Steps Towards a Robust Analysis of Procedure: New Formal Methods for Human-Machine Cooperative Tasks

Peter Bernard Ladkin
Bernd Sieker

Faculty of Technology and
Cognitive Interaction Technology Centre of Excellence (CITEC)
University of Bielefeld

13 July 2009

Abstract

Human error in commercial aviation has recently come again to the fore in three accidents, all within a month of one another in February and March 2009. We focus on a take-off accident in Melbourne. Various tasks involved in pre-flight cockpit preparation are crucial, but vulnerable to error, which may be caused by such phenomena as third-party interruptions in cockpit pre-flight procedures, whose intensity has been remarked by pilots and recently investigated by NASA. Were known techniques for fault-tolerance in computing to be applied to pre-flight cockpit preparation and more rigorously supported by computational devices, the major possibilities for human error could be avoided.

Basic Flying Skills

The basics of flying aircraft are instilled in pilots from their first day of flight. Two of those basics are, first, to calculate the aircraft's weight, and its balance point, before flying to ensure they are within the range for the aircraft to remain controllable in flight, and, second, when flying to ensure the airspeed always remains within range (too fast, and the airplane might break apart; too slow, and the wing stops flying, a condition known as a "stall").

Checking and cross-checking these basics is required, but also becomes routine. Aviators, airlines and regulators are all aware of the importance of these basics to safety, as well as the dangers of complacency and habit when executing critical but routine tasks. Three recent airline accidents, occurring all within a month of each other, have refocused attention on these basics, and how routine can go wrong.

Losing Control in Buffalo

On 23 February 2009, a turboprop aircraft of the "new generation", a Bombardier Q400, crashed on approach to Buffalo airport, New York state, killing 49 people on board and 1 on the ground. The crew had allowed the airspeed to become too slow, and reacted to the automatic warnings inappropriately, putting the aircraft into a climb, which slowed it down further, instead of a descent to gain more flying speed. The aircraft stalled, went out of control and spun into the ground [1]. The

U.S. National Transportation Safety Board convened a public hearing on 12-14 May 2009, which concentrated on studying the possible factors which could have led the crew to behave as they did.

Failing to Cope in Amsterdam

Two days later, the crew of a Turkish Airlines Boeing 737-800 jet on final approach to Amsterdam's Schiphol airport similarly allowed their airspeed to become too low, and although they initially reacted appropriately by putting the engines to full power, they did not ensure that the engines remained at full power: indeed, the aircraft's automation reduced the power back to idle thrust without the crew correcting, and the aircraft stalled, impacting farmland short of the runway. 9 of the 135 occupants were killed, including the three crew. Here, though, a faulty instrument, a radio altimeter, and the reaction of the airplane's autothrottle system to the faulty radio altimeter, played a role. The radio altimeter is one of two on board and is active when the aircraft is relatively close to the ground, to measure its height above the ground. The aircraft also has three barometric altimeters which are used to measure altitude throughout flight. The faulty radio altimeter had also malfunctioned during previous flights of the aircraft, and Boeing manuals (specifically, the "Dispatch Deviations Guide") say that the autothrottle is not to be used during approach and landing when the radio altimeter is malfunctioning. The reasons for this requirement become clear from the accident. At about 2000 ft above the ground on the approach, the faulty radio altimeter suddenly registered that the aircraft was on the ground. The autothrottle system read that data, went into "RETARD" mode and retarded the throttles to idle. As the airspeed became lower, the aircraft's warning systems activated and the throttles were advanced to full power (one presumes, by the crew, but this is not yet definitively established). However, the crew did not ensure the power remained at full: the throttles came back to idle power (one may presume due to the autothrottle, in RETARD mode, retarding the throttles, but this is not yet definitively established), and the aircraft descended and slowed still further. The autothrottle disconnected (or was disconnected). At 420 ft above the ground, the crew attempted a full recovery, putting throttles back to full and pitching down, but the aircraft was already too low to effect the recovery and hit the ground during the attempt [2]. Unlike the Buffalo accident, in which the crew apparently lost control of a perfectly-functioning aircraft, this accident involves malfunctioning systems which were not effectively handled by the crew. The accident was not inevitable. The crew had had some indication earlier in the approach that there was some malfunction – some warnings had sounded, indirect indications, but it appears the crew did not correctly diagnose their significance. Also, a similar malfunction of the radio altimeter had obviously been handled effectively by crews on previous flights of the aircraft. Some major airlines require as Standard Operating Procedure on this airplane, and some other pilots simply as a regular precaution, that at least one pilot has hisher hands permanently on the throttle levers at low altitude, precisely to ensure that the power is set, and stays set, as the pilots want it. Had the accident crew done this and ensured the engines stayed set at full power, their initial recovery would have sufficed and they would likely have proceeded to a routine landing. The investigation by the Dutch Safety Board will want to know how and why the malfunction was not logged on previous flights and the aircraft dispatched on the accident flight without registering with the crew the manufacturer's restriction that the autothrottle was not to be used for approach and landing. The Safety Board will also want to determine as far as possible why, after the crew's initial, correct, reaction to dangerously low airspeed, the crew did not ensure that the recovery continued as it should have done, but allowed the aircraft to revert to its previous, dangerously low airspeed condition, at a lower altitude with no margin for a second recovery. They will also be looking at the design of the automated systems and their interactions with other systems as well as with the pilots.

