressing *Send.*

Use Case

There are two main phases to producing a design concurrently between team members. In the first phase, an initial idea is decomposed into subsystems, and data


Client Major Software Routines
CLIENT
Send
Receive
Parameter List
Server
Current Outputs Sheet
Current Inputs Sheet
Inputs Sheet with updated values
Populated Project Status Sheet
①
②
③
④
⑤
⑥
⑦


Client Project Status Sheet
Microsoft Excel - Power.xls
File Edit View Insert Format Tools Data Window Help Acrobat
Refresh data from server
Screen Size
Back to main sheet
Parameters wanted
I agree to provide:
Send
I also request:
Request
Parameters available
Select parameter you require
Request
Color in the parameters I Publish (green) / Subscribe (yellow)
Color
Main
Project Status
Inputs
Outputs
Ready
①
②
③
④
⑤
⑥
⑦

Name	Units	Comments	Req. by
Cost, LSMD - Orbit	$K		Systems
Cost, LSMD - Orbit	$K		Systems
Time, duration - Mode 02 - Lar ask mission			Propulsion
Cost, LSMD - Prop	$K		Systems
Cost, LSMD - Proj	$K		Systems
Number, cylces, entire life - Orbiter			Power
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	hr		CDH
Time, duration - Mc	min		CDH
Time, duration - Mode 03 - Lar ask mission			Propulsion
Lifetime - Lander		redundant?	Mission Design
Mass, total wet, pl	kg		ADACS
Time, duration - Mode 04 - Lar ask mission			Propulsion
Cost, LSMD - Fligh	$K		Systems
Time - Orbit period	hours	2 hours min / 39 hours max	Power

Name	Units	Comments	Publisher
Temperature, min,	C		Payload&Scienc
Data size - Propuls	kwords	wag	Propulsion
Temperature, min,	C		Payload&Scienc
Pointing, knowledg	degrees	estimate	Telecom
Pointing, control re	degrees	estimate	Telecom
Power, peak w/ cor	W	Assumed 10%	Power
Coefficient, Drag - value, phas		estimate	Structures
Eccentricity, orbit - Final, phase 1 - Orbiter			Mission Design
Power, peak - Ther	W	Error 2015	Thermal
Power, average - Th	W	Error 2015	Thermal
Data size, landed -	Mbits	This is per DAY, Price et al, p5	Payload&Scienc
Cost, LSMD - Tele	$K		Cost.Orb
Margin - Link budget, crosslink, req, from Orbiter to Lander			Systems
Margin - Link budg	dB	calculated	Telecom
Mass, total wet - A	kg	Orbiter and two landers	Systems
Type - Fuel, adacs - Orbiter			Propulsion
Method, separatior	kg		Mission Design
Cost - Launch vehi	$K		Cost.Orb
Date - First Launch Opportunity			Mission Design
Cost - Launch vehi	$K		Cost.Orb
Number - Main engines running - Phase 1			Propulsion
Number - Main engines running - Phase 2			Propulsion
Eb/No - Uplink, req	dB	SMAD p. 561	Telecom
Number - Main engines running - Phase 3			Propulsion

Figure 5 - Relations Between Major Software Functions and The GUI

relationships between the subsystems are formed. In the second phase, representative data values are exchanged between subsystems using the previously established data relationships.

We describe these two phases here for a generic project starting from scratch. Steps are modified or omitted in situations where prior experience of a project or project type exists. For example, spacecraft designs are typically decomposed into the same subsystems from one design to the next, with perhaps one or two changes of niche areas directly relating to the mission specialization, and have similar data relationships between subsystems even for very different types of spacecraft. In these cases the project decomposition can be skipped, as well as most of Phase 1.

