# Session NL: The importance of being Urgent

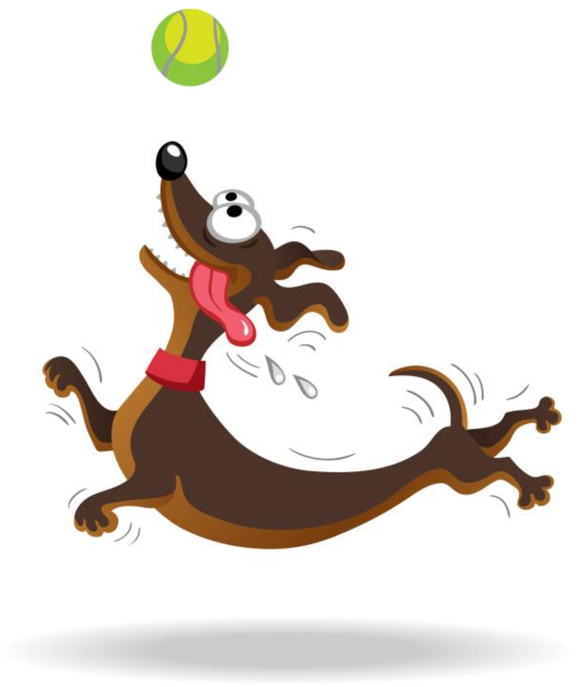
A cautionary tale of an eager puppy called Urgent

## Introducing Urgent

- This is Urgent
- He has Eager Puppy Syndrome
- He’s just like IBM Workload Scheduler for z/OS
- He’s always desperate to run for you

## What is Eager Puppy Syndrome?

- You have two balls in your hand, puppy can see both
  - Ball 1 – Old, ripped to pieces
  - Ball 2 – New and squeaky
- You throw ball 1, puppy runs off with it
- You throw ball 2, puppy still playing with ball 1



## The IWSz Eager Puppy

- IWSz only compares jobs ready at the same time
- Even if another job is about to be ready shortly
- "IWSz will happily run a long running unimportant job
  - even if 1 second later you have a high priority job ready"
- Let's find out why

## Put away prior preconceptions of priority

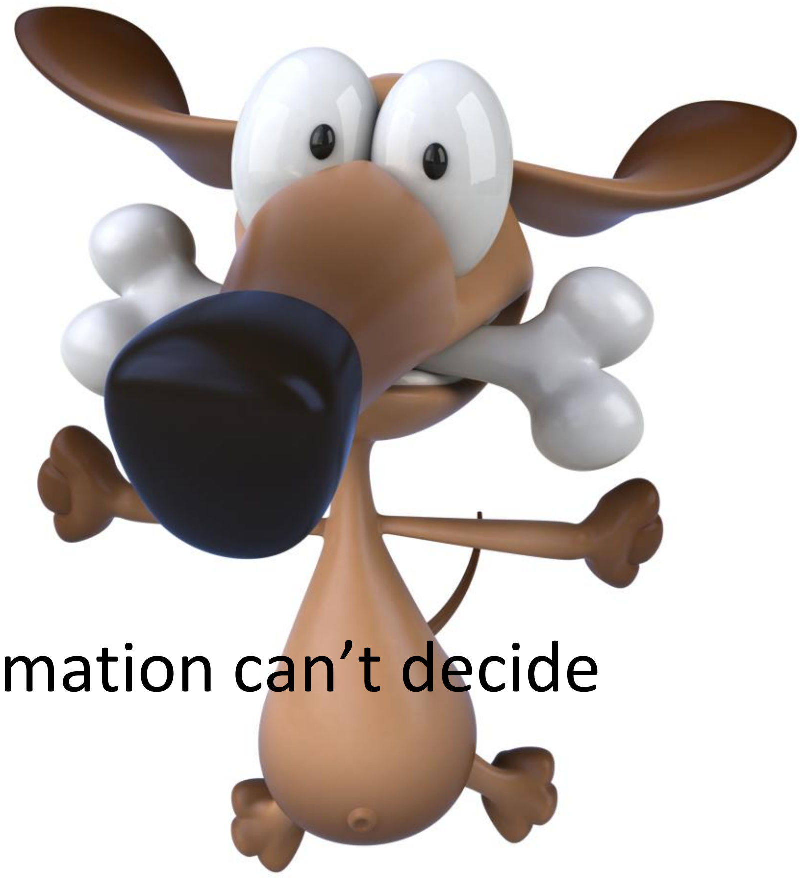
- Priority is purely subjective, not objective
  - It does not take into account absolutes such as when the job MUST end for business success
  - Real business rules, such as deadlines will allow IWSz to objectively manage workload
- Ask your customer which jobs are high priority
  - I think you all know what the answer will be
- The reality is a little different
  - A job can be the most critical one day
  - Other days not so much
- It all depends on when it has to be completed

## What priority really does (Chapter 15)

- The "next job" is selected as follows –

1. Is it urgent ? (priority 9)
2. Is it the earliest latest start time ?
3. Is it the highest priority ? (for priorities 1 – 8)
4. Is it the shortest estimated duration ?