Scraping the Dirt in Melbourne

A third accident occurred in Melbourne, Australia on 20 March, 2009. A long-range Airbus A340-500 of Emirates Airlines almost failed to get airborne on its trip to Dubai. The aircraft, with 275 people on board, became airborne only after it had overrun Melbourne's Runway 16 and left traces in the grass at the end, partly demolishing some low-lying radio equipment belonging to the runway's Instrument Landing System. Unknown to the crew at that point, the aircraft's rump had scraped the ground hard as it pitched up for takeoff (a so-called "tailstrike"), removing some of the fuselage skin, dislodging the flight data recorder, and distorting the rear pressure bulkhead, a structural member essential for pressurising the cabin during high-altitude flight. Immediately after becoming airborne, the crew assessed the situation, and returned for landing after dumping fuel. During their assessment, they discovered incorrect data had been entered for the weight of the aircraft on the laptop computer they had used per procedure to calculate their final take-off parameters, most importantly the speed at which to pitch the aircraft upwards on take-off (the "rotation speed"), and the engine power required to get the aircraft airborne at that speed on that runway [3]. The aircraft landed and everybody deplaned normally, some somewhat shaken, but not all fully aware that they had been involved in what could have been Australia's worst-ever aviation accident. The extent of the damage to the rear fuselage became clear on inspection, and the Australian Transportation Safety Board, who were called, duly classified the event as an accident. Because of the incorrect data entry, the aircraft had used too little engine power on its takeoff roll, and was rotated at too low an airspeed, which creates extra wind resistance ("drag") and inhibits acceleration further. The crew had reacted correctly and promptly to not lifting off. The captain put the throttles to full takeoff power (significantly more power than is normally used during flight, and limited to a few minutes' use only) and the aircraft belatedly became airborne. It became apparent, again, to observers just how little time and space is available for a crew to react to problems on takeoff.

Pre-Flight Calculations

The entry error was significant. The ATSB says an entry error corresponding to 100 tonnes was made [3]. Since the empty weight of the aircraft is over 200 tonnes and the full weight under 400 tonnes, it follows that an error was made in the leftmost position of the number representing the weight (normally entered in kilogrammes); that instead of using the (correct) take-off weight of somewhat over 360 tonnes, the pilot calculating the performance data on the laptop computer had typed a "2" for a "3" and inadvertently entered 260 tonnes. This difference of 100 tonnes represents some 27% reduction of the total weight of the airplane during the power and rotation-speed calculation. No wonder that the airplane couldn't lift off at the computed speed! Such errors are not unknown, and there are procedural checks intended to catch such errors. How this erroneous value made it through the multitude of these checks will be a topic the investigators are expected to look at very closely.

Other Recent Weight Miscalculations

Entering the wrong take-off weight into the flight management computer is a known hazard, In March 2003, a Singapore Airlines Boeing 747 suffered a tailstrike on departure from Auckland airport, similarly the consequence of an undetected data-entry error for the weight, similarly 100 tonnes too light [4]. In Halifax, Nova Scotia, a Boeing 747 freighter operated by MK Airlines crashed on take-off on 14 October, 2004. The crew had apparently used a lower thrust setting and a lower rotation speed than what they needed – a symptom of an incorrect data entry for aircraft weight. The take-off performance data were usually computed by airline crews using a laptop computer (the "Boeing Laptop Tool", BLT). Direct evidence was hard to come by, since much had been destroyed during the accident. The Canadian investigators concluded that incorrect weight data entry into the BLT, and failure to recognise the weight and performance values as incorrect, were likely [5]. Whereas Singapore Airlines is recognised generally in the industry as having superior safety performance, the Canadian investigators were concerned about the apparently substandard safety performance history of MK Airlines.