Phase 1—Forming the parametric model:

1. Choose a team member to administrate the ICE process. Usually this member is designated the *Process Leader* or *System Engineer.*
2. Initial idea
    - 2.1. Brainstorm idea as a team, forming a list of the disciplines and expertise involved.
    - 2.2. Decompose the concept into subsystems, forming interfaces according to tradition or the "simplest interface" rule.
    - 2.3. Set up project folders and generate named subsystem workbooks for each subsystem.
    - 2.4. Usually a *Mission, Project, and/*or *Systems* subsystem is established to manage top-level mission figures of merit such as cost, lifetime, and (scientific or monetary) return. The *System Engineer* usually operates this subsystem.
3. Team discusses information required from each other and forms data relationships.
    - 3.1. Flowing down from the initial project requirements, individual team members list information needed to define a system that meets the given requirements.
    - 3.2. Each piece of needed information is given a name, often according to a naming convention, and entered into the *Inputs* sheet of the subsystem that needs it, to become a wanted parameter.
    - 3.3. All subsystems *Refresh Data From Server,* and the *Process Leader* or *System Engineer* runs down the *Parameters Wanted* category of the *Project Status* sheet verbally 'auctioning off' the parameters to the subsystems most suited to provide them.
    - 3.4. Subsystems *agree to provide* parameters, in turn needing more information, which may or may not already be provided in the *Parameters Available* category. By agreeing to provide a parameter, a subsystem assumes responsibility to calculate that value and is assigned exclusive control of it by the server; the parameter then moves to the *Parameters Available* category.
    - 3.5. The total *Parameters Wanted* initially snowball. Steps 3.2 to 3.4 are repeated until the system 'converges' and there are no more parameters remaining in the *Parameters Wanted* category. This represents the state where the data relationships are fully joined between subsystems with no disconnects. These data relationships can be tabulated by the server, as shown in Figure 4.
4. Blank Subsystems are transformed into design tools by constructing calculations and databases relevant to the subsystem role and needed to fulfill data responsibilities.
5. Without attempting calculation, each subsystem enters initial educated guesses as values in their *Outputs* and *Sends.*

Phase 2—'Steady state' data exchange:

1. With the necessary parameters established, calculations in place, and initial guessed values in the system, a first iteration occurs. In a process akin to Newton iteration, the subsystems *Request,* perform necessary calculations, and *Send,* iteratively until the major system quantities stabilize (converge). For a spacecraft design, major system quantities might be total mass, power consumption, or cost, and could be located in the *Mission* subsystem.
2. The *System Engineer* makes top-level design decisions or modifies requirements, and another iteration is performed. The process continues, and variants of the design are considered and characterized as determined by the *System Engineer,* eventually constituting a thorough examination of the design variations or 'trade space'.

4. IMPLEMENTATION AT JPL

ICEMaker has recently been adopted by Team-X at NASA JPL. JPL is NASA's lead center for robotic exploration of the solar system. Team-X is JPL's advanced project design team. The primary purpose of Team-X is twofold: to improve the quality of and reduce the time in completing JPL mission concepts through a study process with dedicated facilities, equipment, procedures, and tools [7], [8]. Team-X enables mission principal investigators and their design teams to plan new mission proposals efficiently. Team-X consists of 16 different subsystem (discipline) experts, a team leader, and a documentalist. Table 3 summarizes the different subsystems represented in Team-X. Each subsystem expert has a computer workstation to assist in his or her subsystem design. The team leader coordinates and leads the study and is the customer's primary contact before, during, and after study sessions. The team follows the integrated concurrent engineering (ICE) method discussed earlier. The documentalist establishes electronic files, records significant technical discussions, and ensures that study results are properly documented. Team-X products are mission design feasibility studies and reviews. A study lasts one to two weeks and results in a 30–80 page report that includes equipment lists, mass and power budgets, system and subsystem descriptions, and a projected mission cost estimate.

The Project Design Center (PDC) is the dedicated facility for Team-X design sessions. The different subsystem computer workstations in the PDC are connected to one another via a local area network (LAN). Microsoft Excel is the primary design tool used by the different subsystem experts at their workstations. Certain subsystem experts require additional software to assist in their design; for example, the Structures expert uses SolidWorks® 2000 for configuration analysis. From 1996 to January 2001, Team-X used Macintosh computers for all subsystem workstations. The Macintosh version of Excel at that time included the "publish" and "subscribe" functions that could pass data among LAN members. When Team-X switched to personal computers (PCs) in January 2001, a crude set of dummy worksheets and Visual Basic macros were added to each of the subsystem tools to pass data since the version of Excel for PCs did not support the "publish" and "subscribe" functions. Problems with data exchange in Team-X became significant enough during 2001 that the team leader and several subsystem experts began looking into a possible replacement for the dummy worksheets and Visual Basic macros.