- Priority is only the tie-breaker when objective information can't decide
  - and then only for jobs ready at exactly the same time



## What priority doesn't do

- Much ☺
- It has NO influence on the job after submission
- It does NOT give a job more resources on z/OS
  - WLM Service class does that
  - Workload Service Assurance influences that – not priority
- It does NOT make a job run faster on any platform

## What you shouldn’t do with priority

- Don’t EVER define an application with Priority 9
  - This is IWSz’s wiggle room when things go late
  - If you use it yourself, you hinder IWSz being able to react
- Don’t spend a lot of time worrying about setting priority
  - It only ever comes into play when 2 jobs are ready simultaneously
  - Define most of your work with the same priority e.g. 5
  - Only vary from that for fine tuning occurrences ready together
- Why can’t I set priority at job level?
  - Because that just increases things you can waste time with

## Are durations and deadlines important?

Yes!

## Are durations and deadlines important?

- IWSz used to be known as OPC
  - Operations Planning and Control
  - The clue is in the name
- Durations and deadlines have always been important
  - Earliest latest start time is crucial to what runs next
  - Only recently though have they been manageable

## Why are deadlines important?

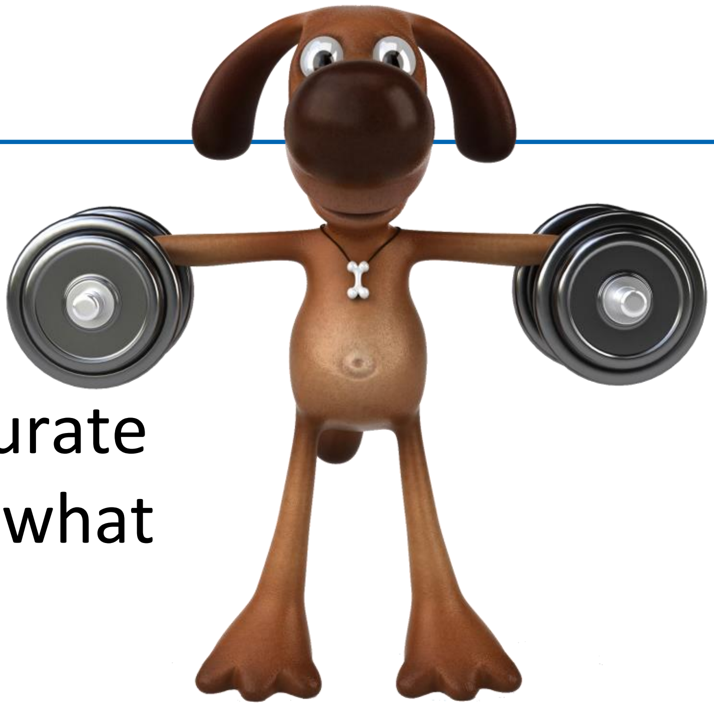
And should only really be set with genuine reason

- Deadlines are business rules
  - They define when things must be complete by
  - Meet your deadlines, have business success

- With accurate deadlines, durations and dependencies
  - IWSz can calculate the latest start time for every job
  - It can then make a valued judgement of what to submit next

## So how do I get it to work?

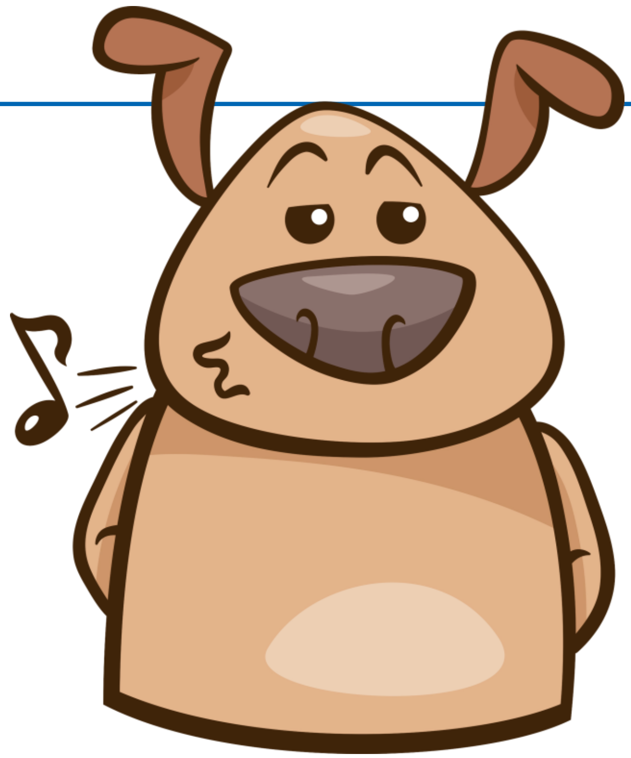
- There are 2 key elements



1. Making sure your durations and deadlines are as accurate as possible so IWSz can make valued judgements for what should come next