Securing the Basics

Similar to all three accidents is that critical basics in what was superficially a normal situation were misperceived by the crew. At Buffalo and Amsterdam, that basic was airspeed and its control. At Melbourne, it was so-called “weight and balance”. At Buffalo and Melbourne, the aircraft was apparently functioning perfectly. At Amsterdam, its malfunction would not normally have been critical (or even so exceptional – many airline pilots have experienced a radio altimeter malfunction and coping with it is routine). At Buffalo and Amsterdam, the crew's reaction to the aircraft's unusual situation was not as procedures require; at Melbourne their reaction appears to have been exemplary.

The major difference between the accidents is that at Buffalo and Amsterdam, the critical parameter was presented to the crew directly on their instruments: namely, the airspeed. It is hard to see what, if anything, formal analysis of procedure would have to contribute to avoiding a repetition. At Melbourne, the critical parameter was some number tucked away out of sight after it was entered in the Flight Management and Guidance Computer (FMGC). It is a mistake that would not have been perceptually apparent to the crew until they rotated the aircraft on take-off – and then they would have just a second or two to react. In contrast to the other two incidents, we gain something with formal procedural analysis, as follows.

Handling Critical Parameters

Those circumstances – critical parameters that are hidden from, or at best only indirectly indicated to, a crew in the critical flight phases of take-off and landing – have been the subject of some of our recent work, using our methods **Ontological Hazard Analysis** (OHA) [6,7,8] and **Causal Flow Hazard Analysis** (CF HazAn) [9]. The second author has successfully used OHA to analyse the train dispatching protocols, the verbal commands exchanged between train driver and controller to control train movement on railway lines which are not equipped with signalling systems, in the German railway laws [8]. He also developed CF HazAn to analyse the procedures for a rejected take-off (RTO, one of the most hazardous manoeuvres in commercial jet transport aircraft) for a specific widely-used commercial transport [9].

OHA and CF HazAn may be used for the safety analysis of complex heterogeneous systems, in which components may be digital-electronic, mechanical, human, and procedural. The methods are formal, using the languages of formal logic and theoretical computer science, and adapt techniques known from formal proofs of computer-program correctness. Modern commercial aircraft can be considered to be systems involving mechanical and electrical/electronic/programmable electronic (E/E/PE) components such as flight computers and autopilots as well as human pilots. The flight computers and the human pilots interact to perform certain flight-critical functions. We call these **engineered multiagent cooperative functions** or EMC functions.

Consider the EMC function of calculating the take-off performance parameters. The most important of these are the so-called “V-speeds”: V_1, the maximum speed at which the take-off may be rejected; V_r ,the “rotation speed”, the speed at which the nose of the aircraft should be raised to lift off from the runway, and so on. Then there is the angle to which the nose of the aircraft should be raised upon rotation; as well as the so-called “balanced runway length”, the length of runway required to accelerate to take-off-reject speed, and then brake the aircraft to a stop - all, one hopes, on the runway. The performance parameters are affected by environmental conditions such as “pressure altitude” (a function of actual altitude and temperature), wind strength and direction, as well as by “weight and balance”, the total weight of the aircraft on take-off, and its center of gravity: these are “input” to the calculation of take-off performance parameters, which are “output” of the EMC function. The agents cooperating in this EMC function include the airline's Load Dispatch and Flight Dispatch departments, which calculate weight and balance from the passenger and fuel data as well as route data (needed to calculate the fuel required), the aircraft's Flight Management Computer (FMC, or as it is called on the Airbus A340 the Flight Management and Guidance Computer, FMGC), the pilots, the laptop computer used to perform certain computations, as well as various minor agents such as the refuelling agent (who must correctly inform Dispatch of the amount of fuel heshe has loaded). Just considering this list of who/what is involved in the EMC function gives a hint of the actual complexity of the function. It is such complexity which makes a formal approach to analysis not only worthwhile but in our view imperative.