ICEMaker was presented to Team-X as a solution in the fall of 2001. From December 2001 to May 2002 several Team-X members began a thorough implementation and testing of ICEMaker with the Team-X tools. ICEMaker clients were created for each of the subsystems listed in Table 3. Each subsystem tool was inserted into its respective client. Implementing ICEMaker involved two major steps: updating the Team-X parameter set to a new naming convention and linking this parameter set to the rest of the tool. Approximately 2000 unique parameters are passed among Team-X members. The parameter naming convention used by Team-X is:

```
Attribute, attribute modifier -
Product, product modifier
```

Many Team-X parameters were simple enough that they did not require the full parameter convention described. Below are some examples that the power subsystem passed to other Team-X members:

```
Assumptions, cost - Power

Mass, CBE dry - Solar array, per wing
with substrate

Efficiency - Solar cell

Number - Batteries, primary
```

The parameter export capability of ICEMaker Server was used to look for disconnects between the parameters requested and sent from one subsystem to another. The bulk of the implementation time was dedicated to linking the parameters on the *Inputs* and *Outputs* sheet to the rest of the tool for each subsystem. Testing the integrated Team-X – ICEMaker system began with two subsystems and was expanded subsystem by subsystem. It should be acknowledged that several of the features and improvements in ICEMaker Server, described earlier, were due to requests from members of Team-X discovered during this testing period. The official switch to ICEMaker occurred in May 2002, and it has been used successfully since.

As of August 2002, ICEMaker has also been licensed to United Technology Research Center (East Hartford, CT), Boeing Satellite Systems – CIEL Laboratory (El Segundo, CA), NASA Glenn Research Center – Design Center (Cleveland, OH), and the California State University, Northridge – Design, Analysis & Simulation Laboratory (Northridge, CA).

Table 3. Team-X subsystems

Attitude control
Command data systems
Configuration
Cost
Ground systems
Instruments
Mission design
Power
Program management
Propulsion
Science
Structures
Systems engineering
Telecommunications – System
Telecommunications – Hardware
Thermal control

5. CONCLUSIONS

As of August 2002, Team-X has completed approximately 20 studies using ICEMaker. Team-X members took on average one month to become comfortable with ICEMaker. There have been some minor implementation issues with Team-X but these have been addressed or circumvented.

At present, Team-X measures its performance by customer satisfaction. Customer satisfaction is based on a number of factors, including the quality, timeliness, and cost of a design study. Prior to adopting ICEMaker, Team-X already adhered to the ICE methodology, and so there was no significant decrease in the number of man-hours required for a design, and hence the cost remained about the same. However, the System Engineer has reported a marked increase in the quality of design studies produced using ICEMaker, and hence customer satisfaction has increased. These increases are subjective and have not yet been studied quantitatively.

In particular, the quality of design studies has improved because sharing data is faster and easier than with the previous system, enabling more in-depth study of design issues. The ability of the System Engineer to display information, such as parameter lists, has led to a greater degree of flexibility in leading the design effort. Unlike Team-X's earlier software, ICEMaker has enabled real-time studies with groups at other locations, further increasing the flexibility of their design process.

In summary, there have been fewer data exchange problems using ICEMaker than the previous system. New parameters and clients were successfully added on the fly during a session. The successful implementation of ICEMaker at JPL has demonstrated that ICEMaker can have a positive impact, even for organizations previously employing ICE methodologies.

ACKNOWLEDGEMENTS

We would like to thank Professor Fred Culick (Division of Engineering and Applied Science, California Institute of Technology); Matt Johnson, Bob Oberto, Theodore Sweetser, and Robert Moeller (Mission and Systems Architecture Section, JPL); Ray Baker (Thermal and Propulsion Engineering Section, JPL); and the many individuals whose suggestions contributed to the development of ICEMaker.

ICEMaker was developed for the Laboratory for Spacecraft and Mission Design (LSMD) at the California Institute of Technology in Pasadena, California. The two primary sponsors of the LSMD during the development of ICEMaker were JPL and TRW Space & Electronics (Redondo Beach, California).