2. Taming the eager puppy to reduce the amount of non-critical jobs coming ready at busy times

## Historical basics (Whistlestop tour)



- Estimated duration is defined per operation in the AD

- When a job runs –
  - The new duration is compared with Limit for Feedback
  - If within limits the smoothing factor applies the difference to the AD
  - Outside the limits the operation hits the Missed Feedback Report

# Limit for feedback

- The point of limit for feedback is to weed out anomalies
  - Lower Limit = Old Duration * 100/Limit
  - Upper Limit = Old Duration * Limit/100
  - Defined globally and optionally per operation
  - Global Upper Limit also sets long duration alerts, local upper limit only limits feedback

## Examples

100 = Ignore new duration (both limits are old duration)

200 = From half old duration to twice old duration

500 = From fifth old duration to five times old duration

## Smoothing factor

- Essentially the percentage of the difference to store
  - Can be set globally and optionally per job

Examples

0 – Take no new duration

50 – Take half the difference between the old and new (average)

100 – Take the whole new duration

## The old catch up game

- It’s a lot of work to get accurate durations
- Long duration alerts can be problematic for short jobs
- If your first duration was out, it may never feedback
  e.g. With LF=999 a 1 sec job only needs to run for 10 sec to miss
- If your job duration varies wildly by run it may never be accurate
- Jobs moved to dummy workstations retain inaccurate duration

## What have the Romans ever done for us?

- All 9.x versions
  - LIMFDBK(0) – Lets you capture all values
  - ALEACTION – Improves long duration alerts
- From 9.3
  - FIRSTFDBK – Takes first run time regardless of LIMFDBK
  - Variable durations – Allows estimated durations to vary by day
  - None reporting workstations all considered 1 second

## Getting in shape

```
JTOPTS FIRSTFDBK(YES)
       LIMFDBK(0)
       ALEACTION(200,300)
       SMOOTHING(50)
```

Means all new jobs will take 1st run time

Takes all durations
No missed feedback

Sets long duration alert

Twice as long
At least 5 minutes

50 = running average
100 = actual values

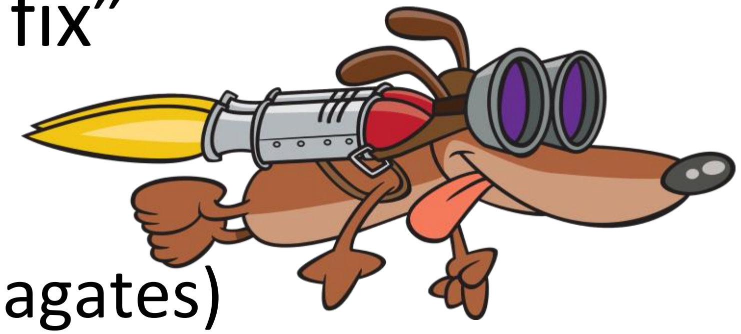
- Create Variable Duration rules for predictable variation
- Lock durations for unpredictable variations
  - Set "worst case" duration
  - Set operation smoothing factor to zero

## Staying in shape

- Option 1
  - Set LIMFDBK to something you feel appropriate
  - Regularly review the missed feedback report
- Option 2 (controversial)
  - Keep LIMFDBK at zero
  - Deal with anomalies when they occur

## Getting deadlines in shape (Quick)

- Historically every application must define a deadline
  - Even though only end points really need them
- Workload Service Assurance required a "Quick fix"
  - BATCHOPT IGNOREDEADL(YES)
  - Only honors deadlines of Critical Operations (& propagates)
  - All others considered "end of tail plan" (7 days hence)
  - WARNING: This prevents setting of deadlines other than critical ops



## Getting deadlines in shape (Long)

- Run cycles groups removed requirement for deadlines
  - If set to blank they go to end of tail plan (7 days hence)

- WAPL can help you find them –

```
//RUNWAPL   EXEC EQQYXJPX
//OUTDATA   DD SYSOUT=*
//SYSIN     DD *
OPTIONS STRIP(Y) SHOWDFLT(N)
OUTPUT ADRUN DATA(OUTDATA)
FIELDS(ADCOM.ADID,ADRDD,ADRDT)
LIST ADCOM SELECT(Y)
```