Inputting the Weight Parameter: A Formal Analysis Using OHA

OHA may be used here as follows. OHA formally compares three models of the critical procedures, the takeoff performance weight parameter entry into the FMGC (the so-called "Zero-Fuel Weight", ZFW), using formal and semi-formal logic:

- the Requirement Model: pilot enters ZFW into FMGC
- the pilot's Cognitive Model: crew enters ZFW from Preliminary Flight Manifest (Prelim.ZFW) into FMGC; on receipt of Final Flight Manifest shortly before takeoff, crew enters Final.ZFW into FMGC, comparing Prelim.ZFW with Final.ZFW
- the Procedural Model: the series of specific reading and entry actions, including button-pushes on the various devices (FMGC, laptop computer) used to get the information electronically (say, from Dispatch via radio data-transfer directly to the FMGC using a data service called ACARS), or manually (say, from a paper Operational Flight Plan by typing it in on the keyboard of the FMGC, or from the Flight Plan to the laptop computer, and then typing the parameters calculated and displayed on the laptop into the FMGC). The Procedural Model is very close to what is written in the Operating Manual of the aircraft.

OHA now attempts a formal proof in logic that the Procedural Model formally emulates ("refines") the Cognitive Model, and that the Cognitive Model formally emulates ("refines") the Requirement Model. The formal proofs are not expected to work straight away. Extra assumptions will need to be introduced to get them to work. If the assumptions introduced for the Cognitive Model are violated, then it may be that the Requirement Model is not emulated, that is, that the correct ZFW is not entered into the FMGC. Similarly, if the assumptions for the Procedural Model are violated, it may be that what the pilot thinks heshe is doing (the Cognitive Model) is not actually fulfilled by the steps undertaken in the Operating Manual procedures (the Procedural Model). The violated assumptions in the Cognitive and Procedural Models, then, represent a formally complete list of all the things that can go wrong when entering the ZFW into the FMGC. They may not be expressed in the most intuitive language, but – and we consider this to be the critical contribution of OHA over other methods – we can be assured that they are all there in some form.

The procedures may be redesigned, if necessary, to reduce the assumptions, and thereby the possibility of their violation, to the minimum, and to introduce a specific focus on those specific handbook actions which are critical to maintaining the assumptions, rather than the more general unfocussed cautions which are found there at present.

We did not in fact perform a full OHA of this part of the take-off performance parameter calculation, because when setting up the analysis it became quickly evident where crucial weaknesses lie, as follows.

The Semi-Formal Analysis

In order to analyse how the Procedural Model might refine the Cognitive Model, we performed a CF HazAn of the *information flow* in the entry of the critical performance parameters. Software tools are available to check information flow, but as part of a commercial toolkit for correct-by-construction computer program development in the computer language Ada. However, already at the first stage of the analysis it became evident where some hazards lie. There are two values for ZFW in the Cognitive Model, but the Procedural Model (drawn from the Operating Manual of the aircraft, for a different airline, anonymised for our use) evidently contains six or so informationally-independent values for ZFW, that is, values that are not physically forced to be the same. There is a ZFW in the ACARS download, one in the FMGC which changes as one follows the procedures, one maybe on the paper flight plan, one transmitted verbally by the Load Dispatcher to the crew as part of the Final Flight Manifest, which is then written down by the crew (informationally-independent, for a mistake might be made), entered into the laptop (also similarly informationally-independent), and then read off the laptop and entered into the FMGC (also informationally-independent). Pilots intuitively understand which of these are supposed to be identical to Prelim.ZFW and which to Final.ZFW, and indeed the Manual cautions that values must be checked with care. But there are no specific procedures to *enforce* this identity, and - what aviation insiders are coming to believe might be crucial - there appear to be no specific ways to handle interruptions in the procedure and to recommence the procedures reliably after interruption.

Such procedures, for handling “abnormal” situations such as interruptions, have been known to computer scientists working with so-called distributed, real-time systems for decades. Such abnormal situations are known as **exceptions** and the methods as **exception handling**. They include methods such as **recovery blocks**, procedures designed to tolerate exceptions through **rollback** to a **recovery point**: when an exception is “raised” (seen), the computation is recommenced from an earlier point, the nearest recovery point, whose state (the values of the critical parameter) was saved precisely for that purpose [10, Chapter 2].