REFERENCES

[1] Presley, S. and Neff, J., "Implementing a Concurrent Design Process 'The Human Element is the Most Powerful Part of the System'," 2000 IEEE Aerospace Conference Proceedings, March, 2000.

[2] Heim, J., et al., "TRW Process Improvements for Rapid Concept Designs," 1999 IEEE Aerospace Conference Proceedings, March, 1999.

[3] Wall, S., "Reinventing the Design Process: Teams and Models," International Astronautical Federation Specialist Symposium on Novel Concepts for Smaller, Faster and Better Space Missions, Redondo Beach, CA, April 1999.

[4] Sanders, G., "The Sky's the Limit for CIEL," *BSS World*, Vol. 3, Number 13, July 12, 2002.

[5] Bayt, R., et al., "Collaborative Engineering in Integrated Aircraft Power System Design," 2002 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference Proceedings, October, 2002.

[6] Sercel, J., "Integrated Concurrent Engineering (ICE): A Case Study in Engineering Productivity Improvement," to be presented at the 2003 IEEE Aerospace Conference, Big Sky, MT, March 2003.

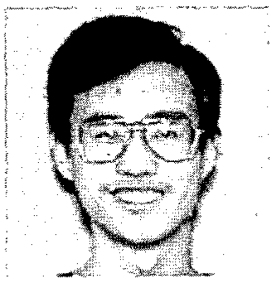
[7] Wall, S., "Team Structures and Processes in the Design of Space Missions," 1999 IEEE Aerospace Conference Proceedings, March, 1999.

[8] Mark, G., "Extreme Collaboration," *Communications of the ACM*, Volume 45, Issue 6, June 2002.

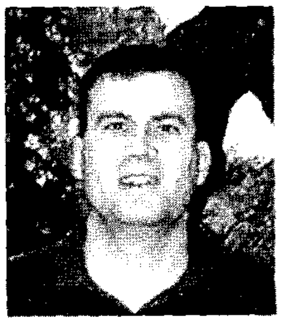
BIOGRAPHIES

Kevin Parkin *is a Graduate Student of Aeronautics at the California Institute of Technology. He holds an undergraduate M.Phys. degree in Physics with Space Science and Technology from the University of Leicester (England), and an M.S. in Aeronautics from the California Institute of Technology. In his first year as a graduate student he developed ICEMaker, later versions of which have been adopted by Team-X at NASA JPL, several large aerospace companies, and by SSPARC, an NRO-funded consortium comprised of Caltech, MIT, Stanford, and The Naval War College. His current research interests include modeling the feasibility and performance of advanced non-chemical launch concepts, particularly microwave rockets and related methods of propulsion.*

Joel Sercel *is the Director of The Laboratory for Spacecraft and Mission Design at Caltech. Sercel recently left JPL where he had a 15-year career in diverse areas including space technology development, system engineering, software development, and management. In addition to his work at JPL and Caltech, Sercel is the founder of ICS Associates. Through ICS Associates, Dr. Sercel teaches professional development courses in Integrated Concurrent Engineering (ICE); project planning and cost estimation; and Space System and Mission Design. Dr. Sercel received his Ph.D. and Master's Degrees in Mechanical Engineering from Caltech. His undergraduate degree was in Engineering Physics from the University of Arizona.*



Michael Liu *received his B.S. in Engineering and Applied Science from the California Institute of Technology in June 2002. His undergraduate coursework focused on areas of computer science such as 3D graphics, computation theory, and programming languages. He created the standalone version of ICEMaker Server in his senior year, and is now continuing development work on ICEMaker as an employee of SpreadsheetWorld, Inc. His current programming interests include C++, XML, and the Microsoft .NET platform.*



Daniel Thunnissen *is a Doctoral Student in Mechanical Engineering at the California Institute of Technology. His research interests include uncertainty in engineering design, probabilistic methods, and multidisciplinary design optimization. He is a former employee of the Jet Propulsion Laboratory (JPL) in Pasadena, California, and is currently an employee at User Technology Associates, Inc. in conjunction with pursuing his Ph.D. He holds a B.S.E in Aerospace Engineering from the University of Michigan (Ann Arbor) and an M.S. in Aerospace Engineering from the University of Illinois (Champaign-Urbana).*