## Finding deadlines to drop

- Which gives you

```
ADRUN ADID=DH#PLANNING ADRDD=0 ADRDT=1300
ADRUN ADID=DH#Z2D001 ADRDD=0 ADRDT=0000
ADRUN ADID=DH#Z2D002 ADRDD=0 ADRDT=0000
ADRUN ADID=DH#Z2D003 ADRDD=0 ADRDT=0000
ADRUN ADID=PERFX ADRDD=0 ADRDT=2359
ADRUN ADID=PERFX100 ADRDD=0 ADRDT=2359
ADRUN ADID=PERFX1000 ADRDD=0 ADRDT=2359
ADRUN ADID=PERFX2 ADRDD=0 ADRDT=2359
ADRUN ADID=PERFX200 ADRDD=0 ADRDT=2359
```

Empty Deadlines

- Then you can use WAPL to drop deadlines from selected ADs

## Dropping deadlines

- This WAPL job generates batch loader

```
//RUNWAPL  EXEC EQQYXJPX
//OUTBL    DD SYSOUT=*
//SYSIN    DD *
OPTIONS STRIP(Y) SHOWDFLT(N)
LOADDEF AD* DATA(-)
OUTPUT ADRUN LOADER(=)
       FIELDS(ADRPER,ADRVALF,ADRVALT,ADRUNDESC,ADRUNRULE,ADRTYPE,
              ADRIAD,ADRIAT,ADRJVTAB,ADRSHTYPE,ADRINPOS,ADRINNEG,
              ADRIADPOS,ADRIADNEG,ADRREPEATEVRY,ADRREPEATENDT,
              ADRSHIFT,ADRSHSIGN,ADRULET)
LIST ADCOM ADID(AD#FERRYMAN) SELECT(Y)
LIST ADCOM ADID(DH#FROGGYMAN) SELECT(Y)
LIST ADCOM ADID(HO#OMABRIDGE) SELECT(Y)
```

ADRDD & ADRDT missing from FIELDS which suppresses deadline from Batch Loader

Code a LIST with SELECT for every application that you want to drop deadlines from

- Creates almost identical AD but without deadlines

## Practical uses of deadline smoothing

## Taming the eager puppy

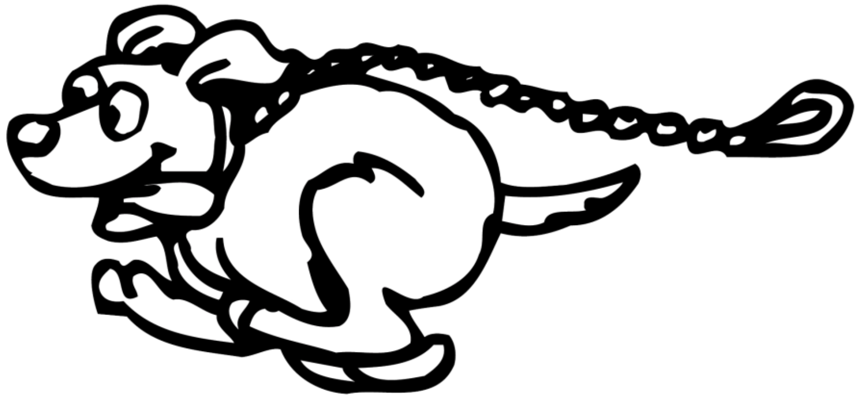
- It doesn't matter how accurate your deadlines are -
  - If a low priority job becomes ready on its own, it runs
  - Even if in peak hours
- You need to find a way to put your puppy on a lead
  - Sadly at present IWSz will not do this for you
  - At present…

## The workstation harness

- Identify your low priority jobs (can’t help you there)
- Move them to a new workstation for the destination
- This workstation can then be constrained
  - Use intervals to reduce parallel servers in busy periods
  - If workload really struggles constrain workstation manually
- A Special resource may be an alternative constraint
  - This is not as visible as a workstation
  - Especially to automation

## Letting the puppy off the leash

- Assuming you've already got deadlines in shape
- When a low priority job goes late, it still needs to run
- Use automation to react to late alerts on this WS
  - Move such jobs to their non-constrained counterpart



## At present…

- Do I need to do this if HCL are going to make it better?
- It’s eager puppy taming that is being investigated
  - To lose the habit of running stuff way before needed
  - To make job selection a little more prescient
- But to decide the best order to run, you still need to know when a job MUST be complete by

Durations and Deadlines ARE important

HCL

Relationship™

BEYOND THE CONTRACT

**$7.6** BILLION ENTERPRISE | **119,000** IDEAPRENEURS | **32** COUNTRIES

# Session NL
# The Importance of being Urgent