A computer scientist analysing an EMC function may classify interruptions and distractions as “exceptions”, but in the pre-flight procedures of commercial transport aircraft they have been the norm, and have been coming recently to the attention of researchers.

Distractions are Increasingly the Norm

Commercial-aviation insiders have been commenting on the level of distractions in an airliner cockpit previous to departure, and there has been considerable speculation in pilot Internet forums about the level of distraction that might have been present before the Melbourne accident. Regulations generally require a “sterile cockpit” process, in which the interactions between the pilots are verbally choreographed and no comments are permitted which are not directly pertinent to the specific task being performed. But such a process does not explicitly address the interruptions that necessarily come during the cockpit check from maintenance services, from the ramp service, from cabin crew, from Load Dispatch and Flight Dispatch, and from Air Traffic Control.

Recent research by NASA Ames Reseach Center researchers Loukia Loukopolos, Immanuel Barshi and Robert K. Dismukes, reported at the Flight Safety Foundation European Aviation Safety Seminar in Nicosia, Cyprus in March 2009 found hundreds of cases in which crews forgot to carry out, or to complete, checks and procedures [11]. They found 55 cases in which the correct configuration of the aircraft for take-off was inadvertently not set, including one case with a Boeing 737-800 aircraft at a major U.S. airport which bears some similarity to the Spanair MD-82 accident in Madrid on 20 August 2008, which killed 154 of 172 people on board when the aircraft failed to lift off the runway during take-off. The researchers proposed that the continual distractions during flight preparation, from the time when the crew arrived at the aircraft to commencing the take-off, appeared to be the common contributory factor in the incidents which they studied.

It is easy to see the significance of interruptions in these EMC functions in terms of information flow. An interruption breaks the flow of critical information from its initial location to its goal location, allowing "rogue" values to enter. Robust procedures would enforce the continuity of this flow, even during interruption.

Designing Robust, "Exception-Tolerant", Procedures

We believe that our formal approach aids the design of cockpit procedures which are more tolerant of interruptions and the procedural failures which interruptions induce than the largely informal design methods which are currently used. For the case of performance parameter calculations, formal analysis of the information flow derived through OHA using CF HazAn indicates that techniques derived from fault-tolerant computer programming could be introduced, as we have indicated above. We give some indication here of how this can be done.

1. The five or six informationally-independent ZFW values that appear at various points in the current Procedural Model must be coerced into just two values in the Cognitive Model: Prelim.ZFW and Final.ZFW. The laptop computer used for take-off performance data calculation could be programmed to contain, and display in succinct fashion, the entire state of the computation from entry of first data (from ACARS via a network connection to the data transfer device, or from a data scan of the paper "load sheet") to final data set (from the Final Flight Manifest). This would satisfy a principle we have identified in other work on EMC functions as an important evaluation criterion, and which is not satisfied by the current Procedural Model for take-off parameter data entry, namely the MCRPC:

- Mutual Cognisance of Relevant Parameters Criterion (MCRPC): Let the set of parameters of which knowledge is required to achieve a specific goal G be R. All agents cooperating to achieve the goal G must explicitly be cognisant of (must have cached) the true values of R [12].

  If one is to use a Recovery-Block approach to this task, then values which are to be continuously displayed would be not only the true actual values, but also the values from the last Recovery Point.

2. The data coercion required for correct emulation of the Cognitive Model can be explicitly enforced through the programming of the laptop.

3. The laptop could perform "run-time bounds checks": it could ensure that updated values of quantities such as ZFW remain within reasonable bounds of previous values, and require specific remedial action if not.

4. Data redundancy can be introduced and exploited. For example, the ZFW can be estimated from the aircraft empty weight (a constant) and the number of people on board, assuming a fixed average value for passenger-with-baggage. The estimated number of people on board can be entered at Preliminary Manifest data entry, an updated number at Final Manifest, and, separately, ZFW. The laptop can calculate and compare weight from ZFW and weight from empty+passengers as a cross check.

5. A "watchdog timeout" can be introduced for data entry to cope with interruptions in the cockpit: during entry of a block of data with specific procedural cross checks, if no action is performed by the computer user in a period of, say, 30 seconds, then the data state reverts to

that at the last recovery point, requiring that the entire block with cross checks be reentered from that point.

As mentioned above, most of these measures have been in the computer-science literature for decades [10]. Such measures enforce the continuity of the information flow. Indeed it would be a part of this formal approach to verify formally that the detailed Procedural Model emulates ("refines") the Cognitive Model, including interruptions.

A Broad Conclusion

It is ironic that introduction of automation has brought with it new hazards, such as incorrect but largely invisible critical data, and that even more automation, such as suggested above, promises a partial answer. One thing seems certain: the level of automation of critical aspects of flying commercial aircraft, including flight and performance data entry and storage, is increasing inexorably.

The recent history of software and algorithm development for critical systems has seen the informal methods of traditional software development yielding to mathematically more rigorous methods of ensuring that the system behaves as designed and intended, along with standards which encourage and in part require these more formal methods. We may expect the development of EMC critical functions to be similar - both EMC functions and software implement algorithms and a crucial aspect of dependability is to ensure that the implementation reflects the algorithm as intended, which can most satisfactorily be accomplished with formal methods.

This note has indicated one approach to the engineering of EMC functions, using the formal/semi-formal methods OHA and CF HazAn and how a relatively simple application of these methods could significantly reduce the opportunities for error in the critical function of calculating take-off performance parameters, even in an interruption-laden environment. Pilots should no longer be blamed for, nor passengers, airports and rescue crews, be subjected to the consequences of pilot "negligence" when it is possible for engineers to design it out.

References

[1] National Transportation Safety Board, Washington, D.C. Public docket on accident to Bombardier DHC-8-402 aircraft, Clarence Center, N.Y., 12 February, 2009.
www.ntsb.gov ⇒ Aviation ⇒ Major Investigations ⇒ Event Date 2/12/2009

[2] The Dutch Safety Board. Preliminary Report, Accident to Boeing 737-800 aircraft TC-JGE, 25 February, 2009. www.onderzoeksraad.nl/docs/rapporten/Prelimenary_EN_def.pdf

[3] Australian Transport Safety Board. Transport Safety Report, Aviation Occurrence Investigation, AO-2009-012, Preliminary. Tail Strike, Melbourne Airport, Vic. 20 March 2009. A6-ERG. Airbus A340-500.
www.atsb.gov.au/publications/investigation_reports/2009/AAIR/pdf/AO2009012_Prelim.pdf

[4] Transport Accident Investigation Commission, Investigation 03-003. Boeing 747-412. 9V-SMT, flight SQ286, tail strike during take-off, Auckland International Airport, 12 March 2003. Final Report. Available from www.taic.org.nz ⇒ Aviation Reports ⇒ 03-003

[5] Transportation Safety Board of Canada. Aviation Reports – 2004 – A04H0004.
www.tsb.gc.ca/eng/rapports-reports/aviation/2004/a04h0004/a04h0004.asp

[6] P. B. Ladkin, Ontological Analysis, Safety Systems 14(3), May 2005.

[7] Jörn Stuphorn, Bernd Sieker and Peter B. Ladkin, Dependable Risk Analysis for Systems with E/E/PE Components: Two Case Studies, in Chris Dale and Tom Anderson, eds., Safety-Critical Systems: Problems Process and Practice; Proceedings of the Seventeenth Safety-Critical Systems Symposium, Brighton, UK, 3-5 February 2009. Springer-Verlag London Limited, 2009.

[8] Bernd Manfred Sieker, Systemanforderungsanalyse von Bahnbetriebsverfahren mit Hilfe der Ontological Hazard Analysis am Beispiel des Zugleitbetriebs nach FV-NE, Doctoral Thesis submitted to the Faculty of Technology, University of Bielefeld, May 2009.

[9] Bernd Sieker, Examples of Reverse Engineering, Causalis Limited Technical Report, www.causalis.com ⇒ Publications ⇒ 200805 Reverse Engineering Examples

[10] Alan Burns and Andy Wellings, Real-Time Systems and Programming Languages, 4th edition, Addison-Wesley, 2009.

[11] David Learmount, Distractions frequently cause flapless takeoffs, NASA reveals, Flightglobal, 24 March 2009.
www.flightglobal.com/articles/2009/03/24/324207/distractions-frequently-cause-flapless-take-offs-nasa.html

[12] Peter Bernard Ladkin, Some Principles of Multiagent Cooperative Behavior, e-mail message to Don Hudson et el., 11:34:46 GMT, 8 March, 2009.